# Anomaly detection using knowledge graphs and synergistic reasoning : application to network management and cyber security

Lionel Tailhardat

EURECOM
Sophia Antipolis

SORBONNE UNIVERSITÉ

# PHD THESIS

In Partial Fulfilment of the Requirements for the
Degree of Doctor of Philosophy from Sorbonne University
Specialization: Data Science

# Anomaly Detection using Knowledge Graphs and Synergistic Reasoning: Application to Network Management and Cyber Security


## Lionel TAILHARDAT


Defended on 30/09/2024 before a committee composed of:

| | |
|---|---|
| Reviewer | **Oscar CORCHO**, Facultad de Informática Boadilla del Monte, Madrid, Spain |
| Reviewer | **Olivier FESTOR**, Lorraine INP - Université de Lorraine, Vandoeuvre-lès-Nancy, France |
| Examiner | **Anastasia DIMOU**, KU Leuven, Louvain, Kingdom of Belgium |
| Examiner | **Adlen KSENTINI**, EURECOM, Sophia Antipolis, France (President of the Jury) |
| Thesis Director | **Ulrich FINGER**, EURECOM, Sophia Antipolis, France |
| Thesis Co-Director | **Raphaël TRONCY**, EURECOM, Sophia Antipolis, France |
| Thesis Co-Director | **Yoan CHABOT**, Orange, Belfort, France |

*Dedicated to my family and friends,*

*as well as to the artists*
*(authors, directors, musicians, scientists)*
*who, through their works, have left traces of*
*this desire to explore the universe one step further,*
*even if it's challenging.*

# Preface & Acknowledgements

The feel of the $20^{th}$ century I had growing up and studying at the university was that we, as humans, could literally achieve any goal, even reaching to the moon. Yet, when I started working in hi-tech during the 2000s, I felt a contrast: although my colleagues and I were mastering the art of deploying and managing broad-scale Information and Communications Technology (ICT) systems, we were still struggling with some dark corners related to the quality of service for complex systems. Improving or changing the configuration of the network here would break it somewhere else, leading to faults potentially unnoticed and hours spent trying to understand the "why and how" of it. So how could it be that reaching to the moon seemed more achievable than dealing with very terrestrial things? At some point in time, it became obvious to me that dealing with complex networks would need more than just predictive maintenance plans, local technical patches or nice looking user-interfaces in monitoring tools. It would rather need something yet to be defined that would allow one to intuitively grasp the overall functioning of the system, including its inter-dependencies, network operations, and the organizational perspective. But what is worth building, and how can we make it sustainable and useful in the long term? Retrospectively, this is where my PhD journey began, several years after completing my initial years of study at the university.

While I did not pursue a doctoral journey immediately after university, many have played a crucial role in shaping who I am and my path towards research. I express my sincere gratitude to them, starting with my colleagues from my early days at GlobeCast France – Christophe Meslot, Christian Tailleux, Gérard Gay, Philippe Guiot, Stéphane Huard, and David Maistre – who supported me in the execution and management of multi-technical projects. At Orange France, Laurent Guillet, Olivier Gbedjeha, Mehdi Lebon, and others introduced me to the intricacies of critical networks while granting me the necessary leeway to develop operational support tools. CNAM[1] professors – Didier Le Ruyet, Mohamed Ketata, Michel Terré, Daniel Roviras and Pascal Chevalier – encouraged me to continue with research while I was doing my later Master's studies. Imen Grida Ben Yahia and Baptiste Olivier (Orange researchers at the time we met) showed interest in my work on network incident diagnosis using graphs and sparked my current research collaborations.

[1] Conservatoire National des Arts et Métiers

From these PhD study years, I would like first to express my gratitude to my PhD supervisors – Yoan Chabot and Raphaël Troncy – who made me experience this incredible adventure by sharing their standards, knowledge, and expertise while making me feel that I was part of a team. Similarly – Sok-Yen Loui, Adam Ouorou, Catherine Chevanet and Eric Gourdin – who also put their trust in me and enabled full-time engagement in my research project as a Researcher at Orange Innovation. Thank you as well to my PhD monitoring committee – Anastasia Dimou and Aurélien Françillon – for your external perspective and advice on how to embrace this journey. And, of course, many thanks to my PhD Jury – Oscar Corcho, Olivier Festor, Anastasia Dimou and Adlen Ksentini – for taking the time to assess and acknowledge my work, and for challenging me with insightful questions during my PhD defense.

I also have a strong thought for the students I supervised during these years – Mihary Ranaivoson, Yash Agarwalla, Dario Ferrero, Benjamin Stach and Ovidiu Pascal – who shared their enthusiasm and energy with me on parts of my project.

A special thanks too to my project team – Perrine Guillemette and Antoine Py – as the strength of the algorithms and solutions I worked on would have meant little without a nice User Interface/User eXperience (UI/UX) to intuitively convey their utility.

Gratitude also goes to the Orange DATA-AI board – Steve Jarrett and Olivier Simon – for funding parts of the project, and to Orange operations experts – Maria Davidson, Denis Lallée, Olivier Nicot and Pierrick Guillou – for rallying decision-makers around our initial concept. To Bertrand Decocq, who then took on him to organize with enthusiasm the future deployment of the project within the company. Lastly, Camille Barboule and Adrien Bufort joined with their research expertise, and I'm excited for the outcomes of their contributions.

Short, fun and informal discussions can be just as meaningful and inspiring as formal ones in research. I can't name everyone, so please don't be upset if I miss you, but I'm thinking of my Orange DATA-AI/MORE team, of Mikael Touati who enjoys "category theory" as much as I do, and Sylvain Allio who makes complex issues seem easy and fun. Thanks to the EURECOM Semantics research group and the W3C Knowledge Graph Construction community group. Thanks also to Pierre-Antoine Champin and Gérard Berry for their valuable feedback that boosted my confidence in my emerging ideas.

Last but not least, I wish to thank my dear family and friends for their unconditional support, pride, and interest, even when I tried to explain concepts far away from everyday concerns. Thomas and Zoé-Jane, it was fun to be a student alongside you during your school years ;-)

The research field and application domain connected to this work can be highly technical and abstract. But ultimately, it is all about helping people grasp the world's complexity, improve mutual understanding, and collaborate more effectively. Making this possible and sustainable,

potentially leveraging technology, can be very challenging and demanding. But I am sure it is a worthwhile and rewarding journey.

Sophia Antipolis, Autumn 2024 ***Lionel Tailhardat***

# Abstract

Incident management on telecom and computer networks, whether it is related to infrastructure or cybersecurity issues, requires the ability to simultaneously and quickly correlate and interpret a large number of heterogeneous technical information sources. In this thesis, we study the benefits of structuring this data into a knowledge graph, and examine how this structure helps to manage the complexity of networks, particularly for applications in anomaly detection on dynamic and large-scale networks. Through an ontology (a model of concepts and relationships to describe an application domain), knowledge graphs allow different-looking information to be given a common meaning. We first introduce a new ontology for describing network infrastructures, incidents, and operations. We also describe an architecture for transforming network data into a knowledge graph organized according to this ontology, using Semantic Web technologies to foster interoperability. The resulting knowledge graph allows for standardized analysis of network behavior. We then define three families of algorithmic techniques for using the graph data, and show how these techniques can be used to detect abnormal system behavior and assist technical support teams in incident diagnosis. Finally, we present a software architecture to facilitate the interactions of support teams with the knowledge graph and diagnostic algorithms through a specialized graphical user interface. Each proposal has been independently tested through experiments and demonstrations, as well as by a panel of expert users using the specialized graphical interface within an integrated solution.

# Abrégé

La gestion des incidents sur les réseaux informatiques et télécoms, qu'il s'agisse de problèmes d'infrastructure ou de cybersécurité, nécessite la capacité de traiter et d'interpréter simultanément et rapidement un grand nombre de sources d'information techniques hétérogènes en matière de format et de sémantique. Dans cette thèse, nous étudions l'intérêt de structurer ces données dans un graphe de connaissances, et étudions en quoi cette structure permet de maîtriser la complexité des réseaux, notamment pour des applications en détection d'anomalies sur des réseaux dynamiques et de grande taille. À travers une ontologie (un modèle des concepts et relations pour décrire un domaine d'application), les graphes de connaissances permettent en effet de donner un sens commun à des informations différentes en apparence. Nous introduisons pour commencer une nouvelle ontologie permettant de décrire les infrastructures réseaux, les incidents, et les opérations d'exploitation. Nous décrivons de même une architecture pour transformer les données des réseaux en un graphe de connaissances organisé selon cette ontologie, en utilisant les techniques du Web Sémantique pour favoriser l'interopérabilité. Le graphe de connaissances résultant permet d'analyser le comportement des réseaux de façon homogène. Nous définissons ensuite trois familles de techniques algorithmiques pour utiliser les données du graphe, et montrons comment ces techniques peuvent être utilisées pour détecter des comportements anormaux des systèmes et aider les équipes d'exploitation dans le diagnostic d'incidents. Enfin, nous présentons une architecture logicielle pour simplifier les interactions des exploitants avec le graphe de connaissances et les algorithmes d'aide au diagnostique par l'intermédiaire d'une interface graphique spécialisée. Chaque proposition a été testée de manière indépendante par des expérimentations et démonstrations, ainsi que par un panel d'utilisateurs experts depuis l'interface graphique spécialisée dans le cadre d'une solution intégrée.

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

**AI** Artificial Intelligence.

**AIS** Alarm Indication Signal.

**API** Application Programming Interface.

**AT** Authoring Test.

**BPMN** Business Process Model Notation.

**CPU** Central Processing Unit.

**CQ** Competency Question.

**CTI** Cybersecurity Threat Intelligence.

**DAG** Directed Acyclic Graph.

**DBMS** DataBase Management System.

**DL** Description Logic.

**DSS** Decision Support System.

**ETL** Extract-Transform-Load.

**FMEA** Failure Mode and Effect Analysis.

**FSA** Finite State Automaton.

**GBAD** Graph-Based Anomaly Detection.

**I/O** Input/Output.

**ICT** Information and Communications Technology.

**IMP** Incident Management Process.

**IoT** Internet of Things.

**IP** Internet Protocol.

**IRM** Incident Response Management.

**IT** Information Technology.

**ITSM** IT Service Management.

**ITU** International Telecommunication Union.

**KG** Knowledge Graph.

**KGC** Knowledge Graph Construction.

**KR** Knowledge Representation.

**KRR** Knowledge Representation and Reasoning.

**LOT** Linked Open Terms.

**MCS** Management and Control System.

**ML** Machine Learning.

**NetOps** Network administration and Operations.

**NLP** Natural Language Processing.

**NMS** Network Monitoring System.

**NOC** Network Operation Center.

**NORIA** machine learNing, Ontology and Reasoning for the Identification of Anomalies.

**OS** Operating System.

**OSS** Operations Support Systems.

**OTN** Optical Transport Network.

**OWL** Web Ontology Language.

**QoS** Quality of Service.

**RCA** Root Cause Analysis.

**RDF** Ressource Description Framework.

**RDFS** RDF Schema.

**RML** RDF Mapping Language.

**SE-SI** Smart Environment & Smart Industry.

**SecOps** Cyber security Operations.

**SIEM** Security Information and Event Management.

**SKOS** Simple Knowledge Organization System.

**SLA** Service Level Agreement.

**SNMP** Simple Network Management Protocol.

**SOC** Security Operation Center.

**SPARQL** SPARQL Protocol and RDF Query Language.

**SPIN** SPARQL Inferencing Notation.

**SUS** System Usability Scale.

**SWRL** Semantic Web Rule Language.

**UEBA** User and Entity Behavior Analytics.

**UI/UX** User Interface/User eXperience.

**URI** Uniform Resource Identifier.

**URL** Uniform Resource Locator.

**VM** Virtual Machine.

# Chapter 1

# Introduction

When managing large-scale IT & telco networks (broadband international backbones, corporate networks, Internet access networks), one is sooner or later involved into handling complex incident situations, such as general IT service disruption because of cascading failures or cyber-attacks. For incident management, technical support teams typically leverage information from decision support tools like Network Monitoring Systems (NMSs) or Security Information and Event Management (SIEM) systems. These tools often use an elementary representation of the network infrastructures and services. Basically, an IT network is a set of computers, routers, and other devices connected and configured to allow data processing and sharing. Similarly, an IT service is the usage of this processing and sharing capability for specific purposes, from the most trivial ones (entertainment, ticket booking, home automation) to more challenging ones (stock exchange, road lights, or nuclear plant management).

Although obvious at first glance, this level of description is not sufficient to scale up for maintaining high-standard quality of service on large-scale networks. This is due to the heterogeneity of the Information and Communications Technology (ICT) systems that compose them, making incident diagnosis and remediation a challenging task: to ensure network services function properly, supervision teams need to understand information from diverse and dynamic technical systems. For example, one can consider a service architecture that combines Virtual Machine (VM) distributed across data centers, which are interconnected through an IPoDWDM[1] network. To achieve efficiency, it is necessary to integrate and correlate data from various sources. This includes data from VM management tools, Optical Transport Network (OTN) layer management tools (which may be managed by a third-party operator), information about scheduled operations, and contact details for local servicing teams. Given the interdependencies between services and infrastructure and the inherent complexity of networks, the importance of a comprehensive and standardized knowledge representation of network assets and events therefore becomes evident for anyone wishing to develop a decision

[1] Internet Protocol (IP) over Dense Wavelength-Division Multiplexing (DWDM).

support system that can capture and analyze an incident context in its full complexity.

At the same time, one might be tempted to solve these complexity and operational efficiency challenges by adding Artificial Intelligence (AI) techniques to monitoring tools. This is already observed in various commercial and open source products for alarm grouping, alarm prioritization, alerting on trend breaks (e.g. sudden increase in network traffic), or alerting on risky user behaviors (e.g. unusually frequent authentication attempts from various sites). It is typically implemented through a business rules system and an overlay of correlation analysis. This approach is effective in that the business rules ensure a form of explainability for the generated alerts and recommendations thanks to their explicit and logical form. However, the operational burden remains significant because rule-based systems are complicated to maintain due to a great number of fine-grained rules and typically react to discrete stimuli, which hinders the generalization of rules for complex networks and often leads to missing the detection of anomalies (false negatives). Another approach is the use of probabilistic models derived from machine learning. It is also effective in that it allows for generalization to different types of stimuli but sacrifices explainability because of untractable model representations (e.g. the weight matrices of a deep neural network) and introduces an operational burden due to the need to qualify falsely generated alerts (false positives). Given that both approaches seem to have complementary advantages and disadvantages, the question arises of identifying principles that can be shared to meet the requirements of explainability and generalization, in line with the earlier mentioned need for standardized representation.

Drawing on the understanding that Knowledge Representation and Reasoning (KRR) are inherently linked, this research work addresses both aspects in tandem by delving into explicit knowledge representations of networks, and exploring their direct utilization or integration with AI techniques for anomaly model learning and detection.

The remaining part of this chapter is organized as follows: Section 1.1 presents a description of the application domain. Section 1.2 formalizes the problem addressed in this thesis work and summarizes it through two research questions. Section 1.3 presents a summary of the contributions made through this thesis work. Finally, Section 1.4 provides an overview of the overall organization of the thesis work and manuscript.

## 1.1 Background

### 1.1.1 Network Design

**What are networks?** Basically, an IT network is a set of computers, routers, and other devices connected and configured to allow data processing and sharing. Similarly, an IT service is the usage of this processing and sharing capability for specific purposes. The standardization

efforts in the telecom and IT industry have led to standards and specifications for data transmission and processing system interfaces, such as the OSI model [161]. The OSI model allows for a description of network infrastructures and services along horizontal and vertical axes, representing the technical assets and protocol stack required for data transmission (Figure 1.1).

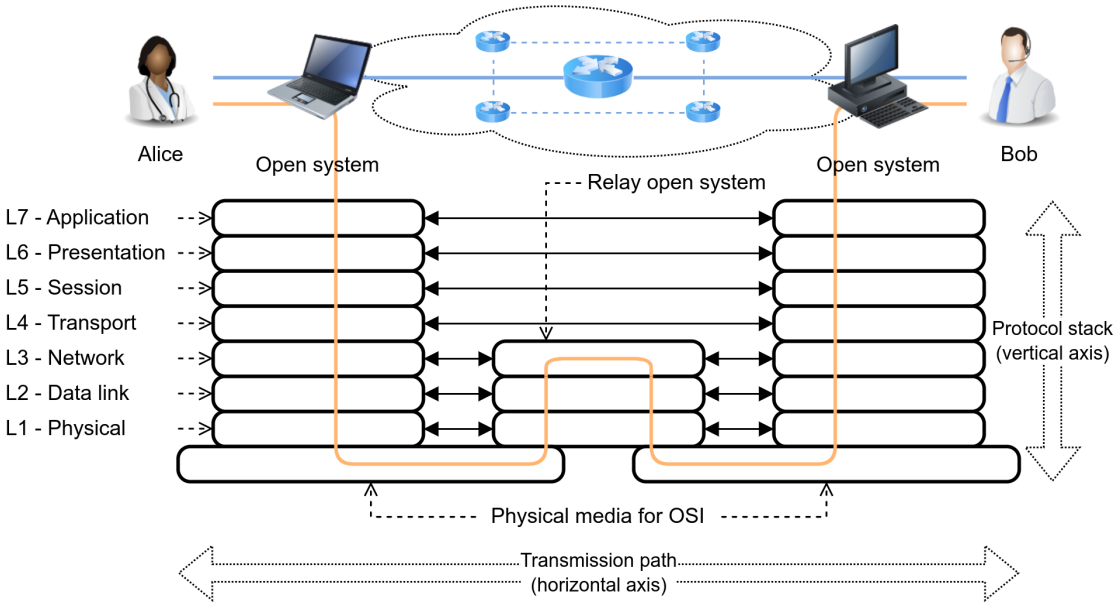



Figure 1.1: A flat view of a network with a data flow, and its equivalent in the OSI model.
Top: schematical view of an IT network. The central bubble represents a simplified network consisting of a mesh network of routers (i.e. network elements that determine and ensure the routing of data packets). The router in the center is an abstract representation of the redundancy group formed by the mesh of routers. Bottom: data path along the OSI model. The horizontal axis gives us a lecture of the technical assets (e.g. laptop, ethernet switch, router, firewall, etc.) involved in handling message data. The vertical axis provides insights into the protocol stack required for data takeover (i.e. encoding, routing, presentation, etc.). Based on the TISO2930-94/d11 diagram from [161].

According to this framework, routing a payload data unit from point $A$ to $B$ involves a horizontal data path formed by a set of network elements (Eq. 1.1, where $tp$ stands for transmission path, and $ne$ for network element), such as when Alice is sending an "hello" message to Bob using an instant messaging service.[2]

$$tp_{A \to B} = A \to ne_1 \to \cdots ne_i \cdots \to ne_N \to B \tag{1.1}$$

The vertical perspective considers the protocol stacking within each network element, including line coding and error correction services of L1, encapsulation, link and medium access control of L2 microcode, L3 routing engine, etc., up to L7 if relevant. Combining both axes allows for stack-to-stack connections between protocol endpoints, such as from the L7 layer on Alice's terminal to the L7 layer on Bob's terminal in Figure 1.1.

[2] Implementation details, such as bidirectional mechanisms for flow control or data link management (e.g. IP/TCP "SYN/SYN-ACK/ACK" sequence), are not discussed here for simplicity.

**Network dynamics.** The previous details describe a scenario where data flows along a fixed transmission path. However, considerations for network dependability introduce the use of a mesh network architecture and failover mechanisms, which bring temporal reachability concerns. In terms of the horizontal axis, a temporarily unavailable network element is automatically replaced by another, ensuring the continuity of the data path from the end users' perspective (Eq. 1.2), where $t1$ and $t2$ are two different instants.

$$\begin{aligned} &\left(tp_{A\to B}(t1) = A \to ne_1^{t1} \to \cdots ne_i^{t1} \cdots \to ne_N^{t1} \to B\right) \\ \equiv &\left(tp_{A\to B}(t2) = A \to ne_1^{t2} \to \cdots ne_i^{t2} \cdots \to ne_N^{t2} \to B\right) \end{aligned} \tag{1.2}$$

This leads to a functional object that remains invariant over time, represented by a pseudo ordered set (Eq. 1.3), where $\{ne_{i,j}\}$ is a set of network elements that forms a redundancy group (i.e. assets within $tp$ that are functionally equivalent).

$$tp^*_{A\to B} = A \to \cdots \{ne_{i,j}\} \cdots \to B \tag{1.3}$$

The dynamics of networks must also be viewed through the multiplexing and virtualization capabilities offered by the networks. These capabilities are primarily motivated by considerations of optimal resource allocation and rapid deployment in relation to the expectations for the network/service functions (e.g. forwarding, filtering, processing) and performance (e.g. throughput, number of simultaneous users, latency). Multiplexing can involve parallelizing links ($tp_{A\to B} = A \to \cdots \left\{tp_{a1\to b1} \parallel \cdots \parallel tp_{aN\to bN}\right\} \cdots \to B$) to increase throughput or path resilience. Multiplexing can also mean partitioning or stacking flows through recursive encapsulation ($\left\{tp_{a1\to b1} \parallel \cdots \parallel tp_{aN\to bN}\right\} \subset tp_{A\to B}$) to maximize the reuse of a given data transmission resource. Finally, virtualization enables temporary and mobile processing capacity based on processing resource sharing ($(ne_i \subset ne_j) \prec (ne_i \subset ne_k)$), like deploying a firewall based on user needs or a cache system based on the localization of users.

### 1.1.2 Network Operations and Incident Management

**What is an "anomaly"?** The term "anomaly" commonly refers to a deviation from the normal state that requires action for recovery. In the context of Network administration and Operations (NetOps), an anomaly is defined with respect to a supervision process[3] where **alarms** are a logical consequence of **errors** (i.e. a deviation of a system from normal operation [165]) caused by persistent **fault causes** (i.e. the physical or algorithmic cause of a malfunction [165]). These faults occur within the atomic functions of network devices [162]. The alarms should be reported to a Management and Control System (MCS), such as a Computerized Maintenance

[3] Quoting [162]: the way in which the actual occurrence of a disturbance or fault is analysed with the purpose of providing an appropriate indication of performance and/or detected fault condition to maintenance personnel.

Management System (CMMS) or Software Defined Network (SDN) controller. The logical sequence of these concepts can be summarized by: $Fault \succ Error \succ Alarm \succ AlarmReport$.

In the context of Cyber security Operations (SecOps), an anomaly is defined based on a business policy: user and device activities that comply with the policy are considered legitimate, while others may be classified as **attack**[4]. Errors, alarms and other automated analysis reports from both the NetOps and SecOps domains serve as input for a behavioral analysis process that utilizes causal entities such as **threats** (i.e. entities that can adversely act on an unwanted asset [104]) and **vulnerabilities** (i.e. weaknesses of assets that can be exploited by threats [104]). This analysis helps detecting attacks and **incidents** (i.e. unwanted events resulting in the loss of confidentiality, integrity, and/or availability [104]).

**Incident management and root cause analysis.** The concept of IT Service Management (ITSM) emerged in the 70s-80s as organizations recognized the importance of Information Technology (IT) for their operational efficiency. Standards and best practices, such as ISO/IEC 20000 [1], ITIL [17], and FitSM [160], were developed to provide guidance to IT organizations in aligning their ITSM processes with business needs and international best practices. These standards emphasize the establishment of a continuous quality improvement loop, which relies on the observability of ICT systems and the accumulation of knowledge, such as the causes of incidents and the corrective actions taken. By adhering to these standards, network operators are well-positioned to achieve and maintain the expected level of quality for end-users, as defined in Service Level Agreements (SLAs). These SLAs often establish demanding performance or reliability requirements, such as achieving “five nines” (99.999%) uptime, which is especially critical for essential systems like power, transportation, and telecom networks.

Security management standards – such as the ISO/IEC 27000 series [1], ETSI TVRA [104], NIST SP 800-53 [183] – distinguish between the business policy topic (i.e. rules for leveraging the information system to detect and track illegitimate activities) and the security implementation topic (i.e. selecting protocols and mechanisms to enforce security). These standards provide guidelines for establishing an Information Security Management System (ISMS), which focuses on risk mitigation through a multilevel iterative approach, such as a plan-do-check-act (PDCA) cycle. Methodological frameworks, such as OCTAVE [304], EBIOS RM [33], and NIST SP 800-61 [277], can help organizations setting up a cyber security management organization and aligning it with the ISMS.

The Incident Management Process (IMP) and Incident Response Management (IRM) processes are designed to meet SLAs and security requirements (Figure 1.2). The terms IMP

[4] Quoting [75]: any kind of malicious activity that attempts to collect, disrupt, deny, degrade, or destroy information system resources or the information itself.

and IRM can be considered interchangeable or have some level of difference depending on the perspective of the individual [295]. Notably, due to the sensitive nature of cyber security incidents, the SecOps incident lifecycle has a natural inclination to be seen as an IRM. However, the underlying concepts apply to both the NetOps and SecOps perspectives. For example, the containment stage in the SecOps context (e.g. disabling a compromised user account to prevent the attacker from accessing endpoints and other resources in the network) is similar to the restoration stage in the NetOps context (e.g. applying a failover activation procedure on a load balancer cluster to prevent the loss of access to an application). In both cases, the concept of an incident (impacting the service or its security) is defined from the perspective of the end user. The decision-making capabilities regarding the actions to be taken for system restoration, attack containment, incident resolution or repair depend on the diagnostic stage. Ideally, a Root Cause Analysis (RCA) leads to a clear diagnosis (i.e. to be able to clearly and unequivocally state which event on which asset is at the origin of the situation), and consequently, an objectively immediate choice of the repair procedure.

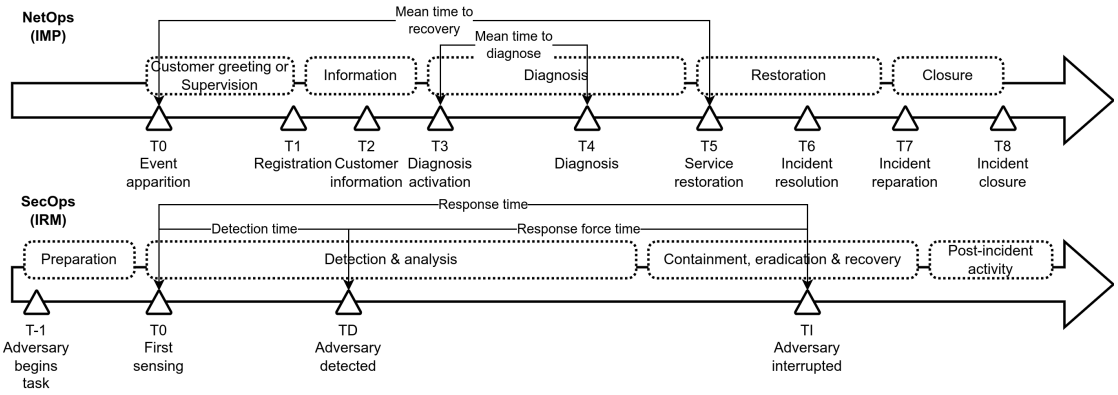



Figure 1.2: Incident Management Process (IMP) *vs* Incident Response Management (IRM).
Drawing a parallel between the alert and incident management processes for NetOps (ITIL Incident Management [334]) and SecOps (Management of Information Security and Improvements [159, Annex A 5.24] & Design Basis Threat framework [154]) contexts, showcasing their fundamental stages and temporal milestones. These milestones can serve as metrics for evaluating process performance and also for modeling and analyzing cyber-physical interactions.

## 1.2 Problem Statement

In this section, we specify the objectives of this research work by analyzing the challenges inherent in ICT systems operations (Section 1.2.1), stating the working hypotheses (Section 1.2.2), and defining two specific research questions (Section 1.2.3). In relation to this analysis phase, a summary of the contributions resulting from this research work is presented in the following Section 1.3.

### 1.2.1 Challenges

**Process performance and knowledge capitalization.** To effectively manage network operations, including incident management, it is essential to have a thorough understanding of the ICT systems' state and operational rules. In the absence of such knowledge, it becomes necessary to develop a method to group and prioritize the multitude of indicators and notifications originating from network operations, which are reported through NMSs and SIEM systems. These notifications encompass various aspects like system and application logs, performance indicators, technical alarms, and service alarms, which are associated with spontaneous faults, configuration changes, normal usage, and malicious activities.

By implementing a systematic approach to managing these informations & notifications, several benefits can be realized. For instance, it would allow to highlight explicit behavioral patterns exhibited by the network, promptly report critical situations, facilitate tractable analysis of incidents, and identify new instances of faults that are identical or similar to previously encountered issues. Moreover, this process could enable the creation of new supervision rules that give priority to relevant notifications until the root cause is pinpointed, thus establishing a connection with appropriate remedial actions.

**Scale effect.** However, as networks grow in size and complexity, organizations tend to adopt siloed structures to maintain high expertise within specialized teams. This dual dynamic leads to Network Operation Center (NOC) and Security Operation Center (SOC) teams becoming overwhelmed with data analysis and alarms prioritization, considering that 1'000 events per day and per security engineer is the practical maximum to deal with [90]. Although automatic alarm prioritization solutions are used to assist, they have limitations: rule-based solutions result in false negatives and are hard to maintain, while machine learning-based solutions lead to false positives and lack explainability.

Another consequence of this dual dynamic is that the knowledge about ICT systems' behavior is fragmented across different teams with varying terminologies and rules, even for similar equipment types. Hence, although the variety of decision support tools and technical solutions to manage networks is a wealth in itself that corresponds to the variety of technical or functional scopes to be managed, it is at the same time a challenge for efficient incident situation understanding: in practice, decision-making on the remediation action to be taken for a given situation must be based on a multiplicity of viewpoints stemming from various specialized tools. This diversity also makes it challenging to consistently and practically account for system behavior, thus undermining the principle of continuous improvement loop recommended in quality management standards like the ITIL Incident Management process [334], ISO/IEC 20000 [1] and NIST SP 800-61 [277].

**Knowledge representation.** We observe that graph technologies are more and more used to monitor complex systems or help to detect anomalies [85, 370, 94, 217]. Using relational models like graphs could be highly beneficial to network and cybersecurity administrators for improving situation understanding and response through User and Entity Behavior Analytics (UEBA) [381] solutions as part of NMSs and SIEMs. Beyond the fact that networks are generally represented as graphs for the intuitive value of this representation mode, employing a relational model is crucial because similar network events can lead to different incidents depending on the technical context in which they occur (i.e. thinking of network configuration in the broad sense, including network topology and equipment & service parameters). In fact, not using graphs eliminates the possibility of considering an incident event as part of a larger whole that may itself contribute to more complex events/incidents (e.g. common cause failures, cascading failures and alarm spreading phenomenon).

**Heterogeneity in concepts and data.** At the same time, using graphs necessitates imposing a certain level of consistency in knowledge representation to ensure effective data utilization. The heterogeneous nature of network data (e.g. technical characteristics of assets, technical logs and alarms, performance measurements as time-series, users and organizations), combined with the fact that graphs are a versatile data structure, may lead to the temptation of directly recording this diversity of data in the graph, potentially overlooking the need for generalization for downstream analysis tools (e.g. interpreting network interface state changes consistently, irrespective of the manufacturer providing the textual notification). Building upon earlier management protocols like SNMP [112], various modeling languages and data models are now available to tackle this heterogeneity challenge. For instance, the TM Forum Open APIs [355] offer interoperable definitions for states and operations in decision support tools, the YANG [268] modeling language describes configuration and state data of network elements, and several prior studies in close relation to knowledge graphs [8] and Semantic Web [352] technologies have shown the value of semantic modeling in network infrastructure monitoring, such as INDL [235], CRATELO [12], UCO [388], ToCo [294], ACCTP [29], and DevOpsInfra [270].

The use of graphs also opens the door to the use of analysis and inference tools directly aligned with graph theory (e.g. risk diffusion using shortest path calculations [380], event clustering using a centrality measure [231]), machine learning (e.g. fraud analytics using graph isomorphism [210]), or automated reasoning (e.g. identifying computer forensic scenarios using pattern matching [30]). However, introducing new AI techniques or adapting existing ones for anomaly detection and root cause analysis within graph structures must prioritize compatibility with this data format. Challenges may arise, such as handling dynamic graphs [231, 222] or data transformations with information loss at the models' input, thus leading to costly inference pipelines for result recontextualization. Similarly, relying on a

single AI technique may prove inadequate for addressing the wide range of failure or attack scenarios that can arise. Typically, detection tools and models specialize in specific data types and detection cases, like using file excerpts to detect viruses by their signatures [365] or IP packet headers to identify trend disruptions in application usages [151]. However, incident characterization often involves multiple artifact types or compromise indicators due to the complex nature of large-scale system failures and the ingenuity of malicious actors in devising attack scenarios. Therefore, unless an inference model capable of handling all data types and detection cases is available, an architecture like synergical reasoning [46] (cooperative decision making) becomes essential. Such an architecture could follow a model stacking scheme [273], or incorporates diverse specialized inference models that collaborate, leveraging each other's outcomes through mechanisms like voting, on-demand inference and re-use of previous inference results.

**Observability and decision-making.** Finally, since ICT systems are inherently interconnected, any action of modification (whether it is intended to evolve or repair the system) must be taken with full knowledge of the consequences. Therefore, having precise and complete knowledge of the system is a desirable ideal, but it is not the general case for various reasons. From a completeness perspective, typical examples include the absence of measurement means, encrypted or protected data, or dropped notification data unit by the network. From a precision perspective, the typical case is the accumulation of layers of data interpretations, each introducing errors that hinder unambiguous decision-making. Taking inspiration from [32], this phenomenon can typically be represented with Eq. 1.4:

$$D_T = T_H(T_S(T_P(D))) \tag{1.4}$$

where $T$ is an interpretation function, $D$ is the data representing a system state, measured by a probe $P$, encoded into the information system $S$, and then understood by a human operator $H$ for potential decision-making.

The first drawback arising from this observability context is that the diagnostic phase corresponds to a special situation of making an inference from vague or fuzzy premises [78], thus necessitating the inclusion of a belief or confidence indicator with the results of the RCA (Figure 1.3). The second drawback is the risk of error in choosing the remediation action. This is theorized in [46] through equation $L \equiv C \wedge P_i \rightarrow G\langle p \rangle$, where $L$ is a logical lemma, $C$ a concept, $P_i$ the $i^{\text{th}}$ procedure stored in a knowledge base, $G$ the goal to be reached (i.e. returning the system to normal operations) and $\langle p \rangle$ a probabilistic measure of confidence over $L$.[5] The typical approach in this kind of situation is to proceed through trial and error until the system is brought back to a viable state. This involves making decisions in uncertainty about a

[5]The equation can be read as "If the context $C$ appears to hold currently, then if I enact the procedure $P$, I can expect to achieve the goal $G$ with certainty $p$" [46].

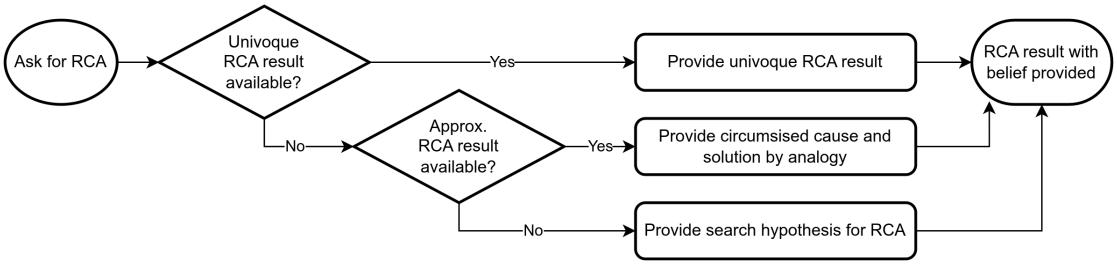



Figure 1.3: Graduated Root Cause Analysis (RCA) computation strategy.
During incident diagnosis, the accuracy of root cause search results determines remediation action selection. In the best case, actions are deduced from the cause; otherwise, abductive reasoning [298, 108] is used to select probable causes/solutions or present research hypotheses.

system whose state evolves over time, influenced by its own dynamics and the consequences of attempted actions for incident recovery (Figure 1.4).

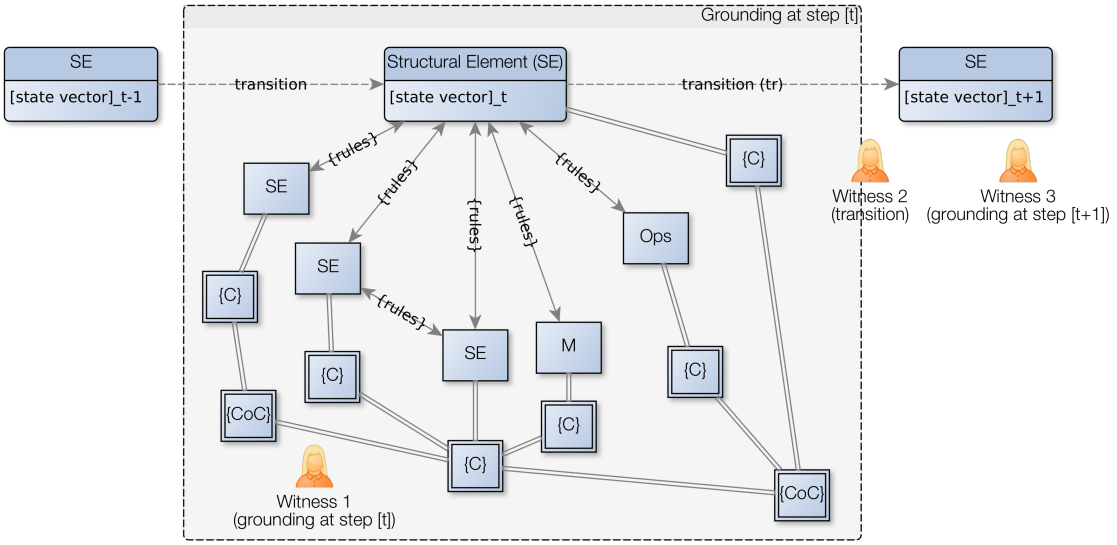



Figure 1.4: Sequential & uncertain decision problem on a hybrid "concrete-conceptual" model.
Witness 1 interprets the network's state (a set of state vectors related to structural elements $SE$ at time $t$) abstractly using concepts ($C$) and combinations of concepts ($CoC$). Each time step has potential future states based on the technical and conceptual neighborhood of $SE$ ($M$ = malicious actor, $Ops$ = planned operations, etc.), and a state transition model (set of $rules$). Witness 2 can predict future state vectors by analyzing transitions or actions on the system, while Witness 3 observes the system with its new set of states and concepts (same as Witness 1) resulting from the application of the state transition model. Lack of knowledge about state vectors or rules leads to considering the inference process (e.g. alerting on undesirable user/system trajectory, predicting next user/system action for corrective maintenance) as a sequential decision-making problem under uncertainty, where states and transitions represent the system's dynamics.

### 1.2.2 Hypotheses

Considering the aforementioned challenges, we aim to provide a constructivist framework towards mastering the complexity of ICT systems behavior in the context of network operations and incident management. Therefore, we propose the following main hypotheses for this research work.

**Hypothesis 1.** The fundamental model for representing ICT systems and their dynamics is a dynamic graph (network topology + notifications = dynamic graph).

**Hypothesis 2.** Alarm spreading and cascading failures are bounded in terms of both time and location.

**Hypothesis 3.** The functioning logic of ICT systems can be (partially) inferred or learned by observing both network topology and notifications.

**Hypothesis 4.** The state trajectory of ICT systems reflects the course of action, including malicious usage.

**Hypothesis 5.** The state trajectory of ICT systems can be mapped to a logic-based abstracted representation of the situation.

Ultimately, by leveraging diverse data sources and drawing insights from domain experts, the research community, and shared/private data models, the framework based on these hypotheses could facilitate understanding the behavior of a complex technical system. This understanding could then be used to reason about the system's trajectory in a transparent manner through a fundamental dataflow abstraction (Eq. 1.5), enabling applications such as anomaly detection and optimal design calculations.

$$\mathscr{E} \xrightarrow{e.t.l.} \mathscr{K}^{\circlearrowright r} \xrightarrow{infer.} \mathscr{P} \tag{1.5}$$

The semantics of Equation 1.5 are as follows: $\mathscr{E}$ the Environment (i.e. primary and secondary data describing the network)[6], $\mathscr{K}$ the Knowledge (i.e. information about the network behavior and business rules guiding network administration tasks), $\mathscr{P}$ the Propositions (i.e. explained inferences about the network states and behavior), $e.t.l.$ an Extract-Transform-Load process or alike, $r$ a reasoning process (i.e. an "internal" inference process aimed at producing facts and knowledge from basic facts present in $\mathscr{K}$) and $infer.$ an inference process.

### 1.2.3 Research Questions

These hypotheses lead to the following main question of this doctoral research:

> *How to define an anomaly model in a dynamic technical environment with various interdependencies, and what form should this model take to be shareable among practitioners (network designers, network administrators, cybersecurity analysts, etc.) and directly usable in anomaly detection tools and decision support systems?*

[6] We use a general definition of *primary data* (data directly collected on the source, may be in raw format or that have been normalized for future processing steps) and *secondary data* (data that has already been collected, processed, or even aggregated).

On the one hand, the anomaly model production/exploitation aspect with heterogeneous data needs to be investigated:

**RQ 1.** *What is an adequate neuro-symbolic AI architecture that can learn logically-constrained behavioral rules from events and topology data of an ICT system, and enable to detect and interpret complex anomalous technical or user-based situations (e.g. network service issue with alarms spread across time, locations and data sources; stealthy cyber security attacks)?*

On the other hand, the constraints on internal representation of data/knowledge aspect needs to be investigated too:

**RQ 2.** *Can human operators and decision support AI agents use the same Knowledge Representation (KR) of ICT systems for anomaly detection and knowledge management, that KR being subject to computation efficiency (e.g. near real time anomaly detection) and interpretability (e.g. decision tree like representation of predefined/learned ICT system's dynamics)?*

## 1.3 Outcomes

This work contributed to the research with the following outcomes.

**Methods and insights.** Four groups of contributions have emerged from this doctoral work in relation to the research questions RQ 1 and RQ 2. They can be summarized as follows, and their interrelationships are shown in Figure 1.5.

- **NORIA-O.** We proposed the NORIA ontology, a Semantic Web-based data model developed in collaboration with network and cybersecurity experts. The ontology enables operators and analysis tools to have a comprehensive and homogeneous view of networks, even with data from diverse sources. Existing data models focus on representing computing resources and their allocation, but there is currently no model that describes the inter-dependencies between the structural, dynamic, and functional aspects of a network infrastructure. NORIA-O addresses this gap.

- **KG-based data platform.** We developed an end-to-end data processing architecture that combines design patterns from NMSs/SIEMs with Semantic Web tools to construct a Knowledge Graph (KG) [8]. This architecture enables the interconnection of heterogeneous data using shared definitions, which is crucial for efficient interpretation of events and incidents. Semantic Web techniques play a vital role in addressing challenges related to data heterogeneity, knowledge sharing, and logical/probabilistic reasoning. The platform includes features such as batch/stream processing, declarative data mapping,

data patching and reconciliation, centralized configuration and data management for provenance auditability, and semantic data transfer using message bus technologies.

- **Anomaly detection framework.** We defined three families of anomaly detection techniques using KGs and the NORIA-O data model: model-based design, process mining, and statistical learning. We showed that heterogeneous ICT system data can be unified for downstream algorithms, allowing broader inference capabilities and reducing the coupling between the techniques and the data sources. We also demonstrated that using multiple detection techniques defined within this framework combines intrinsic explainability and probabilistic reasoning capabilities through the application of the synergistic reasoning principle (cooperative decision-making). This principle involves reinjecting knowledge from each technique into the KG, providing additional contextual information for subsequent techniques. This approach enables broader coverage of abnormal situations compared to a single specialized detection model for specific data types or application domains.

- **UI/UX design.** We proposed a Web-based client-server software architecture using a KG for decision support in complex network situations. Through this, we identified NetOps/SecOps expectations in terms of ergonomics and functions, based on expert interviews. We demonstrated the need to go beyond simple data retrieval from the KG. Synergistic reasoning and interactive analysis of multi-layered systems are essential. Our User Interface/User eXperience (UI/UX) evaluations provide a foundation for future KG-based NMS/SIEM designs with hybrid logical/probabilistic reasoning.

**Outreach.** This PhD thesis' results were accomplished and applied in the frame of the "machine learNing, Ontology and Reasoning for the Identification of Anomalies (NORIA)" research project at Orange[7], an international network infrastructure and service provider. They had significant industrial impact, leading to the active deployment of a knowledge graph-based digital twin infrastructure at the company's level. Additionally, they provided an opportunity to supervise and support four students in their internships and final projects, providing them with training in Semantic Web technologies and Decision Support Systems (DSSs). This PhD thesis' results also contributed to the establishment of the "RML join" community, an expert discussion group within the W3C Knowledge Graph Construction Community Group[8], as part of the effort to standardize the RML ontology [23] and make it a W3C Recommendation.

[7] https://hellofuture.orange.com
[8] https://www.w3.org/community/kg-construct/

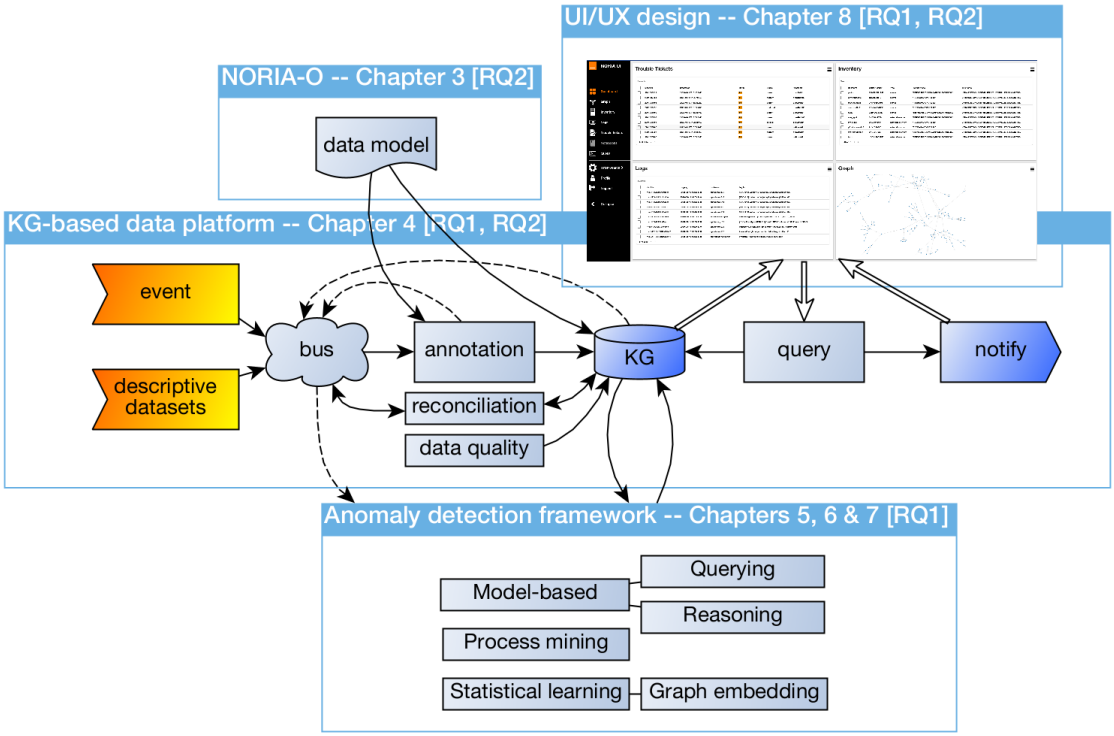



Figure 1.5: Overview of this PhD thesis.
The outcomes of this thesis work and their interrelationships, presented in the form of a block diagram. The diagram includes references to the addressed research questions (RQ) and chapters that detail the associated work.

**Papers, communications and resources.** This PhD thesis led to eight peer-reviewed papers in international conferences, all of which have been accepted [203, 205, 204, 339, 340, 300, 341, 338]. The paper [339] was among the nominees for the Best Paper Award in the ESWC[9] 2024 Resource track, and the paper [341] received the IC@PFIA[10] 2024 Best Paper Award. Additionally, there was one poster, one demo, one blog post, and one keynote presented. Furthermore, eight resources and tools were published as open source. For a comprehensive list of the papers, communications and open-source code published in the context of this thesis, please refer to the end of the manuscript in Chapter A.

## 1.4 Thesis outline

The remainder of this thesis is organized as follows.

Firstly, in Chapter 2, we present the state of the art of the field according to the two facets *Knowledge Representation and Reasoning,* and *Anomaly Detection and Explainability* that we aim to articulate to address the research questions RQ 1 and RQ 2. We demonstrate here

[9] Extended Semantic Web Conference
[10] Ingénierie des Connaissances

that while data heterogeneity and interrelatedness between data entities (i.e. distinct and persistent units of information) appear to be cornerstones for advancing the capabilities of NMS/SIEM DSSs, several semantic models and algorithmic methods for anomaly detection share common properties that could help address these two aspects. This enables us to tackle challenges outlined above in Section 1.2.1 through research on the means of constructing a rich representation of networks and their ecosystem that can be used by one or a combination of several inference techniques.

The presented work is then divided into two main parts.

Part I focuses on "*Knowledge Engineering and Massive Data Integration*". It begins with the design of NORIA-O (Chapter 3), an ontology implemented using Semantic Web techniques that allows describing a network infrastructure, its events, diagnosis, and repair actions during incident management. We then establish the design methodology and principles for an end-to-end knowledge graph-based data platform (Chapter 4), for handling descriptive datasets and events streams.

In Part II, the focus is on "*Exploitation of the Knowledge Graph for Anomaly Detection and Diagnostic Assistance*". This part leverages the data from the knowledge graph built using NORIA-O and the Extract-Transform-Load (ETL) architecture discussed earlier. Firstly, three complementary anomaly detection approaches are proposed and discussed: model-based (Chapter 5), process mining & conformance checking (Chapter 6), and statistical learning with graph embeddings (Chapter 7). Then, we introduce and evaluate NORIA UI (Chapter 8), a Web application co-designed with network operations experts for incident diagnosis using knowledge graphs and the aforementioned anomaly detection approaches.

Finally, in Chapter 9, we provide a summary of this research work with conclusions and perspectives.

# Chapter 2

# State of the Art

With this survey we propose to tackle the challenges defined in Section 1.2.1 by exploring the conditions for developing a network & IT monitoring system with advanced anomaly detection and reasoning capabilities through the lenses of Knowledge Representation and Reasoning (KRR): we provide an end-to-end overview with perspectives about graph-related knowledge representations, available anomaly detection methods, and how these two perspectives can match. Through this, we consider the possibility of improving operational efficiency in incident management situations and enhancing the design of complex network architectures by learning an explicit representation of the context of incidents (i.e. including information about system configuration and events that have occurred).

The remaining sections of this chapter are structured as follows. Section 2.1 outlines the research methodology used for this survey. Section 2.2 examines Decision Support Systems (DSSs) in the NetOps and SecOps fields, and identifies limitations that need to be addressed based on their capabilities and the expectations of these fields. Section 2.3 examines semantic models for storing and managing technical and operational data required for NetOps and SecOps activities. Section 2.4 examines algorithmic methods for anomaly detection from the literature, focusing on application domains close to NetOps and SecOps. It provides an overall analysis of these methods, considering their principles, practicality within the incident management process, and the data structures involved. Finally, Section 2.5 concludes the survey by summarizing the technological and scientific challenges and suggesting future directions for knowledge graph-based monitoring systems.

## 2.1 Research Methodology

In this section, we explain the methodology used for creating this survey. Considering anomaly detection for Information and Communications Technology (ICT) systems as an end-to-end research topic, we followed a two-step research methodology to collect and analyze relevant

scholar and technical articles.

Firstly, in relation to the challenges discussed in Section 1.2.1, we broke down the field into the four following research axes to establish a framework for exploration and analysis: **Knowledge Representation (KR)** as the capability to store and process heterogeneous data; **CompleXity (CX)** as the capability to (efficiently) handle and process data streams as well as older/static stored data; **Anomaly Detection (AD)** as the capability to contextualize events and use them as a basis for complex anomaly detection; **eXPlainability (XP)** as the capability to provide feedback to the decision support system users about the reasoning process that led to a given alert.

Next, we conducted a keyword/topic-based bibliographic research, paying particular attention to the sources such as university libraries (e.g. Cambridge[1], Standford[2]), publishing houses (e.g. ACM[3], IEEE[4]), interest groups (e.g. W3C[5], ITU[6], TM Forum[7]), well-known organizations (e.g. Gartner[8], CISCO[9]), and the targeted application domains (e.g. networks and data centers management, software engineering, cyber security, cyber-physical systems). We also considered the co-citation network, adoption degree, and the availability of implementations.

In order to address the subject comprehensively, our initial set of keywords included "*anomaly detection*", "*failure detection*", "*availability*", "*dependability*", "*self-healing*", and "*process modeling and comparison*". Then, regarding the knowledge representation axis, we further looked for data structures and knowledge representations commonly used in the ITSM and cyber security domains with more specific keywords: "*network and system topology/functional description*", "*event/fault/signaling notification*", "*asset management*", "*communication protocol definition*". This included scrutinizing telco and IT standards, user manuals of products, and code analysis from the research community projects. As graphs have a natural affinity with ICT system diagrams, we primarily targeted knowledge representations with native support for graph structures, notably knowledge graphs [8] based on Semantic Web technologies [352] for their knowledge interoperability and inference capabilities. We also expanded our exploration to more traditional data structures such as relational tables in DataBase Management System (DBMS), decision trees, and Bayesian networks [83], as translations between these structures and graphs can be made back and forth with help of additional tools.

[1] https://www.cam.ac.uk/
[2] https://www.stanford.edu/
[3] https://www.acm.org/
[4] https://www.ieee.org/
[5] https://www.w3.org/
[6] https://www.itu.int/
[7] https://www.tmforum.org/
[8] https://www.gartner.com/
[9] https://www.cisco.com/

Using this method, we have selected ~ 270 references[10] that have been further analyzed according to the specific criteria for the three sections that follow: in Section 2.2 (capabilities of DSSs), we examined a corpus made of approximately 65 industrial tools, file formats, and data exchange standards; in Section 2.3 (semantic models), we examined 95 references that were published between 2004 and 2023; in Section 2.4 (algorithmic methods for anomaly detection), we examined 103 references that were published between 1999 and 2023. Note that some references may address more than one research focus. For example, as shown in the FOLIO work [57], the development of knowledge-based anomaly detection techniques is typically associated with the development of a semantic model that allows for representation and reasoning in the discourse domain.

## 2.2 Detecting Anomalies and Malicious Activity: a Cartography of Industrial Tools

Designing a data processing architecture for incident management of ICT systems involves various research and technical domains, such as data transformation and wrangling, storage and processing architectures, decision making, and business process management. In this section, we review related work focusing on the current architectures of Network Monitoring System (NMS) & Security Information and Event Management (SIEM) systems. First, we examine high-level requirements from the eye of NetOps & SecOps in Section 2.2.1. Then, we analyze the well established capabilities of these tools in Section 2.2.2. Finally, we discuss the remaining limitations in Section 2.2.3.

### 2.2.1 What Do Experts Need To Ask Monitoring Tools and Decision Support Systems?

As mentioned in Section 1.1.2, recommendations apply to both the NetOps & SecOps domains in terms of organization and tooling. These recommendations aim to achieve the dual objective of maximizing service quality/security and minimizing anomaly detection/correction times. In both domains, the incident management process is described as a sequence of iterative steps including the diagnosis of the situation and leading to the remediation and correction of an undesirable situation. As such, it is akin to an "action-observation-reward-goal" process model [27] with the following scenario: *1)* a failure (issue) on an asset induces events and alarms on the asset's neighborhood; *2)* responding to a trouble ticket (an alert), a network or security administrator analyzes events and alarms to distinguish primary events (causes) from secondary events (effects); *3)* contextualizing events and alarms with respect to "in policy" or "out of policy" activity models enables the administrator to select a remediation

[10] The number of references provided is approximate as some sources are spread across multiple documents, particularly technical documentations and standards related to DSSs and their ecosystem.

action; *4)* based on the remediation action results, the administrator closes the trouble ticket (the issue) or loops back for further analysis and corrective actions.

When engaged in this four-step scenario, experts wish to get an accurate insight about some specific event occurring over the network and be able to answer fundamental questions such as: “what are the objects involved?”, “what are the observations?”, “what can we infer from a set of observations and why can we infer this?”, “what are the causes / consequences?” and “what is the remedy ?”.

It is noteworthy that these questions follow an incremental situation understanding task scheme, i.e. from elementary queries on a knowledge base (e.g. using SPARQL queries [368], the DROOLs [228] engine) to queries combined with specialized inference modules (e.g. rule-based, entailment [119], classification, what if model [246], digital twin simulation [102, 87]). It is also noteworthy that these questions lead to secondary requirements in terms of reasoning capabilities upon situations: *dependency calculus* (e.g. what services are at risk whenever this host fails?); *causality inference* (e.g. is this log related to some other log?), *situation awareness* (e.g. is this set of log anecdotal evidence of an attack course of action?). These questions and remarks themselves are guides for the expectations of NetOps and SecOps teams regarding monitoring tools and DSSs.

### 2.2.2 What Are the Well Established Capabilities of Those Tools?

**Umbrella systems for centralized situation understanding.** To support network and security administrators in their task, numerous tools and procedures are available for diagnosing the state of ICT systems (remote access to devices [208], on-site measurements and indicators from probing systems [73], decision support tools such as NMSs [274, 185] or SIEMs [189, 134]), monitoring the life cycle of incidents, and capitalizing on knowledge of the causes and solutions to incidents (help desk ticketing systems [152], knowledge bases [145]). This variety of tools and solutions is a wealth in itself that corresponds to the variety of technical or functional scopes to be managed (e.g. network traffic analysis [72, 347] *vs* malware signature in files [365], IPoDWDM international backbone network *vs* distribution and access datacenter network). However, this is at the same time a challenge for a unified approach to the diagnostic stage: in practice, decision-making on the remediation action to be taken for a given situation must be based on a multiplicity of viewpoints stemming from various specialized tools. NMS and SIEM platforms play a crucial role in addressing this challenge, as their typical role is to centralize and present all technical information and alerts from networks.

**Log recording/management & notification analysis.** NMSs and SIEMs are two different product lines due to the nature of the data processed and the expectations regarding the

incident management processes in which they are involved. For telecommunication networks, alarms (i.e. a persistent or non fugitive fault that happens on an atomic function, as discussed in Section 1.1.2) are first class citizens that should be reported to a Management and Control System (MCS) for performance analysis and service impairment detection. For cybersecurity, technical logs need to be combined with vulnerabilities and threat intelligence in a Log Collection → Log Normalization → Notifications and Alerts → Security Incident Detection component chain for threat response management [90]. Both product lines show two main functional blocks: log recording/management and notification analysis, with the second block dependent on the first. Figure 2.1 illustrates this for SIEMs, highlighting the characteristic sub-functions of each block. It also shows that the implementation of the recording/management block primarily involves technical solutions, while the analysis block primarily involves algorithmic solutions.

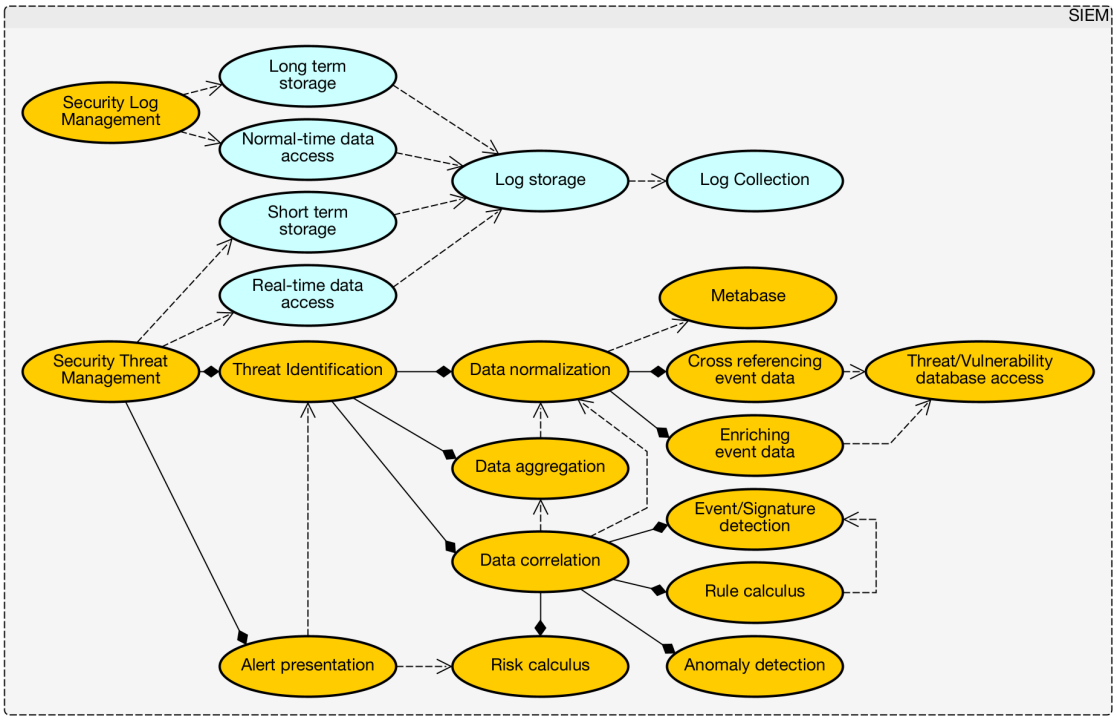



Figure 2.1: SIEM technical and functional capabilities.
Use case analysis for SIEMs, based on [90], using the UML representation standard. Dotted lines indicate a functional dependency, and diamond lines represent a composition relationship. The light blue use cases (top ellipses) are technical-focused use cases, while the other use cases are primarily algorithmic-based.

**Handling heterogeneous data & short/long-term analysis.** As ICT event data falls into the category of big data (e.g. high volume, variety, and velocity characteristics) [308], design choices for the implementation of the recording/management and analysis features are influenced by the need for efficient data ingestion, processing performance, and data retention. These design choices aim to enable both near-real-time anomaly detection (e.g. event correla-

tion indicating the propagation of malware) and long-term behavior analysis of the systems (e.g. calculation of the characteristic propagation method of a given malware).

In both NMS and SIEM contexts (e.g. ZenOSS [389], NetWitness [311]), data processing architectures generally follow the producer/consumer design pattern (a.k.a. observer pattern [101]) and inspirations from distributed computing (i.e. hubs + aggregator).[11] This type of architecture indeed offers a level of modularity that allows for managing the integration of data sources with varied persistence and dynamics characteristics through specialized modules that operate independently. This is particularly the case when monitored systems are heterogeneous, such as a High-Performance Computing (HPC) platform with a data transfer and processing service offer *vs* an Internet service provider with communication service and Platform as a Service (PaaS) offers.

The architectures used also allow for combining multiple storage solutions. The general trend is to maximize Input/Output (I/O) performance and minimize storage footprint, subject to both data semantics and persistence/dynamics characteristics: *1) daemons and web applications* (e.g. dedicated filesystem for raw logs, binaries and libraries); *2) events and node information* (e.g. PostgreSQL[12] for structured notifications and characteristics); *3) performance data* (e.g. RRD[13] for throughput or CPU usage time series). When not combining storage solutions, alternative approaches are to apply rule-based data normalization before storage (e.g. ManageEngine's EventLog Analyzer[14]), store raw logs/events then normalize data before applying rule-based analysis modules (e.g. SolarWinds's Security Event Manager[15]), or apply in-memory real-time analysis of data streams and process the resulting information (e.g. OSSEC Foundation's OSSEC HIDS[16], IBM's QRadar[17]). It is noteworthy that persistent storage formats mainly correspond to relational database management systems or time series databases, rather than graph formats, except for new entrants in the market (e.g. Luatix's OpenCTI[18], EXFO's Nova Context[19]). Overall, the proposed approaches are a compromise between two related trends: how data is conceptually stored and managed, and how hardware/software handle data.

[11] The TM Forum's Open Digital Architecture (ODA) aims to improve user experience and Information System (IS) interoperability in the ICT industry beyond general best-practice approaches for Decision Support Systems design.
[12] https://www.postgresql.org/
[13] http://www.rrdtool.org/
[14] https://www.manageengine.com/
[15] https://www.solarwinds.com/
[16] https://www.ossec.net/
[17] https://www.ibm.com/
[18] https://www.opencti.io/
[19] https://www.exfo.com/

**Information correlation & shared services.** Log centralization in NMS and SIEM systems allows network administrators to focus their monitoring and analysis activities on a single tool. An additional strategy implemented by monitoring tools and DSSs to minimize (cognitive) resources required for analyzing alarms and logs when presented to the operator is based on the concept of semantic distance (i.e. the distance between the goal aimed by the user and the actions/objects of the user interface [172]). Various complementary approaches are observed in this regard, among which *notification contextualization through rendering* and *notification rewriting and enriching* are playing a significant role and typically leverage the DSS operators' skills through implicit characterization of the notification. Regarding rendering, classic examples include flashing a shape on a network map and displaying the alarm in text form alongside other information related to the equipment or service affected by the alarm. For rewriting and enriching, examples include mapping an alarm to a basic supervision category[20] and annotating it with `probableCause` [165] attribute value or confidence score based on inference services (Figure 2.2), thereby relating the notification to the fault interpretation domain and the context for assessment.

Another strategy involves *grouping and hierarchically prioritizing notifications* to facilitate Root Cause Analysis (RCA) and decision-making for remediation actions. RCA for alarm spreading phenomenon in ICT systems is typically approached as an inductive process that distinguishes between primary failures (i.e. a failure that directly indicates the fault location and initiate a repair action, e.g. a broken cable or a misconnection) and secondary failures (i.e. a consequential failure, e.g. an upper level service that is gone down) [162, Section 7.1.1.1]. Therefore, the cause of a data transmission impairment is sought in the first alarming network element of a datapath, and redundant Alarm Indication Signal (AIS) can be silenced using an alarm suppression function or linked to a parent notification using a `correlatedNotifications` [165] attribute.

Similarly, for non-hierarchical or more complex systems, User and Entity Behavior Analytics (UEBA) is typically approached through a doubt removal process using dependency graphs or decision trees. Applied examples of this model-based approach are present in NMSs/SIEMs where it is assumed that the network topology and transaction records (i.e. a sequence of one or more operations – reads or writes – which reflects a single real-world transition [125]) are a model for causality relationships about network elements, services and applications. As this kind of feature is a key differentiator for commercial tools, few detailed papers are freely available about this but online blog posts for demonstration purposes. For examples, see the Zenoss "Layer 2 ZenPack" [191], the Riverbed "APM" [306] or the Cisco AppDynamics

[20] As per [162, Section 7.1.1] for telecommunication networks: transmission (management of the transmission resources in the network), quality of service (degradation in the performance), processing (software or software processing fault), equipment (fault localization and repair of the equipment itself) and environment (ambient conditions within an enclosure in which the equipment reside). In the cyber security domain: Confidentiality, Integrity, Availability (a.k.a. the CIA triad).

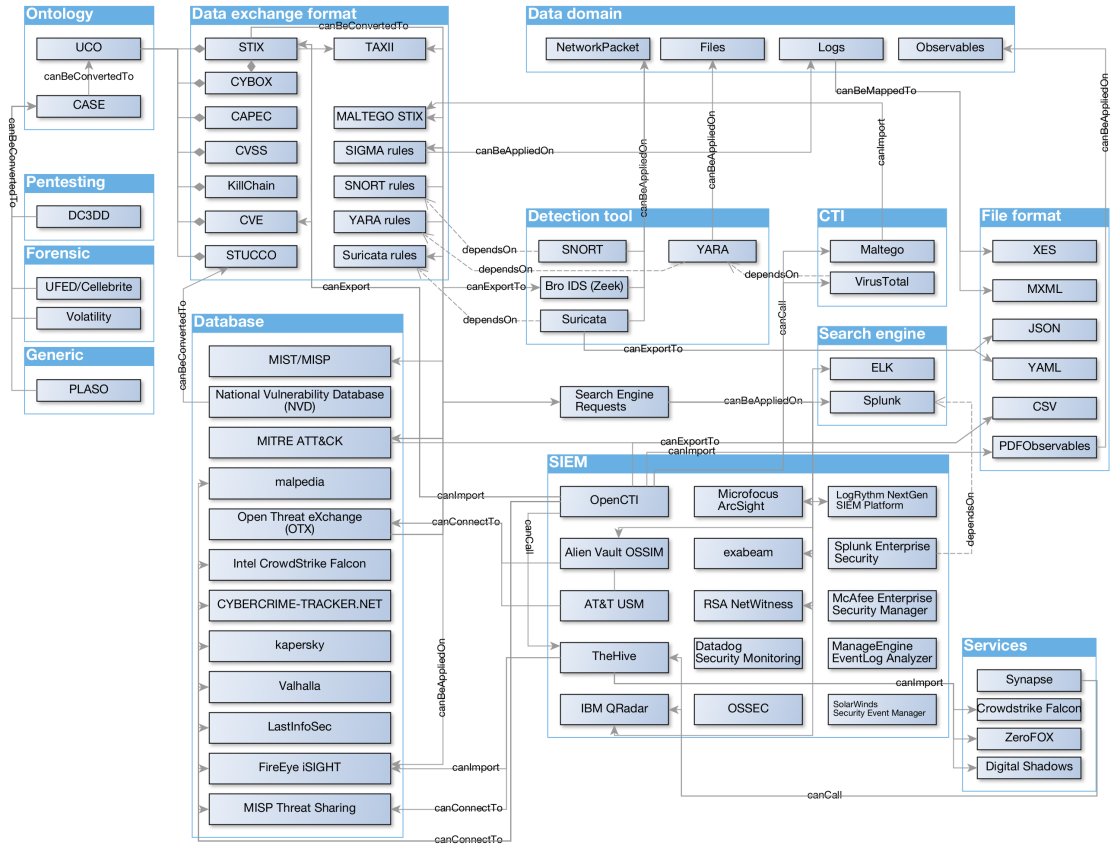



Figure 2.2: Cyber security tools relationships.
This entity-relationship diagram shows how specialized cyber security applications like SIEM (e.g. OpenCTI) or Cybersecurity Threat Intelligence (CTI) tools (e.g. VirusTotal) can compose, with help from shared services (e.g. NVD [257], MISP [248], Maltego [343]) or exchange formats (e.g. STIX [264]), in order to provide contextualized information over detection tools (e.g. SNORT [72], YARA [365]). Regarding relationships around SIEMs, emphasis is placed on the OpenCTI tool due to the easy accessibility of its detailed features, as it is an open source tool. Relationships around the UCO [388] component showcase the data sources and exchange formats used for knowledge discovery and After Action Report (AAR) annotation tasks as mentioned in [388], providing insights into the functional domains covered by the UCO project. The method used to construct this survey consisted of distinguishing, within the exchange formats, the aspects from those that rely on specific analysis tools, and then tracking the interrelationships between these tools.

"Cognition Engine" [307]. In the absence of algorithmic solutions, doubt removal is typically achieved in DSSs by facilitating an exploratory approach to data through the use of hyperlinks for navigating between information elements, display filters, a query system on the DSS database, and even interactive annotation (e.g. IBM "i2 Enterprise Insight Analysis" [148]) and event sharing among the supervision staff.

### 2.2.3 What Are the Limitations Remaining?

Although NMSs/SIEMs are a data focal point for network element characteristics and operational states, it is noteworthy that automated contextualization of notifications does not (or minimally) take into account network topology information (e.g. network links, data flows,

routing tables, failover mechanisms, location), network operation information (i.e. current and past trouble tickets or scheduled operations), agreed-upon work methods (e.g. equipment upgrade or IP address blacklisting procedures), or information regarding services provided to users (e.g. contractual data, business functions affected by a network service). Instead, it is rather the principles of UI/UX ergonomics and easy access to complementary (ad hoc) tools (e.g. search engine of Operations Support Systems (OSS), network diagram inference from traffic dumps, queries to third-party CTI tools) that are deployed, trusting that the DSS user will have and know how to mobilize the necessary skills to perform diagnostics.

Part of the reasons for this state of affairs comes from the fact that networks are complex (please refer to Section 1.1.1 and Section 1.2.1 for details) and open systems (i.e. no *a priori* knowledge of the behavior of users and connected neighboring systems) that are constantly evolving (e.g. new customers, new devices, new services). This, in turn, leads to an administrative effort in managing DSSs, defining alert rules, and ensuring data coherence, with costs exceeding regular operational expenses. For example, regarding the RCA and UEBA techniques evoked just before in Section 2.2.1, it is indeed necessary to consider that learning and using causal models rely on the idea of having complete knowledge of the ICT systems. This requires a complex technological ecosystem composed of network discovery protocols (e.g. LLDP [149]), cross-vendors definitions of managed objects (e.g. non vendor-specific branch in a SNMP Management Information Base or in a YANG model [52]), active network monitoring systems (e.g. flow monitoring with an IPFIX [392] compliant system) running over an in-production network, and even information sharing between network production and operation stakeholders. Regarding information sharing, functional alignment issues can be observed between the data models and vocabularies implemented and used by network production and operation stakeholders, thus entailing poor interoperability (e.g. challenges in interpreting potentially similar facts and concepts) and further complicating the implementation of RCA and UEBA techniques (i.e. causal relationships filtering for unseen yet failure modes or incident situation is generally unavailable unless explicitly implemented by NetOps & SecOps experts responsible for a specific network). Table 2.1 illustrates this with data exchange formats in the field of cyber security, along with the existence of numerous cyber security taxonomies from various sources: European Commission [106, 116], NIST Computer Security Research Center[21], IEEE[22], IFIP[23], ECSO[24], cyberwatching.eu[25], and so on.

Although interoperability of data representations and exchange formats is a cornerstone of current limitations, the heart of the problem lies in the ability of DSSs to effectively support the diagnostic steps performed by experts by providing full and reliable contextualization of

[21] https://csrc.nist.gov/
[22] https://www.ieee.org/
[23] https://www.ifipsec.org/
[24] https://ecs-org.eu/
[25] https://www.cyberwatching.eu/

| | Asset Definition | Configuration Guidance | Vulnerability Alert | Threat Alert | Indicator Sharing | Incident Report | Configuration Guidance Analysis | Vulnerability Analysis | Threat Analysis | Intrusion Detection | Centralized Reporting | System Assurance | System Assessement |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CAPEC | | | | ✓ | | | | | ✓ | ✓ | ✓ | | ✓ |
| CCE | | ✓ | | | | | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| CCSS | | ✓ | | | | | ✓ | | | | ✓ | | |
| CEE | | | | | ✓ | ✓ | | | | ✓ | ✓ | | |
| CPE | ✓ | | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| CVE | | | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | ✓ | | |
| CVRF | | | ✓ | | | | | | | | | | |
| CVSS | | | ✓ | ✓ | | ✓ | | ✓ | ✓ | ✓ | ✓ | | |
| CWE | | | ✓ | ✓ | | ✓ | | ✓ | ✓ | ✓ | ✓ | | ✓ |
| CWSS | | | | | | ✓ | | ✓ | | | ✓ | | |
| CYBEX | | | | | | ✓ | | | | | | | |
| CYBOX | | | | ✓ | ✓ | ✓ | | | ✓ | ✓ | ✓ | | |
| IODEF | | | | | ✓ | ✓ | | | | | | | |
| MAEC | | | | ✓ | ✓ | ✓ | | | ✓ | ✓ | ✓ | | ✓ |
| OCIL | | ✓ | | | | | ✓ | ✓ | ✓ | ✓ | | ✓ | |
| OVAL | ✓ | ✓ | ✓ | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| SBVR | | | | | | | | | | | | | ✓ |
| STIX | | | | ✓ | ✓ | ✓ | | | ✓ | ✓ | | | |
| SWID | ✓ | | | | | | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| XCCDF | | ✓ | | | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |

Table 2.1: Cyber security vocabularies and featured application domains.
Comparison of the application domains for well-established cyber securities vocabularies, based on [346].

events occurring on the ICT systems. According to Cybernetics [261], processing a notification alone cannot increase its informational content, but enhancing the precision of the knowledge about the situation in which it occurs can be valuable, even in ambiguous situations. Similarly, time psychomechanics [133] suggests that a complete understanding of an object involves considering not only its current state but also the various states it has gone through. Based on these principles, a natural approach to enhance the capabilities of NMSs/SIEMs is for analysis algorithms to rely on a comprehensive and integrated view of ICT systems and their ecosystem, rather than aggregating inference results from several algorithms, each using a subset of data. The means to implement this proposal are analyzed in the following sections, starting with the knowledge representation and reasoning perspectives in Section 2.3, and then the anomaly detection perspective in Section 2.4. Ultimately, as insights from NetOps

can benefit SecOps and vice versa, NMSs and SIEMs could then be considered as part of the same DSS solution.

## 2.3 Knowledge Representation and Reasoning: a Cartography of Semantic Models

In order to represent data in graphs for NMSs and SIEMs, the remaining limitations discussed in Section 2.2.3 highlight the need to combine various facets of knowledge. Ontologies – as explicit representations of a discourse domain through concepts and relationships – and their instantiation as knowledge graphs [8], enable data analysis and inference techniques to handle heterogeneous data and reason about the context of represented objects. In this section, we provide a summary of the ontologies we found during the bibliographic research stage, starting with the definition of evaluation criteria in Section 2.3.1, and reporting on our analysis in Section 2.3.2. For more details on the different graph models, including knowledge graphs, the reader is invited to refer to Appendix B.1.

### 2.3.1 Evaluation Criteria

The data collected and evaluation criteria used to analyze the ontologies are as follows:

**Name.** The name of the ontology or, if not available, the title of the document describing it (e.g. research paper, specification, website).

**Primary application domain.** The domain or field of application that originated the ontology, based on the indications provided by the authors, or if not available, deduced by us.

**Bibliographic document category.** We categorize reference documents discussing the data model based on a detailed reading to estimate the effort required for understanding, using, or implementing the model. The six categories are: "*position*" (analyzing domain issues and expressing intentions for future work), "*overview*" (providing a high-level description and use cases), "*specification*" (formally describing the design methodology and model), "*dataset*" (primarily presenting a dataset), "*documentation*" (providing context and usage information), and "*no access to paper*" (due to broken links or access restrictions).

**Main concepts and relations.** We sample concepts and relationships from the data model, either from its implementation or the authors' description. We focus on top-level concepts and their relationships, and sometimes second-level concepts if there are few top-level ones. In complex models with many top/second-level concepts and shallow hierarchy, we consider the centrality of concepts. This centrality indicates their

importance in the domain's conceptualization and can be inferred from graph diagrams in ontology reference documents (research papers, documentation).

**Availability and location.** We assess availability of data model implementation: "*yes*" (publicly accessible), "*broken link*" (unavailable despite reference), "*empty content*" (reachable but empty), "*no reference provided*" (location not indicated or explicitly stated as unpublished), or "*not relevant*" (not related to Semantic Web or data model). If implemented, we record URL and serialization used.

**Bibliographic citation count.** We evaluate the usage or consideration of the proposed data model based on the number of citations of the research paper or reference document describing it. The citation count is obtained from Google Scholar[26], and the corresponding URL is recorded.[27]

**Ontology metrics.** We gather the characteristics of the data models, including the number of classes, object properties, data properties, individuals, and Description Logic (DL) [119] expressivity. This information is collected using the Protégé 5.1 tool [227] or, if we do not have access to the implementation of the model, based on the details provided by the authors (in research papers or online documentation describing the data models).

**Best practice compliance.** We assess the quality of implemented data models using best practices described in [225] and the related OOPS! tool[28]. To do this, we obtain a copy of the data model (from references in research papers, online documentation, or by contacting authors), normalize it into RDF/XML format using Protégé 5.1, send it to the OOPS! Web service for analysis, and aggregate the results. For modular models, we merge the ontologies into a single file using Protégé before analysis with OOPS!.

**Conceptual facets coverage.** We analyze the capability of data models to encompass the four facets of the discourse domain, as established in Section 3.2.2, that are essential for representing networks and their dynamics: *structural* (network assets such as servers and links), *functional* (network services and flows), *dynamic* (events and states changes) and *procedural* (processes and actions). We evaluate this by examining main concepts and relations of the data models using a subset of competency questions (Table 2.2) derived from NetOps & SecOps expert panel interviews.[29] The reader is invited to refer to Section 3.2.1 for details on the origin of these competency questions.

[26] https://scholar.google.com/

[27] To fully understand ontology adoption, detailed usage information such as the number of instantiated classes and its evolution over time would be valuable. However, this information is generally unavailable.

[28] https://oops.linkeddata.es/

[29] We acknowledge that automatic topic modeling could help qualifying a data model for facets. However, based on our experience (e.g. using the ZeSTE tool [157]), we find that it is possible but not reliable due to its dependence on concept naming and description, which is influenced by ontology author bias.

| St. | Fu. | Dy. | Pr. | Competency Questions |
|---|---|---|---|---|
| ✓ | ✓ | | | What assets are shared by a given asset chain? |
| ✓ | | ✓ | | Which entity (resource/application/site) is concerned by a given incident? |
| ✓ | | ✓ | | On which resource did this sequence of events take place and in which order? |
| | | ✓ | | What corrective actions have been carried out so far for a given incident (who, what, where)? |
| | | | ✓ | What interventions were carried out on this resource that could have caused the incident? |
| | | | ✓ | What operation plan (automations, operating procedures, etc.) could help us solve the incident? |
| | | | ✓ | Given all the corrective actions carried out so far for the incident, what possible actions could we still take? |

Table 2.2: Competency questions for analyzing the conceptual facets coverage.
The four knowledge facets to represent (St.: structural, Fu.: functional, Dy.: dynamic, Pr.: procedural) map to a subset of competency questions (i.e. user queries expressed in natural language) derived from NetOps & SecOps expert panel interviews, as described in Section 3.2.1 and [339].

### 2.3.2 Semantic Models

In this section, we present the identified data models from the literature review, their categorization, and evaluation based on the previous criteria. Out of 95 references analyzed, 50 had an implementation based on Semantic Web technologies [352] (i.e. using a RDF-related serialization [305] or OWL functional syntax [54]), while the remaining 45 did not have an implementation. In the following, we focus on the models with available implementations. The complete analysis and resources are open source and can be found at https://w3id.org/noria/docs/sota/.

Table 2.3 shows the results in the form of a list of data models. We observe that the models cluster into six primary application domains (theme), with varying proportions of available models and model characteristics. Table 2.4 summarizes to what extent the set of models for each application domain theoretically aligns with the targeted discourse domain as established in Section 3.2.2. From the proportion of models with identified facets in Table 2.4, we observe that the models developed in the domains of Process modeling and Smart Environment & Smart Industry (SE-SI) are the most represented. This suggests the significance of modeling efforts in representing networks and their dynamics in these domains. However, the models in these domains generally exhibit disparities in terms of the targeted concepts, as revealed by the fact that process mining models focus on the *dynamic* and *procedural* facets without significant overlap with the *structural* and *functional* facets, which are primarily addressed by models in the SE-SI domain. Similarly, the proportion of models simultaneously covering multiple facets (i.e. $Fx$% columns in Table 2.4) indicates that models in the CyberSec domain seek to cover the maximum of facets simultaneously. This suggests a greater expressiveness and complexity of this group of models (e.g. in terms of the number of concepts and relationships between them).

In this line of thought, regarding the low coupling between facets and potential difficulties

in precisely allowing for reasoning on the interplay between network architecture and its operation, a detailed analysis of the data models reveals various opportunities for enrichment, improvement, or linking of the models. For example, the combination of the SEAS and PEP [240, 241] models enables the representation of communication links between technical assets and facilitates the analysis of assets' state changes by tracking commands and results. However, SEAS mainly targets the Internet of Things (IoT) domain and end-user devices, and the semantics of PEP relates to computer process. The DevOpsInfra [270] ontology enables the provisioning of data processing services and tracking computer resources hosting capabilities. However, concepts are missing for a finer grain description of the network topology. Additionally, the ontology mainly focuses on the provisioning activity and is not aligned with other well-known models such as SOSA [194] for sensors and probing systems, and the TM Forum Open API [355] for interoperable definitions of states and operations between DSSs. The CRATELO [12] and PACO [258] models enable the representation and classification of network traffic. However, they lack concepts for network topology and operations, which are necessary for contextualizing network traffic sessions within the network topology itself and day-to-day operations. Regarding the representation of *procedural* knowledge, the detailed analysis of the data models also reveals that Semantic Web-based models have less presence in modeling capabilities for knowledge about processes with conditional branching (e.g. IF-THEN-ELSE decision making) compared to sequential and relatedness knowledge. This suggests that representing networks using the available models or through the knowledge graphs formalism may not be self-sufficient for interpreting their dynamics, unless relying on external knowledge and reasoning tools as proposed by FOLIO [62] through the Failure Mode and Effect Analysis (FMEA) approach and the use of a rule engine (e.g. leveraging the Semantic Web Rule Language (SWRL) [147] or the SPARQL Inferencing Notation (SPIN) [143]).

Finally, regarding the compliance with best practices, all models exhibit non-conformities to varying degrees across application domains, in terms of both quantity and severity. However, it is worth noting that the number of non-conformities generally increases with the size of the model.

Table 2.5 highlights the top five implementation pitfalls found in the set of semantic models listed in Table 2.3. The relatively low severity of these non-conformities (three "Minor", two "Important"), as well as their nature (i.e. primarily related to the characterization of relationships and the description of concepts & relationships), suggest that the proposed models are generally ready for short-term integration into a knowledge graph-based solution. However, it is important to note that their inference capabilities may need to be adjusted based on experiments with practical cases.

Table 2.3: Semantic models comparison table, for models with an implementation, and as per a subset of the evaluation criteria defined in Section 2.3.1. Abbreviations: $Ref.$ = bibliographical reference, $CC$ = class count, $OPC$ = object property count, $DPC$ = data-type property count, $IC$ = individual count, $SE-SI$ = smart environment & smart industry, $n.a.$ = non applicable.

| Theme | Short title | Ref. | St. | Fu. | Dy. | Pr. | Expressivity | CC | OPC | DPC | IC |
|---|---|---|---|---|---|---|---|---|---|---|---|
| CyberSec | ACCTP | [29] | | | | ✓ | ALEROI+(D) | 117 | 52 | 20 | 160 |
| CyberSec | CASE | [100] | | | ✓ | ✓ | AL(D) | 17 | 3 | 7 | 11 |
| CyberSec | D3FEND | [284] | ✓ | ✓ | ✓ | ✓ | SROIQ(D) | 1568 | 198 | 41 | 1740 |
| CyberSec | ICAS ontology | [216] | ✓ | ✓ | ✓ | ✓ | SROIN(D) | 559 | 385 | 144 | 256 |
| CyberSec | MALOnt | [256] | | | | ✓ | ALH(D) | 76 | 11 | 13 | 265 |
| CyberSec | ORD2I | [382] | ✓ | ✓ | ✓ | ✓ | SHOI(D) | 67 | 49 | 82 | 14 |
| CyberSec | OWASP OdTM | [28] | | | | ✓ | ALCROI(D) | 100 | 42 | 4 | 30 |
| CyberSec | SLOGERT log extraction ontology | [26] | ✓ | ✓ | | | AL(D) | 5 | 6 | 4 | 98 |
| CyberSec | SLOGERT log event ontology | [26] | | | ✓ | | ALH(D) | 12 | 15 | 45 | 0 |
| CyberSec | ForensicOntology | [110] | ✓ | ✓ | ✓ | ✓ | SHIQ(D) | 483 | 148 | 51 | 1800 |
| CyberSec | UCO | [388] | ✓ | ✓ | ✓ | ✓ | SRIN(D) | 422 | 177 | 574 | 455 |
| Generic | A Core Pattern for Events | [193] | | | ✓ | | ALERI | 5 | 7 | 0 | 0 |
| Generic | Basic geo Ontology | [376] | | | ✓ | | SROIQ(D) | 61 | 22 | 1 | 4 |
| Generic | EmOCA | [117] | | | | ✓ | AL(D) | 18 | 4 | 3 | 1 |
| Generic | Event Ontology | [386] | | | ✓ | ✓ | ALCHI(D) | 8 | 17 | 5 | 2 |
| Generic | EventKG | [332] | | | ✓ | ✓ | AL(D) | 10 | 3 | 2 | 2 |
| Generic | FARO | [384] | | | ✓ | ✓ | ALRI+ | 5 | 32 | 0 | 4 |
| Generic | FOAF | [81] | | | | | ALCHIF(D) | 22 | 36 | 15 | 0 |
| Generic | GeoSPARQL | [254] | | | | | ALCH(D) | 9 | 37 | 18 | 0 |
| Generic | Geonames | [139] | | | | | ALEROIN+(D) | 15 | 33 | 17 | 701 |
| Generic | Mining Minds Context Ontology | [74] | | | | | ALCF(D) | 52 | 4 | 2 | 0 |

| Theme | Short title | Ref. | St. | Fu. | Dy. | Pr. | Expressivity | CC | OPC | DPC | IC |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Generic | Ontology of units of Measure (OM) | [135] | | | | | ALCON(D) | 815 | 23 | 11 | 2242 |
| Generic | OWL-S | [89] | | ✓ | | ✓ | ALCHOIN(D) | 86 | 88 | 41 | 13 |
| Generic | OWL-Time | [331] | | | ✓ | | SROIN(D) | 21 | 33 | 25 | 15 |
| Generic | PROV-O | [354] | | | ✓ | | ALCRIN(D) | 31 | 44 | 6 | 1 |
| Generic | Procedure Execution Platform (PEP) | [241] | | | ✓ | ✓ | ALCRIF(D) | 6 | 10 | 2 | 1 |
| Generic | QUDT | [297] | | | ✓ | | SHOIQ(D) | 138 | 132 | 102 | 78 |
| Generic | Semantic Sensor Network (SSN) | [38] | | ✓ | ✓ | ✓ | ALRIN | 39 | 34 | 0 | 1 |
| Generic | Vocabulary Status Vocabulary (VSV) | [80] | | | | | n.a. | | | | |
| Health Science | HeLiS | [236] | | | | ✓ | ALCIQ(D) | 281 | 23 | 31 | 30005 |
| Net-IT | INDL | [212] | ✓ | | | ✓ | ALUIF(D) | 26 | 29 | 17 | 0 |
| Net-IT | DevopsInfra (network) | [270] | ✓ | ✓ | | | ALH(D) | 17 | 15 | 25 | 0 |
| Net-IT | DevopsInfra (product) | [270] | ✓ | | | | ALCHOQ(D) | 22 | 20 | 3 | 2 |
| Net-IT | DevopsInfra (workflow) | [270] | | | | ✓ | ALCHOQ(D) | 13 | 16 | 8 | 2 |
| Net-IT | The SEAS Communication ontology | [241] | ✓ | ✓ | | | ALCRIN(D) | 69 | 46 | 5 | 6 |
| Net-IT | ToCo | [294] | ✓ | ✓ | ✓ | | ALCRI(D) | 85 | 41 | 54 | 1 |
| Net-IT | HTTP in RDF | [234] | | | ✓ | | ALH(D) | 15 | 26 | 13 | 0 |
| Process modeling | Activity Ontology | [243] | ✓ | | ✓ | ✓ | SRIQ(D) | 21 | 70 | 12 | 0 |
| Process modeling | BPMN ontology | [219] | | | ✓ | ✓ | ALCRIQ(D) | 158 | 103 | 34 | 0 |
| Process modeling | FOLIO | [62] | | | | ✓ | ALEQ(D) | 29 | 19 | 3 | 0 |
| Process modeling | gist | [288] | ✓ | ✓ | ✓ | ✓ | SROIQ(D) | 138 | 71 | 39 | 15 |
| SE-SI | BOnSAI | [345] | ✓ | | | ✓ | ALCHIN(D) | 99 | 76 | 41 | 1 |
| SE-SI | DogOnt | [84] | ✓ | ✓ | ✓ | | ALCHOIN(D) | 167 | 18 | 26 | 0 |
| SE-SI | IoT-Lite | [220] | ✓ | | ✓ | | ALUI(D) | 20 | 13 | 9 | 0 |
| SE-SI | RAMI Ontology | [155] | ✓ | ✓ | | | ALH+ (D) | 35 | 47 | 19 | 3 |
| SE-SI | SAREF | [239] | ✓ | ✓ | ✓ | ✓ | ALCIQ(D) | 81 | 35 | 5 | 10 |

| Theme | Short title | Ref. | St. | Fu. | Dy. | Pr. | Expressivity | CC | OPC | DPC | IC |
|---|---|---|---|---|---|---|---|---|---|---|---|
| SE-SI | SAREF4SYST | [239] | ✓ | ✓ | | | SRIN | 4 | 9 | 0 | 0 |
| SE-SI | SOSA | [194] | | ✓ | ✓ | ✓ | ALI(D) | 16 | 21 | 2 | 1 |
| SE-SI | The Building Topology Ontology (BOT) | [215] | ✓ | | | | SRIN | 10 | 16 | 1 | 5 |
| SE-SI | mIO! Ontology Network | [226] | ✓ | ✓ | ✓ | ✓ | SROIQ(D) | 624 | 364 | 310 | 520 |

| Theme | MC | St. % | Fu. % | Dy. % | Pr. % | F0 % | F1 % | F2 % | F3 % | F4 % |
|---|---|---|---|---|---|---|---|---|---|---|
| Generic | 18 | 0,0 | 11,1 | 55,6 | 38,9 | **33,3** | 33,3 | 27,8 | 5,6 | 0,0 |
| CyberSec | 11 | 54,5 | 54,5 | 63,6 | 81,8 | 0,0 | 36,4 | 18,2 | 0,0 | **45,5** |
| SE-SI | 9 | **88,9** | **66,7** | 55,6 | 44,4 | 0,0 | 11,1 | **44,4** | 22,2 | 22,2 |
| Net-IT | 7 | 71,4 | 42,9 | 28,6 | 28,6 | 0,0 | **42,9** | 42,9 | 14,3 | 0,0 |
| Process modeling | 4 | 50,0 | 25,0 | **75,0** | **100,0** | 0,0 | 25,0 | 25,0 | **25,0** | 25,0 |
| Health Science | 1 | *100,0* | 0,0 | 0,0 | *100,0* | 0,0 | 0,0 | *100,0* | 0,0 | 0,0 |
| Overall | 50 | 44,0 | 36,0 | **54,0** | **54,0** | 12,0 | 30,0 | **32,0** | 10,0 | 16,0 |

Table 2.4: Number of semantic models and facet coverage ratios by application domain.
This table summarizes Table 2.3 from the perspective of the number of models (*MC*) and the use of facets by primary application domain (*Theme*). The columns *St.*% , *Fu.*%, *Dy.*%, and *Pr.*% correspond to the proportion of models for which the facet has been identified. The columns *Fx*% account for the expressiveness of the models by comparing the proportion of models that meet 0, 1, 2, 3, or 4 facets. Italicized values for the Health Science theme indicate that they may not be representative due to the inclusion of only one model from this family in the sample.

| Code | Importance Level | Description | Count |
|---|---|---|---|
| P13 | Minor | Inverse relationships not explicitly declared | 43 |
| P11 | Important | Missing domain or range in properties | 42 |
| P08 | Minor | Missing annotations | 41 |
| P04 | Minor | Creating unconnected ontology elements | 29 |
| P10 | Important | Missing disjointness | 20 |

Table 2.5: Most common implementation pitfalls in semantic models.
This table highlights the top five implementation pitfalls found in the set of semantic models listed in Table 2.3, as reported by the OOPS! tool.

## 2.4 Anomaly Detection: a Cartography of Algorithmic Methods

NMSs and SIEMs, seen in Section 2.2.2 as umbrella systems for centralized situation understanding, already enable significant operational efficiency for incident management. However, the remaining limitations discussed in Section 2.2.3 show room for improvement in their automated analysis functions towards greater explainability, notification contextualization, and notification grouping and prioritizing. With the aim of enabling these improvements, we provide in this section a summary of the anomaly detection methods we found during the bibliographic research stage, starting with the definition of evaluation criteria in Section 2.4.1, and reporting on our analysis in Section 2.4.2.

### 2.4.1 Evaluation Criteria

The data collected and evaluation criteria used to analyze the anomaly detection methods are as follows:

**Name.** The name of the anomaly detection method or, if not available, the title of the document describing it (e.g. research paper, blog post, tool documentation).

**Primary application domain.** The domain or field of application that originated the anomaly detection method, based on the indications provided by the authors, or if not available, deduced by us.

**Approach.** A short description of the method, and a categorization of the method summarizing its core principle.

**Usage step.** The typical stage of the incident management process – as discussed in Section 1.1.2 and Figure 1.2 – to which the use of the method corresponds, based on the information provided by the authors, or deduced by us otherwise. We define the following three macro stages: *design* (i.e. techniques providing prior knowledge of system operation and enabling optimal deployment based on safety criteria or potential reengineering to meet these criteria), *detection and classification* (i.e. techniques analyzing artifacts related to the system lifecycle to generate an alert for an undesirable situation), and *diagnostic aid* (i.e. techniques enabling the characterization of a given situation).

**Data structure.** The main data structure(s) used within the algorithmic method for its model learning and inference stages. This data structure may differ from that of the input data due to the specificities of the method (e.g. aggregation of heterogeneous data into a knowledge graph, semantic annotation of natural language documents).

### 2.4.2 Algorithmic Methods

In this section, we present the identified algorithmic methods from the literature review, and their categorization based on the previous criteria. Out of 103 references analyzed, 55 emerged with both a primary application domain close to the NetOps and SecOps fields and practicality falling into an incident management stage. We use this shortlist of 55 references in the following paragraphs, providing an overall analysis from the point of view of approaches, usage stage, and data structures. Statistics on the primary application domain for our shortlist, in terms of the number of references, are as follows: CyberSec = 20, Net-IT = 16, Generic = 5, Industry 4.0 = 5, Energy systems = 4, Smart-Cities & Smart-Homes (SC-SH) = 2, Business process = 1, and Software engineering = 1. For complete details on these 55 references, including a summary of each approach, please refer to Table B.1 in Appendix B.2.

**Logic for design and diagnosis *vs* probabilities for detection.** Analyzing the references highlights six families of approaches: **graph-based** where the processing relies on the structure and characteristics of data represented in a graph, with principles drawing from graph theory [6] or

message passing [245]; **knowledge-based** where deductive[30] and abductive[31] reasoning leverages domain knowledge organized in taxonomies or ontologies; **Markov model** [327] where a probabilistic model describing the potential state transitions of a system is used for inference; **ML-based** where the probabilistic model used for inference leverages correlations between multiple observational variables as an indicator; **model checking** where a behavioral model in the form of a Finite State Automaton (FSA) [125, 114] is used for inference; **rule-based** where deductive reasoning applies leveraging a set of business rules typically of the IF-THEN-ELSE form. Table 2.6 shows the distribution of references in these approach families across the three usage stages defined in Section 2.4.1. The proportions reported in this table indicate a predominance of works applicable to the *detection & classification* stage. Additionally, there is a prevalence of logic-based approaches in the *design* and *diagnostic aid* stages, as opposed to correlation-based approaches in the *detection & classification* stage. These trends suggest a current focus for anomaly detection on the ability to capture complex situations without necessarily leveraging prior or expert knowledge of the ICT systems. This contrast also suggests a limited current capability for coupling between logic-based and probabilistic approaches. However, the presence of the graph-based approach in all three usage stages suggests that a significant portion of the addressed problems involves the interconnected nature of the data.

| Approach | System Design | Detection & Classification | Diagnostic Aid |
|---|---|---|---|
| Rule-based | 1 | 5 | 0 |
| Model checking | 1 | 2 | 1 |
| Knowledge-based | **2** | 6 | **6** |
| Markov model | 0 | 1 | 0 |
| Graph-based | 1 | 10 | 5 |
| ML-based | 0 | **14** | 0 |
| Overall | 5 (9,1 %) | **38 (69,1 %)** | 12 (21,8 %) |

Table 2.6: Approach family and incident management stage in analyzed papers.
This table provides information on the distribution (in number and proportion) of the analyzed papers, based on the approach family (the middle line serves as an arbitrary separation between logic-based and correlation-based approaches) and the stage of the incident management process involved. Values in bold highlight the most representative approach for a given stage of the incident management process.

**Partially ordered sets and graphs as key data structures.** Table 2.7 shows the distribution of data structures used in algorithmic solutions, as a function of the solution's approach family and usage stage. The following five types of structures emerge from our analysis: data with an

[30] In an inference process, the formation of a conclusion is based on generally accepted statements or facts (i.e. from general or universal premises).

[31] In an inference process, observational facts (major premise) are evident, but the cause (minor premise) and therefore the conclusion are only probable. Backward chaining on a rule set [228] is a typical implementation of this process, where the system checks if an hypothesis is true or not.

**order** relation, which includes timestamped *sequential data* such as event logs and alarms, *network traffic* captures, and *time series* for regularly sampled measurements such as data throughput or temperature; **graph** (*static* or *streaming*) such as network topology; **tabular** data, such as a list of assets with their characteristics; multi-dimensional **data points**; and so-called **mixed** approaches that simultaneously use a combination of the aforementioned structures. From the proportions in Table 2.7, we observe that data with an order relation are generally predominant across all usage stages. In the *detection & classification* usage stage, approaches primarily utilize data structures – in descending order of preference – such as ordered data, graphs, and tables, with a prevalence of ordered data for ML-based approaches. In the *diagnostic aid* usage stage, approaches make the most use of mixed structures. These observations suggest a general tendency for *detection & classification* approaches to focus on the temporal evolution of systems, while *diagnostic aid* approaches tend to focus on a broader context of the system's state.

| Approach | Seq. data | | Seq. data (network) | | Time series | | *Ordered (1,2,3)* | | Graph | | Graph streams | | Tabular | | Data points | | Mixed seq.+ graph | | Mixed seq.+ tab. | | Mixed seq.+ unstr. | | *Mixed (9,10,11)* | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | [%] | | [%] | | [%] | Σ | [%] | | [%] | | [%] | | [%] | | [%] | | [%] | | [%] | | [%] | Σ | [%] |
| **Design** | | | | | | | | | | | | | | | | | | | | | | | | |
| G.-based | | 0,0 | | 0,0 | | 0,0 | | *0,0* | | 0,0 | | 0,0 | | 0,0 | | 0,0 | 1 | 10,0 | | 0,0 | | 0,0 | *1* | *8,3* |
| K.-based | | 0,0 | | 0,0 | | 0,0 | | *0,0* | | 0,0 | | 0,0 | | 0,0 | | 0,0 | 1 | 10,0 | 1 | **100,0** | | 0,0 | *2* | *16,7* |
| M. check. | 1 | 7,1 | | 0,0 | | 0,0 | *1* | *4,0* | | 0,0 | | 0,0 | | 0,0 | | 0,0 | | 0,0 | | 0,0 | | 0,0 | | *0,0* |
| R.-based | | 0,0 | | 0,0 | | 0,0 | | *0,0* | 1 | 9,1 | | 0,0 | | 0,0 | | 0,0 | | 0,0 | | 0,0 | | 0,0 | | *0,0* |
| **Detection & Classification** | | | | | | | | | | | | | | | | | | | | | | | | |
| G.-based | 2 | 14,3 | | 0,0 | 1 | 16,7 | *3* | *12,0* | 3 | **27,3** | 1 | **50,0** | 2 | **66,7** | | 0,0 | 1 | 10,0 | | 0,0 | | 0,0 | *1* | *8,3* |
| K.-based | 2 | 14,3 | 1 | 20,0 | | 0,0 | *3* | *12,0* | 3 | **27,3** | | 0,0 | | 0,0 | | 0,0 | | 0,0 | | 0,0 | | 0,0 | | *0,0* |
| Markov | 1 | 7,1 | | 0,0 | | 0,0 | *1* | *4,0* | | 0,0 | | 0,0 | | 0,0 | | 0,0 | | 0,0 | | 0,0 | | 0,0 | | *0,0* |
| ML-based | 5 | **35,7** | 1 | 20,0 | 5 | **83,3** | *11* | ***44,0*** | | 0,0 | 1 | **50,0** | | 0,0 | 2 | **100,0** | | 0,0 | | 0,0 | | 0,0 | | *0,0* |
| M. check. | 1 | 7,1 | | 0,0 | | 0,0 | *1* | *4,0* | 1 | 9,1 | | 0,0 | | 0,0 | | 0,0 | | 0,0 | | 0,0 | | 0,0 | | *0,0* |
| R.-based | 1 | 7,1 | 3 | **60,0** | | 0,0 | *4* | *16,0* | 1 | 9,1 | | 0,0 | | 0,0 | | 0,0 | | 0,0 | | 0,0 | | 0,0 | | *0,0* |
| **Diagnostic Aid** | | | | | | | | | | | | | | | | | | | | | | | | |
| G.-based | | 0,0 | | 0,0 | | 0,0 | | *0,0* | | 0,0 | | 0,0 | | 0,0 | | 0,0 | 5 | **50,0** | | 0,0 | | 0,0 | *5* | ***41,7*** |
| K.-based | | 0,0 | | 0,0 | | 0,0 | | *0,0* | 2 | 18,2 | | 0,0 | 1 | 33,3 | | 0,0 | 2 | 20,0 | | 0,0 | 1 | **100,0** | *3* | *25,0* |
| M. check. | 1 | 7,1 | | 0,0 | | 0,0 | *1* | *4,0* | | 0,0 | | 0,0 | | 0,0 | | 0,0 | | 0,0 | | 0,0 | | 0,0 | | *0,0* |
| Overall | 14 | **25,5** | 5 | 9,1 | 6 | 10,9 | *25* | ***45,5*** | 11 | 20,0 | 2 | 3,6 | 3 | 5,5 | 2 | 3,6 | 10 | 18,2 | 1 | 1,8 | 1 | 1,8 | *12* | *21,8* |

Table 2.7: Data structures within algorithmic methods for anomaly detection.

This table provides information on the distribution (in number and proportion) of the main data structures used within the algorithmic solutions in the analyzed papers, based on the algorithmic approach family and the stage of the incident management process involved. Values in bold highlight the most representative approach for a given data structure. The columns in italics represent cumulative values (*ordered* = columns 1 + 2 + 3, *mixed* = columns 9 + 10 + 11) to provide a summary view of similar structures.

## 2.5 Summary of Technological and Scientific Challenges

With the aim of facilitating the capture and interpretation of complex situations occurring on networks, we have proposed a review of NMS and SIEM DSSs capabilities, semantic models, and algorithmic solutions that, through their individual improvement or combination, could allow to achieve both crisp and approximate reasoning on the interplay between a large-scale ICT system architecture and its operation. Table 2.8 summarizes the key findings from the analysis phase of the survey. It highlights that NMS and SIEM DSSs are well-established tools that simplify the analysis of diverse data sources (e.g. assets database, logs and alarms from IPoDWDM and VM management systems, vulnerability scans, etc.). However, their effectiveness is hindered by the implicit heterogeneity of these sources, which limits the ability to contextualize network and service failures and implement comprehensive analysis solutions that consider a broader range of information, including network topology. Furthermore, while there are semantic models that align with the discourse domain of NetOps and SecOps, enabling the utilization of knowledge graphs for knowledge representation, they do not individually fully cover the necessary discourse domain. Additionally, they do not inherently support reasoning about system state changes related to procedures with conditional branching in decision-making processes. Finally, various algorithmic methods exist to address key steps in the incident management process. However, individually, they do not capture and analyze phenomena that involve temporal, structural, logical, and probabilistic aspects simultaneously.

**Advancing through knowledge engineering and massive data integration.** Data heterogeneity and interrelatedness between data entities (i.e. distinct and persistent units of information) appear to be cornerstones for advancing the capabilities of DSSs. In this survey, we assumed that knowledge graphs naturally align with these two notions in the sense that they bring an abstraction level for standard interpretation and logical reasoning over heterogeneous data. Sketching a next-generation NMS/SIEM therefore leads to understand how to bring knowledge graphs to such a system, while considering that NMSs & SIEMs systems rely on a multitude of both streamed and static data sources.

Several tools have been proposed in different application domains for Knowledge Graph Construction (KGC). For streamed data: RMLStreamer [122] applies declarative mapping on the fly to structured data streams (e.g. file, Kafka topic) with RDF Mapping Language (RML) [24] rules; StreamingMASSIF [292] uses basic string substitution for mapping, and allows for real-time reasoning (e.g. SPARQL query processing, Complex Event Time processing); C-SPARQL [91] extends the SPARQL query language for continuous reasoning within a publisher/subscriber platform. For static data: RMLMapper [24] enables data fetching and declarative mapping with RML rules; Ontop [132] creates a virtual graph representation of

| KR | CX | AD | XP |
|---|---|---|---|
| **Decision Support System** – Section 2.2 | | | |
| • Data is generally recorded in a structure close to its original format, typically in a tabular structure, without annotations.<br>• Data normalization can be performed at different stages of the data ingestion process, depending on the design choices of the DSS. | • The data recording/management block utilizes a combination of storage technologies and often follows a hub/aggregator architecture. | • The analysis block can utilize internal resources and/or external services.<br>• Correlation-based analysis on top of rule-based characterization is common.<br>• Automated contextualization may not consider network topology or network operation information. | • Explainability is addressed by contextualizing notifications through rendering, rewriting, and enriching. |
| **Semantic models** – Section 2.3 | | | |
| • There are many models with good overall quality, but none fully cover all aspects required for reasoning on the interplay between network architecture and its operation.<br>• Some models describe network topology but at a high level or for specific domains. | • Current data models reveal various opportunities for enrichment, improvement, or linking of the models.<br>• Inference capabilities may need to be adjusted based on experiments with practical cases. | • Anomaly detection with semantic models currently generaly rely on graph traversal or rule-based techniques.<br>• Semantic Web-based models have limited capabilities for conditional branching knowledge. | • Explainability is naturally addressed for semantic models due to their explicit knowledge representation, roots in logic, and utilization of shared vocabularies. |
| **Algorithmic methods** – Section 2.4 | | | |
| • Data with an order relation is generally predominant across all usage stages.<br>• In the detection and classification stage, approaches primarily use ordered data, graphs, and tables, with a prevalence of ordered data for machine learning-based approaches.<br>• Detection and classification approaches focus on the temporal evolution of systems, while diagnostic aid approaches consider a broader context of the system's state (typically using mixed structures). | • There are few solutions directly designed for streamed analysis, and they generally require a training phase. | • Focus on detection and classification in current works on anomaly detection.<br>• Focus on the capability to capture complex situations without relying heavily on prior or expert knowledge of ICT systems in current works on anomaly detection.<br>• Interconnected data (e.g. graphs) is important, but its usage is not widespread. | • Explainability is naturally addressed for algorithmic methods with roots in logic.<br>• Limited current capability for coupling between logic-based and probabilistic approaches. |

Table 2.8: Key findings from the survey.
Summary of the key messages from the *capabilities of DSSs*, *semantic models*, and *algorithmic methods for anomaly detection* aspects according to the research axes Knowledge Representation (KR), CompleXity (CX), Anomaly Detection (AD), and eXPlainability (XP) defined in Section 2.1.

various data sources via SPARQL queries; and SLOGERT [26] orchestrates log modeling and annotation with Cybersecurity Threat Intelligence (CTI) tags[32].

These solutions provide a foundation towards a KG-based NMS/SIEM. However, additional research effort is required to achieve an end-to-end solution design that bridges the Semantic Web tools with the design patterns observed in industrial DSSs. It includes satisfying requirements for distributed processing, separation of concerns, data sketching (i.e. enabling both early and posterior reasoning on data), openness to third-party databases/tools, and re-use of well-established frameworks (e.g. declarative data transformation, message passing). Indeed, in end-to-end frameworks [290, 377, 315, 58], the KGC step is never considered singular, initial or terminal, but rather is the subject of multiple instances of a similar tool/principle within processing flows depending on the application field. In addition, this step is always placed between heterogeneous non-RDF data and a knowledge graph working sometimes as a main data storage, and sometimes as a support for third-party inference processes. This variety of options in itself constitutes a field of exploration to be pursued.

Regarding the standardized interpretation of data, the variety of available semantic models encourages understanding how to leverage existing implementations of vocabularies without introducing complexity. This can be achieved by avoiding using vocabularies with loosely coupled semantics to the application domain, avoiding the introduction of a new ontology that lacks interoperability with other standard vocabularies, and preventing the creation of an ontology network that would unnecessarily lengthen reasoning paths. Various knowledge engineering methodologies allow for approaching this research, whether it be general approaches such as practical guides to semantic modelling [150] and ontology design patterns [266], or more focused approaches: Competency Questions [385], Dichoscope [49], DOE [299], NeOn [336], OntoClean [129], ontology design with Formal Concept Analysis (FCA) [265], automated ontology learning from raw data with Text2Onto [71], automated derivation of class taxonomies from an already existing knowledge graph [293], and translation of formal models to an ontology [10, 348, 45].

**Advancing in anomaly detection and explainability.** Put simply, anomaly detection methods in the context of NetOps and SecOps aim to identify deviations from normal behavior and flag potentially undesirable/suspicious activities at both the ICT system and user level. While various approaches have been proposed, as discussed in Section 2.4, only a few of them simultaneously combine intrinsic explainability through the use of explicit representations (e.g. knowledge graphs, FSA, Petri nets) and the ability to handle interconnected multi-dimensional data, including the temporal dimension. Assuming the use of knowledge graphs as a foundational formalism for NMS/SIEM DSSs, this prompts us to understand to what

[32] As for SLOGERT v0.9.1: with MITRE CEE categories from http://cee.mitre.org/language/1.0-alpha/

extent the existing algorithmic solutions are suited to it.

Focusing on activity modeling and analysis, trace-based reasoning [21] shows opportunities in creating tools for semantically interpreting digital services artifacts using controlled vocabularies and semantic models. Indeed, the representation of events and activities within knowledge graphs is implemented across a range of data models, encompassing both domain-independent and domain-specific contexts: process modeling and execution (BBO [22], Petri nets-related [97, 186], HTTPinRDF [178, 234]); causal analysis (FARO [384]); cyber-security (UCO [388], MITRE D3FEND [284]); smart cities (iCity ActivityOntology [244]).

At the same time, User and Entity Behavior Analytics (UEBA) corresponds to the temporal dimension of the knowledge graph, which further motivates the search for how an anomaly context capture (e.g. a subgraph centered around the undesirable event) can accommodate historical graphs [107] or a graph that evolves over time. Graph embeddings [312] typically allow for capturing the context of graph entities. However, since learning embeddings (i.e. a vector representation of the subgraph that enables similarity calculations) is a potentially time-consuming process performed on a snapshot of the graph, using such an approach requires defining the content of the subgraphs to capture [169], particularly in relation to the speed of graph evolution. Moreover, since such an approach may also not be generalizable, the simultaneous implementation of various approaches can prove useful, whether by following the principles of cooperative decision making [46] (i.e. a heuristic approach for problem-solving by using results obtained from complementary inference techniques) or by combining properties of inference models into a single one [111, 51, 314, 64] (e.g. by including the power of logical reasoning into ML-based models).

# Part I

# Knowledge Engineering and Massive Data Integration

# Introduction

For NetOps & SecOps teams, efficiency in incident management and situation understanding requires a comprehensive view on how communication devices are interconnected and are performing. However, the information is spread across heterogeneous data sources which triggers information integration challenges. Semantic Web technologies are essential in this context, as they address the problems of data heterogeneity, knowledge sharing and logical/probabilistic reasoning. Existing data models enable to represent computing resources and how they are allocated. However, to date, there is no model to describe the inter-dependencies between the structural, dynamic, and functional aspects of a network infrastructure. Additionally, well-established Network Monitoring Systems (NMSs) and Security Information and Event Management (SIEM) systems do not explicitly use Semantic Web knowledge representation, thus limiting the ability to efficiently analyze incidents in their full complexity.

To address these gaps, we propose the NORIA Ontology (Chapter 3) that has been developed together with network and cybersecurity experts in order to describe an infrastructure, its events, diagnosis and repair actions performed during incident management. We also propose an end-to-end data processing architecture (Chapter 4) that combines NMS/SIEM design patterns with Semantic Web tools to construct a knowledge graph suitable for understanding incident situations using data from various sources relevant to network operations. The platform features batch/stream processing, declarative data mapping with RML, data patching & reconciliation with SPARQL queries and SKOS, provenance auditability with centralized configuration and data management, and semantic data transfer with Kafka. In conjunction with NORIA-O, the proposed architecture has been instantiated and tested in an industrial setting, producing an RDF knowledge graph that shows strong potential for addressing cross-domain anomalies from heterogeneous data.

# Chapter 3

# An Ontology for Anomaly Detection and Incident Management in ICT Systems

## 3.1 Introduction

To achieve efficiency in incident management and situation understanding, NetOps & SecOps teams require the ability to simultaneously and quickly correlate and interpret a large number of heterogeneous technical information sources. This is even more important when operating large-scale networks, which often consist of various ICT sub-systems and have complex inter-dependencies between services and infrastructure (Chapter 1). To tackle these cross-domain data interpretation and incident management challenges, we argue that a graph-based explicit knowledge representation would help capturing complex network situations (e.g. discrepancy of routing metrics with respect to an engineering rule, lack of redundancy in a distributed service) and reasoning on them (e.g. inventory list to scrutinize further, cause and remediation procedure search). Our main contribution in this regard is the NORIA Ontology (NORIA-O) for representing network infrastructures, incidents and operations on networks. This ontology re-uses and extends well-known ontologies such as SEAS [240, 241], FOLIO [62], UCO [388], ORG [86], BOT [215] and BBO [22]. It also includes controlled vocabularies for handling data from various ICT systems and incident situations through a small set of shareable definitions. NORIA-O has been developed within the Orange company. Its long-standing experience on complex network management allows us to back NORIA-O with insightful details from domain experts and to evaluate the model with real-world data. As shown later in Chapter 4 and Chapter 7, NORIA-O has also been successfully used in a knowledge graph construction pipeline in an industrial setting and for capturing and classifying incident contexts using graph embeddings. The ontology, controlled vocabularies and their associated documentation are available at https://w3id.org/noria.

The remainder of this chapter is organized as follows. In Section 3.2, we describe the methodology we follow to design NORIA-O, starting from competency questions that capture the knowledge of experts. In Section 3.3, we deep dive into the different concepts as well as into

the associated vocabularies. We evaluate the ontology with respect to our requirements and competency questions in Section 3.4. We exemplify how NORIA-O is used for the supervision of a network infrastructure in Section 3.5. Finally, Section 3.6 concludes this chapter by summarizing the work done and presenting preliminary conclusions on the opportunities and potential future developments associated with it.

## 3.2 Methodology

In this section, we describe the knowledge engineering methodology we used to develop NORIA-O. First, we capture Competency Questions (CQs) from a panel of experts familiar with network operation issues and derive archetypes from these CQs for further analysis (Section 3.2.1). Second, we show how we designed the conceptual model (Section 3.2.2).

### 3.2.1 Competency Questions and Conceptualization

We gathered experts from several entities in the fields of engineering, operations, supervision, and incident management on networks and data centers, including teams from NOC and SOC. This panel consists of 16 experts who collectively represent 150 operations team members. To effectively capture the knowledge of the experts, we followed a user-centered design methodology combined with ontology engineering methods. From our review of the literature, the Competency Question approach [385] turns out to be the most intuitive and straightforward with respect to how NOC and SOC teams use and talk about their tools. Indeed, this approach involves extracting the conceptual model of the knowledge domain by analyzing user queries expressed in natural language through a set of semantic patterns. During several iterations of knowledge capture meetings on a shared notebook, the experts could validate, invalidate, add and modify Competency Questions (CQs). At the end of this stage, the teams validated 26 CQs presented in Table 3.1. This includes questions on events, resources (e.g. server, router), applications (e.g. Domain Name System, Video-on-Demand platform), log and alarms (e.g. login, CPU overload) and operation plan (e.g. SSL/TLS certificate renew, IS-IS interface re-prioritization).

From the set of CQs, we derived a conceptual model of the domain of discourse by applying the "*Competency Question archetype mapping*" approach [385, Section 4.3]. For example, CQ#1 can be mapped to the "*Which [CE1] [OPE] [CE2]?*" archetype (ID: 1), yielding to breaking down the competency question into the following components: $CE1$ = Asset (resource/application/site), $OPE$ = areContainedIn, and $CE2$ = Incident. We also adhered to the guidelines of the Linked Open Terms (LOT) methodology [224], which notably include reusing or aligning with existing vocabularies.

Upon scrutinizing the conceptual model and candidate vocabularies, two characteristics

are observed. Firstly, concepts derived from the Table 3.1 can be referred to as "atomic concepts" (e.g. application, alarm, resource), representing concepts that are not defined by composition but are considered indivisible in nature. Thus, we can expect to use simple relationships (potentially hierarchical) between concepts during the modeling and implementation phases. Secondly, research domains related to ICT systems management (such as event spreading [217], software engineering [138], knowledge management, and automated reasoning [46]), exhibit abstract concepts common to these domains and the NORIA-O field, such as *physical vs functional* and *cause vs consequence*. Therefore, we suggest structuring the NORIA-O domain concepts using similar facets to leverage the approaches and tools applicable in these research domains, such as finite state automatons and Markov decision processes. For instance, combining the facets allows for a comprehensive analysis of the complexity and observability levels of networks, which we refer to as a hybrid "concrete-conceptual" model (Figure 1.4). In this model, assets' states dynamically vary based on behavioral rules and are interpreted through higher-level composite concepts. Consequently, predicting the next set of states/concepts becomes a sequential decision under uncertainty problem. We further define these facets in the next section.

Table 3.1: NORIA-O Competency Questions (CQs)
In this table, we list the NORIA-O CQs collected during knowledge capture meetings (Section 3.2.1), along with their corresponding archetype (Arch. ID, as defined in [385, Section 4.3]) and authoring tests results (Section 3.4, with the number of implemented queries for the evaluation stage). Facets (Section 3.2.2): $S$ = structural, $F$ = functional, $D$ = dynamic, and $P$ = procedural.

| # | CQs | Facets | Arch. ID | AT Eval. |
|---|---|---|---|---|
| 1 | Which resource/application/site is concerned by a given incident? | S, F, D | 1 | OK (4) |
| 2 | What assets are shared by a given asset chain? | S, F | 6 | OK (1) |
| 3 | What logs and alarms are coming from a specified resource? | S, D | 1 | OK (1) |
| 4 | Which metrics are coming from a specified resource? | S | 1 | OK (1) |
| 5 | To which event family does this log belong and is this event normal or abnormal? | D, P | 3 | OK (1) |
| 6 | What events are associated with a given event? | D | 1 | OK (1) |
| 7 | Which agent/event/resource caused the event under analysis? | S, F, D, P | 1 | OK (3) |
| 8 | What do the various fields in the log refer to? | D | 1, 3 | OK (1) |
| 9 | Is there any pattern in a given set of logs/alarms? | D, P | 1, 6 | AI (1) |
| 10 | What interventions were carried out on this resource that could have caused the incident? | S, D, P | 1, 6 | OK (2) |
| 11 | What was the root cause of the incident? | D, P | 6 | AI (1) |
| 12 | Which sequence of events led to the incident? | D, P | 6 | OK (1) |
| 13 | On which resource did this sequence of events take place and in which order? | S, D | 1 | OK (1) |
| 14 | What past incidents are similar to a given incident? | D, P | 6 | AI (1) |
| 15 | What operation plan (automation, operating procedures, etc.) could help us solve the incident? | D, P | 1, 3 | AI (1) |
| 16 | What corrective actions have been carried out so far for a given incident? | D, P | 1 | OK (1) |
| 17 | What is the list of actions taken that led to the resolution of the incident? | D, P | 1 | OK (1) |
| 18 | Given all the corrective actions carried out so far for the incident, what assumptions covered the actions taken? | D, P | 1, 4 | AI (1) |
| 19 | What has been the effect of the corrective actions taken so far for the incident? | D, P | 1 | OK (1) |
| 20 | Given all the corrective actions carried out so far for the incident, what possible actions could we still take? | D, P | 6 | AI (1) |
| 21 | What is the summary of this incident and its resolution? | D | 1 | OK (1) |
| 22 | Which agents were involved in the resolution of the incident? | D | 1 | OK (1) |
| 23 | What is the financial cost of this incident if it occurs? | D | 2 | Ext. |
| 24 | How long before this incident is resolved? | D | 1 | AI (1) |
| 25 | What are the vulnerabilities and the associated risk levels of this infrastructure? | S, F, D | 1, 2 | AI (1) |
| 26 | What is the most likely sequence of actions that would cause this infrastructure to fail? | S, F, P | 6 | AI (1) |

### 3.2.2 Domain of Discourse and Modeling Strategy

**Facets.** Considering dynamic ICT systems with constrained and multi-level functional behavior, we define the four following facets for structuring the knowledge domain. An illustration of these facets is provided in Figure 3.1.

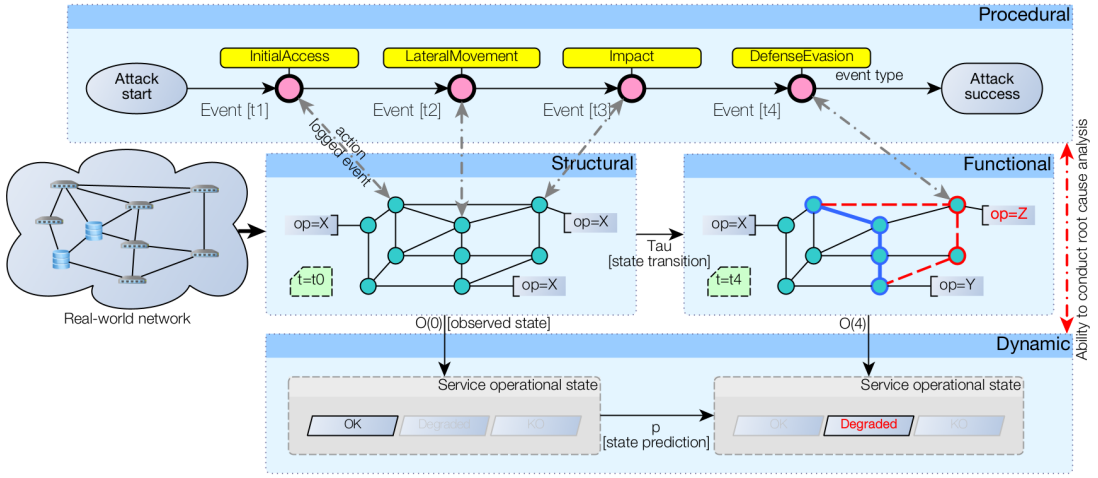



Figure 3.1: ICT system state transition model and relations to the NORIA-O facets.
The representation of a network can be divided into four facets: *structural*, *functional* (the blue path indicates an operational data flow, the red path a faulty flow), *dynamic*, and *procedural* (logged events are related to cybersecurity attack tactics from the MITRE ATT&CK matrix [284]). *Tau* stands for state transition, $O(t)$ for observed state at time $t$, and $p$ for state prediction.

- **The structural facet** describes the physical and logical elements of the network. It allows modeling the equipment classes, connections and compositions. This facet aims to support calculations on network objects and properties (direct or deduced) and calculations on the physical and logical structures (real or patterns).

- **The functional facet** describes services provided and diffusion areas. This facet makes it possible to meet the need for functional isomorphism (e.g. replacing one equipment with another performing the same function). It allows modeling the service types, interactions between them, and compositions. This facet allows calculations on network domains and their properties (direct or inferred) and calculations on services and streams (e.g. “end-to-end” notion).

- **The dynamic facet** describes the sequence of events. It allows modeling the occurrence of an event on a given equipment or service as well as precedence relationships. This facet aims to support time calculations (absolute, relative, membership) and causality calculations (first order or probabilistic).

- **The procedural facet** describes how things work and should be interpreted. Automation principles (e.g. fail-over mechanisms of redundant systems) or operation principles

(e.g. doubt removal procedures) are expected parts of this facet. Associated application goals are deductive/abductive reasoning over facts and reflection over knowledge for automated learning (discovery/recommendation) of procedures (e.g. evolutionary search over targeted goals, composition calculus over sequences of events).

**Modeling strategy.** Considering the domains of anomaly detection and incident management, we consider incidents as a central concept for i) computing and reasoning about anomaly signatures, and ii) linking trouble tickets to anomaly signatures for root cause analysis tasks. We introduce the `noria` namespace, which encompasses the entire set of NORIA-O concepts and relations in a unified manner. To align with risk management and business modeling practices (such as the incident logging and categorization steps in the ITIL's Incident Management Process model – IMP [334]), we adopt a top-to-bottom modeling strategy, starting from the process and extending to objects. Due to the "atomic concepts" characteristic of the knowledge domain (Section 3.2.1) and our top-to-bottom approach, we primarily focus on an axiomatization based on subsumption with RDFS [82], and with OWL [319] for cases that require additional logical structuring (e.g. class disjointness, property qualification)[1].

**Model re-use.** Following the best practices in ontology development, we aim to re-use existing data models and vocabularies as a base and extend them to represent domain-specific classes and properties. From RDF-based ontologies, we interconnect and/or extend the following models: **BBO** [22] for describing activities from the business process modeling perspective in conformance to the Business Process Model Notation (BPMN); **BOT** [215] for describing resource locations and enabling geographical neighboring analysis for root cause analysis tasks; **DCTERMS** for standard management of NORIA-O instances as parts of a catalog; **DevOpsInfra** [270] for enabling potential interactions of NORIA-O with the DevOps perspective; **FOAF** [81] for describing social organizations; **FOLIO** [62] for enabling Root Cause Analysis (RCA) tasks based on the FMEA approach; **ORG** [86] for describing stakeholders and related organizations; **SEAS & PEP** [240, 241] for describing technological systems, measures, commands, and results; **UCO** [388] for enabling cyber-security risk assessment on instances of NORIA-O; **SLOGERT** [26] for describing system logs and enabling potential usage of the SLOGERT log interpretation framework.

From non RDF-based data models, we take advantage of the concept hierarchy and vocabulary definitions from the **TM Forum Data Model**[2] for enabling an interoperable definition of trouble tickets and change requests with third-party OSS and DSSs, **ITU-T** [166, 164] for standard

[1] During implementation, cardinality restrictions were not prioritized as they were not considered crucial. Instead, we see cardinality restrictions as more beneficial for post-implementation data quality tasks using SHACL [142].

[2] https://github.com/tmforum-apis

definitions of notifications and ways to handle them within the telecommunication industry, **IETF** for precise use of terminology in the context of a Request for Comments (RFC)[3].

We propose implementing alignment with third-party data models and vocabularies on a class or property basis when relevant, using the dedicated OWL and RDF constructs such as `owl:equivalentClass`, `rdfs:subClassOf`, or `rdfs:isDefinedBy`. Similarly, we aim to provide guidelines for directly instantiating these vocabularies in cases where aligning a class or property would be redundant.

**Modeling observations.** Considering observables and their state change (e.g. the operational state of a network interface, the temperature measurements from a sensor), we observe that modeling and logging observations can be done:

(a) as a string,

(b) as a concept from a controlled vocabulary,

(c) as an instance,

(d) as an instance with time property or time instance (e.g. using reification, or following the `sosa:Observation` model[4]).

These four options are relevant for the NORIA-O application domain. The concern is not about choosing one option for all situations, but how we can mix them. Hence, we adopt the following selection criterion:

- use (a) and (b) for invariant properties,
- use (c) and (d) for time-dependent and/or specific use-case extensions to NORIA-O (i.e. additional observables are defined in a side vocabulary so the main ontology remains stable).

**Controlled vocabularies.** Because of potentially heterogeneous data incoming from varied ICT systems and incident situations to handle, we take notes of terms from datasets and other ontologies for building up a controlled vocabulary. This aims at efficient management of anomaly detection patterns, rules and methods by reducing the lexical range of possible situations to interpret. For this, we propose a set of domain-specific vocabularies (e.g. Incident Management Process, Application, Notification vocabularies) modeled as SKOS concepts within concept schemes (e.g. the milestones of the Incident Management Process). We

[3] https://datatracker.ietf.org/

[4] https://www.w3.org/TR/vocab-ssn/#SOSAObservation

add, whenever available, alternate definitions of the concepts for reconciliation of similar object attribute values through a single concept reference (e.g. communication devices may report the same status of network interfaces with varied terms such as "*active*", "*up*" or "*enabled*"). We also use the concept scheme approach for enabling multiple interpretation of a similar concept. For example, an event may be categorized as an `integrityViolation` based on the analysis of the event text, which allows us to reason on the event type and infer a `SecurityAlarm` thanks to a dual membership of the `integrityViolation` concept definition. The implementation of the vocabulary reconciliation task (e.g. relating the observed network interface administrative status to the adequate concept reference using natural language processing) is out of the scope of this work and is left to the NORIA-O user's choice.

## 3.3 NORIA-O: Formalization and Implementation

We have implemented the NORIA-O conceptual model in RDFS/OWL-2. NORIA-O consists of 59 classes, 107 object properties, and 71 datatype properties. It is organized with the four facets presented in Section 3.2.2 and illustrated in Figure 3.2. Its expressivity is $ALCHOI(D)$ as per Protégé 5.1. In this section, we introduce some of the main concepts and properties.

### 3.3.1 Resources, Network Interfaces, Network Links and Applications

Within computer science, a resource is some "*part contributing to the functioning of an ICT system.*" Similarly, as per the TM Forum Data Model[5], a resource is "*an abstract entity that describes the common set of attributes shared by all concrete resources in the inventory*". Therefore, we define the `Resource` class for describing any physical or logical manageable entity composing the network. Defining the type of a resource is made possible through object properties such as `resourceType` (i.e. controlled-vocabulary concepts such as rack, server, router, virtual machine, etc.) and `resourceProductModel` (i.e. entity model instances). Additional properties allow for identifying the resources based on their logistic identifier, hostname, installation date, etc.

Locating and reasoning over a physical entity from a geographical standpoint is available with a chain of `bot:containsZone` and `bot:hasElement` properties, starting from a `bot:Site` with `bot:hasZeroPoint` property, down to a `Locus` concept for precise `Resource` location within a `Room` (i.e. a specialization of `bot:Space`). Locating a resource is also available through a dependency relationship with the `seas:subSystemOf` object property from the SEAS SystemOntology[6]. This allows for describing and reasoning with parts from various levels of organization (e.g. a virtual router instance in a router, a hard drive in a server, a server

[5] https://github.com/tmforum-apis/Open_Api_And_Data_Model
[6] https://w3id.org/seas/SystemOntology

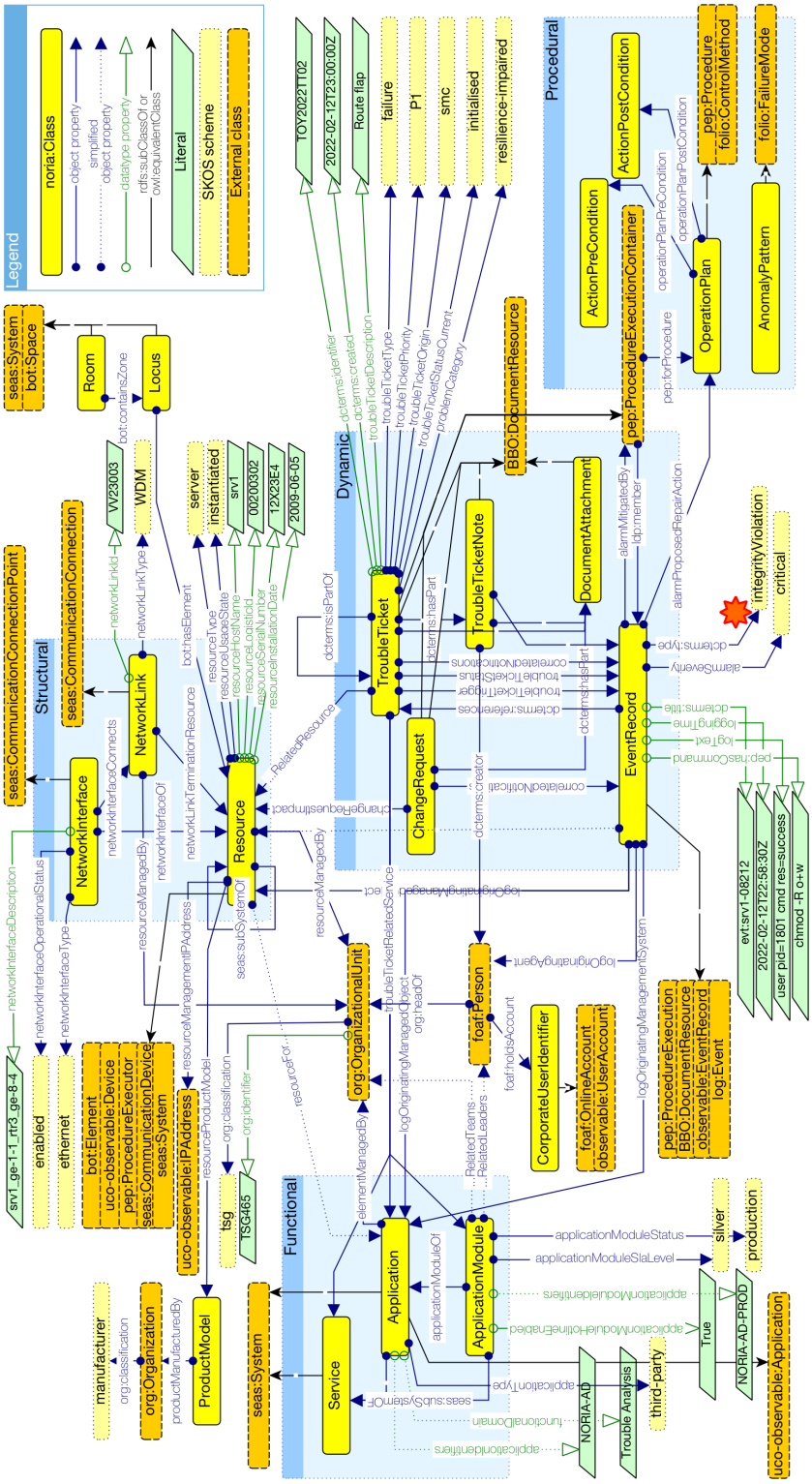



Figure 3.2: Overview of the NORIA-O model.
We depict the most important classes and properties, including related domain of discourse facets and relationships to third-parties models. `noria` is the default namespace. The red star indicates where events are characterized within the data model as incidents or anomalies. Examples are provided for the "literal" and "SKOS scheme" blocks. For the sake of clarity, some object properties are grouped (see "simplified object property") for a light representation of similar properties (i.e. same `rdfs:domain` or same `rdfs:range`). The diagram partly follows the Graffoo specification [330].

in a rack, a rack in a `bot:Site`, etc.).

Describing the network topology itself is defined with the `NetworkInterface` and `NetworkLink` classes. We align with the SEAS CommunicationOntology[7] model through object properties such as `networkInterfaceOf` and `networkLinkTerminationResource`. It should be noted that this approach is compatible with advanced networking features such as sub-interfaces, link aggregation, virtual channels, etc. Operational characteristics for interface and links are available with properties such as `networkInterfaceOperationalStatus` and `networkInterfaceRoutingPriorityMetric`.

The `Application` concept enables to define models of purpose (e.g. internet access, network time, alarm monitoring) for sets of resources, and to categorize these with respect to their nature (i.e. controlled-vocabulary concepts such as infrastructure, service platform, etc.). An `ApplicationModule` is a concrete instance of a given model (e.g. national federated Internet access, corporate network time service, monitoring for in-production devices). This grouping level enables to relate specific technical skill centers, such as a named IP backbone engineering or support team (Section 3.3.4), to a given module for specific expertise (e.g. re-engineering, diagnosis and repair). Additional properties allow for finer grain resources and events management at the module level such as `applicationModuleSlaLevel` for prioritizing servicing teams, or `applicationModuleHotlineEnabled` for triggering night shift support teams.

We also define the `Service` concept in accordance to the TM Forum TMF638 Service Inventory API[8] and the IETF SFC Architecture [167] for grouping instances of `ApplicationModule`, and thus enabling the data path and application composition perspectives of the functional facet (Section 3.2.2). The network topology related to a given service is inferred from the set of resources, network interfaces and network links included in each application that is part of the service. We observe that, although deterministic, the data path granularity calculus for some communication session (e.g. a time-bounded IP/http query with its response) depends on the specificity of the resources included in `ApplicationModule` instances. For example, the resulting granularity for a "*national IP backbone infrastructure*" application instance will correspond to the routing domain.

### 3.3.2 Logs and Alarms

As per the International Telecommunication Union (ITU), "*the log is a repository for records*" (ITU-T Rec. X.735) [166] and an event log record "*represents the information stored in the log as a result of receiving notifications or incoming event reports*" (ITU-T Rec. X.721) [164]. Based on this definition, we define the `EventRecord` class for storing any event coming

[7] https://w3id.org/seas/CommunicationOntology
[8] https://github.com/tmforum-apis

from managed objects (e.g. `Resource`, `Application`) such as system logs [123], SNMP Traps [112] and application specific messages (e.g. user applications, operational support systems, processing platforms). Fundamental properties such as `loggingTime`, `logText`, `logOriginatingManagedObject` and `logOriginatingManagementSystem` allow for keeping track of the event origin and content. Details about the message meaning are managed with the `dcterms:type` property that refers to a controlled-vocabulary for event type tagging[9] (e.g. state change, processing error alarm, integrity violation). The `alarmSeverity` property provides an indication of how it is perceived that the capability of the managed object has been affected, or how serious are the service affecting conditions (including for security alarms). Additional properties related to alarm management and interpretation are available with alignment to the DCTERMS and PEP models, such as: `alarmMitigatedBy` and `dcterms:relation` for aggregating events and building event signatures; `dcterms:conformsTo` for root cause analysis and repair planning; `dcterms:mediator` for responsibility follow up.

### 3.3.3 Trouble Tickets and Change Requests

We define the `TroubleTicket` concept accordingly to the TM Forum DataModel where a trouble ticket is "*a record of an issue that is created, tracked, and managed by a trouble ticket management system*"[10]. It is not an event per se, but a mean to efficiently manage targeted resource/service (e.g. `troubleTicketRelatedResource` property) restoration operations through collaboration. Hence, we also consider trouble tickets as a product of the ITIL's Incident Management process [334], and relate them to ITIL's Problem Management process [335] and the BPMN by alignment to the `bbo:DataResource` class.

Corrective maintenance action details are logged as `TroubleTicketNote` and related to the parent `TroubleTicket` with the `dcterms:isPartOf` property. Actions' accountability is implemented with the `dcterms:creator` in relation to the `foaf:Agent` class (Section 3.3.4). Correlating actions to the digital traces (i.e. `EventRecord`) they produce at the structural and functional level (e.g. login, configuration change, upgrade) is available with the `dcterms:relation` property towards a `pep:ProcedureExecutionContainer` entity.

We provide additional properties for improving the incident diagnosis stage efficiency (e.g. `dcterms:hasPart` for hierarchical grouping of tickets), and moving towards root cause analysis based on the notion of Known Error Database (KEDB) (e.g. `troubleTicket-Category` and `problemCategory` for a priori and a posteriori categorization, respectively) and primary/secondary anomaly (cause/effect) with alignment to the FOLIO model. With greater details, a trouble ticket is a document transitively referencing a set of corrective

[9] Event type tagging can be carried-out at the data integration stage, or through a posteriori language processing of the "logText" property.

[10] http://datamodel.tmforum.org/en/master/Common/TroubleTicket/

maintenance actions that can be abstracted into an issue remediation `OperationPlan` for solving the `AnomalyPattern` at hand. Reaching such abstraction from actions' digital traces is enabled by considering the PEP model with `TroubleTicket` as a specialization of a `pep:ProcedureExecutionContainer`, actions as `pep:ProcedureExecution` and `OperationPlan` as `pep:Procedure`.

Similarly to trouble tickets, we define the `ChangeRequest` concept according to the TM Forum DataModel[11] for tracking scheduled change operations (as sets of `pep:ProcedureExecution` carried-out in correspondence to a given `OperationPlan`) with structural or functional impact, and computing (potential) causality for trouble tickets based on the set of correlated resources/applications and operations start/end time.

### 3.3.4 Agents, Teams and Organizations

From the incident management point of view, finding experts in short time is key for operational efficiency. One common approach is to form teams based on technical expertise (e.g. routing and international backbone, servers and virtual machines, forensics and malware retro-engineering) and assign them to manage specific devices or services. External support and engineering services are also relied upon for specialized cases. To facilitate interoperability with complementary knowledge bases, we utilize the FOAF and ORG data models for representing entities such as agents and users (`foaf:Person`), organizational units (`org:OrganizationalUnit`), and organizations (`org:Organization`). Relationships with IT entities, such as `Resource` and `Application`, are modeled using properties like `elementManagedBy` and `applicationModuleRelatedParty`. We introduce the `CorporateUserIdentifier` class as a specialization of `foaf:OnlineAccount` and provide a controlled vocabulary for detailed role descriptions of agents, teams (e.g. Technical Support Group), and organizations (e.g. Manufacturer). This notably enables applying the cyber security out-of-policy principle (i.e. what is not defined is not allowed) for tracking non-legitimate operations (unless facing an insider) by asserting access control groups as `org:OrganizationalUnit` and scrutinizing observed or declared user actions (e.g. `eventLogOriginatingAgent`, `dcterms:creator`).[12]

## 3.4 Evaluation

We have evaluated the NORIA-O implementation according to the ability of the model to answer the CQs that were collected in Section 3.2.1. The CQs have emerged from an iterative and collaborative process of capturing knowledge from domain experts. Therefore, we consider

[11] http://datamodel.tmforum.org/en/master/Common/ChangeRequest/

[12] We assume that companies' human-resource databases are reliable and accurate sources of truth.

that translating these CQs into Authoring Tests (ATs) [385, 173] and obtaining a satisfactory answer to these SPARQL [368] queries from the knowledge graph constitute a sound evaluation of NORIA-O. This evaluation aims to check that all the concepts and relations that are important for the experts' needs are included in NORIA-O. The set of authoring tests, available at https://w3id.org/noria/evaluation, has been defined and tested on two knowledge graph instances structured by NORIA-O. The first one describes a fictitious case of supervision and is publicly available (Section 3.5). The second one has been generated from Orange internal data (10 data sources encompassing 128 features over 15 tables) using an in-house data pipeline (Chapter 4); the size of the resulting RDF dataset is approximately 4 million triples for 400K entities, including streamed events spanning over 111 days.[13]

After this evaluation, we distinguish three situations depicted in column "AT Validation" of Table 3.1. First, a large number of CQs (16/26) can be answered using a single or several simple SPARQL queries and the ontology ("OK" in Table 3.1). 9/26 CQs ("AI" in Table 3.1) are partially satisfied using SPARQL queries, to which complementary AI techniques should be added to fully answer the CQs. For example, to answer CQ#11 "*What was the root cause of the incident?*", the representation of alarms and logs associated with a given incident needs to be enhanced with root cause analysis algorithms (e.g. using semantic reasoners with failure mode descriptions as discussed in [57] and Chapter 5, or similar incident context search as discussed in Chapter 7). Another example is CQ#25 "*What are the vulnerabilities and the associated risk levels of this infrastructure?*" that can be answered only by looking for non-desirable network topology shapes or relations to cybersecurity knowledge derived from network structure and security scanners (e.g. using the SHACL [142] toolset, or graph-based risk assessment [380] with UCO-labelled data [388]). Third, 1/26 CQs requires the introduction of new concepts or relations via an extension of NORIA-O ("Extension" in Table 3.1). The CQ #23 "*What is the financial cost of this incident if it occurs?*" involves information about the cost of an incident (e.g. leveraging the SEAS Failable System ontology [241] and calculating the number of users affected by a service impairment).

## 3.5 Use Case: Modeling a Complex IT Infrastructure

We illustrate the usage and expressiveness of NORIA-O through a fictitious case of network infrastructure supervision. The Figure 3.3 summarizes this use case by showing both the network topology and the corresponding entities. The dataset for this scenario is available at https://w3id.org/noria/dataset. 660 triples are needed for representing the full scenario with additional resources, organization and root cause analysis details.

Based on this scenario, we observe that NORIA-O enhances anomaly detection and analysis

[13] Due to confidentiality, this large dataset is not made public but the fictitious one has been created with the purpose of being a shareable resource.

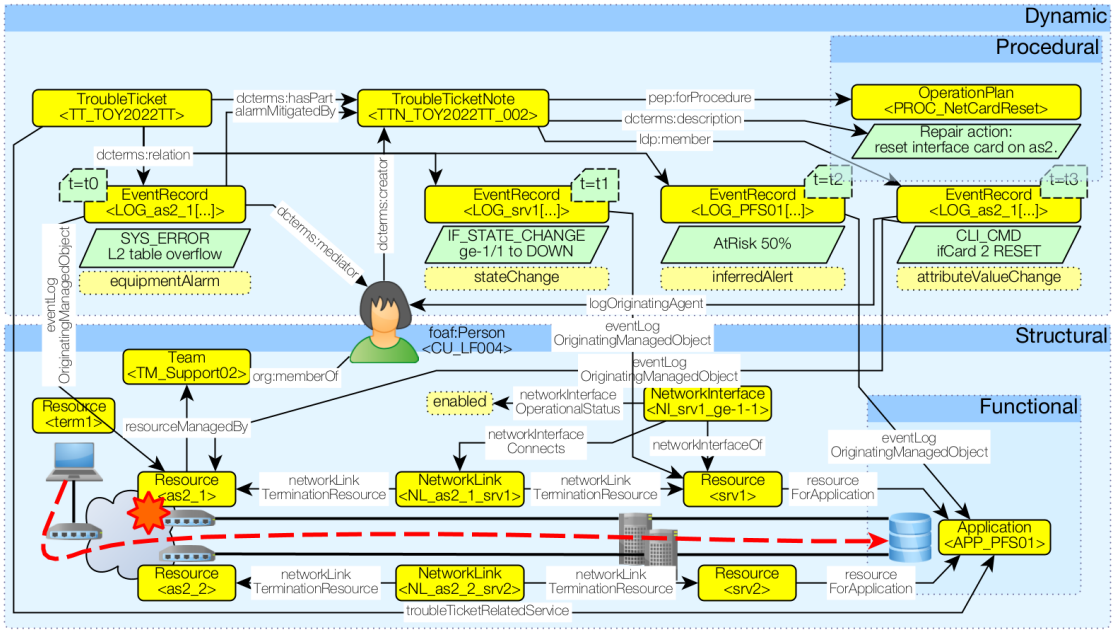



Figure 3.3: NORIA-O instantiation example.
A fault on an access switch (red star) impacts redundancy for critical application resources. The technician identifies the issue through an inferred alert ($t = t2$), traces it back to the faulty equipment, and takes corrective action ($t = t3$). The process is documented in a trouble ticket, including the cause and repair action. `noria` is the default namespace for classes and properties. "SKOS scheme" block values (e.g. "*equipmentAlarm*", "*inferredAlert*", etc.) are coming from the NORIA-O controlled vocabulary.

tasks with the following capabilities.

**Data integration:** it consolidates data from various sources and provides a standardized interpretation using the `rdf:type` and `skos:Concept` constructs. For example, this allows entities in the structural and functional facets to combine data from network discovery tools, VM hypervisors, and the company directory. Events can also be looked up by their type, regardless of their originating signaling system.

**Data querying:** it enables facet-wise checking of the network state and configuration in a rule-based approach using data retrieval. For example, a SPARQL query can derive the scenario's inferred alert (a resilience problem) using a k out-of n graph pattern on `Resource` and `Application` entities (Section 5.4.1). This would not have been possible with a single data source or without standardized interpretation of the data using controlled vocabularies.

**Situation understanding & classification:** it enables gaining facet-wise insights about a situation and its ecosystem using graph traversal. For instance, a SPARQL query can calculate the network neighborhood for a specific incident or identify the teams involved in its resolution. Categorizing event patterns and root cause analysis is addressed using reasoning through complementary ontologies [57, 255] or rule engines [292, 228]. Additionally, the context of an incident, represented by the graph structure related to

a `TroubleTicket` entity, can be captured and categorized using graph embeddings (Chapter 7).

## 3.6 Conclusion and Future Work

In this chapter, we presented NORIA-O, an ontology for representing network infrastructures, incidents and maintenance operations, that relies on and extends well-known semantic models such as BBO, BOT, FOAF, FOLIO, SEAS and UCO. NORIA-O is available at https://w3id.org/noria under a BSD-4 License, along with its documentation and a sample dataset. We conducted an evaluation of NORIA-O using the Competency Questions & Authoring Tests methodology [385], demonstrating its suitability according to the expert needs. We also illustrated the usage and expressiveness of the data model using a fictitious case of network infrastructure supervision. Additionally, we discussed the importance of this model by showcasing the complementary use of graph traversal techniques, reasoning, and incident context capture through graph embeddings. These techniques are discussed in detail in Part II of this manuscript.

Future work will focus on experimenting with NORIA-O for cross-domain alarm correlation and aggregation. First, event logs from heterogeneous data sources depicting an identical phenomenon need to be parsed and categorized in the same way. This can be achieved by incorporating specific technological domains, such as OTN or 5G mobile network specifications, into the NORIA-O controlled vocabulary. Techniques such as log parsing [175, 325] and semantization [26] can be applied, either before or after the data integration stage. NLP-related techniques, including named entity recognition [37], topic modeling [156], and vocabulary reconciliation [276], are crucial in this process. Second, relating events to anomaly models (Chapter 7) or attack scenarios [42], requires to filter-out event logs and alarms on both trouble tickets' timespan and impacted resources characteristics. Recent research on dynamic graphs with event streams has shown promising results in estimating the useful spreading of observables [68, 379, 351, 92].

We also note that network resilience and cybersecurity application domains will benefit from extensions of a NORIA-O knowledge graph with third-party data collection tools. For example, network topology anti-patterns and semantic interpretation of the ICT resources configuration [371] could be related to the network performance and issues. Similarly, integrating data from vulnerability scanners and cyber threat intelligence tools could enable cybersecurity risk evaluation and minimization (e.g. combining CVSS [285] data from OpenCTI[14] with optimized countermeasure placement techniques [380]).

Finally, to maximize the opportunities for improving NORIA-O through its use in various

[14] https://www.opencti.io/

application contexts, as well as to facilitate the sharing of operational knowledge among network operators by using NORIA-O as a common model, we have referenced NORIA-O in the LOV[15] [291] and ENIT Industry Portal[16] [20]. Improving the implementation of NORIA-O or adjusting the model to new application contexts can be achieved by monitoring the indicators and metrics provided by these portals, such as the names of projects integrating NORIA-O, third-party ontologies reusing NORIA-O, and the NORIA-O's FAIR score [99].

[15] https://lov.linkeddata.es/dataset/lov/vocabs/noria
[16] https://industryportal.enit.fr/ontologies/NORIA-O

# Chapter 4

# A Knowledge Graph-based Platform for Anomaly Detection and Incident Management in ICT Systems

## 4.1 Introduction

Incident management for broad scale ICT systems implies scrutinizing massive amounts of heterogeneous data for proper definition of remediation strategies. In the best case, crisp reasoning over situations at hand brings fast root cause analysis and high level of confidence for selecting the corrective maintenance actions to carry out. Anomaly detection within decision support systems, such as NMSs and SIEM tools, typically rely on expert knowledge translated into logical rules for catching specific situations based on the systems activity traces. However, uncertainty arises whenever the ICT system's activity shows unexpected values or behaviors poorly fitting known activity models. A typical solution would be to fine-tune the decision support system stack, for example by extending the detection rule set with the new values, or retraining the anomaly detection model. This unfortunately brings computational complexity and overfitting to the diagnosis stage.

A better solution is to keep rules and models consistent by working on semantically equal data. The notions of "ontology" and "data model" solve the challenge of reasoning upon composite alerting signals at the semantic level (Chapter 3). Indeed RDF Knowledge Graphs (KGs) bring an abstraction level for standard interpretation and logical reasoning over heterogeneous data.

Leveraging on Semantic Web tools could bring decision support systems for ICT systems to a next level of diagnosis and recommendation capabilities. However, few feedback exist about the design of such data processing platform from an end-to-end point of view. This chapter reports on the design experiments for a knowledge graph-based data platform made at Orange. It comprises the following key features:

1. Building a RDF knowledge graph from static (IT resources lists, organization) & streamed (trouble tickets, logs) data;

2. Providing data & inferences provenance and confidence indicators;
3. Enabling inline & posterior entity patching and reconciliation;
4. Enabling multi-level & synergical reasoning.

Our main contributions with this work are: setting design methodology and principles for an end-to-end KG-based data platform, providing ETL architecture details and code for handling descriptive datasets and events streams, and sharing lessons learned while building the platform about data mapping strategies and configuration deployment.

The rest of the chapter is organized as follows. Section 4.2 explores design challenges and requirements through a tool chain model proposal. Section 4.3 details our knowledge graph-based data platform architecture. References to contributed open source code are provided there. Section 4.4 evaluates the platform features and discusses lessons learned. Finally, Section 4.5 concludes this chapter by providing a summary of the work that has been done and offering some preliminary conclusions regarding the opportunities and potential future developments associated with it.

## 4.2 Design Methodology and Challenges

This section outlines the methodology followed in constructing a knowledge graph-based data platform with stream processing and reasoning capabilities for incident management over ICT systems. It proposes a conceptual tool chain and examines design challenges and requirements to frame the high-level design and implementation work discussed in Section 4.3.

**Conceptual tool chain & design principles.** Looking at data integration theory and generic data transformation processes (e.g. "Extract, Transform, Load", CRISP-DM [323]), we remark that none directly take into account the abstraction and reasoning capabilities brought by the Semantic Web technologies and Knowledge Graphs. Furthermore, these design patterns set apart decision making concerns where informed-decisions potentially involve gradual understanding of data (i.e. raw data → information → knowledge) combined with synergical reasoning [46].

Extending on these, we propose a tool chain model (Figure 4.1) to guide design thinking steps: *unstructured data* (e.g. event logs) enters the tool chain and becomes *structured data* by application of a defined/learnt *structure model*. Semantic mapping is applied for making *annotated data*. These can benefit of additional knowledge from some *enrichment service* (e.g. mapping assets to organization or vulnerability knowledge). *Reasoning service* (e.g. rule-based inference, confidence propagation, link/entity prediction) works from *annotated data* for producing further knowledge (i.e. *interpreted data*). Downstream agents (e.g. operational

teams, information system) get informed (e.g. situation awareness) by querying *interpreted data*. This conceptual tool chain is open to complementary processes, such as a direct feed of structured data or recursive loops of the *inference* step.

Based on the above, we posit the following design principles (further discussed below) to streamline integration and improve user adoption: *1) Minimize transformation needs at ingress*: data encoding (serialization & structuration) must be backward compatible early in the processing chain to limit the number of technologies used; *2) Independent downstream usage*: parallel downstream applications may focus on different data facets, so the serialization/structure should allow easy separation of data from meta-data without imposing specific remote procedure calls; *3) Implementation independent*: abstractly describing transformation and processing rules enables system behavior description and transposition independent of implementation; *4) Integrate, customize or build*: prioritize integrating existing frameworks that meet requirements, extend partially meeting frameworks, or develop specific solutions if neither of the previous options apply.

Note that knowledge engineering methodologies (e.g. Competency Questions [385] and Linked-Open Terms [224]) are separate from our proposal. Data models resulting from these methodologies are used in the *annotation* step, but our tool chain is not affected by changes in data models from a functional and technical perspective.

**Dataset characteristics organize the processing architecture.** Scrutinizing Orange internal datasets and third party datasets based on their TAM Domain/Sub domain[1], we took note of the data structures and technical characteristics (e.g. number and type of features, serialization syntax, schema definition, access protocol, update period) for devising a data integration strategy. The *update period* and *data access method* emerged as key design factors: descriptive datasets (e.g. assets database, network topology, organization) have a low refreshment pace (one day to one week period) and are generally available through file-based platforms (e.g. database API, file dumps), while network operations and events (e.g. interface status change, applications logs, alarms, trouble tickets) are stream feeds with fast-paced time-stamped data (real time to quarter-hour period).

**Data wrangling with syntax heterogeneity.** When transforming data, we need to simultaneously consider syntax heterogeneity and batch/stream processing. This can be represented using the $ET[P1]L[P2]L$ model embedded in Figure 4.1, where $P1$ components are for per feed processing and $P2$ components apply at the dataset level. The $P1/L$ interface should comply with standard data transport solutions (e.g. JSON for Kafka messages) and knowledge

[1] i.e. their parent application/research domain within the TM Forum's GB929 Application Framework (TAM), see https://www.tmforum.org/application-framework/

graph data ingestion methods (e.g. SPARQL Update, periodic/on-demand bulk load), while the $L/P2/L$ interfaces require data representation transformation to match $P2$ requirements (e.g. Turtle to JSON-Graph) and integrate results (e.g. time-stamped confidence as a RDF triple) into the knowledge graph-based application data model.

**Data wrangling with annotation, patching & reconciliation tasks.** We assume data must have meaning (e.g. data is about a hostname or a date, and not just a string of characters) and structure to be useful (in our case, a relational graph structure). Therefore we introduce the concept of data patching & reconciliation to do in-place update of the graph data and link entities from different sources. This includes substituting equivalent literals with controlled vocabulary and normalizing terms and relationships.

**Post-processing constraints on the ETL stages.** First, it is important to track the data origin for trustworthiness. From a practical standpoint, this allows for:

1. Isolating/correcting contaminated (intentionally or not) data sources;
2. Accessing the original data to restore its original meaning and context;
3. Exposing data characteristics (e.g. freshness, validity period) for refined decision making.

Second, it is necessary to make post-processing efficient by considering both the composition of post-processing (e.g. sequential, parallel) and the form of the data for lossless transformation, such as from graph to table. The third goal is to determine how post-processing results are utilized. This involves considering the compatibility of processing blocks and whether the results can be interpreted beyond their original context. It also involves reintroducing the results into the base data space to serve iterative or synergistic reasoning. The nature of the result must also be considered in terms of form and value, as it can add information to an existing object (e.g. assigning a cyber security risk level to a network asset) or create a new object (e.g. an alert). Provenance and trust are necessary here, but with different semantics since they affect the product of data interpretation, not the original data.

## 4.3 KG-based Platform and Data Processing Architecture

Thinking through the design methodology of Section 4.2, we developed two data integration pipelines and a mechanism for data interpretation (Figure 4.1) based on well-known open

source frameworks (e.g. Apache Kafka[2], Apache Airflow[3], OpenLink Virtuoso[4]), academic projects (e.g. RMLMapper [24], StreamingMASSIF [292], string2vocabulary [276], grlc [9], RDFUnit[5]) and ad hoc code (Table 4.1). The overall system is akin to a Lambda data processing architecture [95].

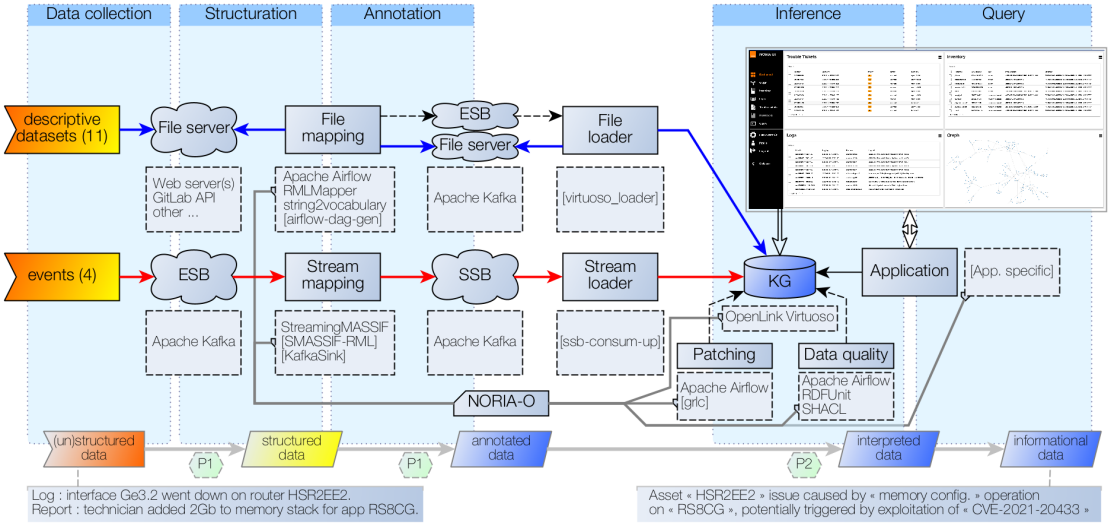



Figure 4.1: Conceptual tool chain & data platform overview.
Acronyms: ESB = Enterprise Service Bus, SSB = Semantic Service Bus, KG = Knowledge Graph. Plain arrows are for data flows, dotted arrows for control and query flows. Arrows start from the component initiating the flow/transaction. Numbers for the "descriptive datasets" and "events" blocks refer to the number of sources in Table 4.2. Component names within brackets relate to ad hoc code from Table 4.1. $P1$ and $P2$ stands for Processing components groups in an $ET[P1]L[P2][L]$ reading of the tool chain (i.e. matching the tool chain with the ETL process model).

**Knowledge graph management.** We manage the knowledge graph using a Virtuoso quad store, with enabled SPARQL endpoint and Faceted Browser services. Named graphs enable fast data access and help track the source of triples in RDF datasets. We use predefined URI [50] patterns that closely match the NORIA-O data model (see below), following the *Graph per Source* and/or *Graph per Aspect* data management patterns [200]. This prior knowledge simplifies the creation of linked data using a declarative transformation approach before inserting mapped data into the graph (see below for implementation details).

**Batch processing for descriptive datasets [airflow-dag-gen, virtuoso_loader].** A set of Apache Airflow Directed Acyclic Graph (DAG)[6] periodically triggers data downloading, mapping and inserting tasks. DAGs are defined on a per $\langle Source, View \rangle$ basis and

[2]https://kafka.apache.org/
[3]https://airflow.apache.org/
[4]https://virtuoso.openlinksw.com/
[5]https://github.com/AKSW/RDFUnit
[6]https://airflow.apache.org/docs/apache-airflow/stable/core-concepts/dags.html

Table 4.1: Complementary developments for the KG-based platform

| Component name | Role |
|---|---|
| *airflow-dag-gen* | Parametric Airflow DAG generator for data integration and patching. |
| *KafkaSink* | StreamingMASSIF [122] component for JSON-LD output to Kafka, available at https://github.com/Orange-OpenSource/SMASSIF-RML. |
| *grlc* | Enhanced grlc [9] with GitLab connector and SPARQL Update, available at https://github.com/Orange-OpenSource/grlc. |
| *NORIA-O* | RDF data model for IT networks, events and operations information, available at https://w3id.org/noria/. |
| *SMASSIF-RML* | Modified RMLMapper [24] for StreamingMASSIF [292] component, available at https://github.com/Orange-OpenSource/SMASSIF-RML. |
| *ssb-consum-up* | Kafka to SPARQL gateway, available at https://github.com/Orange-OpenSource/ssb-consum-up. |
| *virtuoso_loader* | Event-triggered (Kafka) bulk load of remote RDF datasets into Virtuoso. |

are configured using a limited set of parameters: schedule interval, a reference to a `noria:ETL_process_node` entity, and templated ETL tasks to schedule.

`noria:ETL_process_node` entities are configuration nodes stored in the platform's knowledge graph. They include a reference to the data source to download via a `dcat:downloadUrl` property, and a relationship to the RML mapping rules (also stored in the platform's knowledge graph). This allows for centralizing information (configurations and mapped data) with a homogeneous representation, resulting in simplified interrogation and audit of data provenance. Because RML is RDF data, making these rules available from the knowledge graph is as simple as uploading an RML file into the graph store once the mapping implementation done.

Prior starting a mapping thread (i.e. a local `rmlmapper-java` instance), *1)* a `fetchRules` task queries the knowledge graph for the mapping rules and stores them in a temporary file, and *2)* a `fetchData` task downloads the raw data to map. Then mapping is started with complementary output configuration parameters asking for RDF Trig[7] serialization (enables targeting specific named graphs in the downstream graph store with `rr:graph` attributes in the mapping implementation) and provenance metadata generation at the dataset level [25] (relates the mapping activity to the `rml:source` used in the mapping implementation with a `prov:used` attribute). Because `rmlmapper-java` does not include target graph data in the provenance metadata file, we rewrite the file with an `adjustProvenance` step. Once the mapping thread is over, a `loadRequest` signal triggers `fetchMappedData` and `loaderRun` threads on a downstream `virtuoso_loader` component listening to a specialized Kafka

[7] https://www.w3.org/TR/trig/

topic. This ends the batch processing for descriptive datasets by inserting mapped data and provenance metadata into the knowledge graph.

**Speed processing for events [SMASSIF-RML, KafkaSink, ssb-consum-up].** A set of specialized StreamingMASSIF pipelines continuously consume, map and forward data for insertion into the knowledge graph by a downstream `ssb-consum-up` component (a Kafka to SPARQL gateway). Equation (4.1) shows the typical form of the pipelines:

$$KafkaSource \rightarrow RMLMapper \rightarrow (OptionalProcessing) \rightarrow KafkaSink \tag{4.1}$$

where $KafkaSource$ is a StreamingMASSIF native component, $RMLMapper$ is a modified version of the `rmlmapper-java` tool (to handle streamed data as a StreamingMASSIF component), $OptionalProcessing$ can consist of any combination of StreamingMASSIF's reasoning components[8], and $KafkaSink$ is a contributed component that sends the data stream in JSON-LD syntax to a Semantic Service Bus (SSB). Pipelines are defined on a per $\langle Source, View \rangle$ basis and are configured using a limited set of parameters: input topic, reference to an RML rules implementation, and output topic.

We posit that RDF data downstream of the mapping pipelines can serve multiple purposes as distinct streams (e.g. direct update of the knowledge graph, intermediary vocabulary reconciliation, multi-source event logs co-occurrence alerting, notification-triggered dependency calculus). To manage the distribution of streams, we developed the concept of SSB using the Kafka event streaming technology and considering the following features: *1)* forwarding RDF data messages in standard RDF serialization; *2)* providing provenance metadata; *3)* providing named-graph compatibility; *4)* enabling the use of SPARQL Update actions[9].

The Kafka platform uses $\langle key, value \rangle$ pair-based messaging with native JSON compatibility, hence JSON-LD is a natural serialization choice for forwarding RDF data through the SSB. We identified two strategies for mapping JSON-LD with the $\langle key, value \rangle$ message model while handling potentially RDF dataset-shaped streams:

1. Build Kafka messages with `key = NULL`, and `value = <JSON-LD payload>` (with or without named graph).
2. Assuming named graph at the subject level, build Kafka messages with `key = <JSON-LD metadata>` (e.g. `{[provenanceMetadata], [updateAction], [otherMetadata]}`) and `value = <JSON-LD payload>` (with named graph).

[8] As for StreamingMASSIF v0.0.1: Windowing, Filtering, Abstraction, Complex Event Processing (CEP).
[9] https://www.w3.org/TR/sparql11-update/

The strategy #1 is the most straightforward and versatile approach, as it delegates all decision making regarding RDF triples to the downstream consumer(s), including which graph update operations to apply (such as `INSERT` or `DELETE`). Further, it directly complies with the fact that the Kafka's $key$ is for partitioning and compaction[10], which means that $key$ is akin to a primary key and may not be used for metadata unless made compatible with the partitioning and compaction principles (note that $key$ can be left empty). The strategy #2 enables providing guidance to downstream consumers on how to handle RDF triples through metadata. We remark that `updateAction` can leverage on well-known vocabularies such as `schema:UpdateAction`[11]. Similarly, `provenanceMetadata` can include a `prov:wasGeneratedBy` attribute to keep track of the successive modifications made within a tool chain. A downside of the second strategy is that the scope of the `updateAction` should match all `<JSON-LD payload>` triples, both in terms of the operations to apply and target named graph. Because the strategy #2 can be later built upon the #1, we chose to implement the #1 as a starter, that is: straightforward payload mapping at the `KafkaSink` level and graph update operations defined at the `ssb-consum-up` level.

Since we are using named graphs at the knowledge graph level and that the Virtuoso graph store requires defining the target graph parameter for data insertion using SPARQL update queries, we explicit the design of the `ssb-consum-up` component with Equations (4.2) & (4.3):

$$(s, p, o, g) \underset{Op}{\longrightarrow} Op\{\text{GRAPH } g\ \{s\ p\ o\}\} \tag{4.2}$$

$$(s, p, o) \underset{Op,g}{\longrightarrow} Op\{\text{GRAPH } g\ \{s\ p\ o\}\} \tag{4.3}$$

where Eq. (4.2) states that incoming messages from the SSB are RDF triples along with a target graph information, thus data insertion into the downstream SPARQL endpoint is akin to translating the messages into SPARQL update queries subject to a user-defined action $Op$ (e.g. `INSERT`); and Eq. (4.3) reflects the same with an additional user-defined target graph parameter $g$ whenever incoming messages are raw RDF triples. Assuming many StreamingMASSIF pipelines pushing mapped data with target graph information into a same SSB topic, then a single `ssb-consum-up` component instance is sufficient for continuous data insertion (e.g. with $Op = INSERT$, and $g = \langle DefaultGraphURI \rangle$ for filling any gaps).

**Data model [NORIA-O].** The NORIA-O data model, presented in Chapter 3, is used as the main data model for the data integration and exploitation work described in this chapter as it can model complex ICT system situations and serve as a basis for anomaly detection and root cause analysis. The NORIA-O data model also provides a set of controlled vocabularies

[10] https://kafka.apache.org/documentation/#compaction
[11] https://schema.org/UpdateAction

useful for standard interpretation of the knowledge graph entities; for example, with reconciliation (see below) on the network device alarms through the <Notification/EventTypeGroup/SecurityAlarm> concept scheme.

**Patching & reconciliation [airflow-dag-gen, grlc].** The per source and per concept mapping approaches discussed above entails handling data ingest interdependencies with complementary patching & reconciliation tasks. We make use of an Airflow DAGs-based periodic run of ordered patching queries in SPARQL syntax via a enhanced grlc [9] tool instance. For this approach to work, we assume that the NORIA-O data model is available in the data store, including the controlled vocabularies (a.k.a. NORIA-O KOS).

We observe that patching requests follow a limited number of forms that can be expressed as (arche)types of patching queries, thus leading to a standard approach to patching (Eq. 4.4, where $P$ stands for Patching Queries and $O$ for Ontology):

$$P_{templates} \times P_{definitions} \xrightarrow{\text{query generator}} P \quad ; \quad O \times P \xrightarrow{\text{patching}} O' \tag{4.4}$$

We define the three following archetypes, hence making the mapping definition process faster and easier to maintain via patching requirements set in a definition file (e.g. YAML syntax) and query generation (e.g. Python script + templated SPARQL queries in JINJA2 syntax[12]):

1. $literal2KOS := \langle Literal \rangle \rightarrow \langle skos : Concept(Subject) \rangle$. We implement it with SPARQL queries as an exact string match via a `LCASE(STR(x)) = LCASE(STR(y))` statement in order to avoid declaring redundant `skos:altLabel` in the NORIA-O vocabulary files. For example, from Figure 4.1: "interface went down" → EventRecord.type (<kos/Notification/EventType/StateChange>).
2. $literal2URI := \langle Literal \rangle \rightarrow \langle Subject \rangle$) (but not a KOS URI). Likewise $literal2KOS$, we implement it as an exact string match. For example, from Figure 4.1: "router HSR2EE2" → Resource.resourceHostName('HSR2EE2').
3. $addShortcut := \{\langle Predicate, Object \rangle\} \rightarrow \langle Subject \rangle$, i.e. a direct property between a subject and an object when these two nodes are related by a given longer path. For example, from Figure 4.1: "issue potentially triggered by" → EventRecord.conformsTo( Vulnerability('CVE-2021-20433')).

**Complementary iterative processing.** Complementary Airflow DAGs trigger data model and data quality audits (e.g. querying the knowledge graph against the NORIA-O competency

[12] https://jinja.palletsprojects.com/

questions, checking data ingest conformance with the RDFUnit tool), performance evaluations (e.g. queries velocity *vs* NORIA-O expressivity), and application-specific code (e.g. querying the IT network topology from the knowledge graph and then running a graph-based risk assessment method).

## 4.4 Lessons Learned

Our design currently runs on Orange internal data (10 data sources encompassing 128 used features over 15 tables, see Table 4.2). Batch processing generates, updates and patches the knowledge graph on a hourly basis. Speed processing works on generated data until further integration within the data ecosystem. The size of the resulting RDF dataset at hand is approximately four million triples for 400K entities, including streamed events spanning over 111 days[13].

Table 4.2: Data sources and mapping overview, with generic names of Orange internal data sources, along with features and mapping statistics. Acronyms: AAA = Authentication, Authorization and Accounting; VM = Virtual Machine. Unused features are due to data redundancy, or data without correspondence with the NORIA-O data model.

| Nature | Data source (table) | Features (total) | Features (used) | Features (used ratio) | rr:TriplesMap (count) |
|---|---|---|---|---|---|
| Events | Trouble Tickets | 28 | 21 | 75,00 % | 6 |
| | Change Tickets | 124 | 11 | 8,87 % | 2 |
| | Alarm monitoring | 152 | 8 | 5,26 % | 1 |
| | Logs monitoring | 164 | 3 | 1,83 % | 1 |
| Descriptive | AAA groups | 6 | 4 | 66,67 % | 2 |
| | Applications | 25 | 15 | 60,00 % | 2 |
| | Teams | 14 | 8 | 57,14 % | 3 |
| | Users | 12 | 6 | 50,00 % | 2 |
| | Logistic database | 51 | 19 | 37,25 % | 8 |
| | Backbone logical links | 14 | 5 | 35,71 % | 2 |
| | Backbone physical links | 14 | 4 | 28,57 % | 3 |
| | Applications types | 63 | 9 | 14,29 % | 1 |
| | Network topology | 16 | 2 | 12,50 % | 1 |
| | VM management | 74 | 9 | 12,16 % | 3 |
| | VM clusters | 57 | 4 | 7,02 % | 2 |

By applying the Authoring Test approach [385] with the NORIA-O competency questions[14], the resulting RDF dataset proves to be of great interest from a business perspective as it enables the handling of incident management needs across heterogeneous data through the use of

[13] Due to confidentiality, this dataset is not made public.
[14] https://w3id.org/noria/cqs/

data retrieval techniques and complementary AI-based algorithms. Indeed, out of the 26 competency questions defined with expert panel interviews, a significant number of them (16/26) can be answered using simple SPARQL queries and the NORIA-O ontology, while the remaining questions (9/26) may require additional techniques such as relational learning or anomaly detection algorithms.

From a more technical standpoint, the software infrastructure is deployed using an Infrastructure as Code approach. A main project installs and configures the platform using templated scripts based on a host feature inventory. Components can be individually started, stopped, or upgraded thanks to a microarchitecture design. The platform uses nine virtual machines hosted in Orange's private cloud with varying hardware setups (e.g. 1-4 vCPU, 8-16 GB memory, 20-80 GB storage). CI/CD is further used for granular version control and performance evaluation, particularly for the NORIA-O data model, where pre-publishing review and expressivity evaluation are enforced as per the LOT methodology [224]. The data model is automatically loaded into the data store when there is a change. The same approach is used for orchestration DAGs, where data integration tasks can change based on data source changes. The latest DAGs releases are downloaded and scheduled via an update signal sent to the Apache Airflow instance.

From the Apache Airflow DAGs and Virtuoso logs, we measure that the *map data* and *adjust provenance* tasks are from far the longest tasks of the DAGs (Table 4.3). As we implemented simple RML rules (i.e. without `rr:joinCondition`), the mapping time can hardly be lowered as it depends on the input file size and `rmlmapper-java` tool implementation. We remark from complementary experiments that `rr:joinCondition` entail a ×2 to ×5 increase of processing time. However, improving provenance data generation for the *adjust provenance file* step (e.g. at the `rmlmapper-java` or file rewriting level) may bring better overall performance with a ×4 increase in throughput.

For speed processing, we confirmed the effectiveness of our `SMASSIF-RML` → `ssb-consum-ub` → `Virtuoso` tool chain based on local experiments with generated data (related to the "events" category from Table 4.2). However, a thorough load study is yet to be conducted with real data sources. Evaluation is left for future work, which will depend on the evolution schedule of the streaming sources to provide standardized connectors. Besides performance, we observed that although Kafka allows data replay for overcoming subsystems failures, it is a complex system; so materializing mapped data in files (as in our batch processing approach) seems more reliable.

For patching & reconciliation, we make use of 42 SPARQL queries ($literal2KOS = 16$, $literal2URI = 19$, $addShortcut = 7$). From our experience on reconciliation, $literal2KOS$ with exact match is sufficient in a great majority of cases, but will misses advanced text analysis situations, such as for `noria:logText` parsing (e.g. `noria:EventRecord.log-`

Table 4.3: Batch processing performance for three representative (small/medium/big) sources from Table 4.2. Preprocessing may include: convert to csv, delete first line, convert to UTF-8, crop columns.

| | AAA groups | | Users | | Logistic database | | Unit |
|---|---|---|---|---|---|---|---|
| Input data size | 0.16 | | 2.4 | | 45.5 | | [Mb] |
| Download data | 0.44 | 6.63 % | 0.95 | 1.54 % | 3.32 | 0.69 % | [s] |
| Dump rules | 0.14 | 2.11 % | 0.19 | 0.31 % | 0.15 | 0.03 % | [s] |
| Preprocessing | 0.19 | 2.86 % | 9.46 | 15.37 % | 8.66 | 10.83 % | [s] |
| Map data | 3.27 | 49.25 % | 8.54 | 13.87 % | 79.97 | 16.70 % | [s] |
| Adjust provenance | 2.27 | 34.19 % | 40.66 | 66.05 % | 374.26 | 78.16 % | [s] |
| Notify for loading | 0.27 | 4.07 % | 0.29 | 0.47 % | 0.29 | 0.06 % | [s] |
| Data bulk load | 0.05 | 0.75 % | 1.46 | 2.37 % | 12.17 | 2.54 % | [s] |
| Prov. bulk load | 0.01 | 0.15 % | 0.01 | 0.02 % | 0.02 | 0.00 % | [s] |
| Total time | 6.64 | | 61.56 | | 478.84 | | [s] |
| Output data | 0.52 | | 21 | | 222 | | [Mb] |
| | 5 110 | | 244 532 | | 2 415 676 | | [Triples] |
| Throughput | 769.58 | | 3 972.25 | | 5 044.85 | | [Triples/s] |

`Text("LINK-3-UPDOWN: Interface GigabitEthernet0/0/1, changed state to up")` to enrich with `dcterms:type <kos/Notification/EventType/stateChange>`). Hence we developed two complementary approaches:

1. We extended the String2Vocabulary tool [276] with named graph processing capabilities for enabling vocabulary reconciliation with a fuzzy match approach as a DAG task consecutive to data mapping.
2. We experimented with the Slogert framework [26] for complementary `noria:logText` structuring and annotation in a $KG \xrightarrow{Slogert} KG$ fashion through a DAG.

More generally, we remark that combining Airflow with grlc [9] allows quick development of knowledge graph-based applications of the extract-process-report type, and friendly access to data and operations for non-technical/non-expert users. Furthermore, our two step "simple mapping *vs* posterior patching" approach allowed us maximizing direct graph traversal capability with URIs while minimizing duplicates although handling them. This has notably allowed us to keep the knowledge graph's complexity low by avoiding the use of `owl:sameAs` predicates. Finally, we remark that, thanks to a report table such as Table 4.2, tracking the characteristics of the source files and TripleMaps for comparison is simplified, resulting in time savings for exploring and cross-referencing information. Building this table is possible through scripted process akin to $RMLs \times DataSources \rightarrow \{SourceFile, features_{total}, features_{used}\}$. A nice consequence of this programmatic analy-

sis is the natural emergence of mapping management patterns; this notably led to design a DAG generator tool for convenient management and deployment of the ETL and patching DAGs. Table 4.2 is also valuable for complementary analysis. First, the "used ratio" indicator reveals sparse/dense data sources for our application domain, raising concerns about information redundancy and database design practice. Second, it suggests potential for additional concepts/relationships, depending on clever understanding of the necessary and sufficient features for a given domain. Third, it enables direct reading of data flow from sources to concepts and graphs, revealing design principles and characteristics behind them.

## 4.5 Conclusion and Future Work

In the work described in this chapter, we aimed to design and implement a data processing architecture for knowledge graph-based incident management of broad scale ICT systems. We firstly hypothesized that the cross-referencing of semantic representations from multiple sources would enable the evolution of decision support systems for ICT systems to a next level of diagnosis and recommendation capabilities. Next, we developed and deployed a Lambda data processing architecture combining well-known open source frameworks (Apache Kafka, Apache Airflow, OpenLink Virtuoso), academic projects (RMLMapper [24], StreamingMASSIF [292], string2vocabulary [276], grlc [9], RDFUnit) and ad hoc code released in open source (grlc[15], SMASSIF-RML[16], ssb-consum-up[17]). The proposed architecture has been instantiated and tested in an industrial setting, producing an RDF knowledge graph that shows strong potential for addressing cross-domain anomalies from heterogeneous data.

The solution notably minimizes the effort for data quality and trust audit thanks to the generalized use of RML, and the centralized storage of both data and mapping configuration within the knowledge graph. Additionally, we open up the possibility of distributed processing or event-triggered processing through the generalization of RDF data transfer by a message-broker software. However, the data provenance tagging at the dataset level leads to a loss of information granularity after the data patching/reconciliation steps and introduces a heavy file adjustment step. Further, a thorough load study is necessary to consider deploying the stream processing pipeline on massive data (e.g. telemetry data from broadband network routers or a fleet of IoT devices).

Future work on the NORIA platform will consider both improving the Knowledge Graph Construction (KGC) process and using the resulting knowledge graph for efficient incident management. Focusing on KGC, future work will explore how the full description of the ETL processes could be stored within the knowledge graph with RDF pro-

[15] https://github.com/Orange-OpenSource/grlc
[16] https://github.com/Orange-OpenSource/SMASSIF-RML
[17] https://github.com/Orange-OpenSource/ssb-consum-up

cess models [22, 186]. This would allow auditing data platforms through a single language, thanks to a joint representation of data and processing mechanisms. In the same line of thought, automated patching generation can be enabled by browsing RML files for `rr:predicateObjectMap [rr:objectMap [rml:reference "<someRef>"]]`, potentially with an additional `toPatchWith(<someGraphPattern>)` property for better end-to-end process automation and automated URI template checking. For stream processing, we envision comparing our approach with other frameworks [122, 91]. This should help identify decision boundaries, particularly in terms of energy efficiency and network overhead, in order to move towards a Kappa architecture [95]. This should also provide insights on the signaling mechanisms to be implemented for opportunistic processing (e.g. SKOS reconciliation as a service, in-line graph clustering) and synergical reasoning [46] (cooperative decision making). Finally, scrutinizing knowledge graph pruning and summarization techniques will prevent from ever expanding datasets (e.g. ICT systems situation models *vs* an accumulation of logs), although using generic RDF data models for knowledge representation is already a mitigating factor.

Beyond future work on the NORIA platform itself, we also envision conducting a reflection on the generalization of KG-based approaches at the level of enterprise-scale IT urbanism. Firstly, as a KGC pipeline, such as the NORIA platform, provides a unified view of various data silos at the scale of a specific resource management domain (e.g. a regional or national business unit, a technological domain such as OTN or 5G networks), one might indeed consider the deployment of a higher level unified view (Figure 4.3) while considering constraints on data ownership, data access authorization, and data hosting infrastructures. The SPARQL Federated Query extension[18] is a typical option for this research, assuming data stored as RDF or viewed as RDF via middleware (e.g. OnTop VKG [132]) and distributed over different SPARQL endpoints. A second axis of reflection on IT urbanism involves leveraging high I/O performance technologies and frameworks for specific types of data (Appendix B.5) while remaining aligned with the idea of providing a unified and KG-based view of the data. This is the case, for example, with Time Series DataBase (TSDB) and column-oriented DBMS, optimized for time series and sequential data. Figure 4.2 illustrates a development option where the data is distributed both to the KGC pipeline and a TSDB, maintaining a possible relationship with shared references. Through experiments on this architecture, insights can be gained on the storage footprint of time series data in KG-DBMS compared to TSDB, based on KGC policies. Additionally, trade-offs between training time and accuracy of machine learning models for multidimensional time-stamped data in KG-DBMS versus TSDB can be explored.

[18] https://www.w3.org/TR/sparql11-federated-query/

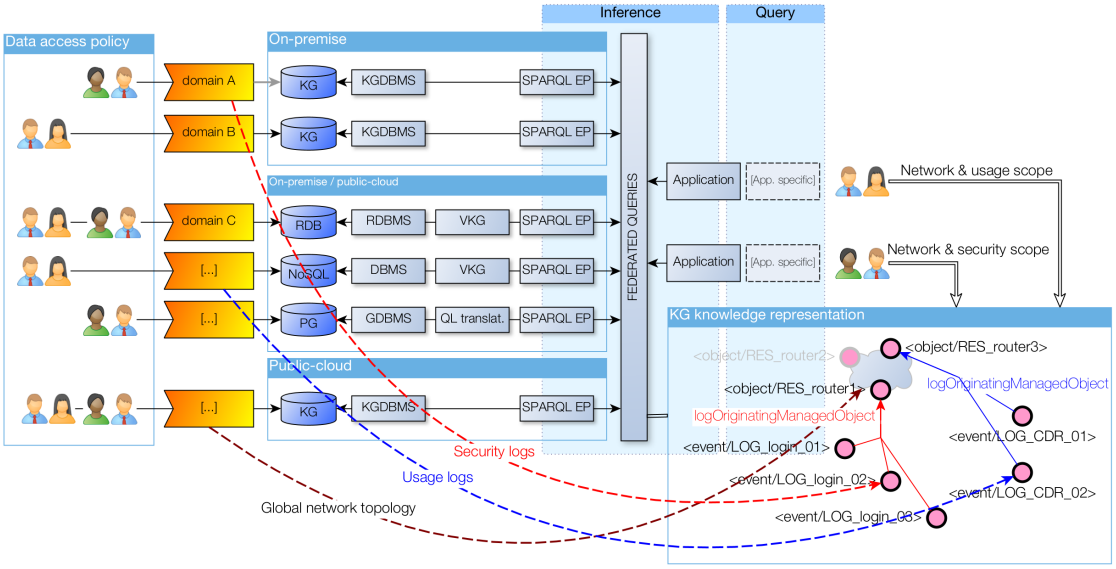



Figure 4.2: Unified access to data distributed across various technological platforms.
This diagram illustrates a technical architecture where data domain owners within an organization, with data hosted on different platforms and locations, provide access to a reconciled view through a mediation layer using the SPARQL protocol and its Federated Query extension. A SPARQL endpoint allows access to data in each silo, potentially through a middleware if the DBMS is not compatible with SPARQL. Common data models across the organization enable the creation of a unified view. Data authorization policies at the silo level determine the subgraphs returned to end users by federated queries. Acronyms: KG = Knowledge Graph; RDB = Relational database; PG = Property Graph; xDBMS = DataBase Management System; VKG = Virtual Knowledge Graph; QL translat. = Query Language translator; SPARQL EP = SPARQL endpoint; CDR = Call Detail Record.

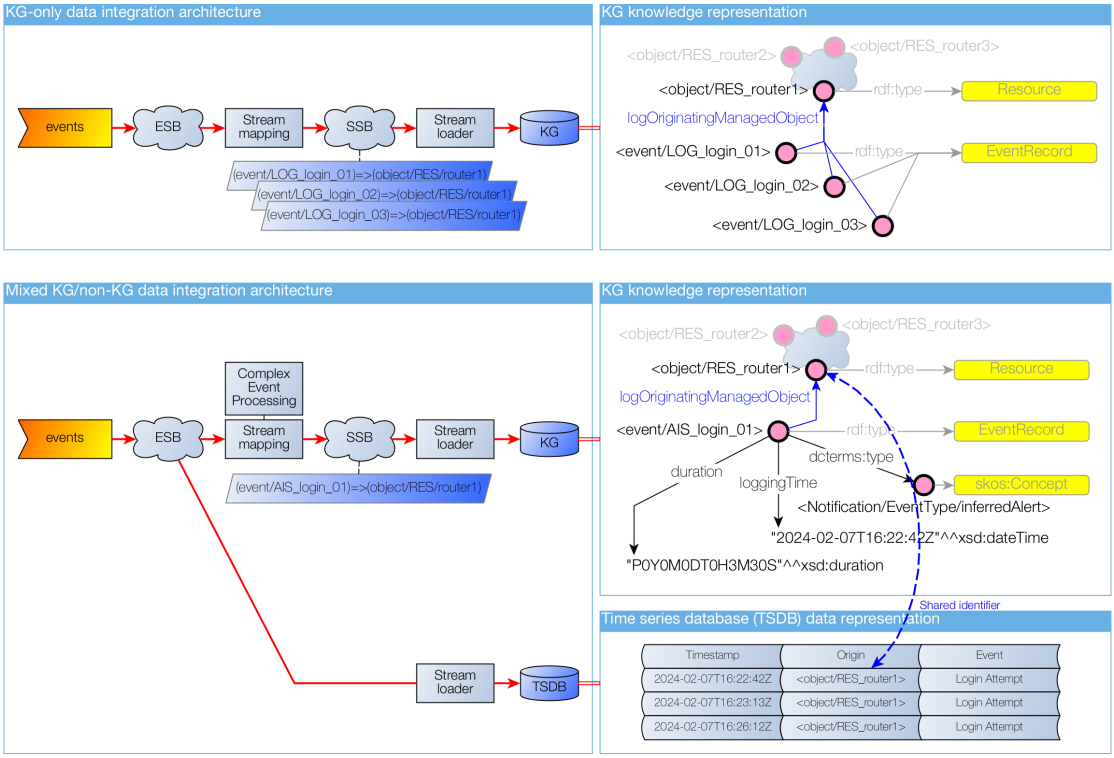



Figure 4.3: KG-only *vs* mixed KG/non-KG data integration for event data streams.
This diagram shows two data integration strategies using KGs to represent streaming data (such as cybersecurity logs) on a network topology. In the "KG-only" pipeline, all events are materialized in the graph. In the "mixed" pipeline, the stream is replicated to a time series database (TSDB), and only the start and duration of event sequences are materialized in the KG using an aggregation function in the KGC pipeline. In this second case, the homogeneity of identifiers allows for transitioning from one representation to another (dashed blue arrow) while benefiting from the advantages of each formalism and the performance of each storage technology, such as analyzing an incident context in KGs and calculating trend breaks in the TSDB. Other acronyms: ESB = Enterprise Service Bus; SSB = Semantic Service Bus.

# Part II

# Exploitation of the Knowledge Graph for Anomaly Detection and Diagnostic Assistance

# Introduction

Securing IT & telco networks is of utmost importance across industries. A solid understanding of user and equipment behavior is crucial for effectively handling network incidents, managing security risks, and comprehending the impact of user activities on the network infrastructure. While having a complete and inherently explainable behavioral model of systems is an ideal that every network designer and operator strives for, it is unfortunately almost impossible to achieve due to the complexity of Information and Communications Technology (ICT) systems and the inherent combinatorial nature of technical configurations. Furthermore, accessing network traffic and system/application logs to obtain an accurate system status can be difficult due to encryption or decentralized systems. This complexity hampers the ability to learn precise user and equipment action sequences and behavioral patterns, thereby introducing uncertainty in causal reasoning and making efficient incident management challenging.

To effectively address complexity and challenges in anomaly detection and situation understanding, we first approach the problem by conducting expert interviews to identify key use cases (Chapter 5). These use cases are analyzed towards a formal representation, allowing exploration of different approaches. We demonstrate that the use case "circumscribe assets and causes search space for multi-applications incident situations" defines a general framework of exploration that corresponds to three families of anomaly detection approaches: model-based design (Chapter 5), process mining (Chapter 6), and statistical learning (Chapter 7). By utilizing NORIA-O and the Knowledge Graph (KG) produced in the earlier steps of this thesis (Part I), we showcase how each approach can be implemented and used individually or in combination to handle a wide range of situations. We evaluate their effectiveness using annotated real-world data and feedback from network administrators. We understand that network administrators prioritize interpreting network behavior and making quick decisions during incident management, rather than delving into the intricacies of data integration and anomaly detection techniques. Therefore, we also explore the expectations of network administrators, obtained through expert interviews, to enhance the usability of the knowledge graph and these techniques. We present the implementation of a graphical interface that addresses these expectations and conduct a field evaluation to assess its performance (Chapter 8).

# Chapter 5

# Model-based Anomaly Detection

## 5.1 Introduction

Deploying ICT systems is based on design rules and expectations in terms of functionality and performance, which means that once operational, the behavior of networks is supposed to follow an underlying rationale and conform to pre-established models. These rules and models are in themselves *a priori* knowledge of what could be observed by NetOps and SecOps teams through Network Monitoring System (NMS) and Security Information and Event Management (SIEM) systems, and therefore serve as a basis for defining alert mechanisms [55] to detect undesirable ICT system states and explaining system behavior.

Description Logic (DL) [119] are a family of formal knowledge representation languages in which KGs take their roots [8, 119]. DL, as a formal system, could allow for addressing the description of anomaly detection cases for IT networks, providing a guarantee of consistency in case descriptions, as well as a capacity for explainable reasoning due to their logical foundation. Thereby, they bring an interesting general framework to address the challenges in incident management and envision a unified approach to the anomaly detection and diagnostic stages (Chapter 1). However, considering the combinatorics of large-scale ICT systems, further understanding of key anomaly detection cases is required to scope the description effort. Additionally, NetOps and SecOps teams face challenges in rule management in NMS/SIEM DSSs, thus bringing the objective to minimize the number of detection models or provide fundamental principles for expansion. This includes exploring opportunities to make the representation of networks through the KG formalism self-sufficient in interpreting their dynamics, for instance, by relying on simple queries on the KG (e.g. using SPARQL [368]) and avoiding the use of extensions such as Semantic Web Rule Language (SWRL) [147] or SPARQL Inferencing Notation (SPIN) [143] for cause-effect calculation if possible.

In the rest of this chapter, we consider the formal description of anomaly detection cases as an essential working foundation for knowledge capitalization and management, and subse-

quently the implementation of a KG-based solution for anomaly detection and diagnostic.[1] To accomplish this, we begin in Section 5.2 by defining a hierarchy of anomaly detection cases to consider. This hierarchy is based on interviews conducted with domain experts to identify the most complex diagnostic scenarios. In Section 5.3, we analyze to what extent DL allow for representing emblematic detection cases, leading us to define three complementary families of anomaly modeling techniques: model-based design, process mining, and statistical learning. Additionally, we discuss how they could practically help in key incident management process stages. Then, we focus on model-based design techniques in Section 5.4 with experiments on anomaly detection with graph retrieval and reasoning techniques (we discuss the process mining and statistical learning in Chapters 6 and 7, respectively). Finally, Section 5.5 concludes this chapter by providing a summary of the work that has been done and offering some preliminary conclusions regarding the opportunities and potential future developments associated with it.

## 5.2 Scoping the Diagnostic Phase with Experts

Assuming that the above-mentioned unified approach to the anomaly detection and diagnostic stages is an achievable goal, decision-making depends heavily on how the operating parameters of the systems are obtained and represented. We argue that observables (i.e. artifacts of the network assets' events and states) result from a generative process in a semi-open world: as assets' states dynamically vary with respect to other agents (e.g. neighboring assets, servicing technicians, end users, randomness) based on behavioral rules (e.g. failover mechanisms, remediation procedures), sets of states are interpreted through higher level (composite) concepts (Figure 1.4). Although these perspectives provide indications on the entities to represent and how to do it, the nature of the processing carried out on these concepts for the diagnostic phase remains a broad subject.

To get more specific on the nature of the analysis and responses that are performed, we conducted interviews with a panel of NetOps & SecOps experts, inviting the experts who were present for NORIA-O (Chapter 3) to participate again. We used the following methodology:

1. Ask experts for "pitfalls and wishes";
2. Analyze responses with clarifying questions following ideas from the Agile framework [318] and ISO/IEC/IEEE 29148:2011 guide [2];
3. Write exemplified anomaly detection use cases following the Cockburn-style template [18].

As a result, six use cases were defined to serve as a framework for the implementation of an

[1] The correspondence between computer programming and logic, as emphasized in [252], supports this intent.

automated reasoning system. Table 5.1 provides a short version of these use cases.

Table 5.1: List of anomaly detection use cases from expert panel interviews

| # | Description |
|---|---|
| 1 | Circumscribe assets and causes search space for multi-applications incident situations |
| 2 | Alert on impaired service situations occurring on (distributed) fail-over architectures |
| 3 | Assess legitimacy of a given network flow |
| 4 | Track single identity from a set of various activity traces |
| 5 | Analyze false-positive and recurrent cyber security alerts |
| 6 | Analyze compliance of web navigation traces from institutional website |

We notice that the use case #1 is the most challenging and encompasses the other use cases in that it generalizes the heuristic established in the incident diagnostic phase. During the interviews, the experts notably regularly raised the need for a confidence indicator alongside the inference results, which supports the need to search for a general mechanism. Therefore, we will consider the use case #1 as the ultimate goal to achieve in the remainder of this thesis work.

The experts also emphasized that the confidence indicator, seen as uncertainty about the interpretation to be given to a situation, should not be systematically used to reject inferences because it can itself contain information about complex situations (common cause failures, multi-application failures) for relating trouble tickets. To illustrate this, let us consider that each trouble ticket holds for a single independent incident or problem. However, it may occur that several tickets are linked, thus covering related incidents. Network assets being linked, mutual information arise about causes and consequences from the knowledge of the faults at hand. Hence, it is possible to infer and exploit a parent/child relationship reflecting the hierarchy of incidents from this confidence indicator: the child tickets describe incidents which are considered as consequences of the incident described in the parent ticket. Eq. (5.1) expresses this with modal logic in order to give substance to this idea:

$$\begin{aligned} \exists x, \exists y: \quad & (F.x \rightarrow T.x) \wedge (F.y \rightarrow T.y) \\ \forall x, \forall y: \quad & \Diamond (F.x \rightarrow F.y) \\ \models \exists x, \exists y: \quad & T.x \rightarrow T.y \end{aligned} \tag{5.1}$$

where $(x, y)$ are network assets (e.g. router, server, application), $F$ a fault indicator, $T$ a trouble ticket, and $\Diamond$ a possibility operator.

In fact, the notion of probability in the association of tickets for common cause failures directly relates to the need to capture incident contexts broadly, which would make it possible to generate a single incident ticket for a complex situation instead of soliciting various teams simultaneously and without coordination through multiple tickets. We remark that to further

develop on this approach with tractable computational complexity, we may hypothesize that alarm spreading and cascading failures are bounded with respect to time and space (Hyp. 2).

## 5.3 Approach

In this section, we first analyze the extent to which DL allow for representing emblematic detection cases (Section 5.3.1). We demonstrate that the principle of causality, essential in the diagnostic process, influences the modeling of detection cases. We also highlight the need for set-returning functions to fully express expectations regarding detection cases, going beyond the simple framework of DL. We then analyze the potential form of these functions through anomaly detection techniques that leverage KGs (Section 5.3.2). This leads us to define three complementary families of anomaly modeling techniques: model-based design, process mining, and statistical learning. We also discuss the practical implementation of these techniques through reasoning services, analyzing their role in decision support for incident management.

### 5.3.1 Getting Formal: on Description Logics for Anomaly Detection

**Implicational logic involves negating roles in Description Logics.** The abstraction of IT networks along a horizontal (transmission path) and vertical axis (protocol stack) – as discussed in Section 1.1.1 – embeds the functioning of networks in a performative implicational logic (Eq. 5.2).

$$antecedent_{\{OperationalState,FaultState\}} \Rightarrow consequent_{\{NormalBehavior,FaultBehavior\}} \quad (5.2)$$

For example, intermittent faults on the packet forwarding process of a network router result in sporadic packet retransmit requests at the data receiver side. Similarly, an anomaly on the Session layer of a connection between two terminals prevents the activation of the Presentation and Application layers. In short, by reformulating in a logical language, inferring the normal/fault (anomaly) state of ICT systems is the conclusive part of a logical formula.

In [16], event sequence mining and Horn clauses (i.e. a rule-like logical formula in disjunctive form with at most one positive literal [229]) are presented as means for detecting decision processes and anomalies from KGs. Interestingly, Horn clauses enable the representation of antecedents with multiple facts, which is a common case in the fields of NetOps & SecOps using fault signatures and attack signatures (Figure 5.1).

Because Horn clauses involves negation of concepts (a.k.a. “atomic negation” in DL), special care should be provided to knowledge representation in KGs as DL implement negation at the role level (i.e. the role complement “$\neg R$” [119, Section 2.4.1]). As a consequence, the “$\neg$”

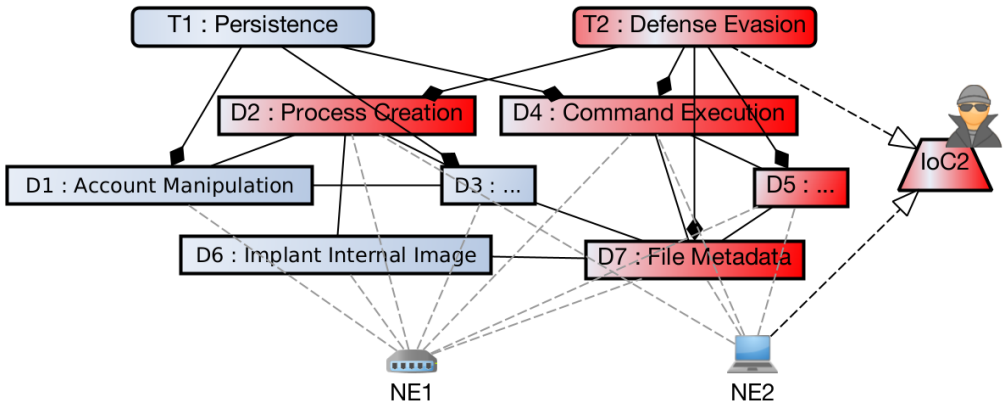



Figure 5.1: Indicator of Compromise (IoC) as a combination of multiple observables
The fault/attack signature concept refers to a combination of observables, where these observables depend on the supervised technical component. In this diagram, the evidence of malicious activity $IoC2$ of type $T2$ on $NE2$ is the logical consequence of the detection of $D2$, $D4$, $D5$, and $D7$ implying $T2$, which is expressed in Horn clause using the formula: $\neg D2 \vee \neg D4 \vee \neg D5 \vee \neg D7 \vee T2$. The following abbreviations apply: $NEx$ = Network Element (e.g. switch, router, server, virtual machine); $Dx$ = Detection (observable); $Tx$ = Technique (signature); $IoCx$ = Indicator of Compromise. A belief network between observables is established based on co-occurrence analysis, thereby providing the opportunity to refine a diagnosis in the absence of direct evidence.

construct on KG relations (i.e. object properties and datatype properties) expresses difference of relations rather than complement: drawing from the DL view, negating a role amounts to defining an inference rule from which we can search for entities that have all roles except the one that is negated.

In the following paragraphs, we present two detection cases in relation to use cases of Table 5.1, and analyze the impact of implicational logic on their modeling. This analysis allows us to explore the rule expressivity used in anomaly detection and determine whether limiting ourselves to Horn clauses is sufficient or if there is a need to express more complex rules.

**Detecting a network interface state inconsistency.** To start, we aim to address the question "*Which hosts have a NetworkInterface Admin/Operational status inconsistency?*" This question pertains to situations where the network interface of a network device is configured (administrative status) for transmitting data but lacks connectivity (operational status) due to signal loss with the neighboring device or misconfiguration. Figure 5.2 depicts this situation as a graph pattern using the NORIA-O data model (Chapter 3), and Equation 5.3 provides the logical form of this detection case.

$$\frac{\dfrac{}{\exists x, \exists y : R.x \wedge N.y \wedge P.xy \wedge F.y \rightarrow F.x}\ \text{pr.1} \qquad \dfrac{}{\exists y : (A.y \wedge \neg O.y) \vee (\neg A.Y \wedge O.y) \rightarrow F.y}\ \text{pr.2}}{\exists x, \exists y : F.y \rightarrow F.x} \tag{5.3}$$

Premise #1 ($pr.1$) in (5.3) implements the principle of fault signaling by the parent element, and reads as:

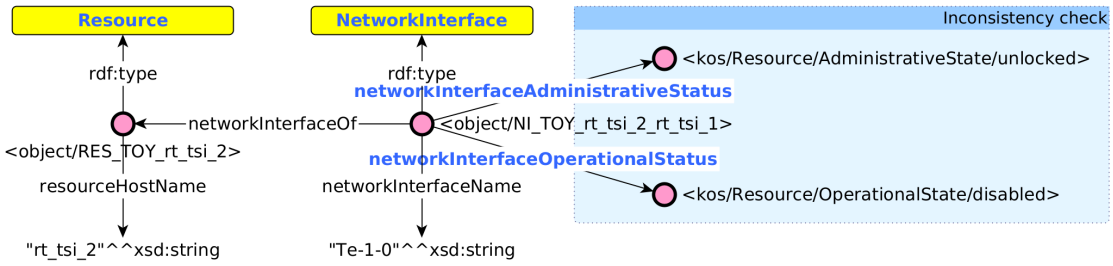



Figure 5.2: Network interface state inconsistency graph pattern
Graph pattern example for the "*Which hosts have a NetworkInterface Admin/Operational status inconsistency?*" detection case. `noria` is the default namespace for concepts and relationships. Other namespace: `rdf` = http://www.w3.org/1999/02/22-rdf-syntax-ns#. The diagram conforms to the Graffoo graphical notation [330].

- If $x$ is a `noria:Resource`,
- If $y$ is a `noria:NetworkInterface`,
- If $y$ is a part of $x$, i.e. a $\langle y, noria:networkInterfaceOf, x\rangle$ subject-predicate-object triple,
- If $y$ is in a faulty state, then $x$ is in a faulty state.

Premise #2 ($pr.2$) in (5.3) implements the business rule, and reads as:

- $A$ holds the `noria:networkInterfaceAdministrativeStatus` value,
- $O$ holds the `noria:networkInterfaceOperationalStatus` value,
- If there is an inconsistency between the states $A$ and $O$ of $y$, then $y$ is in a faulty state.

The conclusion in (5.3) implements the proof of reliability of the alert system (observability), and reads as: if $y$ is in a faulty state, then $x$ is in a faulty state.[2]

Several remarks can be derived from (5.3). Firstly, $pr.1$ restricts the domain of interpretation of the business rule (i.e. network interfaces attached to a resource). This easily translates into a graph pattern in a SPARQL query to query a KG using NORIA-O, and incidentally allows for addressing the "circumbscribe assets and causes search space" use case (Table 5.1). Secondly, (5.3) describes the alarm triggering mechanism itself through the observation of $F.x$. This also easily translates into a SPARQL query (e.g. using a `CONSTRUCT` or `INSERT` form). However, it overlooks the notification rewriting and enriching mechanisms (such as alarm masking and severity modification, as discussed in Section 2.2), which require additional logical clauses. Thirdly, the business rule ($pr.2$) assumes the ability to express the negation of predicates[3],

[2] The completeness of the logical system was validated with ProofTools (https://creativeandcritical.net/prooftools), resulting in a proof tree with $1+3\times32+5=102$ leaves.

[3] As discussed earlier: the role complement, defined by $(\neg R)^I = \top_n^I \setminus R^I$, which reads as "any role except the one mentioned".

which in DL would correspond to defining the concept *F.x* as (Eq. 5.4):

$$\begin{aligned} FaultyInterface \equiv & \\ & (networkInterfaceAdministrativeStatus \\ & \sqcap \neg networkInterfaceOperationalStatus) \\ \sqcup & (\neg networkInterfaceAdministrativeStatus \\ & \sqcap networkInterfaceOperationalStatus) \end{aligned} \tag{5.4}$$

Equation (5.4) sets expectations for anomaly detection at the Knowledge Graph Construction (KGC) process level (Chapter 4) with respect to the two following cases:

1. Deriving $FaultyInterface$ requires that, for example, the $\langle x, networkInterface-AdministrativeStatus, * \rangle$ triple is not materialized while $\langle x, networkInterface-OperationalStatus, * \rangle$ is (and vice-versa). This involves implementing conditional data mapping at the data integration pipeline level, both to match the truth table of the business rule, which is equivalent to using an `XOR` operator, and to ensure the absence of false alerts (i.e. false negatives, false positives) in case a materialization has not been performed due to lack of access to information about the network interface.
2. Describing the state of interfaces using NORIA-O is achieved through an object property. Hence, to align with True/False values of a logical representation, the following assumptions must hold:

$$\begin{aligned} True(Ax) &\equiv \langle x, networkInterfaceAdministrativeStatus, Enabled \rangle \\ False(Ax) &\equiv \langle x, networkInterfaceAdministrativeStatus, Disabled \rangle \\ True(Ox) &\equiv \langle x, networkInterfaceOperationalStatus, Enabled \rangle \\ False(Ox) &\equiv \langle x, networkInterfaceOperationalStatus, Disabled \rangle \end{aligned}$$

   Therefore, it is not possible to handle other interface states using a logical rule, such as $< x, networkInterfaceAdministrativeStatus, StandBy >$, or the absence of a triple unless we deviate from the *Open World Assumption* of the knowledge graph world and work according to the *Negation By Default* principle [229].

**Detecting a "k out-of n" resilience issue.** Next, we aim to address the question "*Which Applications have k out of n (50%) Resources with Alarm (EventRecord)*?" This question corresponds to a typical resilience problem, such as ensuring the data forwarding functionality of a network router redundancy group configured in an active-passive failover setup, or ensuring a minimum processing capacity in a Web server cluster with load balancing. Figure 5.3 depicts this situation as a graph pattern using the NORIA-O data model. Equations (5.5) and (5.6) are two logical formulations of the problem, as a Kripke structure in Tarski notation [56] and Set

theory within First-Order Logic respectively – the most concise forms for the given problem – where $F$ is a predicate for faulty states, and $k$ is the decision threshold.

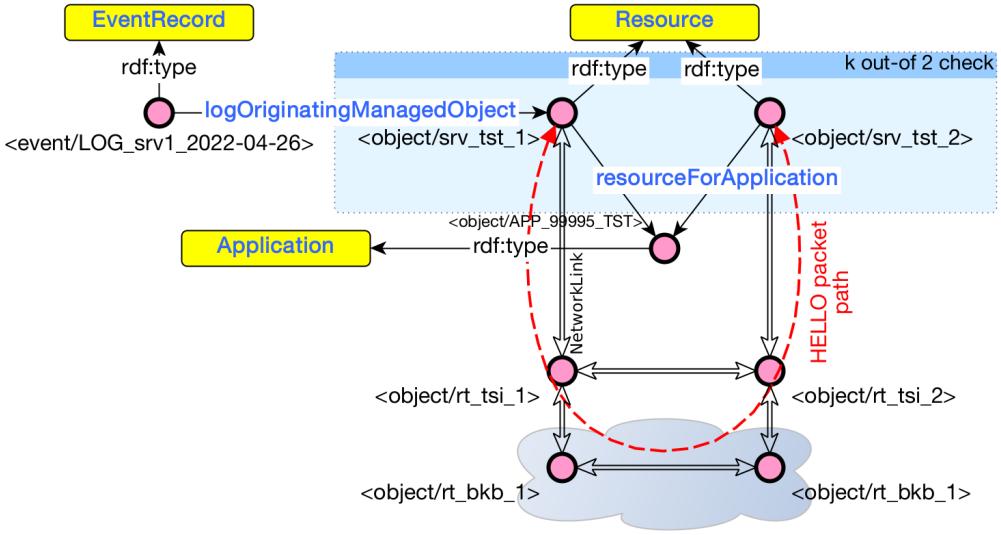



Figure 5.3: "k out-of n" graph pattern
Graph pattern example for the "*Which Applications have k out-of n (50%) Resources with Alarm (EventRecord)?*" detection case. The diagram includes a simplified view of the network's topology (access network and backbone network) through which servers communicate to exchange Hello messages. `noria` is the default namespace for concepts and relationships. Other namespace: `rdf` = http://www.w3.org/1999/02/22-rdf-syntax-ns#. The diagram partly conforms to the Graffoo graphical notation [330].

$$\exists x_1, \exists x_2 \ldots \exists x_n : \quad \left( \bigwedge_{i=1}^{k} F(x_i) \wedge \bigwedge_{j=k+1}^{n} \neg F(x_j) \right) \tag{5.5}$$

$$\exists X : \quad \Big( card(X) \geq k \wedge \forall y : \big( y \in X \rightarrow F(y) \big) \Big) \tag{5.6}$$

Several remarks can be derived from (5.5) and (5.6). Firstly, both equations account for the ability to trigger an alert once we are able to define a group of entities on which the logical test applies ($\{x_1, \ldots, x_n\}$ in the case of (5.5), and $X$ in the case of (5.6)). This easily translates into a graph pattern in a SPARQL query leveraging the fact that the entities under test have a relation to a grouping entity (e.g. the `<object/APP_99995_TST>` in Figure 5.3). However, equation (5.5) highlights the importance of testing for the absence of faults on the entities (as indicated by clause $\neg F(x_j)$), while equation (5.6) is tolerant to the absence of this information (i.e. it only relies on the number of faulty entities as indicated by the clause $card(X) \geq k$). Missing fault information could be due to data collection or materialization issues. In both cases, the functioning of the anomaly detection puts back-pressure on the KGC process, similar to the previous case of network interface state inconsistency.

Secondly, neither of the two equations allows for pinpointing the cause of the incident since the test provides a logical value encompassing all the entities in the group. Therefore, a second step of KG querying is required for cause investigation. In Figure 5.3, diagnosing a `Resource`

redundancy group using Hello packets, which are messages exchanged between members of a cluster indicating their operational state (e.g. according to the Gateway Load Balancing Protocol[4] or a similar protocol), may require querying the KG to check for connectivity between `Resource` entities or determine the network paths taken by the packets to diagnose intermediary `Resource` entities. This can be addressed using SPARQL queries, as SPARQL 1.1 [368] allows for defining property path (i.e. constraints on the graph pattern specific to relationships between the KG entities), which includes the * operator that stands for "any number of". However, the * operator only determines the existence of a path, not the specific path or the length of the shortest path [115]. This reflects the limitations of DL, and therefore, providing the support team members with the means to target their diagnostic intervention by giving them the list of faulty devices between two `Resource` entities (Eq. 5.7) involves introducing a set-returning function into the problem definition, such as $shortestPath$ in (5.7):

$$\Big\{\exists x, \exists y : R.x \wedge R.y \wedge \forall z\big(z \in shortestPath(x,y) \rightarrow R.y \wedge F.z\big)\Big\} \tag{5.7}$$

The semantics of (5.7) are as follows: $R$ is a predicate for identifying `Resource` entities, $F$ is a predicate for identifying entities with a faulty state, and $shortestPath$ is a function name referring to algorithms such as $A^*$. It is noteworthy that works from parallel research communities provide similar answers to this approach. For example, [39] introduces the concept of a set-returning function for spatial reasoning on a KG, and [198] uses shortest path calculations on a KG[5] to reveal semantic relationships within it.

**Concluding remarks.** To conclude this section, the causal interpretation used in anomaly detection and incident management implies the need to have control over the KGC process to ensure the effective implementation of detection cases. Indeed, it is important that KGC practice should not solely focus on materializing all the available data – as it might be suggested in Chapter 4 by the ease of implementing and deploying RML rule set – but is considered through the inference methods that NetOps/SecOps need to apply afterwards. The causal perspective also implies going beyond a strictly logical framework by introducing set-returning functions to model the detection cases and provide the support team members with the means for situation understanding. We further analyze the potential form of these functions in the following Section 5.3.2.

[4] https://www.cisco.com/en/US/docs/ios/12_2t/12_2t15/feature/guide/ft_glbp.html
[5] https://github.com/IDLabResearch/eice

### 5.3.2 Anomaly Modeling Techniques and Reasoning Services

**Families of anomaly modeling techniques.** While the logical foundations of KGs provide a solid basis for implementing anomaly detection, details from previous sections encourage going beyond a strictly logical framework. For instance, NetOps/SecOps experts emphasized the need for fuzzy reasoning and a confidence indicator for incident diagnosis (Eq. 5.1). Additionally, circumscribing assets and causes search space (use case #1 in Table 5.1) holds an implicit sub-graph computation requirement not available from the SPARQL 1.1 standard (e.g. using $A^*$ search algorithm in the case of Eq. 5.7). This means that relying solely on KG queries is not sufficient for our purpose. To address these apparent limitations, we define three families of anomaly modeling techniques (Table 5.2) that incorporate the notion of time and explainability capabilities thanks to the use of explicit representation:

- *Model-Based Design* assumes that the knowledge graph holds the necessary and sufficient data to infer unwanted situations with information retrieval (e.g. SPARQL queries);
- *Process Mining*, including conformance checking tools and Petri nets (P/T nets) representation, is effective for situations tied to a decision-model and bounded in time and location;
- *Statistical Learning* with graph embeddings [312, 153] assumes that anomaly models (i.e. the generalizing context of a set of situations) derive from the structure of the knowledge graph.

**Towards reasoning services.** Let's assume, for the sake of understanding how we could orchestrate the aforementioned families of techniques, that the greatest increase in operational efficiency in human-driven incident management would result from providing assistance for the automatic filling of trouble tickets. Such incident situation categorization can be done at several stages: before the ticket creation (early detection), at the ticket opening (cause/solution similarity based on ticket descriptors and context), during the resolution (cause/solution refinement and proposal of next action based on the actions taken). According to the use cases list defined in Table 5.2, we summarize the above analysis by proposing the following set of reasoning and inference services:

1. Predicting the category of a trouble ticket (i.e. the initial nature or technical impact, such as "isolated customer site", "traffic disruption", "integrity violation"). This is a classification problem, with classes defined in a user-provided controlled vocabulary represented as SKOS concepts.
2. Predicting the probable cause of a trouble ticket. We can imagine that this would also be a classification problem with references to a controlled vocabulary.

Table 5.2: Anomaly modeling technique families

| Principles | Strengths | Weaknesses |
|---|---|---|
| **Model-Based Design** | | |
| Query the graph to retrieve anomalies and their context. | Detecting anomalies "recorded" somehow in the graph thanks to the alarm system; straightforward translation of simple anomaly detection rules; multiple abstraction levels (subsumption). | Relies on expert knowledge; lack of probabilistic reasoning; hard to represent sequential decisions; may require to infer more prior information about the anomaly, e.g. its type using classification. |
| **Process Mining** | | |
| Align a sequence of entities to activity models, then use this relatedness to guide the repair. | Detecting anomalies with multiple alerting signals and sequential decisions; replayable models. | Relies on expert knowledge; may require denoising models; probabilistic relatedness. |
| **Statistical Learning** | | |
| Relate entities based on context similarities, then use this relatedness to alert and guide the repair. | Detecting anomalies with multiple alerting signals. | Requires fine tuning of the context definition depending on use case and temporality requirements; probabilistic relatedness. |

3. Detecting anomalies before a trouble ticket is even created. This service could be implemented with Link Prediction techniques [170]. The link being predicted is not necessarily related to incident remediation initially, but can rather be a link that would lead to the creation of an alarm or a trouble ticket.
4. Adding comments to a given trouble ticket, namely, a comment proposing the next best action to undertake based on the observations of a given situation.
5. Calculate the $n$ closest anomalies given an observed anomaly (with the ambition of then transposing/adapting the remediation plans that have worked in the past).

We notice that these five services have a common point concerning the capture of the context related to the nature of the incident. In the *model-based* experiments described in the following Section 5.4, we assume that we already have a model of the anomaly, and the detection result provides additional knowledge that indirectly enriches the trouble ticket during the associated incident diagnostic phase. This partially addresses reasoning services #1, #2, #3, and #4. Chapters 6 and 7 will cover both the capture and inference aspects through the *process mining* and *statistical learning* approaches respectively, thus inherently covering the five types of

services discussed. In general, we will assume that implementing service #1 will allow for the development of the others through incremental design.

## 5.4 Experiments

In this section, we explore the *model-based design* techniques family through three experiments with Semantic Web related techniques on data stored in a KG using the NORIA-O data model. We start, in Section 5.4.1, by directly applying the graph retrieval approach described above using SPARQL queries on the graph. Then, in Section 5.4.2, we extend graph retrieval with reasoning for root cause analysis using the FOLIO [62] ontology. Finally, in Section 5.4.3, we extend graph retrieval with reasoning for sequence analysis using the CFGOwl [255] approach.

### 5.4.1 Graph Retrieval

As mentioned in Table 5.2, graph retrieval refers to querying the graph to retrieve anomalies and their context based on formal expectations of a situation from experts. It assumes that all the necessary and sufficient data for the detection case is present in the KG. Since the KG is the result of a KGC process in which data is structured according to one or more ontologies, we consider that the implementation process of the detection case specified by the experts corresponds strictly to writing a SPARQL query that represents a graph pattern using the concepts and properties of the ontologies. In the following paragraphs, we use the detection cases studied in Section 5.3.1 as the basis for experimentation and conduct an evaluation based on the dataset defined in Appendix B.3.

**Detecting a network interface state inconsistency.** To begin with, we translate the question "*Which hosts have a NetworkInterface Admin/Operational status inconsistency?*" into the query of Listing 5.1 to match the graph pattern of Figure 5.2.

Using the Apache Jena SPARQL command line tool[6], we execute the query in Listing 5.1. This query returns `ResHostName="rt_tsi_2"` and `NI_Name="Te-1-0"`, satisfying the requirement for the detection case. However, the query is focused on a single pair of states and does not generalize to the various combinations of states shown in Table 5.3.

To address this limitation and achieve generalization, we implement and execute a new query that looks for incorrect state equivalences (Listing 5.2). To function properly, this query requires the inclusion of complementary state correspondence relationships in the KG (Listing 5.3), as the current NORIA-O v0.3 data model implementation lacks these relationships.

[6] http://jena.apache.org/documentation/query/

```
# === PREFIXES ===================================================================
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX noria: <https://w3id.org/noria/ontology/>
BASE <https://w3id.org/noria/>

# === QUERY ======================================================================
SELECT ?ResHostName ?NI_Name {
  ?NI a noria:NetworkInterface ;
    noria:networkInterfaceOf ?Res ;
    rdfs:label ?NI_Name ;
    noria:networkInterfaceAdministrativeStatus <kos/Resource/AdministrativeState/unlocked> ;
    noria:networkInterfaceOperationalStatus <kos/Resource/OperationalState/disabled> .
  ?Res noria:resourceHostName ?ResHostName .
}
```

Listing 5.1: SPARQL query for the “Network Interface State Inconsistency” case.

Table 5.3: Network interface state inconsistency truth table

| | OperationalState/enabled | OperationalState/disabled |
|---|---|---|
| AdministrativeState/unlocked | OK | Fault |
| AdministrativeState/locked | Fault | OK |

```
# === PREFIXES ===================================================================
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX noria: <https://w3id.org/noria/ontology/>

# === QUERY ======================================================================
SELECT ?ResHostName ?NI_Name ?NI_Admin ?NI_Oper ?NI_Admin_related_candidate {
  ?NI a noria:NetworkInterface ;
    noria:networkInterfaceOf ?Res ;
    rdfs:label ?NI_Name ;
    noria:networkInterfaceAdministrativeStatus ?NI_Admin ;
    noria:networkInterfaceOperationalStatus ?NI_Oper .
  ?Res noria:resourceHostName ?ResHostName .
  ?NI_Admin skos:relatedMatch ?NI_Admin_related_candidate .
  FILTER (?NI_Oper != ?NI_Admin_related_candidate)
}
```

Listing 5.2: SPARQL query for the “Network Interface State Inconsistency” case (generalization).

```
# === PREFIXES ===================================================================
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@base <https://w3id.org/noria/> .

# === DEFINITIONS ================================================================
<kos/Resource/AdministrativeState/locked>
  a skos:Concept ;
  skos:inScheme     <kos/Resource/AdministrativeState> ;
  skos:relatedMatch <kos/Resource/OperationalState/disabled> .

<kos/Resource/AdministrativeState/unlocked>
  a skos:Concept ;
  skos:inScheme     <kos/Resource/AdministrativeState> ;
  skos:relatedMatch <kos/Resource/OperationalState/enabled> .
```

Listing 5.3: Complementary SKOS definitions for the “Network Interface State Inconsistency” case, in Turtle syntax.

**Detecting a "k out-of n" resilience issue.** We now consider the question "*Which Applications have k out of n (50%) Resources with Alarm (EventRecord)*?" and translate it into the query of Listing 5.4 to match the graph pattern of Figure 5.3. In this implementation, we nest two SPARQL queries. The inner query lists the resources associated with an application and matches those that are linked to an event using an `OPTIONAL`[7] clause to calculate the ratio of resources in fault. The outer query, of type `CONSTRUCT`[8], creates a new triple associated for at-risk applications (i.e. new knowledge that can be integrated into the KG to alert and assist in diagnosis).

```
# === PREFIXES ========================================================================
PREFIX noria: <https://w3id.org/noria/ontology/>

# === QUERY ===========================================================================
CONSTRUCT { ?App noria:atRisk "K out-of N (50%)" . }
WHERE {
  SELECT
    ?App (COUNT(DISTINCT ?Res) AS ?ResTotal)
    (COUNT(DISTINCT ?ResImp) AS ?ResWithImpact)
  WHERE {
    ?Res a noria:Resource ;
      noria:resourceForApplication ?App .
    OPTIONAL {
      ?Event a noria:EventLog ;
        noria:eventLogOriginatingManagedObject ?Res .
      BIND (?Res AS ?ResImp)
    }
  } GROUP BY ?App HAVING ( (?ResWithImpact / ?ResTotal) >= 0.5)
}
```

Listing 5.4: SPARQL query for the "k out-of n" resilience issue.

Using the Apache Jena SPARQL command line tool, we execute the query in Listing 5.4, which returns the two following triples of Listing 5.5, thereby fulfilling the requirement for the detection case. However, we observe that the results do not adhere to the modeling for `inferred-alerts` proposed by the NORIA-O data model (i.e. the `noria:atRisk` property is not included in the data model), nor do they follow best practices for attribution (e.g. lack of time-stamping, lack of reference to the signaling system).

```
# === PREFIXES ========================================================================
@prefix noria: <https://w3id.org/noria/ontology/> .

# === Alarm ===========================================================================
<https://w3id.org/noria/object/APP_99995_TST> noria:atRisk  "K out-of N (50%)" .
<https://w3id.org/noria/object/APP_99996_TSI> noria:atRisk  "K out-of N (50%)" .
```

Listing 5.5: Results for the "k out-of n" resilience issue, in Turtle syntax.

[7] https://www.w3.org/TR/2013/REC-sparql11-query-20130321/#optionals
[8] https://www.w3.org/TR/2013/REC-sparql11-query-20130321/#construct

To address these two remarks, we extend the initial SPARQL query to the one in Listing 5.6. This new query generates an entity of type `noria:EventRecord` with a time-stamp, criticality and alarm nature information, and origin. Listing 5.7 shows the result of executing this query. By inserting the triples from Listing 5.7 into the knowledge graph, it becomes possible to consider resilience issues as any other type of alert, whether through a SPARQL query or any other KG-based technique.

```
# === PREFIXES ========================================================================
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX prov: <http://www.w3.org/ns/prov#>
PREFIX noria: <https://w3id.org/noria/ontology/>
BASE <https://w3id.org/noria/>

# === QUERY ===========================================================================
CONSTRUCT {
  ?EventURI a noria:EventRecord ;
    prov:wasDerivedFrom "noria-sdlri2023-toy-example" ;
    noria:alarmMonitoredAttribute "Resilience" ;
    noria:alarmSeverity <kos/Notification/Severity/PerceivedSeverity/major> ;
    noria:logOriginatingManagedObject ?App ;
    noria:logOriginatingManagementSystem <object/APP_NORIA_AD> ;
    noria:loggingTime ?NotificationConstructTime ;
    dcterms:description "Application at risk: K out-of N (50%)" ;
    dcterms:type <kos/Notification/EventType/inferredAlert> .
} WHERE {
  SELECT
    ?App (COUNT(DISTINCT ?Res) AS ?ResTotal)
    (COUNT(DISTINCT ?ResImp) AS ?ResWithImpact)
    (NOW() as ?NotificationConstructTime)
    (IRI(CONCAT("event/AIS_TOY_", strUUID())) AS ?EventURI)
  WHERE {
    ?Res a noria:Resource ;
      noria:resourceForApplication ?App .
    OPTIONAL {
      ?Event a noria:EventRecord ;
        noria:logOriginatingManagedObject ?Res .
      BIND (?Res AS ?ResImp) }
  } GROUP BY ?App HAVING ( (?ResWithImpact / ?ResTotal) >= 0.5)
}
```

Listing 5.6: SPARQL query for the “k out-of n” resilience issue, with generation of alerts as an EventRecord entity.

```
# === PREFIXES ========================================================================
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix noria: <https://w3id.org/noria/ontology/> .
@base <https://w3id.org/noria/kos/> .

# === Alarm ===========================================================================
<event/AIS_TOY_bb7818a2-c15e-4e33-9655-dc45e5c4d8c3>
  a                               noria:EventRecord ;
  prov:wasDerivedFrom             "noria-sdlri2023-toy-example";
```

```
  noria:alarmMonitoredAttribute  "Resilience" ;
  noria:alarmSeverity            <kos/Notification/Severity/PerceivedSeverity/major> ;
  noria:logOriginatingManagedObject <object/APP_99995_TST> ;
  noria:logOriginatingManagementSystem <object/APP_AM049124> ;
  noria:loggingTime              "2023-09-08 04:00:0.0" ;
  dcterms:description            "Application at risk: K out-of N (50%)" ;
  dcterms:identifier             "AIS_TOY_bb7818a2-c15e-4e33-9655-dc45e5c4d8c3" ;
  dcterms:type                   <kos/Notification/EventType/inferredAlert> .

# [...] similar for <object/APP_99996_TSI>
```

Listing 5.7: Results for the “k out-of n” resilience issue, with alerts as an EventRecord entity, in Turtle syntax.

### 5.4.2 Reasoning with FOLIO for Root Cause Analysis

In this experiment, we utilize reasoning to extend graph retrieval for RCA in a scenario where a network service is affected due to a fault in the upstream data transmission path. Figure 5.4 provides an example of a knowledge graph representing this situation.

To establish a connection between separate observables (e.g. `EventRecords` entities on network resources making the data transmission path) and provide insights into the probable cause, we incorporate expert knowledge from an Failure Mode and Effect Analysis (FMEA) study (Table 5.4). The model from this study establishes a causal chain between an `OpticalLinkCutOrUnpluggedOrDirty` cause at the `NetworkLink` entity level and the `ServiceImpairement` effect at the `Resource` entity level.

Table 5.4: Deduction rules from FMEA study
This table presents domain knowledge structured through FMEA for the “optical link continuity loss” issue. It is akin to a set of two deduction rules that can be read as: when *Cause* occurs on *Entity* then observe *Effect*; this is a *Mode* situation that can be solved by *Containment* action. The chaining of the two rules (i.e. the `EndpointLostConnection` effect of rule #1 is the failure mode of rule #2) enables to infer causal relationship of the `<event/AIS_ServiceImpairement>` alert of Figure 5.4 to a fault at the level of the `<object/NL_TOY_rt_tsi_1_rt_tsi_2>` network link.

| # | Entity | Mode | Detect & Respond |
|---|---|---|---|
| 1 | NetworkLink | OpticalLinkContinuityLoss | *Effect*: EndpointLostConnection<br>*Cause*: OpticalLinkCutOrUnpluggedOrDirty<br>*Containment*: CleanReplugReplace |
| 2 | Resource | EndpointLostConnection | *Effect*: ServiceImpairement<br>*Cause*: OpticalLinkContinuityLoss<br>*Containment*: TracerouteRestart |

To implement the attribution mechanism, which assigns responsibility to each entity in the situation, we utilize the FOLIO ontology [62]. This ontology enables the classification of an incident situation through OWL-based reasoning, particularly by identifying the role of each observable through concepts such as `FailureMode` (the characteristic observable of a given fault), `LocalEffect` (refers to the first detected effect on the system), and `FailureCause`

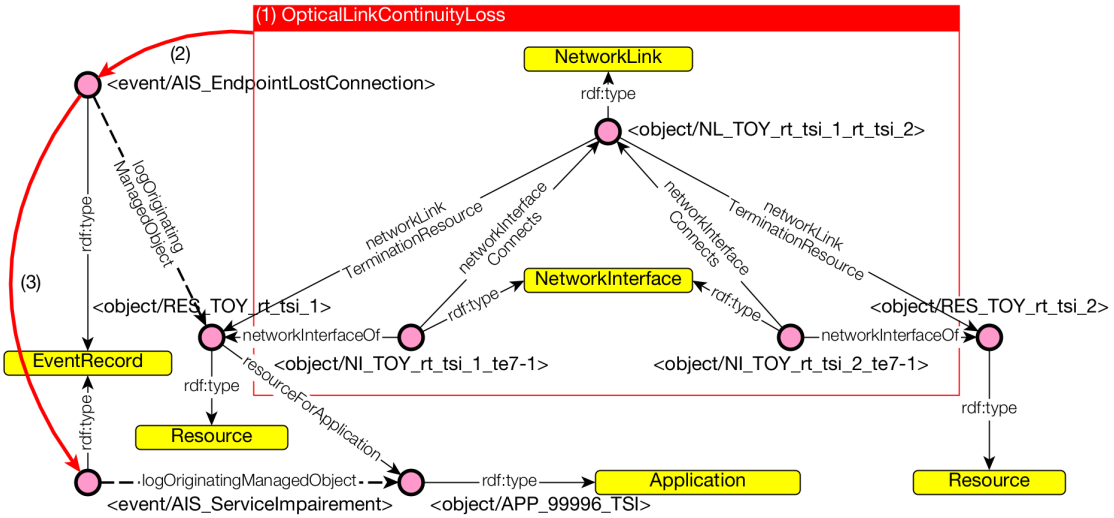



Figure 5.4: Optical link continuity loss graph pattern
Graph pattern example for the RCA experiment with the FOLIO ontology with two network devices connected by network link, and an application relying on the connectivity state of this network link. The diagram includes two alarms reflecting on a fault occuring on the device on the left and the application it is a resource for. The red box highlights the location of the probable cause of the issue, and number (1), (2) and (3) the temporal relatedness of the events. `noria` is the default namespace for concepts and relationships. Other namespace: `rdf` = http://www.w3.org/1999/02/22-rdf-syntax-ns#. The diagram conforms to the Graffoo graphical notation [330].

(defines a concept with no further `hasNextEffect` relations).

**Experiment & results.** For this experiment, we followed the steps outlined below using the FOLIO's toolset, which was shared in open source[9] by the authors of the ontology: *1)* we transcribed the FMEA from Table 5.4 into a CSV file that aligns with the FOLIO data model; *2)* we adapted the original FOLIO mapping rules to accommodate the experiment's FMEA CSV file, and then used the `yarrrml-parser`[10] tool to generate the corresponding RML mapping rule set in Turtle syntax; *3)* we converted the experiment's FMEA definitions into an OWL/FOLIO-based ontology using the `rmlmapper-java`[11] tool and the RML mapping rule set generated in the previous step; *4)* we loaded the experiment's dataset into the Virtuoso graph store of the KG-based platform described in Chapter 4, along with the FMEA ontology created in the previous step, the FOLIO ontology, and the NORIA-O ontology; *5)* we looked for means to infer causes and effects with the native capabilities of the Virtuoso DBMS.

The Virtuoso platform offers reasoning mechanisms accessible via SPARQL queries that are initially limited to subsumption.[12] Unfortunately, the use of FOLIO requires the ability to reason through a sequence of restrictions on class equivalences (i.e. finding entities satis-

[9] https://github.com/IBCNServices/Folio-Ontology
[10] https://github.com/RMLio/yarrrml-parser
[11] https://github.com/RMLio/rmlmapper-java
[12] https://docs.openlinksw.com/virtuoso/virtuosotipsandtricksrdfschowlinfrl/

fying a combination of [owl:equivalentClass/owl:onProperty] property paths). This functionality cannot be achieved using the native capabilities of Virtuoso without employing pragmas[13] in the SPARQL queries or implementing custom inference rules using the SPIN vocabulary [192].

As reasoning over [owl:equivalentClass/owl:onProperty] property paths typically corresponds to tableau calculus, we followed the additional steps outlined below to overcome the above apparent limitations: *1)* we loaded the experiment's dataset into the Protégé 5.1 ontology editor [227], along with the FMEA ontology created in the previous steps, the FOLIO ontology, and the NORIA-O ontology; *2)* we merged the set of ontologies to create a single reasoning space for the HermiT reasoner [127] (a Protégé plugin enabling tableau calculus); *3)* we ran the HermiT reasoner for inferring causes of the <event/AIS_ServiceImpairement> entity of Figure 5.4.

Unfortunately, when activating the HermiT reasoner, we encountered an "error occurred during reasoning: Java heap space" exception after more than ten minutes of execution, regardless of the memory allocated to the process (i.e. from 400 MB to 1.5 GB).

**Discussion.** Although inconclusive due to the failure to implement the inference mechanism, these experiments highlight the interesting aspect that modeling the detection case according to an FMEA study can align with representing an abnormal situation on a network topology represented as a knowledge graph, using properties from the noria:EventRecords entities as observables. It is also noteworthy that the FMEA, in addition to being a proven and well-established method in industrial environments, has an intrinsic form of explainability due to the representation of deduction rules in the form of a standardized table. Furthermore, despite the lack of results to support this claim, we observe that the use of reasoning with FOLIO lacks the capability for forward prediction of the next ICT system state. However, this is partially offset by the fact that the ontology resulting from the conversion of domain knowledge can incorporate expert-provided advice (i.e. the Containment attribute in Table 5.4). Finally, we remark that although the use of SWRL is highlighted by the authors of FOLIO in [62] for rule-based inference (e.g. IF (CPU load > 75%) AND (process progress = 0) THEN alert hanging process), this can be substituted by utilizing SPARQL CONSTRUCT queries to minimize the need for additional engines involved in the overall RCA process.

### 5.4.3 Reasoning with CFGOwl for Sequence Analysis

In this experiment, we assume that there is no prior knowledge of the temporal relationship between events that would allow them to be associated into a logical trace, except for the

[13] Language constructs that specifies how a given parser/compiler should process its input.

fact that they are all emitted by the same source. Hence the question arises of what are the means for identifying a pattern in the absence of materialization of these relationships in the knowledge graph, including for sequences of events nested at multiple levels such as illustrated in Figure 5.5 with logged events of user actions on a given host.

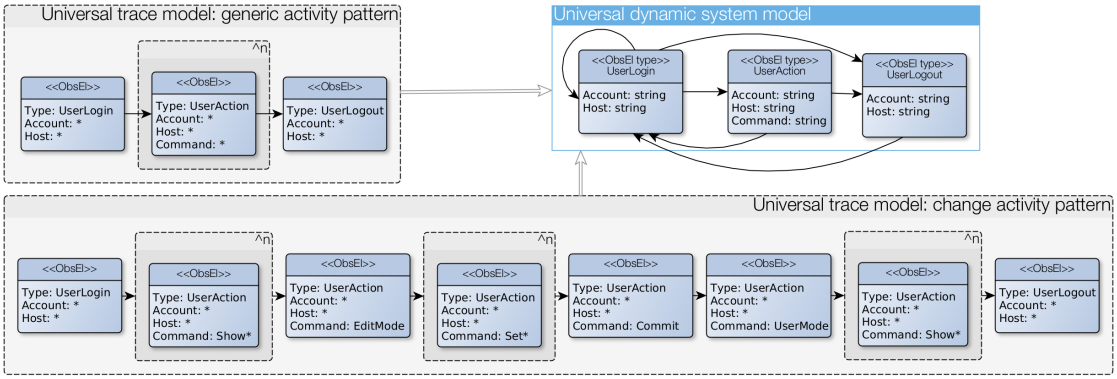



Figure 5.5: Universal trace model for logged user actions
This diagram presents three perspectives for trace-based reasoning [290] on user actions logged as observable elements (`ObsEl`). The "generic activity pattern" captures the fact that login and execution of commands on a remote device (should) typically correspond to a $\{UserLogin, n \times UserAction, UserLogout\}$ sequence. Finer-grain analysis of the $UserAction$ should also allow to detect a configuration change activity, which corresponds to the "change activity pattern". The combination of both patterns leads to considering a "universal dynamic system model" for activity detection and categorization at multiple levels (i.e. at the most abstract level of the generic pattern, as well as at the level of configuration change).

In the following, we utilize formal-grammar-based reasoning to extend graph retrieval for alerting on cybersecurity risk. Figure 5.6 depicts a situation where a user fails to connect via Secure Shell (SSH) [208] to a device due to too many authentication attempts, which is considered an indicator of intrusion attempts on a system.

**Experiment & results.** For this experiment, we followed the steps outlined below using the CFGOwl approach [255]: *1)* we executed the query of Listing 5.9 to retrieve the SSH service messages from the OOTD 2023 dataset (Section B.3); *2)* we implemented the formal grammar of Listing 5.8 in Lark syntax[14] for the classification of groups of messages; *3)* we concatenated the messages and processed them with the command of Listing 5.10 to generate a CFGOwl-based ontology. This ontology was then further analyzed by browsing it and running the HermiT reasoner to infer sequence classification.

```
// A bunch of logs to scrutinize through rules
start: rule+

// Detection rules
?rule: login_success
| login_attempt
| login_max_auth_tries
```

[14] https://www.lark-parser.org/ide/

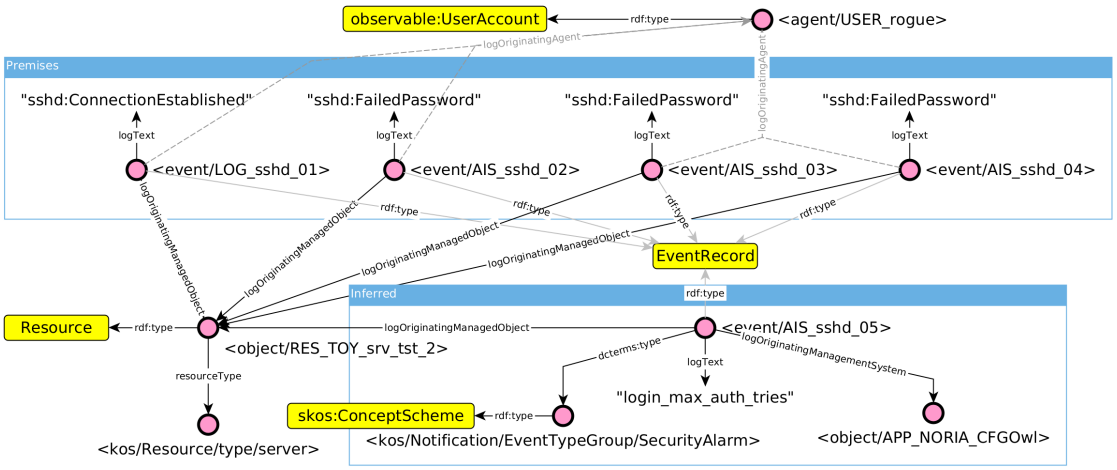



Figure 5.6: EventRecords related to a malicious user attempting to log in through Secure Shell
Graph pattern example depicting a set of logged events from the SSH service of `<object/RES_TOY_srv_tst_2>` following login attempts by the `<agent/USER_rogue>` user. The diagram includes an inferred alarm notifying a `login_max_auth_tries` situation. `noria` is the default namespace for concepts and relationships. The other namespaces are the following: `dcterms` = http://purl.org/dc/terms/; `observable` = https://unifiedcyberontology.org/ontology/uco/observable#; `skos` = http://www.w3.org/2004/02/skos/core#; `rdf` = http://www.w3.org/1999/02/22-rdf-syntax-ns#. The diagram conforms to the Graffoo graphical notation [330].

```
8
9   // Situations
10  login_success: LOG_CONNECTION_ESTABLISHED LOG_LOGIN_FAIL~0..2 LOG_LOGIN_SUCCESS
11  login_attempt: LOG_CONNECTION_ESTABLISHED LOG_LOGIN_FAIL~0..2 LOG_LOGIN_GRACE_TIME
12  login_max_auth_tries: LOG_CONNECTION_ESTABLISHED LOG_LOGIN_FAIL~3
13
14  // Vocabulary
15  LOG_CONNECTION_ESTABLISHED: "sshd:ConnectionEstablished"
16  LOG_LOGIN_SUCCESS: "sshd:AcceptedPassword" | "sshd:AcceptedPublickey"
17  LOG_LOGIN_FAIL: "sshd:FailedPassword"
18  LOG_LOGIN_GRACE_TIME: "sshd:ConnectionClosedPreAuth"
19
20  // Disregard spaces in text
21  WHITESPACE: (" " | "\n")+
22  %ignore WHITESPACE
```

Listing 5.8: Context free grammar for the “login classification” use case, in Lark syntax.

```
1   # === PREFIXES ===================================================================
2   PREFIX noria: <https://w3id.org/noria/ontology/>
3
4   # === QUERY ======================================================================
5   SELECT ?logText WHERE {
6     ?Log a noria:EventRecord ;
7       noria:alarmMonitoredAttribute "Secure remote-login and remote-execution" ;
8       noria:logText ?logText ;
9       noria:loggingTime ?logTime .
10  } ORDER BY ?logTime
```

Listing 5.9: SPARQL query for the “login classification” use case.

```
1   python cfgowl.py \
```

```
2  -g ./sshd_01.lark \
3  -s "sshd:ConnectionEstablished sshd:AcceptedPassword
4    sshd:ConnectionEstablished sshd:FailedPassword sshd:ConnectionClosedPreAuth
5    sshd:ConnectionEstablished sshd:FailedPassword sshd:FailedPassword sshd:
       ConnectionClosedPreAuth
6    sshd:ConnectionEstablished sshd:FailedPassword sshd:FailedPassword sshd:FailedPassword" \
7  -f turtle \
8  > ./sshd_01_seq_all.ttl
```

Listing 5.10: Generating a CFGowl-based ontology from a grammar and a event sequence, using the CFGowl tool set.

Scrutinizing the resulting ontology (i.e. the `sshd_01_seq_all.ttl` file in Listing 5.10) in the Protégé 5.1 ontology editor, we find a correct classification for each sequence (lines 4 – 6 of Listing 5.10) by looking at the parent class under the `<http://w3id.org/cfgowl/start>` entity where each terminal entity (i.e. vocabulary strings in Listing 5.8) would appear grouped. For example, the entities for the fourth sequence are found under `<http://w3id.org/cfgowl/login_max_auth_tries>`. Looking at the object properties and other class hierarchies that are produced by the `cfgowl.py` tool, we find the sequence of the terminal entites in the (Eq. 5.8) form.[15] Using the HermiT reasoner, we infer new relationships of the (Eq. 5.9) form.

$$Terminal_1 \xleftarrow{sequence{:}directlyFollows} Terminal_2 \tag{5.8}$$

$$Terminal_1 \xrightarrow{sequence{:}directlyPrecedes} Terminal_2 \tag{5.9}$$

**Discussion.** From these results, we observe the correct identification of the `login_max_auth_tries` sequence, even when it is hidden among other logged events, thereby satisfying the detection use case. For example, one could retrieve the classification of such results directly in the context of a knowledge graph using a SPARQL query. Furthermore, we observe the inference of temporal relationships, which are useful for RCA and deducing when the adversary began its task.

It is also noteworthy that using a reasoner is not required to parse the sequence, but it is useful in cases where the ontology defines high-level axioms based on the classes we parse the sequence on. For instance, in the case of Figure 5.6, we might define a class that classifies a sequence of events as "unsafe attempts" using an inference pattern of the form $(loginAttempt \sqcap loginMaxAuthTries) \sqsubset UnsafeLoginAttempt$, which a reasoner can infer reasonably quickly.

Additionaly, considering the expressivity of context-free grammars, we can use recursive

[15] Where the namespace `sequence` refers to http://www.ontologydesignpatterns.org/cp/owl/sequence.owl#

functions to compose different classification settings, enabling the identification of nested sequences of events through rule set factorization. We can also handle partial observations of a situation, which would manifest as missing terminals, using the "?" symbol in the Lark syntax. For example, the `login_success` rule can be rewritten as `login_success: LOG_CONNECTION_ESTABLISHED? LOG_LOGIN_FAIL~0..2 LOG_LOGIN_SUCCESS`.

Finally, we also observe that the CFGowl approach does not allow forward prediction of the next ICT system state. Therefore, anticipating a future state or making a remediation action recommendation will involve fuzzy reasoning over the grammar vocabulary or associating domain knowledge (whether learned or defined by an expert) at the level of the entity defining the semantics of the `login_max_auth_tries` alert.

## 5.5 Conclusion and Future Work

In this chapter, we assumed the existence of fundamental and logical formal models to describe the dynamics of ICT systems. The principles used for the design and deployment of networks are good candidates for this, as well as the incident summaries written by experts during the post-incident activities. The availability of these models is not, however, the general case, and the diversity of forms of those available raised the question of what basis to use to develop a generic approach to anomaly detection that would use knowledge graphs.

We then chose to attempt to express some detection cases using DL, given the expressiveness and intrinsic explainability of DL, as well as the strong links that KGs have with DL. To assist us in choosing the detection cases to study, we interviewed a panel of NetOps & SecOps experts. The discussions with them led us to conclude on the prioritization of detection cases, with the ability to capture an incident context or the ability to determine an investigation scope as ultimate goals to implement.

In parallel to this, the modeling attempts of two detection cases using DL allowed us to understand that, even though leveraging a knowledge graph for anomaly detection can be seen as a subsequent step to KGC, the causal perspective used in anomaly detection and incident management implies the need to have control over the KGC process to ensure the effective implementation of detection cases. For example, simply materializing all available data is not sufficient to ensure the detection of anomalies with logical rules. Similarly, the ability to implement a contextual KGC process (e.g. using logical rules at the KGC process level to introduce default values in case of missing data) also determines the strategy of anomaly detection to be implemented. Indeed, the absence of some data at the KG level can be considered as contributing to the detection mechanism or as misleading the mechanism, depending on the expressiveness of the chosen logical rules for anomaly detection. Furthermore, we have also observed that the causal perspective implies going beyond a strictly logical frame-

work by introducing set-returning functions to model the detection cases and provide the NetOps/SecOps teams with the means for situation understanding. This last point has notably led us to define three complementary families of anomaly detection techniques: model-based, process mining, and statistical learning.

The model-based family was further explored in this chapter through three experiments using graph retrieval as a basis, which involves querying the KG for anomaly detection using the available data and translating the expert knowledge (the model) directly into the detection case query. Using graph retrieval, we demonstrated how to implement SPARQL queries that fulfilled the requirements for the detection cases. We then complemented the graph retrieval with reasoning for RCA and event sequence classification. The RCA experiments with the FOLIO approach [62] showed, despite the inability to successfully conduct the experimentation, the need to add deduction rules to the KG-DBMS in the platform from Chapter 4 in order to effectively use expert knowledge derived from an FMEA study. Similarly, the experiments showed that this approach allows for recommending remediation actions but does not enable forward prediction of the ICT system states. The experimentation of event sequence classification using the CFGowl approach [255], on the other hand, showed the possibility of identifying nested or incomplete sequences based on an abstract definition of detection cases in a grammar. Similarly, forward prediction of the ICT system states is not possible in this case either.

Based on these results, we are considering two complementary development axes. The first concern is to allow designers and support engineer to formalize their domain knowledge into SPARQL queries, an FMEA-related format, or a context-free grammar. It is noteworthy that scrutinizing how to write detection cases in DL has brought the intuition that the proof trees resulting from the validation of the anomaly detection formulas could be a means to estimate the computational complexity of the anomaly detection rule in logical form. Therefore, it would be interesting to include this estimation in the knowledge capture phase in order to anticipate the response times of the implementations of the detection cases. In addition to providing support to experts in the definition effort, we also envision the use of Natural Language Processing (NLP) techniques to (partially) extract these models from procedures and descriptions of network behavior in text format [390], or eventually infer these models from the network itself, including both its structure and configuration, but also its behavior, which is discussed then in Chapters 6 and 7. Finally, the second axis aims to understand how to complement the model-based techniques presented in this chapter with those of the process-mining and statistical-learning families. The idea is that a combined operation of the three families would allow leveraging the explainability properties of each while opening up detection capabilities to cases that are not (or are difficult to) covered by model-based techniques. These reflections are also supported by Chapters 6 and 7, and an illustration of combination according to the principle of synergistic reasoning is proposed in Chapter 8.

# Chapter 6

# Relational Learning and Anomaly Detection on Web Navigation Traces

## 6.1 Introduction

Behavioral analysis in cyber-security is a technique used to identify and mitigate security threats by analyzing the activities of users, systems, and network entities. It focuses on understanding normal patterns of behavior and detecting anomalies that may indicate malicious activities. As significant portion of user interactions with applications now takes place through a Web interface, let us consider, for example, an exploit of a public-facing application[1]. A relatively simple scenario could be that, after a phase of reconnaissance of the targeted system, the malicious user directly accesses the homepage of a service platform, deceives the authentication system (e.g. using SQL injection technique[2]), exfiltrates data, and then exits the service by navigating directly to another webpage. Detecting this kind of situation involves analyzing both the application logs and the network traffic content. Unfortunately, application logs can be inaccessible or unusable due to their privacy or improper formatting. Similarly, network traffic can be encrypted or out of reach for collection, resulting in the loss of information about the user's interaction with the platform and the attack scenario [151].

Due to these observability issues, it is difficult for NetOps/SecOps teams to ensure a solid learning of a behavioral model based solely on network topology and usage data stored as a knowledge graph. Additionally, the study of data models (Chapter 2) has shown shortcomings in their ability to represent decision models with branching conditions, which is important for accurately characterizing a course of action and its potential next steps.

In this chapter, we posit that in the absence of being able to learn these behavioral models from network-side logging, learning them at the user-side can provide a useful – albeit partial – basis for characterizing network-side observables. To achieve this, we extend the Dynagraph project [203] (a system combining trace dumping tools with a Web app for rendering graph

[1] https://attack.mitre.org/techniques/T1190/
[2] https://en.wikipedia.org/wiki/SQL_injection

data from traces) with *process mining* techniques to learn interpretable activity models based on linked data: the Graphameleon Web extension collects user activity traces (network traffic, interactions with the Web browser) during a Web navigation session and serializes this data in RDF using the UCO [388] vocabulary. The resulting data is processed to interpret the activity traces at a semantic level and derive activity patterns, notably in the form of Petri nets. These activity models can then be used to identify similar activities from events recorded via the KG-based platform (Chapter 4), and even assist with root cause analysis by reordering a set of events (Chapter 8).

The remainder of this chapter is organized as follows. In Section 6.2, we introduce our approach to capture knowledge from Web navigation traces. This involves a three-layer activity modeling (HTTP, micro-activities, macro-activities) based on the UCO vocabulary. We also describe the Graphameleon data capture component and the utilization of Petri nets for anomaly detection in Web navigation traces. Section 6.3 describes our experiments and evaluations. Finally, Section 6.4 concludes this chapter by providing a summary of the work that has been done and offering some preliminary conclusions regarding the opportunities and potential future developments associated with it. The Graphameleon code is available at https://github.com/Orange-OpenSource/graphameleon, along with the Graphameleon dataset at https://github.com/Orange-OpenSource/graphameleon-ds.

## 6.2 Approach

Our approach focuses on learning Web navigation activity templates at the Web browser level. Using these learned templates, our goal is to analyze new activity traces within the context of the network in which they occur. For example, weighting the risk level of a malicious activity based on the nature of the targeted servers, or segmenting a long trace into subparts by observing that non-connected network segments are being used.

To learn these activity templates, capturing data at the Web browser level is essential. Additionally, an explicit representation of activities with controlled vocabulary is necessary for an explainable generalization. Finally, the representation of these activities should seamlessly integrate with network topology data, which can be modeled as graphs. This integration could help contextualize activities, like assessing risk according to infrastructure criticality or separating actions when no network link exists between resources.

To achieve these objectives, we propose a three-step approach. First, we utilize semantic modeling of activities by leveraging a knowledge graph and the UCO vocabulary (Section 6.2.1). Second, we develop a Web browser plug-in called Graphameleon, which captures data and serializes it in RDF (Section 6.2.2). Finally, we derive activity models using Petri nets for detecting Web navigation activity scenarios (Section 6.2.3). Figure 6.1 provides an overview of

our approach. The experimental results and findings of our approach are discussed in Section 6.3.

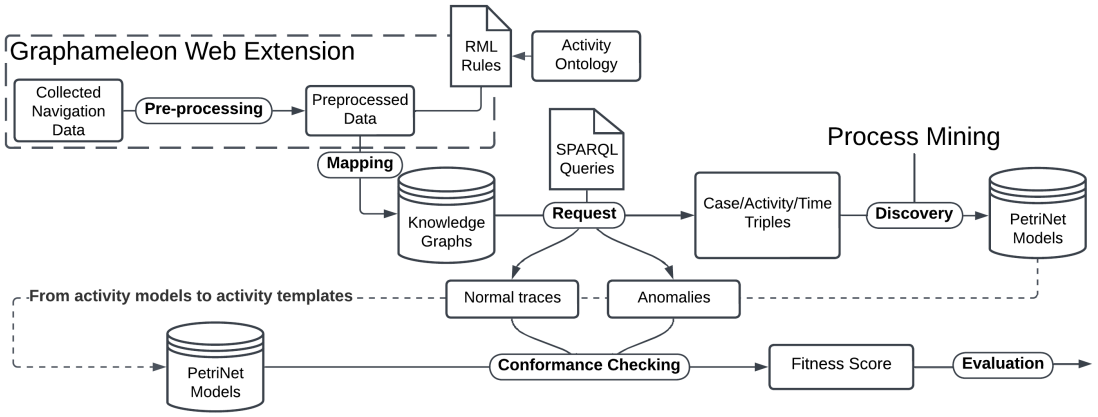



Figure 6.1: Overview of the Graphameleon data processing pipeline.
The Graphameleon Web extension captures and annotates user activity at the Web browser level. A process mining component derives activity models from the resulting RDF KG. These models can be used to build a library of activity templates, which are then used by a conformance checking component to classify new activity traces as normal or abnormal activities.

In terms of application context, our overall approach targets the following uses: an average user could use the framework as a tool for transparency regarding Web services, particularly concerning tracking practices and privacy compliance (the experiment described in Section 6.3.1 illustrates this); a cybersecurity analyst could install the Graphameleon component to capture activity traces with the aim of creating a detection model by interacting within a sandbox with an infected system (the experiment described in Section 6.3.2 illustrates this); a platform engineer could analyze the performance factors of managed service platforms in relation to activity traces, both of which are described in the same knowledge graph.

### 6.2.1 Semantic Modeling of User Activity

In this section, we delve into activity modeling within the context of Web navigation. We introduce the concepts of "micro-activity" and "macro-activity" and provide implementation details using knowledge graphs and the UCO vocabulary [388]. Figure 6.2 illustrates this implementation by presenting the data model.

**Concept of activity.** In the context of Web navigation, the concept of activity lacks a precise definition, as its interpretation heavily relies on the specific data and observation scale chosen. The understanding of activity varies depending on the field of study, ranging from the microscopic level of molecular interactions to the macroscopic scale of societal dynamics. For example, let us consider the activity of buying a car online. From the highest scale, "buy a car" can be seen as an atomic activity. If we delve deeper and examine the process, we

realize that it is composed of several smaller activities that contribute to the overall objective. Indeed, the activity of buying a car online involves various steps. It starts with researching different car models, comparing prices, and reading reviews. At this lower level, we are still considering activities that are perceptible to humans. However, it is important to note that from the perspective of a Web browser, which handles requests and interactions, those explicit concepts may not be directly observable. In order to adopt the perspective of the Web browser, at first, we will focus on the lowest level of activity representation. If we go back to the previous scenario, a person wishing to buy a car online will navigate to a seller's website. An HTTP connection is established and a user-initiated request from the browser client is sent to the server hosting the website. However, most of the time, the requested document needs some additional resources like scripts, images, videos, or other documents. Those dependencies will lead to a set of sub-requests. From our perspective, this action consists in navigating through a link in one single click (or accessing to the page through a URL), whereas from the Web browser it can be seen as a sequence of requests. In this chapter, we consider this sequence as a so called "Micro-activity" trace. Then, exploring the level above, we could simply consider a trace as a set of requests and interactions. An interaction can be defined as any user-initiated action that has an impact on a webpage or application (click on link, form filling, form submission, etc.). We call such a trace a "Macro-activity".

**Semantical mapping.** Knowledge graphs model the relationships between different concepts and represent knowledge on a semantic level. In this work, we perform a semantic mapping to leverage the advantages of this representation for anomaly detection. This mapping relies on the UCO (Unified Cyber Ontology) vocabulary. UCO is a popular community-developed ontology built around the cyber security domain, covering a wide range of important concepts such as agents, resources, or actions. Since UCO is also used in NORIA-O (Chapter 3), we could establish meaningful connections and access additional contextual information within the knowledge graph.

Firstly, knowledge graphs help to homogenize different data sources and address the challenge of interoperability. While the conventional functioning of Web browsers already relies on established standards and protocols, knowledge graphs can be beneficial when integrating data from sources outside of the Web browser context. Secondly, while a key-value structure used by Web browser provides a simple way to store and retrieve information, it lacks the ability to express semantic connections between the data elements. By leveraging a structured knowledge graph, we can uncover hidden relationships, such as the hosting relationship between a URL and an IP address, and gain a deeper understanding of the interconnectedness of entities. Lastly, the graph structure facilitates exploration of relationships, allowing for the discovery of unexpected connections. By identifying anomalies within a specific part of the graph, it becomes possible to investigate related entities and detect potential issues

more accurately. For example, identifying an anomaly in a particular resource could lead to examining the network topology and identifying other nearby resources that may be affected.

Building upon these ideas, we observe that the UCO vocabulary appears to be well-suited for the semantic representation of activities as it enables the representation of Web navigation at various scales, including action cycles, individual actions, connections, protocols, resources, domains, and IP addresses. For the design of the mapping, the principle is to maximize the reuse of concepts and properties defined in the UCO vocabulary, and to match the fields and values captured at the Web browser level with these concepts and properties whenever their semantics align. Figure 6.2 summarizes the data model in the form of a class diagram, and the corresponding mapping rules in RML syntax [24] are available in open source at https://github.com/Orange-OpenSource/graphameleon. For example, notably in the context of capturing micro-activities, an HTTP request is represented by an entity of the class `ucobs:HTTPConnectionFacet`, and its headers are represented by specific properties such as `ucobs:startTime` and `ucobs:endTime` for timestamps, and `core:tag` for fetch metadata request headers [367]. Since an IP address or URL can be common to multiple requests (e.g. a user repeating the same call to a website, a website with various services hosted on the same server), these elements shall be materialized through the `ucobs:IPAddressFacet` and `ucobs:URLFacet` classes respectively, and cross-references between entities is built through properties such as `ucobs:hasFacet` and `ucobs:host`. For macro-activities, we consider the user interactions (e.g. click on a hyperlink, on a Web browser button) as `ucoact:ObservableAction` class instances, with relations to the above `ucobs:HTTPConnectionFacet` and `ucobs:URLFacet` entities for describing the context in which they occur. Further, we consider the `types:threadNextItem` and `types:threadPreviousItem` properties from UCO for modeling the chronology of activity traces.

### 6.2.2 Data Collection with Graphameleon

A Web browser serves as the primary interface between a user and the Web, thus considering interactions and requests is crucial as it reflects direct and indirect user intent. Table 6.1 summarizes the type of data captured by the Graphameleon Web extension. Capturing requests enables us to extract wealth of meaningful information from the headers such as URLs, associated IP addresses and domains, providing an essential context for anomaly detection. Additionally, we collect fetch metadata request headers [367] which further enriches this context (Figure 6.3). For example, `Sec-Fetch-Site` indicates the relationship between a request initiator's origin and its target's origin, thus providing indirect knowledge about the network topology. Then, we characterize user interactions by capturing events, identifying the relevant DOM elements, and associating them with the corresponding resource URLs, thereby enabling a comprehensive understanding and global identification of individual elements

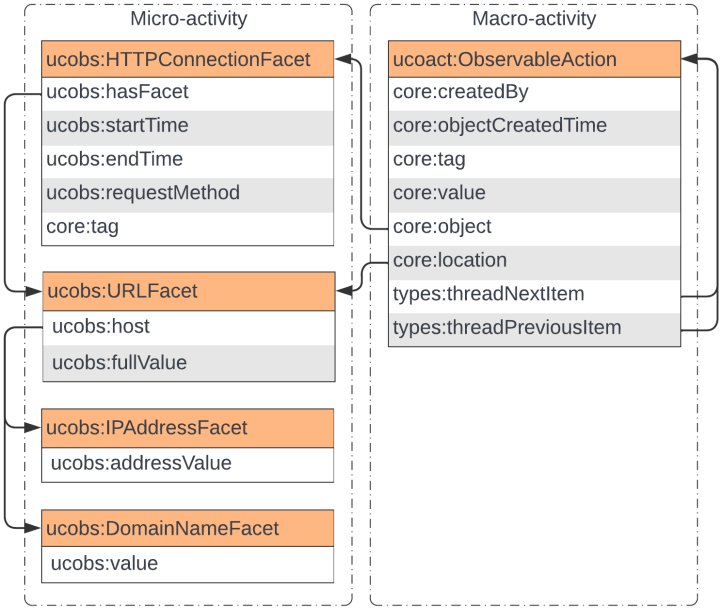



Figure 6.2: Data model for user activities.
The class diagram defines the concepts and properties used for the semantic representation of micro-activities (left) and macro-activities (right) as described in Section 6.2.1. For micro-activities, the presented classes and properties accurately describe a sequence of requests captured at the Web browser level. Macro-activities further enhance the modeling by allowing the description of interactions. The names of concepts and properties used here are defined within the UCO vocabulary, the following namespaces apply: *core* = https://ontology.unifiedcyberontology.org/uco/core#, *ucobs* = https://ontology.unifiedcyberontology.org/uco/observable# and *types* = https://ontology.unifiedcyberontology.org/uco/types#.

within the Web browsing context. Finally, we collect the user-agent header and timestamp values for each capture. The user-agent header provides valuable information about the user's device and browser environment, and serve as a primary characteristic utilized in the user fingerprinting domain for user identification [196]. Moreover, the collection of timestamp allows for precise chronological analysis, facilitating the detection of time-based anomalies.

**Request collection.** Graphameleon applies request listeners to both sending and receiving processes at the background script level. All browser requests are intercepted to retrieve information from the headers. Depending on the collection mode, user-initiated requests are filtered by focusing on the `Sec-Fetch-User` variable. In order to abstract certain contextual elements and avoid excessive diversity in activities for similar cases, we have chosen to tokenize the URLs associated with the requested resources. Thus, all arguments contained in the URLs are replaced by the names of their respective parameters. For example, considering the URLs `https://www.shop.com/?client_id=2313` and `https://www.shop.com/?client_id=346`, regardless of the user initiating the request, it reflects the same behavior. Therefore, it will be tokenized to become `https://www.shop.com/?client_id=-[client_id]`.

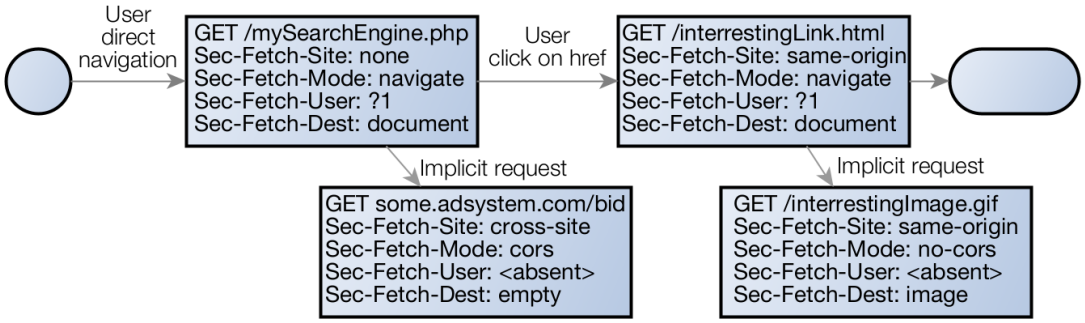



Figure 6.3: Fetch metadata for inferring the user/equipment activity
This sequence diagram illustrates the principle for inferring the user/equipment activity from the four fetch metadata HTTP headers through a fictional example of a Web browsing session where the user logs into a search website and follows a hyperlink. The semantics of fetch metadata summarize as follows: `Sec-Fetch-Site` = relationship between a request initiator's origin and the origin of the requested resource (e.g. same site, cross site); `Sec-Fetch-Mode` = mode of the request (e.g. user navigating between HTML pages *vs* secondary requests to load images and other resources); `Sec-Fetch-User` = only sent for requests initiated by user activation, and its value will always be "`?1`" (e.g. identify whether a navigation request from a document, iframe, etc., was originated by the user); `Sec-Fetch-Dest` = where and how the fetched data will be used for better request handling on the server side (e.g. iframe, video component). The sub-documents of each Web page (implicit requests) are identified based on the absence of value for the `Sec-Fetch-Dest` header.

**Interaction collection.** In order to capture interactions between the user and the browser, content scripts are injected into each of the active tabs. These scripts apply listeners to all interactive elements on the page, such as links, buttons, forms, etc. This approach reduces the impact on the browser performance and prevents capturing unwanted interactions, such as miss-clicks on non-interactive elements. Multiple listeners are applied for various types of events, such as a simple click, a value change or a focus. In order to globally identify interactions, we consider the recorded event type, the element and the URL of the corresponding resource. When an element has an $id$ attribute, its identification is straightforward. However, in most cases, webpage elements lack an identifier, necessitating the use of an alternative method. Thus, we identify an element based on its absolute position within the Document Object Model (DOM) [366] hierarchy. This involves calculating the absolute path of the element from the root of the DOM, resulting in an identifier such as `body > maindiv[2] > div > div > a`. While this method provides a deterministic means to identify elements without an $id$ attribute in a given webpage rendering context, it is important to note that the resulting identifiers in the form of hierarchical paths are often complex and difficult to interpret. Indeed, this technique makes it challenging to understand which element is being referred to solely based on its path identifier, unless a capture of the webpage itself is provided along with the interaction collection for posterior context analysis by looking up for the DOM path. An opportunity here could be to inject $id$s into webpage elements using listeners, but this would not necessarily solve the problem of $id$ stability between browsing sessions for pages with content that changes between each visit.

Table 6.1: Data collected with Graphameleon.
Type of data collected by the Graphameleon Web extension as a function of the capture mode (micro-activity *vs* macro-activity), and grouped by their scope (request *vs* interaction *vs* both).

| Scope | Feature/header name | **Micro** | **Macro** |
|---|---|---|---|
| Request | Method | ✓ | ✓ |
| | URL | ✓ | ✓ |
| | IP | ✓ | ✓ |
| | Domain | ✓ | ✓ |
| | Sec-Fetch-Dest | ✓ | ✓ |
| | Sec-Fetch-Site | ✓ | ✓ |
| | Sec-Fetch-User | ✓ | ✓ |
| | Sec-Fetch-Mode | ✓ | ✓ |
| Interaction | EventType | - | ✓ |
| | Element | - | ✓ |
| | Base URL | - | ✓ |
| Both | User-Agent | ✓ | ✓ |
| | Start time | ✓ | ✓ |
| | End time | ✓ | ✓ |

### 6.2.3 Anomaly Detection with Petri Nets

Three families of anomaly detection techniques are presented in Chapter 5 for analyzing network data represented using a knowledge graph: *model-based design*, where the knowledge graph holds the necessary and sufficient data to infer unwanted situations with information retrieval; *process mining*, including conformance checking tools and Petri nets (P/T nets) representation, for situations tied to a decision-model and bounded in time and space; *statistical learning* with graph embeddings [153] where anomaly models (i.e. the generalizing context of a set of situations) derive from the structure of the knowledge graph.

In this chapter, we focus on the *process mining* approach, considering that data collection using Graphameleon corresponds to relatively well-defined Web navigation sessions in terms of duration and activities: a single user generates an activity trace captured at the Web browser level, which can be directly annotated by the user in terms of activity purpose at the end of the Web navigation session. We posit that activity traces are akin to decision models, as the sequence of user actions during a Web navigation session (such as clicking on a link, using the Web browser's back button, or filling an input box) aims to achieve a specific goal (i.e. the activity purpose) based on the information presented on the webpages. We assume that P/T nets are a suitable representation for analyzing and classifying Web navigation traces due to several reasons:

1. They possess intrinsic explainability through their graphical representation;
2. Decision models associated with P/T nets can generalize to various situations regardless

of the underlying knowledge representation;

3. Decision models can be readily derived from network and service designers' specification documents and implemented as P/T nets using tools like TINA [48].

Using P/T nets enables to leverage anomaly detection techniques commonly referred to as "conformance checking", that is, evaluating the fitness of a trace with respect to a given model or replaying a trace through a model to analyze any inconsistent steps within the activity.

Thereon, we define the following concepts to clarify the notion of anomaly in relation to what is observed and what is expected from the perspective of activities. Firstly, we define an "activity model" as the translation of any activity trace (obtained during the data collection phase) into a P/T net representation using a process mining algorithm. Following this definition, data collections conducted with the Graphameleon Web extension (Section 6.2.2) lead users to establish a catalog of activity models. Next, we define an "activity template" as a universal model of activity represented with P/T nets. A template is established based on either a specification of the expected behavior of the user-Web system for a specific situation or an ideal behavior derived from aggregating and refining multiple activity traces from the catalog of activity models. We remark that it is the responsibility of the user to manage and convert the activity models into templates (analysis, selection, refinement), which falls outside the scope of this thesis work.

Anomaly detection is therefore defined by the comparison of an activity model to an activity template. Hence, assuming a "normal activity template" (e.g. authentication to a webmail is followed by a phase of email browsing), a fitness value below an acceptance threshold is akin to detecting an abnormal situation (Eq. 6.1, with $\eta$ a threshold parameter):

$$abnormal \equiv fit_{\{alignment|replay\}}(model, template) < \eta \tag{6.1}$$

In such case we can trigger an alert but without being able to provide further details about what is abnormal. In practice, we consider that testing against a set of complementary "abnormal activity templates" is necessary in a second stage referred to as the "qualification" phase for classifying the abnormality.

## 6.3 Experiments and Evaluation

This section details the experiments conducted based on the approaches described in Section 6.2. First, we analyze the correlation between the volume of RDF triples generated by Graphameleon and the nature of the visited website (Section 6.3.1). Second, we model and identify three Web navigation scenarios using Graphameleon and P/T nets in a controlled environment (Section 6.3.2). The experiments are conducted using Graphameleon v2.1.0. The

results are presented and discussed in each section. We release the data associated with these experiments at https://github.com/Orange-OpenSource/graphameleon-ds.

### 6.3.1 Website Complexity Clustering

In this initial experiment, we seek to understand to what extent the behavior of a website during a first connection is crucial in creating a usable footprint subsequently for anomaly detection. For this, we focus on the ability to study the complexity of websites in terms of the number and the size of the resources to be loaded. We study this complexity by measuring the number of RDF triples generated by Graphameleon during the initial connection to a website, across a set of websites. Table 6.2 presents the recorded measurements.

To our knowledge, there is currently no study describing well-known website complexity groups (clusters) but from a marketing perspective [350] (e.g. industry sector *vs* average request count per landing page, average page weight bytes, average speed index). Moreover, with over one billion websites as of now [188], existing website classification tools mainly focus on competitive analysis [275]. This highlights the challenge of selecting representative examples from each group. For the experiment, we consider establishing a set of website landing pages for analysis according to three arbitrary complexity categories. This categorization is based on the idea that complexity aligns with the extent of editorial content to be rendered. For each category, we select a subset of three reference websites based on third-party expert opinions:

***One-Page*** where the "Swappa Bottle"[3], "Garden Studio"[4] and "Mark My Images"[5] (MMI) are identified in [214] as the top three best examples of one-page websites to get inspiration from for website design projects;

***Encyclopedia*** where "Encyclopedia Britannica Online"[6] (EBO), "Scholarpedia"[7] and "Encyclopedia.com"[8] are discussed in [190] as the top three alternatives to Wikipedia from the information trustworthiness perspective;

***Content-Heavy*** where "RTI International"[9], "PrintMag"[10] and the "International Women's Media Foundation"[11] (IWMF) are identified in [197] as the top three engaging websites with large amounts of written content while creating an intuitive experience.

[3] https://swappabottle.com/
[4] https://gardenestudio.com.br/
[5] https://www.markmyimages.com/
[6] https://www.britannica.com/
[7] http://www.scholarpedia.org/
[8] https://www.encyclopedia.com/
[9] https://www.rti.org/
[10] https://www.printmag.com/
[11] https://www.iwmf.org/

Next, we proceed with the collection and analysis of navigation traces data for each website landing page as follows:

1. Within a Firefox desktop instance (anti-tracking $\in \{strict, standard\}$), load Graphameleon and enable data capture (collect mode $\in \{micro, macro\}$, output mode = $semantize$);
2. Open a navigation tab and the Network Monitor console[12] (caching = $deactivated$);
3. Navigate to the target website by entering its URL in the Web browser's navigation bar;
4. Cease Graphameleon capture 10 seconds post full page load event detection in the Network Monitor console for consistent runtime of embedded page scripts (i.e. `DOMContentLoaded` event[13]);
5. Save the data to a file (serialization = $Turtle$);
6. Gather the data collection statistics (requests count, responses count, interactions count, vertices count, edges count) from the Graphameleon user interface, and those of the resulting knowledge graph through a set of SPARQL queries on the serialized data (triples count, subjects count, class instances count).

**Results & discussion.** Using this procedure, 27 data collections were performed (three categories × three sites × three data collection setups), with 23 enabling data analysis and four encountering issues (a server-side `SSL_ERROR_NO_CYPHER_OVERLAP` access error for "Swappa Bottle" on both the micro and macro mode, and a Web extension undetermined processing error for "Scholarpedia" in macro mode). Table 6.2 presents the related statistics regarding RDF triples. For macro mode data (CM = M), we observe that the statistics over the RDF triples remain consistent regardless of the visited website. An analysis of the resulting Turtle files also reveals that the RDF data structure adheres to the data model depicted in Figure 6.2. Regarding the micro mode (CM = $\mu$), the counts for a given anti-tracking policy configuration exhibit significant variability within each complexity category. In this line of thought, Table 6.3 focuses on the mean entity count for the `ucobs:HTTPConnectionFacet` (UHC) and `ucobs:IPAddressFacet` (UIP) object classes, and for each scenario. The comparison of the mean entity count values based on the anti-tracking policy ("Std. / Str." column in Table 6.3) reveals an increase in the average number of connections and remote servers accessed when the anti-tracking rules are relaxed, regardless of the complexity level. Based on these measurements, we conclude on the correct functioning of Graphameleon and its suitability for studying initial connection behaviors. While the current proposed complexity categories might not be pertinent for website grouping due to limited sample size and content variability,

[12] https://firefox-source-docs.mozilla.org/devtools-user/network_monitor/
[13] https://developer.mozilla.org/en-US/docs/Web/API/Document/DOMContentLoaded_event

Table 6.2: Statistics for the Website complexity experiment.
Statistics based on the "micro" (CM = $\mu$) and "macro" (CM = M) data collection mode, and as a function of the Web browser's anti-tracking policy. The following abbreviations apply: CM = collection mode, Trk. = anti-tracking policy (strict *vs* standard), TC = Triples count, SC = Subjects count, UOA = *ucobs:DomainNameFacet* entities count, UDN = *ucobs:DomainNameFacet* entities count, UHC = *ucobs:HTTPConnectionFacet* entities count, UIP = *ucobs:IPAddressFacet* entities count, UURL = *ucobs:URLFacet* entities count, n.a. = non applicable.

| Website | CM | Trk. | TC | SC | UDN | UHC | UIP | UURL |
|---|---|---|---|---|---|---|---|---|
| **One-Page** | | | | | | | | |
| Swappa Bottle | $\mu$ | Str. | n.a. | - | - | - | - | - |
| | $\mu$ | Std. | n.a. | - | - | - | - | - |
| | M | Str. | n.a. | - | - | - | - | - |
| Garden Studio | $\mu$ | Str. | 886 | 163 | 5 | 84 | 5 | 69 |
| | $\mu$ | Std. | 985 | 189 | 11 | 89 | 11 | 78 |
| | M | Str. | 21 | 5 | 1 | 1 | 1 | 1 |
| MMI | $\mu$ | Str. | 427 | 81 | 3 | 38 | 3 | 37 |
| | $\mu$ | Std. | 423 | 80 | 3 | 38 | 3 | 36 |
| | M | Str. | 21 | 5 | 1 | 1 | 1 | 1 |
| **Encyclopedia** | | | | | | | | |
| EBO | $\mu$ | Str. | 599 | 122 | 13 | 54 | 13 | 42 |
| | $\mu$ | Std. | 2195 | 472 | 71 | 194 | 70 | 137 |
| | M | Str. | 21 | 5 | 1 | 1 | 1 | 1 |
| Scholarpedia | $\mu$ | Str. | 452 | 111 | 4 | 55 | 4 | 48 |
| | $\mu$ | Std. | 579 | 143 | 11 | 64 | 11 | 57 |
| | M | Str. | n.a. | - | - | - | - | - |
| Encyclopedia | $\mu$ | Str. | 350 | 66 | 2 | 31 | 2 | 31 |
| | $\mu$ | Std. | 1483 | 320 | 44 | 125 | 144 | |
| | M | Str. | 21 | 5 | 1 | 1 | 1 | 1 |
| **Content-Heavy** | | | | | | | | |
| RTI | $\mu$ | Str. | 381 | 76 | 6 | 33 | 6 | 31 |
| | $\mu$ | Std. | 562 | 118 | 14 | 48 | 14 | 42 |
| | M | Str. | 21 | 5 | 1 | 1 | 1 | 1 |
| PrintMag | $\mu$ | Str. | 552 | 111 | 9 | 47 | 8 | 47 |
| | $\mu$ | Std. | 1143 | 234 | 25 | 101 | 24 | 84 |
| | M | Str. | 21 | 5 | 1 | 1 | 1 | 1 |
| IWMF | $\mu$ | Str. | 362 | 72 | 5 | 31 | 5 | 31 |
| | $\mu$ | Std. | 388 | 78 | 6 | 33 | 6 | 6 |
| | M | Str. | 21 | 5 | 1 | 1 | 1 | 1 |

the rise in network exchanges under varied anti-tracking policies provides a basis for future categorization by employed analytics strategies and network topology.

| | Strict | | Standard | | Std. / Str. | |
|---|---|---|---|---|---|---|
| | UHC | UIP | UHC | UIP | UHC | UIP |
| One-Page | 61.0 | 4.0 | 63.5 | 7.0 | 1.04 | 1.8 |
| Encyclopedia | 46.7 | 6.3 | 127.7 | 41.7 | **2.73** | **6.6** |
| Content-Heavy | 37.0 | 6.3 | 60.7 | 14.7 | 1.64 | 2.3 |

Table 6.3: Average number of entities in micro mode.
Comparison of the average UHC and UIP entities count from Table 6.2 as a function of the complexity level and of the anti-tracking policy. Only "Garden Studio" and "MMI" values considered for "One-Page" category. The following abbreviations apply: UHC = *ucobs:HTTPConnectionFacet* entities count, UIP = *ucobs:IPAddressFacet* entities count.

### 6.3.2 Navigation Trace Classification

In this second experiment, our goal is to classify Web navigation traces as either normal or abnormal behaviors. Using macro-activity modeling (Section 6.2.1) and Petri nets (Section 6.2.3), we analyze the following three scenarios (outlined in Figure 6.4) and report on the ability to identify anomalies:

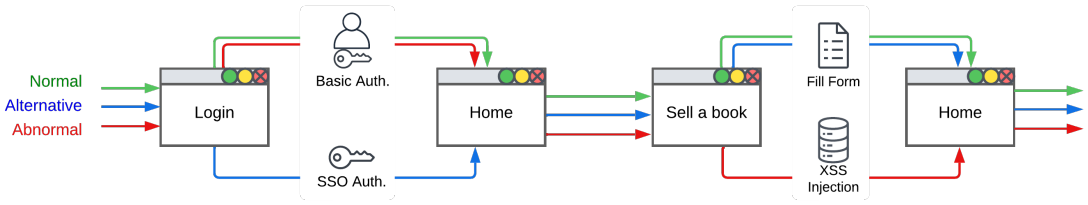



Figure 6.4: Overview of navigation scenarios for the navigation trace classification experiment.
This flow diagram summarizes the three navigation scenarios implemented in Section 6.3.2. From left to right, the key steps correspond to *1)* user authentication on the simulated online bookstore website, *2)* navigation to the homepage, *3)* purchasing a book, *4)* returning to the homepage. The authentication step has two options: basic authentication or SSO authentication. The purchase step alternates between normal input of the author's name in a form and input with XSS injection. The arrows correspond to the sequence of steps for each scenario: Normal = basic authentication + fill form; Alernative = SSO authentication + fill form; Abnormal = basic authentication + XSS injection.

**Base scenario (normal behavior):** a user accesses the website, logs in to their account using their username and password, navigates to the "Sell a Book" page, fills out the form, and then returns to the homepage where they find their book.

**Alternative scenario (different behavior):** a user accesses the website, logs in to their account using Single Sign-On (SSO), navigates to the "Sell a Book" page, fills out the form, and then returns to the homepage where they find their book.

**XSS attack scenario (abnormal behavior):** an attacker accesses the website, logs in to their account using their username and password, navigates to the "Sell a Book" page, and performs a code injection in the "Author" field. Finally, they return to the homepage where the injected script is executed.

To maintain complete control over the experiment, we use a simulated online bookstore website which brings multiple benefits. Firstly, it allows intentional exposure of the website to various security vulnerabilities. In this experiment, we introduce a common form of attack, an XSS vulnerability. Secondly, before the study, we can thoroughly label each webpage element, enhancing the interpretability of collected data.

We proceed with the collection and analysis of navigation traces data for each scenario as follows:

1. Within a Firefox desktop instance, load Graphameleon and enable data capture (collect mode = $macro$, output mode = $semantize$);
2. Open a navigation tab and browse the simulated online bookstore website according to the navigation scenario;
3. Cease Graphameleon capture and save the data to a file (serialization = $Turtle$);
4. Gather the data collection statistics (requests count, responses count, interactions count, vertices count, edges count) from the Graphameleon user interface;
5. Compute the activity model from the saved trace using the PM4PY Process Mining library [11] (process discovery $\in \{Inductive, Alpha, Log\text{-}Skeleton, Heuristic, AlphaPlus\}$);
6. Compute the fitness of the activity model against the activity template using the PM4PY Process Mining library (process mining $\in \{TokenBasedReplay, Alignment\}$).

The base scenario, which corresponds to the behavior we define as "normal," is used to create our reference model (activity template) using the PM4PY Process Mining library[14].

**Evaluation & discussion.** Using this procedure, three data collections were performed. Table 6.4 presents the related statistics regarding the resulting navigation graph, and Table 6.5 compares the results of different fitness evaluation techniques on the "normal" activity model. From a graph statistics perspective and with reference to the "base" scenario, we observe that the "alternative" scenario involves fewer interactions but more network transactions. This aligns with the fact that the user has only one button to click for authentication, and authentication is delegated to various external entities as a service. For the "XSS attack" scenario, the number of interactions remains the same, but the number of requests increases by one. This corresponds to the authentication path aligning with the baseline scenario, yet for the additional query involved by the SQL injection step. Still for this same scenario, we notice a slight variation in the fitness scores (average of 98% fitness, calculated with the "normal"

[14] https://pm4py.fit.fraunhofer.de/

activity model as reference), which also corresponds to the single additional request caused by the SQL injection step. We further observe that this additional query is easily identifiable through sequence alignment using the developed vocabulary applied at the level of the Web extension to standardize the interpretation of the traces.

Table 6.4: Data collection statistics for the navigation trace classification experiment.
Statistics in terms of the number of network requests, user interactions with the Web browser, nodes and edges of the resulting navigation graph, as reported by the Graphameleon user interface for the navigation scenarios defined in Section 6.3.2.

| | **Base** | **Alternative** | **XSS Attack** |
|---|---|---|---|
| Request | 10 | 13 | 11 |
| Interaction | 18 | 14 | 18 |
| Vertice | 263 | 283 | 277 |
| Edge | 404 | 431 | 426 |

Table 6.5: Fitness scores evaluated on normal activity model.
Comparison of fitness scores from deviant activity traces on the normal activity model. Different conformance checking and process comparison techniques are used to provide fitness indicators. Token-Based and Alignment checking methods require model discovery. Alpha/Alpha+, Inductive and Heuristic miner algorithms are used to generate normal activity model. The Log Skeleton technique provides fitness scores directly using traces.

| | | **Alternative** | **XSS Attack** |
|---|---|---|---|
| Token-Based | Alpha | 0.886 | 0.968 |
| | Alpha+ | 0.890 | 0.969 |
| | Inductive | 0.923 | 1.000 |
| | Heuristic | 0.923 | 1.000 |
| Alignement | Alpha | - | - |
| | Alpha+ | - | - |
| | Inductive | 0.718 | 0.976 |
| | Heuristic | 0.718 | 0.976 |
| Log Skeleton | | 0.684 | 0.999 |

Therefore, although our approach provides a standardized and interpretable representation of navigation traces, we observe that its direct use is not suitable for anomaly detection when micro-changes occur in comparison to an activity template (i.e. when differentiating elements to qualify deviations are relatively rare within the sequence). Similarly, we remark that, while discovery algorithms typically use multiple trace samples to generate a generalized model of activity, we considered a single-trace perfect model in our case. It is important to recognize that a real-world model of normal behavior in such scenario is much more complex. For instance, when completing a form, the conventional reading order is usually followed. Yet, due to cognitive biases, a user might complete it in a different sequence while still staying within the bounds of actual normal behavior.

Finally, reflecting on data collection and semantic processing, we find that there is minimal lexical compression of navigation trace data due to consistent formatting (e.g. the request URL is consistently located using the "`url`" header). However, this compression pertains more

to interaction semantics. Indeed, one challenge in aligning activity models stems from the lack of a reliable method to identify HTML elements (especially when lacking an explicit $id$) across browsers, sessions, and users. This challenge becomes apparent when the DOM of page content changes with each site visit, especially when dynamic ad insertions occur.

## 6.4 Conclusion and Future Work

In this chapter, we sought ways to analyze Web navigation activity traces in order to characterize user and application activities, particularly for the purpose of detecting cyberattacks. Expanding on DynaGraph [203], we hypothesized that knowledge graphs would bring structure to data collected at the Web browser level during a Web navigation session for further analysis through activity modeling with Petri nets and conformance checking techniques. To test our approach, we developed the concepts of micro-activity and macro-activity on the basis of the UCO vocabulary [388] for the semantic representation of user activities. We also developed Graphameleon, an open source Web extension available at https://github.com/Orange-OpenSource/graphameleon for live data collection and semantization of Web navigation traces at the Web browser level, and analyzed the activity traces collected by this component using a two-part experimental plan. Firstly, with the website complexity clustering experiment, we showed that the rise in network exchanges under varied anti-tracking policies provides a basis for future categorization by employed analytics strategies and network topology. Secondly, with the navigation trace classification experiment, we showed the limits of the conformance checking technique for anomaly detection when micro-changes occur in comparison to an activity template. We also highlighted the challenge in aligning activity models from the lack of a reliable method to identify HTML elements across browsers, sessions, and users.

Building upon the Graphameleon Web extension component and the utilization of knowledge graphs, future work will focus on Web cartography, behavior analytics and anomaly detection. Regarding Graphameleon, specific technical aspects require deeper development, such as graph streaming, activity labeling through the user interface, and handling multiple Web navigation sessions simultaneously. Regarding conformance checking, there are several options to reduce the sensitivity of our approach. One option is to divide the reference trace into smaller sub-models, thus enhancing the fitness score's variation in case of non-conformity. Using sub-models, another option is to employ action-qualifying patterns to confirm an anomaly and pinpoint the pattern through sequence alignment (e.g. a pattern for SQL injection rather than normal input). A third approach could involve amplifying the alignment mismatch weight in the fitness score using context from the knowledge graph (e.g. infrequent IP address in an SQL injection request, impossible network hop due to the lack of connectivity, a same user connected in two places simultaneously). The aim is to give

these weights enough significance to mask minor variations due to “noise” in comparison to variations caused by genuine errors. Finally, regarding knowledge graphs, we also aim to incorporate activity models as part of anomaly context data for decision support applications using graph embeddings as proposed in Chapter 7, where P/T nets can be serialized into RDF graphs with suitable vocabularies [97, 186].

# Chapter 7

# Capturing and Categorizing Incident Contexts with Graph embeddings

## 7.1 Introduction

The complexity of ICT systems, entails uncertainty in causal reasoning for incident management (Chapter 1 & Chapter 5). Mastering this complexity has been a long-standing effort by the industry and academic communities, including developing NMS/SIEM DSSs to assist operators with incident management (Chapter 2), and logical and statistical modeling of the network dynamics [202, 358, 130, 320, 217, 76]. However, there is still a step to be taken to learn explicit anomaly models from incomplete knowledge of ICT systems and to use these models outside the initial learning context. With such an approach, we could consider assisting NetOps/SecOps teams in decision-making without prior knowledge of the characteristics of a given system, including its multiple stacks of configuration and interactions with other external systems; thereby going beyond the sole local understanding of a subsystem's state and observable artifacts of a situation.

To tackle this challenge, we propose to better capture the context of labeled anomalies through a multi-faceted knowledge graph and to use it to classify incident types with *statistical learning* techniques. More precisely, we make use of an RDF knowledge graph structured by the NORIA-O ontology (Chapter 3 & Chapter 4) and we explore how graph embeddings [312, 153] provide a suitable representation for categorizing trouble tickets. We notably introduce a method for capturing the context of anomalies and examine how it can partially correspond to logical formulations. This analysis includes exploring whether logical formulations could suffice and how statistical approaches can compensate for their limitations. Overall, these contributions lead to the development of a graph embeddings-based classification method for categorizing trouble tickets, and the identification of SPARQL query patterns for anomaly detection based on a qualitative analysis of trouble tickets.

The remainder of this chapter is organized as follows. Section 7.2 details our approach that makes use of the aforementioned knowledge graph. Section 7.3 describes our experiments and

evaluations. Finally, Section 7.4 concludes this chapter by providing a summary of the work that has been done and offering some preliminary conclusions regarding the opportunities and potential future developments associated with it.

## 7.2 Approach

In this section, we present two approaches to explicitly represent anomaly models. Firstly, we approach decision support in Section 7.2.1 as a classification problem and develop a model to predict the category of a trouble ticket using graph embeddings. Secondly, we assume that the anomaly models learned by the classifier have a correspondence, possibly partial, with a logical representation. We analyze trouble tickets qualitatively in Section 7.2.2 and highlight corresponding SPARQL queries for comparison with the classifier. We intentionally set aside the process mining approach discussed in Section 5.3.2 and Chapter 6 because it only captures local processes and therefore misses out on the need for learning from a larger context that is enabled by graph embeddings. We present the related experiments and results in Section 7.3.

### 7.2.1 Multiclass Classifier with Graph Embeddings

Understanding a network-impacting incident based only on network monitoring functions is an ill-posed problem (Section 5.2). We propose that using graph embeddings could help solve the inverse problem. Indeed, we assume that a trouble ticket represents an approximate dual of the anomaly structure in the network's parameter space. Thus, we can create an anomaly model by aggregating the graph representations of each incident in the network's parameter space that have the same characteristics (e.g. problem category, probable cause). In what follows, we focus on the task of predicting the category of a trouble ticket by building a multiclass classifier upon the context of trouble ticket entities (Figure 7.1).

For knowledge representation of ICT systems, we leverage on NORIA-O. It is used as the main data model for the experiments described in this chapter as it can model complex ICT system situations and serve as a basis for anomaly detection and root cause analysis.

From a NORIA-O vocabulary perspective, building the classifier involves learning the relational model on events near the resource that is reported in a given incident (i.e. walking the graph

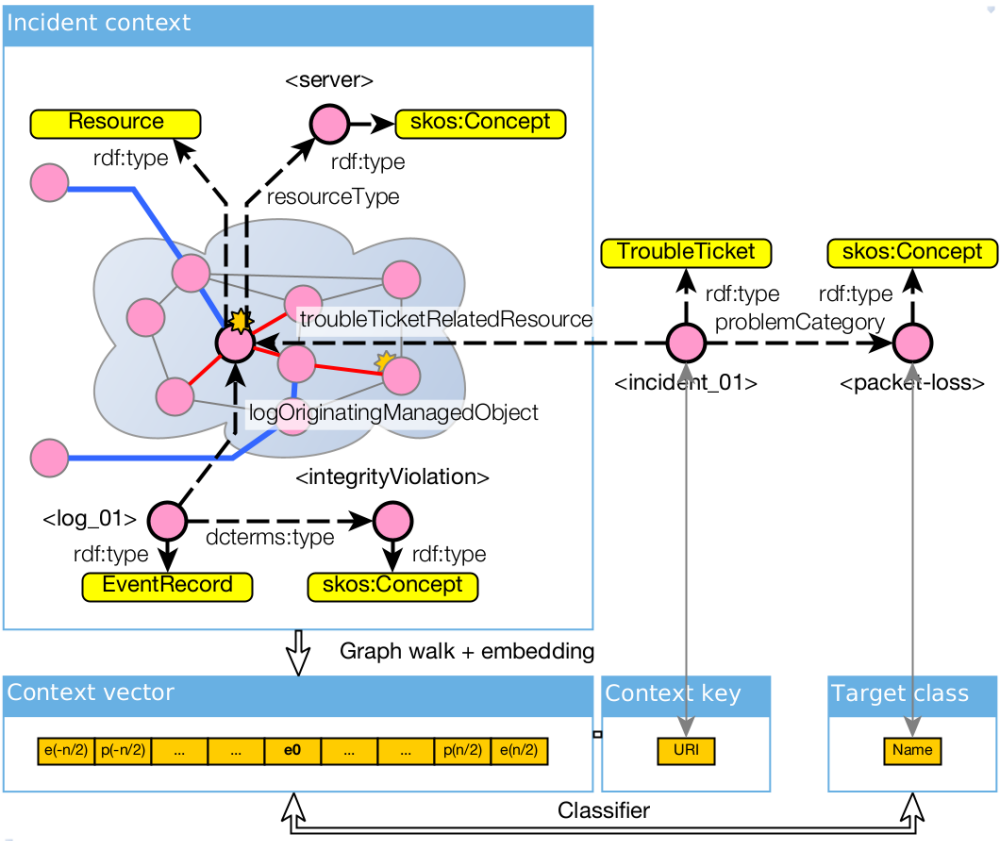



Figure 7.1: Learning an incident context with graph embeddings
This diagram illustrates the process of building a multiclass classifier upon the context of trouble ticket entities by highlighting the relationship between an incident context (i.e. a subgraph centered around a `Resource` entity concerned by a given `TroubleTicket`), the context vector (i.e. the embedding of the subgraph into a vector space), and the classification model that is trained on the relationship of the context vector to the problem class defined at the `TroubleTicket` entity level. The URI [50] of the `TroubleTicket` entity is used as the context key (i.e. the identifier of the vector in the embedding vector database) in order to retrieve the context vector from the `TroubleTicket` entity identifier, and even to search for one or multiple `TroubleTicket` entities from a context vector. `noria` is the default namespace for concepts and relationships. Other namespaces: `rdf` = http://www.w3.org/1999/02/22-rdf-syntax-ns#, `skos` = http://www.w3.org/2004/02/skos/core#. The diagram partly conforms to the Graffoo graphical notation [330].

and computing embeddings), and then linking the relational model to a category (Eq. 7.1):

$$\begin{aligned} &\Big\{EventRecord.logOriginatingAgent\big(Resource(i)\big) \\ &\qquad .logOriginatingAgent\big(Resource(i)_{neighbors}\big)\Big\} \\ \sim TroubleTicket.&\Big(relatedResource\big(Resource(i)\big) \\ &\qquad \sqcap problemCategory\Big) \end{aligned} \tag{7.1}$$

where `noria:problemCategory` is an attribute describing the final nature or technical impact of a `noria:TroubleTicket` entity. This attribute is part of the key fields used in the trouble ticket system to fully qualify incident resolution upon closure[1]. It is an `owl:ObjectProperty`

[1] Other fields are `noria:problemResponsibility`, `noria:troubleTicketCause` and `pep:forProcedure`.

with values from the `kos/TroubleTicket/trouble-category` controlled vocabulary, a SKOS Concept Scheme defining nine concepts.

We make use of the *pyRDF2Vec* library [61] and the *scikit-learn* library [279]. *pyRDF2Vec*[2] is a Python implementation of the RDF2Vec algorithm [283], which captures the context of RDF graph nodes (properties and neighboring nodes) as latent feature vectors and is inspired by Node2Vec and Word2Vec. The data processing steps for our approach are the following:

1. We create graph walks (i.e. sequence of vertices) by traversing the RDF dataset with `noria:TroubleTicket` entities as the starting points, and filtering out the `noria:problemCategory` property as it is used for classification;
2. We compute embeddings for the `noria:TroubleTicket` entities by training a Word2Vec model on the graph walks;
3. We train a random forest classifier with the embeddings as training input samples, and `noria:problemCategory` ranged entities as target values.

The choice of the random forest classifier is based on its intrinsic interpretability as it is using decision trees.

A potential interest with this approach is that for any new trouble ticket matching a certain set of characteristics, projecting the anomaly model onto the rich graph representation of the network would be equivalent to circumscribe the search space in the network's parameter space (i.e. assets and features of interest to further scrutinize for anomalies). This meets the circumbscribe assets need raised for network and security operations (use case #1 in Table 5.1). With network administrators' commands also logged as `noria:EventRecord` entities, projecting the model could also recommend remediation and repair actions to perform (Eq. 7.2):

$$TroubleTicket_{similar} \times TroubleTicket_{actual} \rightarrow \{Resource, Action\}_{actual} \tag{7.2}$$

### 7.2.2 Model-Based Anomaly Detection

In addition to the statistical learning approach that has been presented above, we further develop in this section the anomaly detection retrieval approach from Chapter 5.

A limited set of queries is available beforehand due to the lack of comprehensive knowledge about the situations to be detected. We posit that similar queries apply for similar trouble tickets. Thereon, we use the following analysis scheme on a user-provided dataset to identify the form of the queries and gain additional insight on similarities:

[2] See INK [59] for an alternative to pyRDF2Vec.

1. For each trouble ticket, display it and provide an expert analysis about the SPARQL query to implement for anomaly detection;
2. Retrieve the $k$ most similar trouble tickets based on their embeddings using a cosine distance;
3. Display these $k$ tickets with all known attributes (e.g. creation date, services or resources involved) and provide an expert analysis as whether we could consider these tickets indeed related to the origin ticket and according to which dimension.

We analyze the reciprocal alignment between trouble tickets grouped according to the `noria:problemCategory` attribute and grouped according to the clusters obtained from a similarity graph on the embeddings (Algorithm 1).

**Algorithm 1** Similarity graph of entity embeddings

$E \leftarrow$ embeddings of entities
$k \leftarrow$ number of entities for similarity
$SG \leftarrow \emptyset$ ▷ Empty similarity graph (SG)
**for all** $e \in E$ **do**
  $SG \leftarrow e$ ▷ Add vertex
  $SIM \leftarrow MostSimilar_{cosine}(e, E, k)$ ▷ Similarity on embeddings
  **for all** $e_{sim} \in SIM$ **do**
    $SG \leftarrow e_{sim}$ ▷ Add vertex
    $SG \leftarrow (e, e_{sim})$ ▷ Add edge
  **end for**
**end for**
$SG \leftarrow P_{\text{Louvain modularity}}(SG)$ ▷ Vertex partitioning
$SG \leftarrow R_{\text{Centrality}}(SG)$ ▷ Vertex ranking

The potential interest for this approach is twofold. First, it brings to enumerate query patterns for anomaly models, including for trouble tickets describing complex situations like a network outage impacting multiple applications. Second, it enables to explore how the embedding space is correlated with the semantic similarity [144] based on the logical form of the anomaly model (i.e. the SPARQL query).

## 7.3 Experiments and Results

This section details the experiments conducted based on the approaches described in Section 7.2 and analyzes their results.

### 7.3.1 Dataset

For our experiments, we use a RDF dataset generated using the NORIA knowledge graph construction platform (Chapter 4). The input data of the platform is based on 15 tables distributed across 10 sources such as: trouble tickets, change requests, logs & alarms monitoring,

network topology, applications, teams, users, etc. The size of the resulting RDF dataset is approximately 4 million triples for 400K entities, including streamed events spanning over 111 days. The Table 7.1 provides an overview of the dataset.[3]

Table 7.1: Dataset overview.
Class names are provided in the $\langle prefix \rangle : \langle Class \rangle$ form. Percentage is the ratio of the entities count for a given class over the total number of entities.

| Class name | Entity count | Percentage |
|---|---|---|
| noria:Resource | 236'318 | 54.535 |
| noria:EventRecord | 89'606 | 20.678 |
| foaf:Person | 26'879 | 6.203 |
| noria:CorporateUserIdentifier | 26'879 | 6.203 |
| noria:Locus | 22'662 | 5.230 |
| noria:ApplicationModule | 9'314 | 2.149 |
| noria:ProductModel | 4'306 | 0.994 |
| org:OrganizationalUnit | 3'677 | 0.849 |
| noria:Application | 3'170 | 0.732 |
| bot:Storey | 2'869 | 0.662 |
| noria:Room | 2'869 | 0.662 |
| bot:Building | 1'656 | 0.382 |
| bot:Site | 1'374 | 0.317 |
| org:Organization | 366 | 0.084 |
| noria:NetworkInterface | 346 | 0.080 |
| prov:Activity | 324 | 0.075 |
| noria:ChangeRequest | 190 | 0.044 |
| noria:NetworkLink | 181 | 0.042 |
| noria:TroubleTicket | 150 | 0.035 |
| noria:TroubleTicketNote | 110 | 0.025 |
| noria:AnomalyMode | 75 | 0.017 |
| pep:Procedure | 9 | 0.002 |
| TOTAL | 433'330 | |

### 7.3.2 Multiclass Classifier with Graph Embeddings

**Computing embeddings and training the model.** Prior to computing embeddings, we generate nine sets of walks with a random walk strategy [60], walk depth $WD \in \{4,8,10\}$ (vertices) and walk counts $WC \in \{10,20,30\}$ (per entity). Then, the Word2Vec training for embeddings holds on ten epochs for each set of walks. These sets of walks are referred to as *WDxx/WCyy* in the Table 7.3.

We use the random forest algorithm as the classifier. The input values of the model are the

[3] Due to confidentiality, this dataset is not made public.

embeddings. Target classes are values from the `noria:troubleTicketCategory`[4] property. The Table 7.2 presents the possible values and their distribution in the dataset. We use a stratified fixed-split strategy to build the training dataset while taking into account the target class imbalance, with a proportion of 25% of the dataset to include in the test split. Tuning the model's hyper-parameters relies on a grid-search heuristic, with parameters: the number of trees ∈ {10, 20, 30, 50, 70, 100}, split criterion ∈ {gini, entropy}, maximum depth of the trees ∈ {3, 5, 10, pure leaves}, and feature selection weight ∈ $\{sqrt(\#features), log_2(\#features), (\#features)\}$. We use a weighted F1 score for model selection.

**Evaluation & discussion.** Table 7.3 reports on the classifier performance with respect to the weighted F1 score and the model parameters for each *WDxx/WCyy* set of walks. The WD08-WC30 shows the best performance for the classification task with a 0.81 weighted F1 score. Table 7.2 reports on the per class weighted F1 score for the best model (WD08-WC30) in order to discover if some classes are harder to predict than others, regardless of their frequency.

Table 7.2: Target class distribution
Class labels relate to the NORIA-O `skos:Concept skos:prefLabel` for the `noria:troubleTicketCategory` property. Percentage is the ratio of the `noria:TroubleTicket` entities count for a given class over the total number of entities. F1 weighted and Support are classification performances for the WD08-WC30 random forest model over the test data.

| **Class label** | **Entities** | **Percentage** | **F1 weight.** | **Support** |
|---|---|---|---|---|
| Interrupted service | 77 | 55.8 | 0.97 | 19 |
| Degraded QoS | 22 | 15.9 | 0.75 | 5 |
| No service impact | 22 | 15.9 | 0.62 | 6 |
| Defect to be qualified | 13 | 9.4 | 0.57 | 3 |
| Equipment failure | 4 | 2.9 | 0.00 | 1 |
| TOTAL | 138 | 100.0 | 0.81 | 34 |

Table 7.3: Classifier performance
F1 weighted score and random forest best model parameters as a function of graph walks parameters. WC = Walk Count, WD = Walk Depth, model parameters in the form *<criterion>-<max depth>-<max features>-<n estimators>*.

| | WC10 | WC20 | WC30 |
|---|---|---|---|
| WD04 | 0.64<br>gini-05-SQRT-030 | 0.59<br>gini-05-SQRT-020 | 0.73<br>gini-05-SQRT-030 |
| WD08 | 0.49<br>gini-05-SQRT-100 | 0.75<br>gini-05-SQRT-050 | **0.81**<br>gini-05-SQRT-020 |
| WD10 | 0.52<br>gini-05-SQRT-020 | 0.60<br>gini-05-SQRT-020 | 0.76<br>gini-05-SQRT-020 |

We observe from Table 7.3 that the model performance globally increases with the walk counts ($WC$) parameter. The performance does not appear to increase proportionally to the walk depth ($WD$) parameter. However, the F1 score reaches a peak at $WD = 8$. In-depth analysis of

[4] https://w3id.org/noria/ontology/troubleTicketCategory

the dataset to better understand the phenomenon shows that the available context for trouble ticket entities is not systematically consistent. For example, some `noria:TroubleTicket` entities refer to `noria:Resource` entities out of the scope of the knowledge graph construction process, hence the context from the network neighborhood is absent from the embeddings. Similarly, the time frame of some `noria:EventRecord` entities (e.g. alarms, device logs) does not overlap with the creation date of the `noria:TroubleTicket`. We also observe some non-standard values for the `noria:troubleTicketCategory` (i.e. values absent from the NORIA-O controlled vocabulary), hence the $150 - 138 = 12$ delta between the number of `noria:TroubleTicket` from Table 7.1 and Table 7.2.

Overall, we conclude that the classifier works relatively well but that the dataset is too small (for some classes in particular) and inconsistent for generalizing and tackling the circumbscribe assets need (use case #1 in Table 5.1). The typical approach to overcome this issue is to improve the knowledge graph construction stage with broader data sources and data quality assessment.

### 7.3.3 Model-Based Anomaly Detection

**Qualitative analysis of trouble tickets.** To identify query patterns for anomaly models, we first retrieve all attributes values associated with `noria:TroubleTicket` entities from the RDF dataset with a SPARQL query. Next, we employ the model-based anomaly detection analysis scheme developed in Section 7.2. In a second step, we compute the similarity graphs (Algorithm 1) over the WD08-WC30 embeddings with parameter $k \in \{3,4,5\}$, and then compare the overlap of query patterns with the partitions resulting from the Louvain community detection algorithm. We use the Szymkiewicz-Simpson coefficient for analyzing the overlap.

**Results & discussion.** From the qualitative analysis step, we identified 12 query patterns over 139 trouble ticket entities. The details of these patterns are detailed below with indications on the involved classes and properties and examples of corresponding situations[5]. Table 7.4 reports on the overall distribution of the patterns, and how they were captured by the community detection algorithm for the $k = 3$ similarity graph. Running the Algorithm 1 with $k \in \{3,4,5\}$ led to generate $|P(SG,k)| = \{7,6,5\}$ partitions respectively.

We observe that a significantly lower number of patterns emerge from the dataset compared to the number of tickets considered ($12/139 \simeq 0.09$ reduction factor). Furthermore, it appears that some patterns, such as “AlarmState” and “HeartBeat”, can capture diverse situations while remaining very specific by using restrictions on the objects and values of properties. This provides valuable insights on the detection and implementation of new patterns, in

[5] See https://w3id.org/noria/dataset/ for query examples.

Table 7.4: Retrieval patterns distribution and overlap coefficients
"C&O" stands for Count & average Overlap coefficient. Columns $[C0, \ldots, C6, None]$ report on the pattern count and overlap coefficient for the WD08-WC30 / $k = 3$ similarity graph. *None* stands for entities that were rejected by the Algorithm 1 due to syntax issues.

| **Pattern name** | C0 | C1 | C2 | C3 | C4 | C5 | C6 | None | C&O |
|---|---|---|---|---|---|---|---|---|---|
| **AlarmState** | 2 | 2 | 1 | 1 | 25 | 15 | 3 | | **49** |
| | 0.13 | 0.13 | 0.06 | 0.07 | 0.96 | 0.88 | 0.14 | 0.00 | 0.30 |
| **AuthError** | 1 | | 4 | | | | 2 | 2 | **9** |
| | 0.11 | 0.00 | 0.44 | 0.00 | 0.00 | 0.00 | 0.22 | 0.22 | 0.13 |
| **CoFailure** | 3 | | | | | | | | **3** |
| | 1.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.13 |
| **Complex** | 7 | | 6 | 1 | | | | 1 | **15** |
| | 0.47 | 0.00 | 0.40 | 0.07 | 0.00 | 0.00 | 0.00 | 0.08 | 0.13 |
| **Debug** | | 1 | 1 | 2 | | | | 2 | **6** |
| | 0.00 | 0.17 | 0.17 | 0.33 | 0.00 | 0.00 | 0.00 | 0.33 | 0.13 |
| **ErroneousRes.** | | | 2 | 6 | | 1 | | | **9** |
| | 0.00 | 0.00 | 0.22 | 0.67 | 0.00 | 0.11 | 0.00 | 0.00 | 0.13 |
| **HeartBeat** | | 11 | | | | | 2 | | **13** |
| | 0.00 | 0.85 | 0.00 | 0.00 | 0.00 | 0.00 | 0.15 | 0.00 | 0.13 |
| **Overbilling** | | | | | | | | | **6** |
| | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| **RecurringFai.** | | 1 | | | | | | | **1** |
| | 0.00 | 1.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.13 |
| **RequestForIn.** | 1 | | 1 | | 1 | 1 | 5 | 8 | **17** |
| | 0.06 | 0.00 | 0.06 | 0.00 | 0.06 | 0.06 | 0.29 | 0.62 | 0.14 |
| **RiskPreventi.** | 2 | | 1 | 4 | | | 10 | | **17** |
| | 0.13 | 0.00 | 0.06 | 0.29 | 0.00 | 0.00 | 0.59 | 0.00 | 0.13 |
| **RMA** | | | | | | | | | **10** |
| | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| C&O | 16 | 15 | 16 | 14 | 26 | 17 | 22 | 13 | **139** |
| | 0.16 | 0.18 | 0.12 | 0.12 | 0.09 | 0.09 | 0.12 | 0.10 | 0.12 |

the perspective of an increase in the size of the dataset. However, as discussed in Section 7.3.2, the inconsistency of the data prevents us from directly validating the queries and their relevance on the dataset. As a consequence of this, we are currently not able to establish a correspondence between the patterns, the incident categories reported by the classifier, and the relevant anomaly models. Despite this, we can observe from Table 7.4 a connection that, although not entirely clear, suggests that there might be $n$-to-1 or $n$-to-$m$ relationships between the patterns and the trouble ticket categories.

**AlarmState:** Alarm state w.r.t. `noria:EventRecord.*` and (`noria:Resource` or `noria:Service`). *Examples:* Service disruption on Optical Network Terminal (ONT). Unable to access http://example.org

**AuthError:** User role conformance w.r.t. `noria:EventRecord.type()` and

`noria:EventRecord.logOriginatingAgent()` and `org:OrganizationalUnit`. *Examples:* Authentication error. User does not have access to the 'xxx' role. Please check my rights.

**CoFailure:** Alarm state w.r.t. `noria:TroubleTicket.troubleTicketRelatedResource()` and `seas:connectedTo` and `noria:EventRecord.logOriginatingManagedObject()` and `noria:EventRecord.type()`. *Example:* Co-occurring alarm in a network device neighborhood and creation of a parent/child relationship between trouble tickets for SLA tracking.

**Complex:** Requires further expertise for providing a pattern.

**Debug:** Non-relevant trouble ticket entity, present for debugging purposes of the ticketing system.

**ErroneousResourceInOperationPlan:** Non-existing resource in operation plan with respect to `noria:EventRecord.type()`. *Example:* The processing flow references a resource that does not exist.

**HeartBeat:** Value conformance and event frequency w.r.t. `noria:EventRecord.type()` and `noria:EventRecord.alarmMonitoredAttribute()` and `noria:EventRecord.logText()`. *Examples:* The number of failed calls has increased significantly. No response to SNMP polling and Ping. Agent not running or cannot communicate. Extreme slowness or even unavailability of the service when opening and closing documents on the platform.

**Overbilling:** Value conformance w.r.t. `noria:EventRecord.logOriginatingManagedObject()` and `noria:EventRecord.type()` and `noria:EventRecord.alarmMonitoredAttribute()` and `noria:EventRecord.logText()` and `foaf:Person` and `org:OrganizationalUnit`.

**RecurringFailure:** Repeated situation w.r.t. `noria:TroubleTicket.troubleTicketRelatedResource()` and `noria:TroubleTicket.problemCategory()`. *Example:* Repeated occurrence of the same type of failure on a device within a short period of time.

**RequestForIntervention:** Resource or service w.r.t. `noria:ChangeRequest.changeRequestPlannedStartTime()` and `noria:ChangeRequest.changeRequestStatusCurrent()`. *Examples:* Please decommission the 'xxx' system. The Customer is calling about the Request For Change (RFC) status.

**RiskPreventionNotification:** Presence of en event w.r.t. `noria:Resource` and `noria:OperationPlan`. *Example:* Automated deployment flow triggered on resource.

**RMA:** Alarm type w.r.t. `noria:EventRecord.*` and `noria:Resource.resourceProductModel()`. *Example:* Return Merchandise Authorization (RMA) for redundant Power Supply Unit (PSU).

## 7.4 Conclusion and Future Work

In this chapter, we aimed to simplify incident management activities related to broad scale ICT systems, using knowledge graphs and learning an explicit representation of the context of each incident. We notably tackle the emblematic "common cause failures" and "alarm

spreading phenomenon" cases since they require considering simultaneous situations that are seemingly unrelated as a whole.

Providing knowledge about the potential relationships between events and incidents occurring in networks is a way to simplify the diagnostic phase. Based on this idea, we firstly hypothesized that relating situations requires a common language to describe them, both in their variety and in the context that groups them together. We posit that RDF knowledge graphs are adequate knowledge representation formalism as they bring an abstraction level for standard interpretation and logical reasoning over heterogeneous data. We also consider that learning the relational structure for each type of incident, to a greater or lesser extent, would allow us to gain an explicit understanding of the complex phenomena that occur in network operations.

We developed a dual statistical learning/model-based approach for modeling anomalies to overcome the lack of exhaustive knowledge of the situations to be addressed. The statistical learning approach led to the development of a graph embeddings-based classification method for categorizing incident tickets using a random forest model. For our experiments, we used a RDF dataset generated using the NORIA knowledge graph construction platform (Chapter 4) and making use of NORIA-O (Chapter 3). Based on our evaluation of the classifier, we have determined that while it performs reasonably well (0.81 weighted F1 score), the dataset is too small and inconsistent (particularly for certain classes) to effectively address advanced inference services, such as projecting back the anomaly models onto the knowledge graph for guiding support teams on similar incidents. The model-based approach led to the identification of 12 SPARQL query patterns for anomaly detection based on a qualitative analysis of 139 incident tickets from the RDF dataset. The rather small number of patterns and the versatility of some of them provided valuable insights on the detection and implementation of new patterns, notably in the perspective of an increase in the size of the dataset. We also explored the limits of the model-based approach and the complementarity of the statistical approach through the projection of the query patterns onto the embeddings. We observed that there might be $n$-to-1 or $n$-to-$m$ relationships between the patterns and the trouble ticket categories. This should be further investigated as we found some data inconsistency during the classifier evaluation step.

Future work will first focus on addressing data quality issues to improve the performance of the classification model and continue our research efforts in connecting query patterns to the latent space of embeddings. We also plan to continue exploring ways to improve the capture of the context of incidents at the knowledge graph embeddings stage. Hence, we aim to automatically process the description of incidents or the description of recovery interventions written in natural language. Indeed, traditional knowledge graph embeddings techniques just go through object properties to build the embeddings of a node since a datatype property

breaks the graph (i.e. a literal cannot be a subject). Alternative knowledge graph embeddings approaches exist to handle literals [7]. However, this gives rise to further challenges, including the requirement to discretize the literals (e.g. determining a general criterion for discretizing dates, numbers, etc.). An alternative consists in annotating datatype properties with semantic entities using a language model. Next, we aim to explore additional sampling and walk strategies [169]. This could notably allow us to guide the extraction of network topology data from the knowledge graph according to infrastructure or time criteria. Finally, we would like to test our approach using dynamic graph generation tools [223] to open it up for comparative studies, particularly regarding scalability. Ultimately, future research could explore automating IT support functions by reasoning on these shared explicit models of infrastructure and service behavior and past incidents management.

# Chapter 8

# Designing a User Interface for Graph-based Advanced Anomaly Detection

## 8.1 Introduction

Ensuring a high level of Quality of Service (QoS) on telecommunications and data processing services requires a constant human and technical commitment (Chapter 1), whether it is for the most trivial applications or most critical ones. The ICT systems supporting these applications are indeed complex systems. As a consequence, despite the availability of well-established NMS/SIEM DSSs (Chapter 2), supervision experts face cognitive overload due to numerous indicators and alarms. The heterogeneity of ICT systems' configurations and potential lack of information about neighboring systems further complicate their tasks, leading them to decision making under uncertainty in incident management context.

Providing a comprehensive and unified view of ICT systems and their dynamics, while facilitating access to anomaly detection algorithms and solution recommendation tools, appears to be a critical path to enable increased operational efficiency in the incident management process. RDF KGs have proven to be flexible for data integration and logical reasoning over heterogeneous data, notably thanks to shared semantics provided by ontologies. However, we notice that the use of KGs has not yet become widespread in NMS/SIEM DSSs (Section 2.2) although various ontologies related to NetOps/SecOps are already available for describing network systems and for cybersecurity (Section 2.3). Additionally, the multiple knowledge facets that need to be represented for situation understanding pose a limit to intuitive and efficient exploration of KGs (i.e. quick and limited access to only relevant information) without a deep understanding of the ontologies at work, especially when short response time is imposed by Service Level Agreements (SLAs). Regarding the use of AI, various approaches demonstrate practical interest for anomaly detection or diagnostic assistance (Section 2.4). However, as suggested in Section 5, it is important for supervision experts to have the ability to combine and call different approaches and models in order to cover a wide range of system behaviors and support efficient decision-making.

In this chapter, we posit that the combination of knowledge graphs and automated inference tools offer promising prospects for improving NMS/SIEM DSSs, assuming that the DSSs' ergonomics meet the NetOps/SecOps teams' business requirements in terms of information accessibility (e.g. breaking down technical silos), incident situation contextualization (e.g. reducing cognitive load through alarm grouping), and continuity of operational tasks. To tackle this, we present NORIA UI, a Web-based client-server software architecture for network anomaly management based on data stored in a KG, with its ergonomics (UI/UX) being the result of collaboration with a panel of NetOps/SecOps experts from Orange. Our main contributions are the following. Firstly, we provide details of the business requirements for a next-gen NMS/SIEM DSS in terms of ergonomics and functions. Secondly, we present the technical details of the architecture implemented to meet these requirements, going beyond the simple data exposition from a knowledge graph by implementing the principle of synergistic reasoning for the combination of various diagnostic AI techniques, and the implementation of interaction mechanisms for exploratory analysis of multi-layered systems. Finally, we provide a feedback on the proposed solution and its prospects based on performance analysis and UI/UX evaluation by users in operational situations.

The remainder of this chapter is organized as follows. Section 8.2 presents the methodology to capture the business requirements. Section 8.3 presents the set of UI/UX features designed to meet these requirements. Section 8.4 provides details on the client-server architecture supporting these features. Section 8.5 illustrates diagnostic assistance with synergistic reasoning. Section 8.6 reports on user evaluation of the proposed architecture and features. Section 8.7 concludes this chapter by providing a summary of the work that has been done and offering some preliminary conclusions regarding the opportunities and potential future developments associated with it.

## 8.2 Personas and Requirements

To meet the operational needs of NetOps and SecOps experts, we used a methodology based on personas and interviews to develop the NORIA UI solution, including its UI/UX perspective. NORIA UI allows different user profiles to interact with it, catering to various usage scenarios and objectives. Personas, as described in [221], are archetypes of user classes that capture goals, behavior patterns, skills, attitudes, and environment. These personas are designed to be effective for the specific design problem at hand.

**Gathering requirements.** To capture these different perspectives, we firstly pre-defined a set of professional profiles that could potentially contribute to the expression of needs, and then launched a call for participation, ensuring that we had a sufficiently broad and representative panel. Due to a good match of profiles, we have once again called upon the

panel of experts who contributed to the design of NORIA-O (Chapter 3) and the definition of six fundamental anomaly detection cases (Chapter 5). In addition to these two set of interviews, we organized a series of UI/UX co-design workshops. To do so, we initially derived preliminary UI/UX requirements from the Precondition/Success Guarantees/Trigger of the six fundamental detection cases. These requirements were then sketched and challenged with the panel of experts during the workshops. The outcomes of these workshops served as the basis for the UI design presented in this chapter.

All experts have consistently expressed a recurring need and frustration regarding the heterogeneity of network infrastructures that are monitored. This directly relates to the necessity of handling and analyzing multiple data sources to effectively comprehend network situations. Users, therefore, hold a positive perception of a tool that could standardize and consolidate all data into a single platform with correlation capabilities. However, it is important to note that these users represent different skill sets and activity fields, each with their own distinct objectives and approaches on the comprehensive overview. Typical persona are agents from NOC and SOC. For the definition of the personas, we have chosen to abstract from the specificities of the network and cybersecurity domains in the sense that NORIA can be considered as a platform for supervising network infrastructures and detecting anomalies, whether they are the result of a malfunction, human error or a malicious act.

**Defining personas.** We can define four different personas that will be interacting with the tool. The first is the **incident manager** in charge of the investigation of a given incident. This contributor manages the reception of the alert and its context, assembles an investigation team adapted to the type of situation, coordinates efforts and facilitates communication between investigators, and then communicates the results of the investigation. The main need of this user is to quickly obtain an overview of a situation as well as all the sharing functions enabling him to facilitate work within his team of responders. The team of the incident manager is composed of several **supervision experts** and/or **cybersecurity analysts**, our second and third personas, depending on the needs and the perimeter impacted. The **network supervision expert** use NORIA UI to correlate events, in particular alarms, from different network equipment. This persona has in-depth knowledge of a technical perimeter and of the events generated by the various systems in the network. This type of user is particularly interested in analysis functions such as root cause analysis and the ability to quickly identify faulty equipment within a defined technical perimeter. The identification and description of the network infrastructure, the accuracy of event information and the correlation of indicators from different sources are important aspects for these users.

The **cybersecurity analyst** uses NORIA UI to identify possible malicious activity (i.e. forensic). These users have a good knowledge of the services offered by a given infrastructure and the

types of events that can occur. Similarly to the network supervision expert, they are particularly interested in the ability to correlate indicators from different sources. They are particularly attentive to analysing the events that occur and the information linked to these events (in particular the textual description of the logs).

The fourth persona is the **system architect** who aims to understand how a system works in order to improve its performance through re-engineering or upgrading (e.g. adding a function) or its security (by installing devices such as countermeasures, firewalls, etc.).

## 8.3 NORIA UI Features

In this section, we present the set of UI/UX features designed to meet business requirements derived from expert panel interviews in Section 8.2. Specifically, the analysis of user stories by personas leads us to identify the following five groups of features:

1. Cross-consultation of information on network topology, events and alarms,
2. Display of 2D/3D network topology enriched with indicators,
3. Aggregation and analysis of information in a dedicated digital investigation space,
4. Use of analysis and anomaly detection tools,
5. Access to community functions for sharing information between collaborators.

In the following, we describe from a functional perspective how we have implemented these groups of features. The overall design, called NORIA UI, is a KG-based network monitoring tool & decision support system that offers NetOps/SecOps teams a comprehensive view of multiple data sources and integrated analysis functions for diagnostic assistance and knowledge sharing. NORIA UI leverages data from an RDF knowledge graph generated through the KGC pipeline described in Chapter 4, and structured by the NORIA-O data model described in Chapter 3. In addition to the information provided in this chapter, a demonstration video showcasing the tool's main features is available online in the "*NORIA: Network anomaly detection using knowledge graphs*" blog post (Section A).

### 8.3.1 Dashboard: the Network in One Glance

The *dashboard* page (Figure 8.1) consists of four panels providing access to information about the network's life based on four complementary facets derived from the knowledge graph: trouble tickets, events and alarms, resources and applications, and an enriched view of network topology. It is the main page of the application and serves as the primary tool for NetOps/SecOps experts to explore the data in the knowledge graph and gain an overall understanding of the network's status or a specific event.

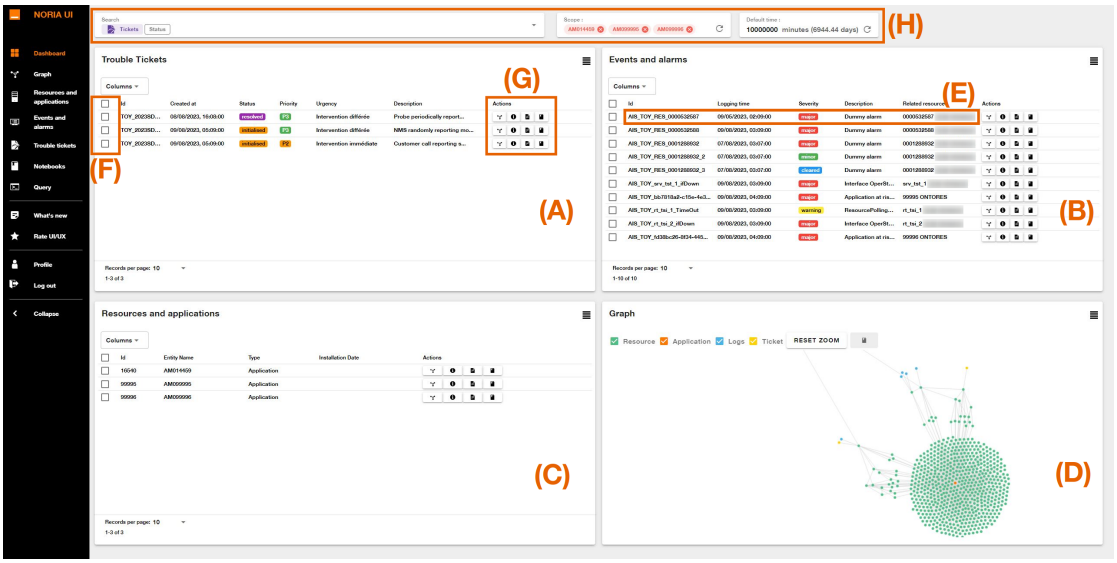



Figure 8.1: NORIA UI: the dashboard page
The dashboard page consists of four panels providing access to information about the network's life based on four complementary facets derived from the knowledge graph: (*A*) trouble tickets, (*B*) events and alarms, (*C*) resources and applications, and (*D*) the enriched network topology. Entities are displayed along with their main properties (*E*). Checkboxes (*F*) allow for a display pivot that applies to all panels. Each entity has contextual commands (*G*). An input field (*H*) allows defining a display scope and searching for entities based on their properties.

**Interacting with the panels.** Out of the four panels, three present the information in the form of a list. To match the needs of personas and avoid potential cognitive overload, we apply customized rendering for certain properties of the KG entities (e.g. color-coding the trouble ticket status for quick understanding of its priority) and enable users to select which properties to display. Furthermore, to facilitate quick interaction with the KG entities, we provide entity-specific action buttons on the right side of each table. The current version of NORIA UI offers the following four actions: *1) Focus:* clicking on this button focuses the entity in the graph visualization panel, allowing users to quickly center their view for studying the network context of the entity. *2) Information:* when more information is needed, clicking on this button redirects to a specialized page (Section 8.3.2) for the entity type (e.g. trouble ticket, events, etc.) and applies pre-filtering to highlight the entity. *3) Linked Data:* for advanced users, this button redirects them to the entity in the KG-DBMS's frontend (Section 8.4). *4) Notebook:* to assist users in conducting in-depth analysis of specific situations, the NORIA UI provides a notebook feature (Section 8.3.4). Clicking on this button, the entity is added to the notebook.

The last panel embeds a graph visualization library to provide an interactive view of the knowledge graph. It represents the entities from the three previously mentioned panels, along with the physical and logical network connections between resources and applications, as well as the composition relationships between these resources (e.g. a network packet filtering card as part of a network router device). This feature allows users to have a multi-faceted view of the network (i.e. topology, events, etc.) Each node in the graph is color-coded based on its

type. This view is interconnected with the other three panels, enabling users to analyze the context of each entity. For instance, clicking on an item in the "Trouble Tickets" panel brings the corresponding node to the center of the graph view, displaying its relations to the linked resource/application in the knowledge graph. This provides a comprehensive understanding of the incident context at a glance, by examining the neighboring resources/applications in the network and the associated alarms. See Section 8.3.3 for more details on the graph visualization feature.

**Searching and filtering.** The user has access to various functions for searching and filtering information. As a first one, the user can use the *pivot* function, which enables dynamic cross-referencing between different information facets [88]. This function allows sequential filtering in the "Trouble Tickets," "Resources and Applications," and "Events and Alarms" data panels. Checkboxes in these panels (Figure 8.1.F) are used to select source entities for the pivot, creating a pivot that filters the content of the other two panels. For example, selecting trouble tickets in the first panel will display entities linked to those tickets in the other panels. The user can then choose to apply a second pivot (based on the first one at the trouble tickets level) by selecting items in one of the remaining panels. For instance, selecting a set of resources in the "Resources and Applications" panel will result in filtering being applied to the "Events and Alarms" panel based on the entities selected in the two pivots. This feature supports exploratory searches by sequentially filtering correlated entities in the three data panels based on the relationships stored in the knowledge graph.

A second set of search and filtering functions is available at the top of the UI (Figure 8.1.H). The *global search* allows users to search all data, regardless of the entity type. Users can use tags (i.e. shorthands to the knowledge graph concepts and properties) to structure their search query and specify search criteria. For example, users can select the "Trouble Ticket" entity type (i.e. an `owl:Class`)[1], the "status" field (i.e. an `owl:ObjectProperty`), and the attribute value "current" (i.e. a `skos:Concept`)[2] to search for all open trouble tickets. Users can also enter any value, such as a network device hostname or logistic identifier, for a broader search across all data. The second part of the search is the *scope*. Users can define a functional/technical perimeter (e.g. core network, mobile network, email services) by specifying applications, and the presented data will be filtered accordingly. Similarly, users can filter data using a time scope. For example, selecting a time scope of 1000 minutes will display data from the last 1000 minutes.

[1] The `owl` namespaces refers to http://www.w3.org/2002/07/owl#.
[2] The `skos` namespaces refers to http://www.w3.org/2004/02/skos/core#.

### 8.3.2 Entities, Details and Comparison of Objects

To obtain comprehensive information on entities, NORIA UI provides a dedicated page for each type of entity (trouble tickets, resources and applications, events and alarms) using a shared page layout template. The template includes a data table to display all occurrences of the entity type in the knowledge graph. The layout, rendering options, and action buttons are consistent with the *dashboard* page (Section 8.3.1). Users can switch to a card layout for better readability and comparison, such as placing resource entity cards from different technical scopes side by side. The template also includes a panel on the right side to display detailed information about a selected entity, allowing users to keep this information visible while browsing the entity list. Additionally, the template includes a *synthesis* panel to provide statistics on selected entities based on their properties. This panel allows users to quickly identify commonalities within a subset of entities, such as the number of resources deployed in a specific building or the distribution of alarm types.

### 8.3.3 Graph: Exploring an Incident Context

To explore the relationships between the entities of the knowledge graph, the NORIA UI also provides a graph visualization component called *noriaConnect*. It allows users to have an multi-faceted and interactive view of the network (i.e. topology, events, etc.) in pages where graph visualization is needed, such as the *dashboard* page (Section 8.3.1), the *graph* page, and the *notebook* page (Section 8.3.4).

*noriaConnect* is based on the `force-graph`[3] library, which we enhanced to meet the needs of NORIA UI, notably in terms of interactions with the graph. For example, considering the *graph* page (Figure 8.2), the *noriaConnect* component occupies the majority of the window and is linked to two side panels for displaying details on the currently selected node, and *synthesis* view when multi-selection occurs in the graph. It enables displaying the entities from the three “Trouble Ticket”, “Events and Alarms”, and “Ressources and Applications” facets, along with the physical and logical network connections between resources and applications, as well as the composition relationships between these resources. Each node in the graph is color-coded based on its type.

Each click on a node triggers an action. A *left-click* on a node zooms in on it and displays the labels of neighboring nodes (i.e. nodes linked to that node by an arc). For example, clicking on a trouble ticket node highlights its linked resource/application and related events, allowing the user to quickly focus on the incident context. A *right-click* opens a context menu that provides access to ad hoc functions. One function sends the node to the notebook, allowing the user to gather items of interest as they explore the network context. Another function is the

[3] https://github.com/vasturiano/force-graph

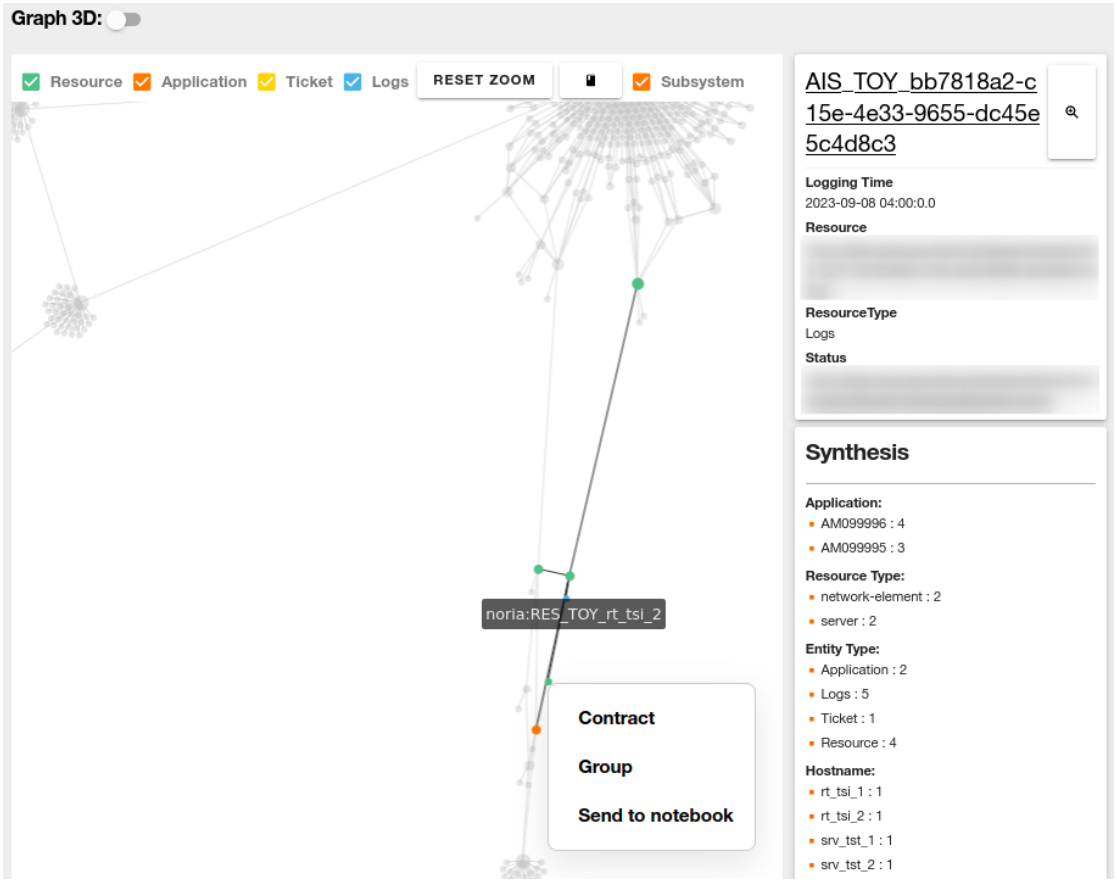



Figure 8.2: NORIA UI: the graph view for analyzing the network context.
The *noriaConnect* component on the *graph* page allows users to explore and interact with the knowledge graph. Checkboxes at the top of the component enable to select the type of entities and relationships to display . A "Graph 3D" toggle command allows to switch to 3D rendering of the graph, providing spatial clustering for comprehending the network behavior as suggested in [203]. Hovering the mouse over a node displays its shortened URI in a tooltip. Right-clicking on a node displays a context menu. Two side panels provide details on the active node and a synthesis of the properties for multi-selection.

graph contraction/expansion function, which reduces/expands the amount of information displayed by *noriaConnect* on the screen (by default, no nodes are contracted).

We have implemented the three following contraction strategies. *1) Directed graph contraction by depth:* contracts the graph by searching for neighboring nodes along directed edges starting from a reference node. This is useful for gathering all constituent nodes of a technical infrastructure linked by a hierarchical relationship or gathering all linked and succeeding temporal nodes from a reference moment. It involves implementing a `depth` parameter to the function (i.e. number of edges in depth), which we set to one by default. *2) Undirected graph contraction by depth and attributes:* contracts the neighbors of a reference node based on a criterion provided by the user, without considering the direction of the edges. This is useful for servers hosting a given application, incident tickets related to a specific server, or servers of an application with a specified operating system. *3) Graph node grouping by attributes:* groups

the nodes of a graph based on an attribute, regardless of the presence of an edge between them. This is useful for nodes of a specific type, such as servers, applications, or sites.

*noriaConnect* also allows for adjusting the graph actions and rendering based on the specific usage context and needs of the users. For example, additional commands can be added to the context menu to execute stored SPARQL queries using the URI [50] of the active node as an argument. Additionally, the size of the nodes can increase based on their degree (i.e. the number of connected nodes) to facilitate focusing on dense parts of the graph (e.g. a resource node with multiple alarms). Finally, nodes and arcs can blink to highlight a path in the graph (e.g. indicating the temporal relatedness of events for root cause analysis).

### 8.3.4 Notebooks: Analyzing an Incident Context

The *notebook* is a place where users can save entities of interest (trouble tickets, resources, alarms, etc.) for an on-going diagnosis process. For example, the root cause analysis of an application performance degradation issue may involve examining the servers hosting the application and the network routers bringing connectivity to these servers. This can potentially require conducting a long-term analysis, during which the user may lose track of their working hypothesis. To facilitate this investigation into the anomaly, users can create a notebook (they can have multiple notebooks at any given time) to gather the entities that are likely to guide them towards understanding and resolving the anomaly.

For a given notebook, the *notebook* page provides a set of display tricks and analysis functions enabling the user to analyze the context independently of the many entities that are not relevant to the case under investigation. For example, the entities in the notebook are displayed as cards that the user can browse and reorganize to find similarities.

Similarly, a three-tab side panel provides access to three groups of tools that use the content of the notebook to generate insights. The first tab, the *notebook synthesis* tab, calculates an aggregated list of entity properties from the notebook to highlight commonalities in the observables. It functions in the same way as in the entities views (Section 8.3.2). The second tab, the *notebook graph* tab, displays the subgraph composed of the elements in the notebook, similar to what is offered in the *dashboard* and *graph* pages. The contextual menu associated with the nodes allows for automatically adding the neighboring nodes of the targeted node to the notebook, thereby facilitating the extension of the scope of diagnosis. The third tab, the *notebook analysis* tab, allows users to call stored inference models (Section 8.4) to categorize the situation depicted by the elements of the notebook using a knowledge base. In the current NORIA UI version, we have implemented the process mining approach discussed in Chapter 6 to enable searching for procedural models in the knowledge base that best fit the events of the notebook. By projecting these models onto the data in the notebook, it is then possible

to reorder the events to trace back the initiating event or highlight a candidate behavior of the network in the *notebook graph* tab. These last two points are illustrated further down in Section 8.5.

In order to maximize the opportunities for sharing the implicit knowledge contained in a notebook about an incident situation, we assign a unique identifier to each notebook upon creation. This identifier is used in the URI that opens the notebook page, allowing different users to share an analysis context by exchanging the URIs of their notebooks. Similarly, the notebook can be exported in JSON format to feed third-party SPARQL algorithms.

## 8.4 NORIA UI Architecture

NORIA UI presents itself to the user as a Web application. In detail, it is a Web-based client-server architecture [310] with four main components (Figure 8.3).

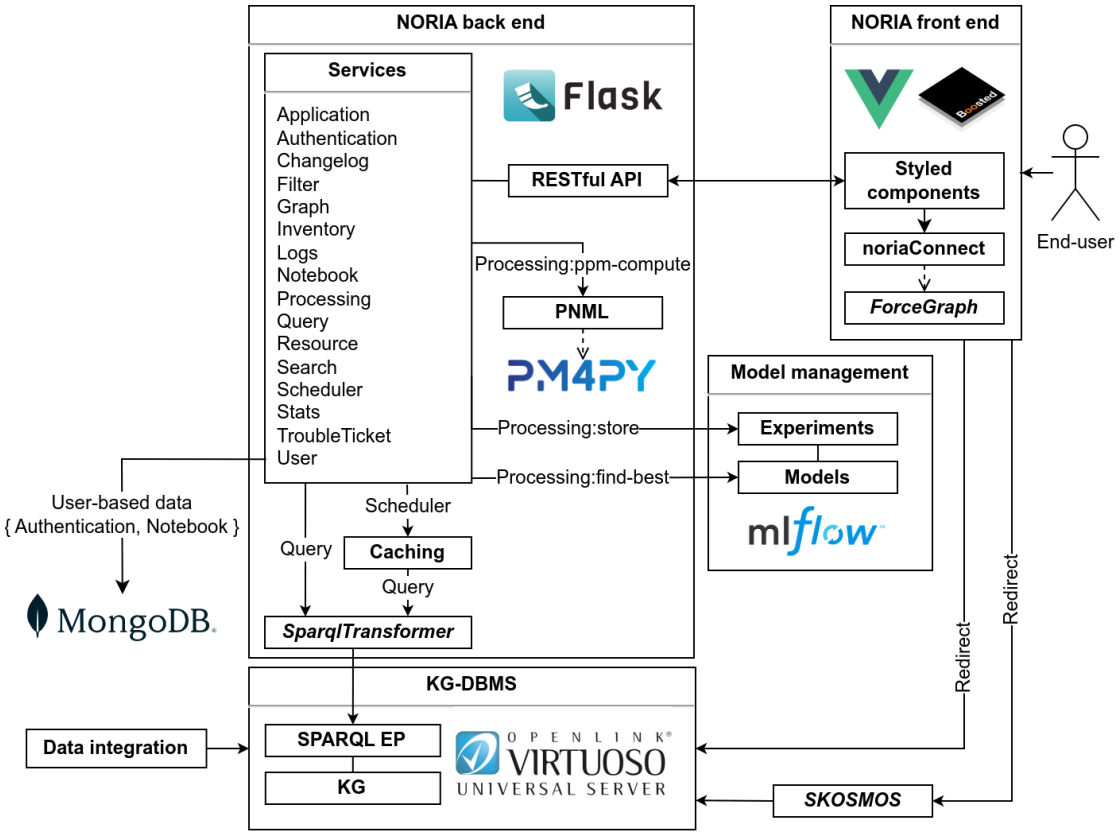



Figure 8.3: NORIA UI architecture overview
This functional block diagram provides an overview of the NORIA UI Web-based client-server architecture. The four main blocks are the *front end*, the *back end*, the *graph database* (KG-DBMS), and the *model database*. The end-user interacts with the knowledge graph data through the back end API. The *back end* implements and orchestrates a set of services, typically of the data fetching/processing/forwarding kind.

Firstly, the *front end* handles data rendering and user interactions through reusable components (e.g. templated list for entities, graph view, etc.), and accesses the knowledge graph data through the back end API. It is developed using the VueJS[4] framework for the codebase, and the Orange Boosted[5] toolkit for styling. Access to the Web application is protected through an OpenID Connect (OIDC)[6] connection managed by a single-sign-on (SSO) service, ensuring that only authorized users can access the NORIA UI. All user-related data, such as preferred display scope and notebooks, are stored separately in a MongoDB[7] database through the back end API.

Secondly, the *back end* implements and orchestrates a set of services, typically for data fetching, processing, and forwarding. Some services are made available through a RESTful API to directly meet the needs of the front end. For example, the *graph* service returns a JSON Graph[8] data structure that provides the network topology data based on the user's display scope, ready to be consumed by the *noriaConnect* component in the front end for rendering. This structure is prepared at the back end level from a pre-established parameterized query to the KG-DBMS (graph database). The *query* service relies on the SPARQL Transformer package [206], which manages interactions with the SPARQL endpoint of the graph database. To minimize network and processing load, we implemented a caching strategy: we minimize requests to the KG-DBMS using a pre-fetch approach for various types of entities, with specific cache durations based on the frequency of data changes. For example, entities of type "resources and applications" have a cache duration of 24 hours as they change infrequently, while "events" and "trouble tickets" have a cache duration of 15 minutes as they are ingested quasi real-time into the graph. The back end is developed using Flask[9] to enable rapid and efficient development of the necessary features for end users. Just like the front end, access to the API is protected by OIDC.

Thirdly, the *knowledge graph database management system* (KG-DBMS), namely an OpenLink Virtuoso[10] graph store instance, handles the data as an RDF knowledge graph, including the ontologies and controlled vocabularies, and gives access to these through a SPARQL endpoint. An independent data integration pipeline, separate from the NORIA UI solution, ensures the KGC process. This includes the periodic execution of SPARQL queries to enrich the graph with `inferred-alert` events for anomaly detection based on business rules corresponding to graph patterns (Chapter 5). Doing so implements a synergistic reasoning approach to anomaly detection and situation understanding at the NORIA UI level (Section 8.5). Indeed, diagnosis actions carried out in the *notebook* page using ad hoc analysis functions leverage

[4] https://vuejs.org/
[5] https://boosted.orange.com/
[6] https://openid.net/developers/how-connect-works/
[7] https://www.mongodb.com/
[8] https://jsongraphformat.info/
[9] https://flask.palletsprojects.com/
[10] https://virtuoso.openlinksw.com/

knowledge graph data from both the field sources and the output of algorithms running at the KGC platform level. The data integration pipeline described in Chapter 4 serves as a basis for the KGC process and periodic execution of anomaly detection queries.

Finally, a *model store*, specifically an Apache MLFlow[11] instance, is used to store and retrieve inference models. For example, in the *notebook* page (Section 8.3.4), a `processing:ppm-compute` service of the back end structures the notebook's data, calls a process discovery function from the PM4Py library [11], wraps the resulting Petri net model into a Python object, and sends the object to MLFlow. To enable this process, we implemented a Python class that inherits from MLFlow's `mlflow.pyfunc.PythonModel`[12] class definition. This allows us to store and version procedural models computed by users, as well as retrieve or run stored models for situation classification. It is important to note that this approach is not limited to procedural models but can also be applied to any type of inference models (e.g. random forest) whose serialization is compatible with MLFlow's functionalities.

In addition to these core components, we enhance the NORIA UI solution with a Skosmos instance [337] connected to the KG-DBMS. This integration allows users to easily browse the controlled vocabulary used in the knowledge graph. The controlled vocabulary is rendered in the front end as hyperlinks to the Skosmos UI.

## 8.5 A Synergistic Reasoning Scenario

In this section, we apply the concept of synergistic reasoning (i.e. the cooperation of multiple algorithms to provide reasoning) [46] to test the NORIA UI proposition. We start by defining a fictitious incident situation, then we show how the diagnosis can be performed via the user interface. The scenario described here serves as the basis for the user evaluation described in the following Section 8.6. The dataset used is described in Section B.3.

**Combining anomaly detection techniques.** We consider the following incident context, shown in Figure 8.4: a fault in the technical environment of a network router causes it to stop, degrading user access performance to an application and triggering various alarms on the supervision system. A supervision technician receives a call from an application user reporting a drop in performance, but with no further details.

As discussed in Section 5.3.2, we assume that the anomaly detection approaches developed in this thesis work could combine to provide efficient situation understanding, each approach being related to a specific scope of inference.

[11] https://mlflow.org/
[12] https://mlflow.org/docs/latest/python_api/mlflow.pyfunc.html

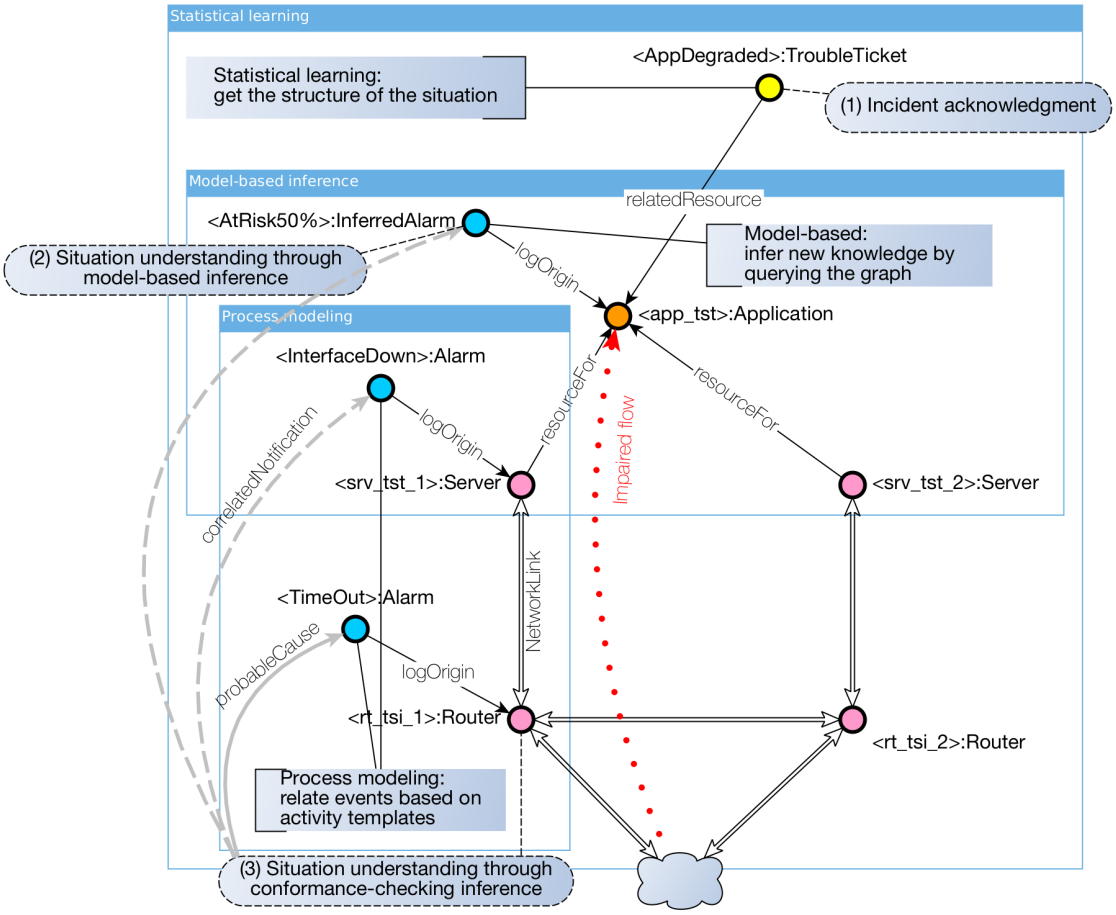



Figure 8.4: Combining anomaly detection techniques for situation understanding
This diagram highlights how the *model-based*, *process mining*, and *statistical learning* approaches combine for anomaly detection and situation understanding on a fictitious incident situation, each approach being related to a specific scope of inference. The graph is a simplified version of a knowledge graph structured by the NORIA-O data model, depicting a network topology with alarm events and a trouble ticket. The incident situation summarizes as follows: a "time out" alarm is emitted by the supervision system because of a hardware fault on `rt_tsi_1` making it unreachable; consequently, an interface down alarm is raised by `srv_tst_1` as its link to `rt_tsi_1` is down; an inferred alarm, computed thanks to a SPARQL query, is attached to the `app_tst` to alert on a resilience problem because of a fault on `srv_tst_1`; a trouble ticket is attached to `app_tst`, following a user complaint about its performance. The dashed boxes with numbering indicate the diagnosis steps through the NORIA UI.

More precisely, and firstly, *model-based* (Chapter 5) relates to querying the graph to retrieve anomalies and their context. Basic examples, derived from NetOps/SecOps expert interviews, are: "*k out-of n devices with faults*", "*user with unusual account rights*", and "*absence of traffic on an interface supposed to be active.*" In the case depicted in Figure 8.4, an inferred alarm, defined as a graph pattern based on expert knowledge and computed thanks to the SPARQL query of Listing 5.4, is attached to the `app_tst` entity to alert on a resilience problem because of a fault on `srv_tst_1`. Besides alerting on the resilience issue at the `noria:Application` entity level, the description of the inferred alarm is also a guide for the diagnostic stage of the incident management process by encouraging to further scrutinize the lower layers of the network (i.e. the server resources hosting the application). It is also noteworthy that, because

the inferred alarm is new knowledge enriching the KG, it could serve as a basis for subsequent use of other anomaly detection techniques (as discussed right below with the *process mining* and *statistical learning* approaches).

Secondly, *process mining* (Chapter 6), relates to aligning a sequence of timestamped entities (e.g. `noria:EventRecord`) to activity models, then use this relatedness to guide the repair. Basic examples are of the `(LoginFail)=>(LoginFail)=>(LoginFail)` and `(EnergyLoss)=>(TimeoutAlert)=>(LossOfSignal)` kind. In the case of Figure 8.4, an enriched version of the second example, such as `(EnergyLoss)=>(TimeoutAlert)=>-(LossOfSignal)=>(AtRisk50%)` will relate the time out alarm as a probable cause to the resilience alarm.

Finally, *statistical learning* (Chapter 7) captures the incident context (e.g. the sub-graph centered on a given `noria:TroubleTicket` entity), thereby enabling to relate entities based on context similarities (e.g. capturing the presence of a `(<InterfaceDown>:Alarm)→(Server)` relationship instead of the too much precise `(<InterfaceDown>:Alarm)→(<srv_tst_1>:Server)` one) and then use this relatedness to alert and guide the repair. In the case of Figure 8.4, we would capture the signature of resilience issues occuring on applications hosted close to a core network, and thereby be able to detect similar issues on other locations of the network even if the network topology is not exactly the same. It is noteworthy that, because the `noria:TroubleTicket` entities hold information about how a given issue has been solved, it becomes then possible to suggest how to remediate new situations based on context similarities. Akin to this, another example derived from NetOps/SecOps expert interviews is "*the hidden cause of the trouble ticket on the application is a 'data leak' attack that started on server 2*", suggested by the idea that complementary CTI knowledge and that events over a long period of time are part of the learning process for incident contexts.

**Situation understanding.** We now illustrate how the diagnosis of the previous scenario is implemented in NORIA UI. The process follows three main steps as indicated in Figure 8.4: (1) incident acknowledgment, (2) situation understanding through model-based inference, and (3) situation understanding through conformance-checking inference.

At step (1), the NORIA UI user searches for a given `noria:TroubleTicket` based on its identifier (Figure 8.5) and filters out useless information from the *Ressources and applications* and *Events and alarms* panels by pivoting on it (Figure 8.6), thereby highlighting the presence of an alarm attached to the related `noria:Application` entity.

At step (2), clicking on the alarm, the user is forwarded to the details of the `noria:EventRecord` entity, revealing the `inferred-alert` type of it and the resilience na-

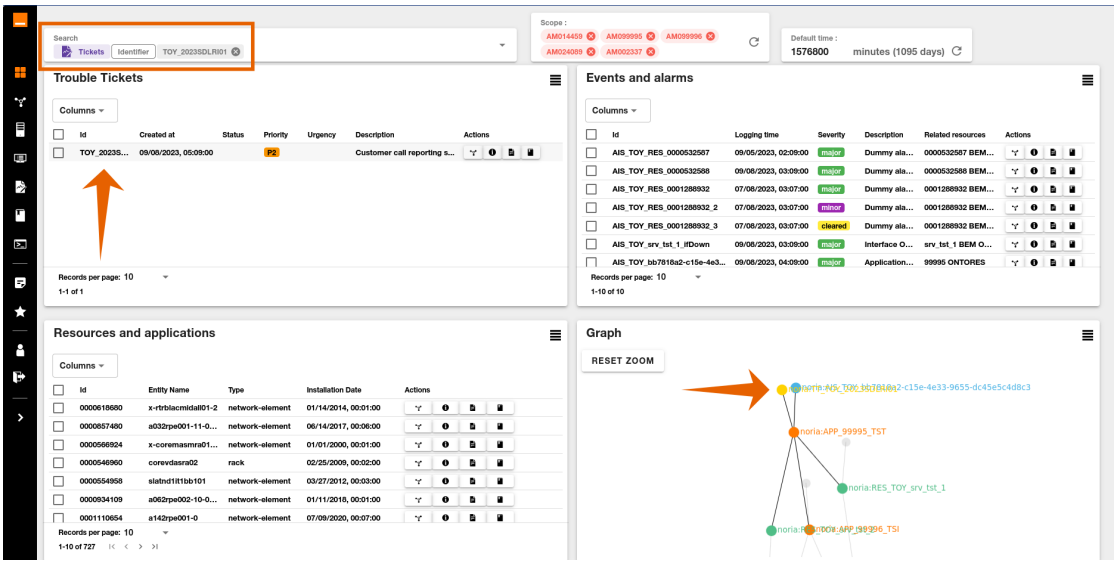



Figure 8.5: Diagnosing with NORIA UI: searching on TroubleTickets based on an $id$.

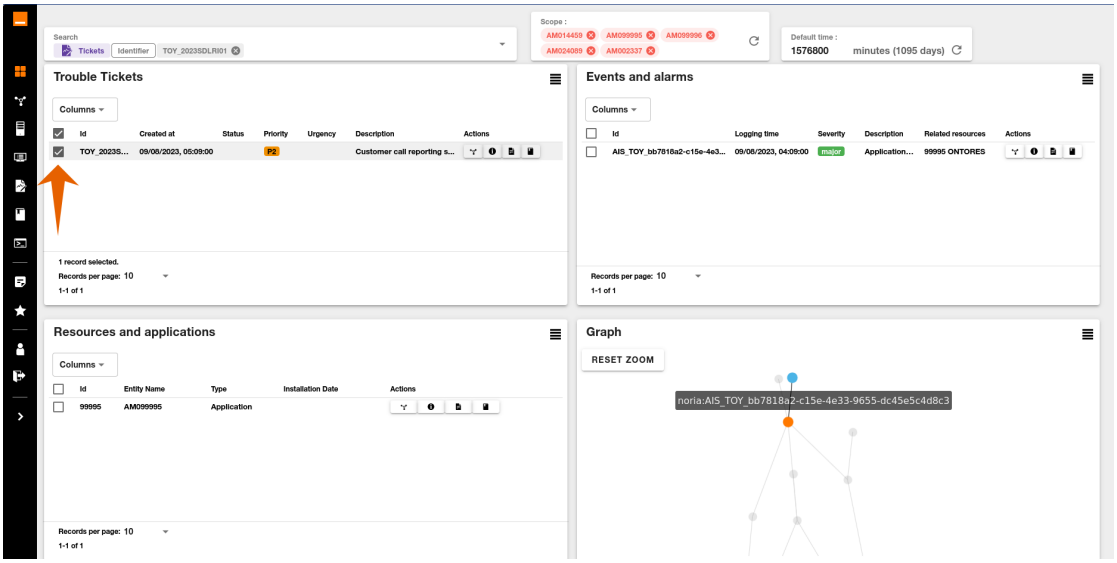



Figure 8.6: Diagnosing with NORIA UI: pivoting on TroubleTickets.

ture of the alert (Figure 8.7). Clicking on the details, the user is then forwarded to the enriched graph view of the network (Figure 8.8), from which he selects a set of entities to send to a notebook for further analysis.

At step (3), the user obtains a summary of the properties of the entities in the notebook (Figure 8.9), then requests a search for a procedural pattern that can explain the situation. This search returns the model in the knowledge base with the best fitness, allowing the user to get an ordered list of the `noria:EventRecords` of the notebook against the model to trace back the probable cause, to get an annotated network context to visually understand the dynamics of

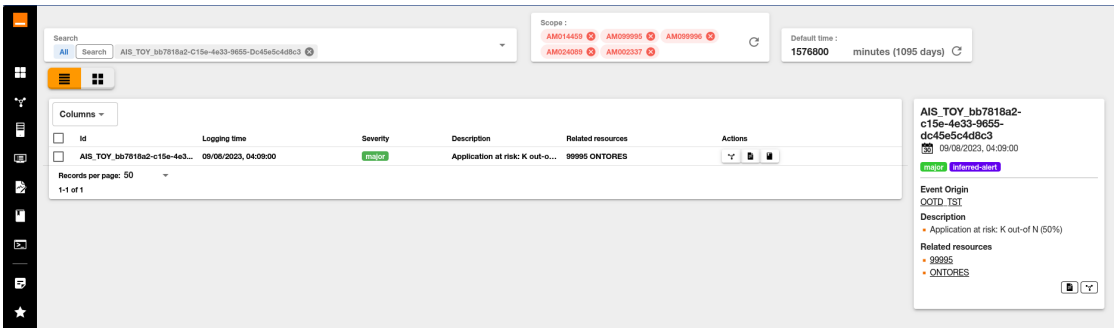



Figure 8.7: Diagnosing with NORIA UI: getting details on EventRecords.

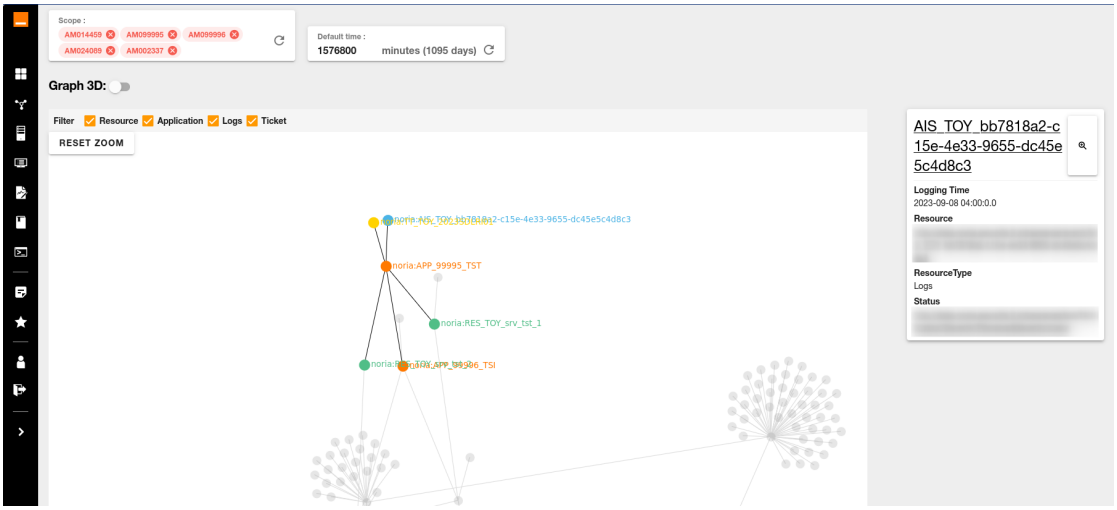



Figure 8.8: Diagnosing with NORIA UI: analyzing the network context.

the situation (Figure 8.10), and to view the formal model in the base (Figure 8.11).

**Discussion.** This three-step scenario highlights an exploratory approach for situation understanding. The model-based approach is shown transparent to the user as the execution of SPARQL anomaly detection queries is performed periodically in the background by an orchestration tool associated with the NORIA platform (Chapter 4). The process mining approach, on the other hand, is activated upon user request within the notebook context. Although not illustrated, the statistical learning approach could be automatically activated periodically to generate inferred alerts, or activated on-demand as a reasoning service (Section 5.3.2), similar to the process mining approach.

This scenario was used to prepare the "*Semantical anomaly sensing*" demonstration of the overall NORIA approach at the OOTD 2023 exhibition (Section A). The demonstration received very enthusiastic feedback live from numerous visitors from various backgrounds. In the following Section 8.6, we further analyze the NORIA UI proposition based on this scenario

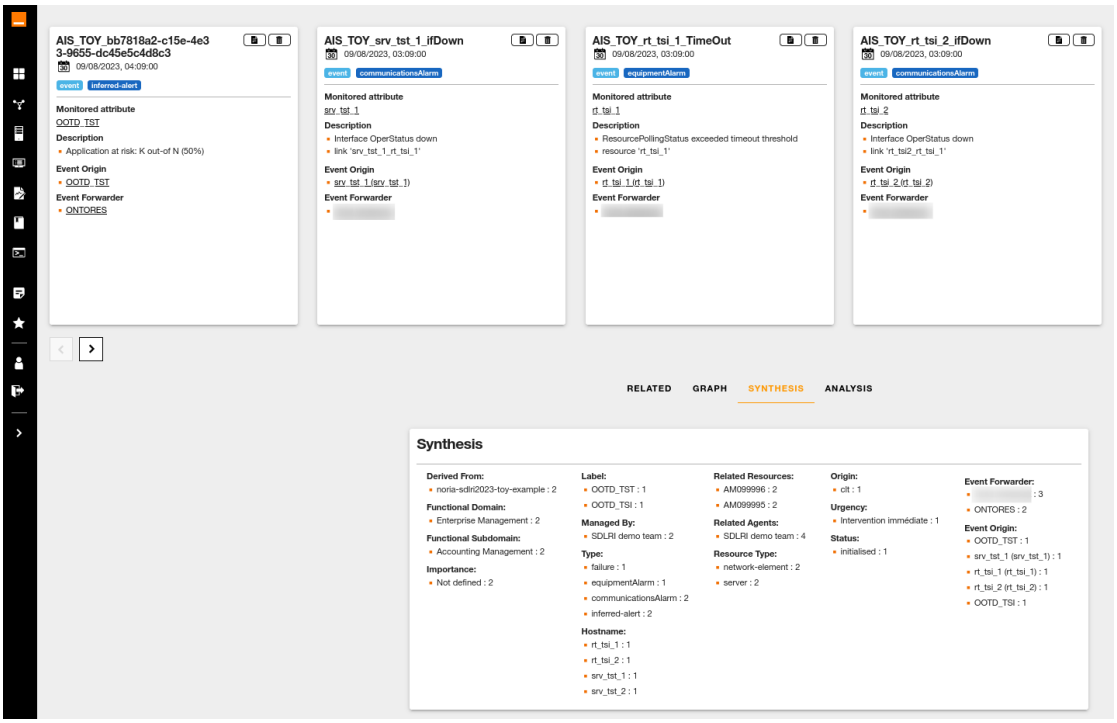



Figure 8.9: Diagnosing with NORIA UI: getting a synthesis on the entities in a notebook.

through a formal evaluation by a panel of users.

## 8.6 Evaluation

In this section, we first present the evaluation process and the corresponding usage context for the proposed NORIA UI solution. We then analyze the evaluation results to draw conclusions about the effectiveness of the design and identify any potential areas for improvement.

**Evaluation process.** The overall evaluation approach involves engaging a panel of beta testers with a profile closely aligned with network/cybersecurity operations. They are invited to connect and use the NORIA UI solution and provide feedback by responding to a questionnaire based on the System Usability Scale (SUS) method [181]. The SUS method evaluates the usability and user experience through a weighted score derived from ten closed-ended questions, assessing the potential adoption and relevance of the tool. We used the SUS questions as proposed in the seminal paper describing SUS [180], and their French version from [131] for French-speaking participants. To ensure reader convenience, we have included the SUS statements as defined in [180] in Table 8.1. To capture users' expectations more accurately, we have included two additional questions in the base SUS questionnaire, which were presented at the end of each evaluation session: "*In a few words, what were you looking for the last time*

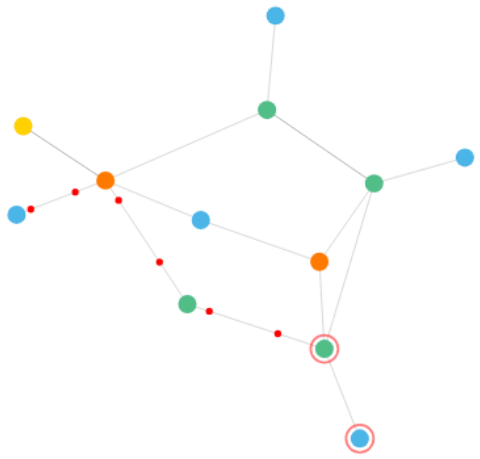


Figure 8.10: Diagnosing with NORIA UI: graphical root cause analysis
A prototype of the graphical root cause analysis view obtained by projecting the procedural model from Figure 8.11 onto the entities in the notebook. The circled nodes highlight the `noria:Resource` and the `noria:EventRecord` likely responsible for the incident. The dotted lines emphasize the temporal sequence.

*you used NORIA?*", and "*Do you have any feedback, positive/negative opinions about NORIA UI, or any suggestions?*" The questionnaire is anonymous by default, but beta testers had the option to provide their email for personalized follow-up. Additionally, we collected usage data of NORIA UI through a Matomo[13] instance, which captures telemetry data such as the browser used, hardware configuration of the workstation, user journeys, and page loading times. Each beta tester has the possibility to connect to the solution and evaluate it as many times as desired during the evaluation period. At the end of the period, we extracted the results from the enriched SUS questionnaire, calculated the SUS scores per user and per question, and analyzed the correlation between these scores and the respondents' profiles in relation to the verbatim responses.

**Evaluation scenario.** We designed an incremental evaluation campaign, with a first phase focused on evaluating the ergonomics of the NORIA UI Web interface with its core features (Section 8.3) using synthetic data, and then iterating the evaluation by adding features to the solution, incorporating real operational data, and introducing new diagnostic tools. We report below on the first phase of the evaluation campaign, which is based on a scenario that corresponds to a typical incident detection and diagnosis chronology for a technical support engineer. The usage scenario, evoked in the previous Section 8.5, summarizes as follows: *1)* the user logs in and authenticates to NORIA UI through a Web browser from a workstation; *2)* configures his user profile by setting the display and search scope; *3)* looks for a given trouble ticket in the *dashboard* view using the search bar based on its $id$; *4)* pivots on the trouble ticket entity to filter out all non-related entities from the *dashboard* alternative panels

[13] https://matomo.org/

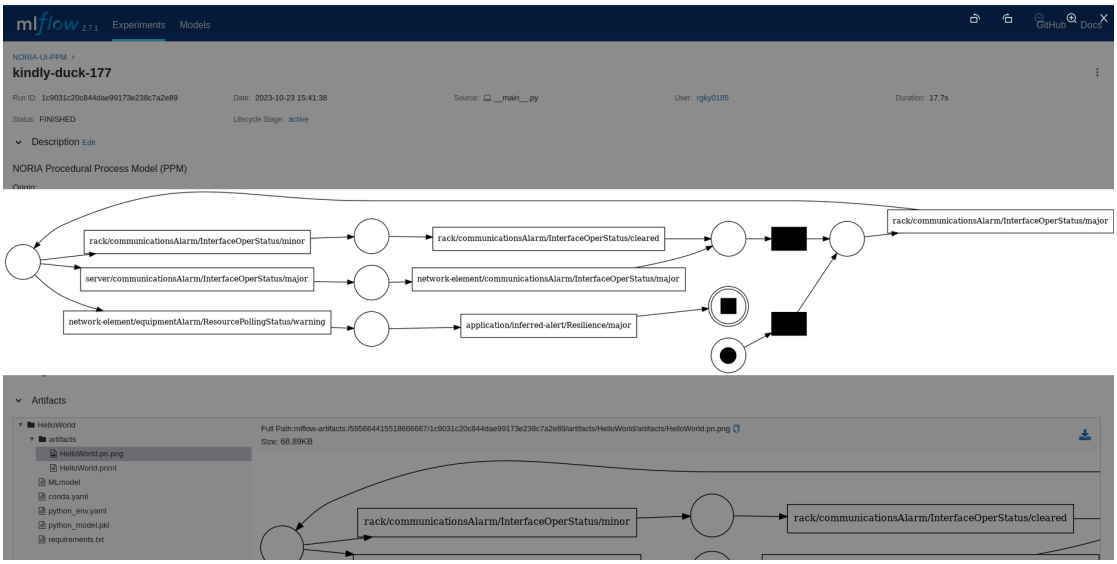



Figure 8.11: Diagnosing with NORIA UI: viewing a procedural model in the knowledge base
The procedural model, stored as a Petri net, can be viewed in the Apache MLFlow's user interface (an open source model store).

(i.e. events and alarms, and resources and applications); *5)* send the related alarms to the *notebook* view; *6)* check for the details of alarm that triggered the trouble ticket, this reveals an `Application at risk: k out-of n (50%)` event of the `inferred-alert` type; *7)* get the network context of the alarm in the *graph* view; *8)* select the nodes of the network context (i.e. `noria:Resource`, `noria:Application`, and `noria:EventRecord` entities) and send them in the notebook; *9)* get an overview of the incident context browsing the *notebook* view, with a first level level of situation diagnosis using its analysis features.

**Evaluation and performance results.** For the first phase of the evaluation campaign (2024-02-10 to 2024-03-10), we recruited a panel of 25 beta tester, upon which ten actively engaged into the evaluation process, with the following distribution of personas: two cybersecurity analysts, two incident managers, one network supervision expert, five system architects. Out of the ten participants, two did not complete the test scenario for unknown reasons. Table 8.2 summarizes the SUS scores for these ten participants. The overall average SUS score is 68.4 when considering SUS questionnaire answers related to complete tests, and 59.0 when considering all the answers.

Four topics emerge from the "*what were you looking for*" verbatims, and can be summarized as follows: *1)* **Dependency Visualization:** users want to search for equipment and understand their connections; examples or technical diagrams would be helpful during the initial use of the tool to ensure a clear understanding of the displayed graphs; users also want to understand which applications are installed on which servers. *2)* **Incident Correlation:** users seek assistance in correlating incidents and events; they suggest that an AI could automate the

Table 8.1: Statements from the SUS method
This table provides the ten statements for subjective assessments of usability, as defined in [180].

| | Statements |
|---|---|
| Q.1 | I think that I would like to use this system frequently |
| Q.2 | I found the system unnecessarily complex |
| Q.3 | I thought the system was easy to use |
| Q.4 | I think that I would need the support of a technical person to be able to use this system |
| Q.5 | I found the various functions in this system were well integrated |
| Q.6 | I thought there was too much inconsistency in this system |
| Q.7 | I would imagine that most people would learn to use this system very quickly |
| Q.8 | I found the system very cumbersome to use |
| Q.9 | I felt very confident using the system |
| Q.10 | I needed to learn a lot of things before I could get going with this system |

Table 8.2: Notes and overall SUS scores by personas
The *Q.x* columns provide the ratings for SUS questions as presented in [180, 131] on a scale of 1 to 10, with the +/− sign indicating whether it is a positive question (the higher the better) or a negative question (the lower the better). The *SUS* column is the overall SUS score calculated by weighted sum according to the formula from [180]. The values by personas are separated between respondents who completed the test scenario fully and those who completed it partially. The values in bold highlight the highest scores. *N* stands for the number of respondents.

| Persona | *N* | Q.1 + | Q.2 − | Q.3 + | Q.4 − | Q.5 + | Q.6 − | Q.7 + | Q.8 − | Q.9 + | Q.10 − | SUS $w.\Sigma$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Cybersecurity analyst | 2 | **10.0** | ***0.5*** | 7.5 | 4.0 | **9.0** | 2.0 | **9** | 2.0 | 8.5 | 2.5 | 78.8 |
| Incident manager | 2 | **10.0** | ***0.5*** | **8.0** | 8.0 | 8.5 | 9.0 | 7 | 2.5 | **9.5** | 2.5 | 63.1 |
| Network supervision expert | 1 | **10.0** | 2.0 | **8.0** | ***2.0*** | 8.0 | 1.0 | 8 | ***1.0*** | 8.0 | ***1.0*** | **81.3** |
| System architect | 3 | 7.3 | 6.7 | 6.0 | 4.3 | 8.0 | ***0.7*** | 8 | 2.7 | 8.3 | 4.7 | 60.8 |
| Average (complete) | 8 | **9.0** | 3.0 | 7.1 | 4.9 | 8.4 | 3.1 | 8 | ***2.3*** | 8.6 | 3.1 | 68.4 |
| System architect (partial) | 2 | 5.5 | 7.0 | 3.0 | 7.5 | 4.0 | 5.0 | 3 | 7.0 | 4.0 | 6.0 | 21.3 |
| Average (all) | 10 | **8.3** | 3.8 | 6.3 | 5.4 | 7.5 | 3.5 | 7 | ***3.2*** | 7.7 | 3.7 | 59.0 |

correlation of events related to a root cause and automatically provide suggestions in the notebook. *3)* **Utilization for Incident Diagnosis:** users are interested in exploring how NORIA can assist in incident diagnosis, particularly for transmission supervisors; they want to envision how the tool would be used by supervisors/operators; users note that if the context is saved, it could accelerate the automation of diagnostics. *4)* **Ontology and Automated Processing:** users are looking for an illustration of the NORIA-O data model to utilize automatic processing algorithms for detection, diagnosis, and remediation proposals.

Regarding the "*feedback and opinions*" verbatims, the respondents provided feedback on the overall relevance of the proposal (e.g. "*the tool could be very useful for ICT systems supervision to quickly identify the root cause of an incident, calculate incident impact, and analyze incidents retrospectively*"), while suggesting some improvements in the sequence of actions (e.g. "*the concept of a notebook to pin relevant elements is interesting, but the manipulations are*

*somewhat tedious*" and "*the tool appears to be designed as a navigation tool for domain experts, requiring many clicks and not suitable for real-time incident handling*"). Respondents also requested evaluation in further usage contexts (e.g. "*a more realistic test scenario would have been helpful to fully grasp the interface and data*") and highlighted minor ergonomic issues (e.g. changes in colors and missing tooltips for increased readability, page and parameter display/refresh problems on a specific Web browser).

Finally, regarding the NORIA UI usage analytics, the platform (consisting of two RHEL7.6 120 GB disk, 16 GB RAM virtual machines on Ericsson HDS 8000 - Intel Xeon DP 2.8GHZ hosts [356]) received 134 visits from authenticated users during the evaluation period. The average time on the website per visitor was 16 minutes, with an average of 14.3 actions per visitor. All visitors accessed the UI from a desktop or laptop workstation, with a screen resolution of 2560 × 1440 for 36% of them, 1920 × 1080 for 20%, and lower for the rest. The UI navigation was done through standard Web browsers, with Chrome accounting for 44%, Microsoft Edge for 29%, and Firefox for 27%. The average page loading time observed was 2.89 seconds (2.03 seconds for generating the DOM), with the following average times per specific page (in decreasing order of the top four): Graph = 4.62 seconds, Home page/Dashboard = 2.97 seconds, Inventory = 2.79 seconds, Notebook = 1.94 seconds.

**Analysis and discussion.** The analysis of values and SUS scores in Table 8.2 suggests a correlation between the respondents' profiles and their evaluations. As per [44], the scores indicate an acceptability level ranging from good (Incident manager, System architect) to high (Network supervision expert, Cybersecurity analyst) of the solution for beta testers who completed the test scenario fully, and not acceptable for those who tested it partially (System architect ×2). The joint analysis of verbatims provides additional insights into the evaluations and scores by persona. For example, considering $Q.6$ answers, respondents rated the proposed solution based on the available data (synthetic data + partially consolidated operational data from [205]) and unresolved display bugs at the time of evaluation. This heightened their overall expectations regarding their capability to act quickly in an incident management context with perfect data quality and faultless tools. Furthermore, the multi-faceted, homogenized, and graphical view of the network and its dynamics is seen as beneficial for operational efficiency ($Q.1$ and $Q.5$ answers). Similarly, the co-design of a graphical interface to leverage the framework consisting of a knowledge graph and anomaly detection algorithms is also seen as beneficial for diagnostic aid ($Q.7$, $Q.8$, and $Q.9$ answers). Enriching the proposition by integrating additional synthesis and recommendation algorithms, as well as the ability to interact with systems from the UI through contextual commands, seems essential to take operational efficiency to the next level. This suggestion is supported by certain verbatims and the $Q.2$, $Q.8$, and $Q.10$ scores, notably of personas (Incident manager and System architect) whose role typically involves using analysis rather than producing it. Focusing on usage

analytics, we notice that respondents generally tested NORIA UI with display conditions that allowed them to take advantage of simultaneous display of maximum information, enabling them to explore and analyze data without the need for window resizing or extreme use of responsive design functions. Regarding page loading time, we notice that the average time is above the theoretical "under a second" threshold [250], and that the high values (Graph and Dashboard views) correspond to the most demanding pages in terms of rendering and data fetching. User feedback does not indicate any responsiveness issues, so the loading and display delay caused by interactions with the graph database and visual rendering in the graph component seem acceptable as it is. However, it will certainly need to be closely observed under intensive usage conditions and with optimization of the service implementation and strengthening of cache usage.

## 8.7 Conclusion and Future Work

In this chapter, we considered the combination of knowledge graphs and automated inference tools as crucial for advancing NMS/SIEM DSSs to the next level. This enhancement aims to improve operational efficiency in incident management on ICT systems, assuming that the ergonomics of the DSSs align with the business requirements of NetOps/SecOps teams. However, since the use of KGs is not yet widespread in the design of NMS/SIEM DSSs, various challenges needed to be addressed in order to achieve this objective, notably in terms of timely access to relevant information and integration of multiple anomaly detection algorithms for the analysis of large-scale network incident contexts.

To tackle these challenges, we designed NORIA UI, a Web-based client-server architecture for network anomaly management based on data stored in a knowledge graph structured by the NORIA-O data model. The ergonomics (UI/UX) of NORIA UI are the result of collaboration with a panel of NetOps/SecOps experts from Orange. We conducted a user evaluation campaign using an incident diagnosis scenario to validate the effectiveness of our proposal. The results confirmed the value of NORIA UI in terms of providing rapid navigation through various aspects of a network by aggregating heterogeneous data. It also demonstrated the ease of use of the synergistic approach for establishing a diagnosis through the successive combination of various analysis techniques. Additionally, the evaluation highlighted the specific expectations of NetOps/SecOps user profiles in terms of functions and usage actions.

Based on this, two axes of evolution for future work emerge from the proposal. The first axis concerns the analysis functions that need to be enhanced, for example by integrating the calculation of incident context similarity using graph embeddings discussed in Chapter 7. This would enable users to perform comparative case analyses, raise `inferred-alerts` on diffuse situations by periodically executing models, and even propose alarm grouping before their presentation [92].

Furthermore, similar to the intrinsic explainability of knowledge graphs and procedural models in the form of Petri nets, the integration of new algorithms less anchored in logic or explicit representation should be accompanied by a reflection on how to present the results of inference in the UI in order to maintain the user's confidence in the system. One option worth considering is to slightly modify the UI to allow the user to consult a trace of the inference process, either in a text format close to natural language or in a graphical format [237]. Another option would be to implement a collaborative filtering approach, which involves a feedback mechanism from users [42]. Indeed, this would allow, for example, using voting buttons in the UI, to gradually improve the accuracy of recommendations from statistical models while giving the user a sense of maintaining some control over the functioning of the models. It is noteworthy that such a feedback mechanism could also be useful for addressing data quality issues. It would enable the user to tag non-relevant or inaccurate information on the fly.

The second axis of evolution focuses on enhancing the performance of the NORIA UI solution, specifically in terms of data loading time and interface responsiveness. One approach to consider is offloading more computation to the backend, including at the KG level. This could involve precomputing aggregating nodes or "short cut relationships" [187] in the knowledge graph (e.g. for graph contraction). Additionally, tracking user activities more closely could help identify common knowledge graph traversals, leading to insights on optimizing I/O performance at the KG-DBMS level (e.g. with additional indexing) and improving the data model (e.g. reworking excessively long property paths).

# Part III

# Conclusion and Appendices

# Chapter 9

# Conclusions

This thesis contributed in research application to the specific domain of Knowledge Representation and Reasoning (KRR) for anomaly detection and incident management on large-scale ICT systems. In the following, we summarize the content of this thesis, report some initial implications of the obtained results, and suggest future research perspectives on this topic.

## 9.1 Summary of the Research

In small and simple IT networks, when failures or service impairments occur, Root Cause Analysis (RCA) and decision-making for containment procedures and restoration to normal operations are usually straightforward. Technical alarms reported by NMS and SIEM systems directly reflect the cause, and the number of system states to consider is small and manageable. However, when scaling up to critical IT networks, we observe that a significant portion of the difficulties in effectively managing network incidents, despite the use of monitoring tools and diagnostic aids, stems from the variety of situations and technical details to be considered at once. Indeed, RCA becomes harder for complex and heterogeneous systems because of cascading failures and alarm spreading phenomenon. To tackle this situation, we delved into the fundamental question of:

> *How to define an anomaly model in a dynamic technical environment with various interdependencies, and what form should this model take to be shareable among practitioners (network designers, network administrators, cybersecurity analysts, etc.) and directly usable in anomaly detection tools and decision support systems?*

To further enhance our investigation, we divided this question into two sub-research questions. The first one (RQ 1) focuses on examining anomaly model production and utilization with heterogeneous data. The second one (RQ 2) scrutinizes the constraints on the internal representation of data and knowledge.

Thinking about networks generally involves considering a meshed system, or more abstractly, considering a graph structure. Adding to this the fact that a large number of notifications (alarms, technical logs) and indicators (OS names, file hashs, equipments' temperature, CPU load, network traffic throughput) are reported in monitoring tools, one can naturally arrive at the intuition of a dynamic graph (Hypothesis 1). Assuming a strictly logical formulation of the network's ecosystem, then the precise description of the network's behavior can be compared to deriving a history of events in the lifecycle of the network from a formal grammar (Hypothesis 5). In other other words, network events and states result from a generative process defined by the network's configuration and its ecosystem. Based on such a grammar, anyone would be able to explain the past and future behavior of a system, thus providing the mean to anticipate abnormal network behavior or trace back to the root cause of network incidents. However, in the general case, it is necessary to consider an imperfect understanding of the network and its ecosystem due to limited observability of some indicators, challenges in comprehending identical facts, the lack of observability of neighboring networks, the non-deterministic behavior of users, and other factors (Hypotheses 2 & 3). As a consequence, to effectively learn and utilize anomaly models (Hypothesis 4), it is essential to combine knowledge representations and analysis techniques that draw their expressiveness from both the realm of strict logic and the realm of probabilities.

Based on these initial insights, the approach adopted in this research involved utilizing an explicit representation of networks (including assets, topology, events, and alarms) and their ecosystem (such as servicing teams, asset locations, incident management artifacts like trouble tickets, etc.) in the form of a Knowledge Graph (KG). Additionally, algorithmic techniques heavily reliant on formal representation at the level of generated models or their results were experimented with. Implementing this approach involved taking a holistic perspective on the application domain, which encompassed defining the domain of knowledge and detection cases in collaboration with network operation experts, as well as developing a data processing pipeline and evaluating the proposed techniques based on their performance and usability by operational teams.

## 9.2 First Implications

The contributions of this work are manifold (see Table 9.1 for an overview). Focusing on RQ 1 (anomaly model production & utilization with heterogeneous data), the first main outcome is the proposal of a **KG-based data platform architecture** (Chapter 4) where 15 data sources are used for constructing the KG, whether it be from streamed sources or using batch processing. Beyond the technical details and processing components shared in open source, a key implication of this result is that data, through the KG, can be presented and interpreted in a unified manner to downstream algorithms. As a benefit, the same anomaly detection

Table 9.1: Research Questions and Outcomes
This table summarizes how the outcomes of this thesis work have addressed the research questions RQ 1 (anomaly model production & utilization with heterogeneous data), and RQ 2 (constraints on the internal representation of data and knowledge).

| Outcome | NORIA-O | KG-based data platform | Anomaly detection framework | UI/UX design |
|---|---|---|---|---|
| Ref. | Chapter 3 | Chapter 4 | Chapters 5 – 7 | Chapter 8 |
| RQ 1 | | ✓ | ✓ | ✓ |
| RQ 2 | ✓ | ✓ | | ✓ |

technique can take into account a broader spectrum of facts for making inferences, in terms of manipulated concepts such as assets, events, sites, teams, etc., thereby validating Hypothesis 1. Moreover, and more importantly, the same technique will remain independent of the data source, as syntactically different but semantically identical facts will be considered in the same way (e.g. a network interface status reported as `UP` for a CISCO device and `Enabled` for a JUNIPER device). This capability is enabled by the **NORIA-O ontology** (Chapter 3), which is the main outcome of RQ 2 (constraints on the internal representation of data and knowledge). NORIA-O is an RDFS/OWL-2 ontology available in open source and that re-uses and extends well-known ontologies such as SEAS, FOLIO, Devops Infrastructure, UCO, ORG, BOT and BBO. This model allows to describe a network infrastructure, its events (e.g. user login, network route priority reconfiguration), diagnosis and repair actions (e.g. connectivity check, firmware upgrade) that are performed during incident management. The NORIA-O data model also provides a set of controlled vocabularies, based on the SKOS data model, that facilitates the standardized interpretation of entities within the knowledge graph entities, such as the network interface status – as evoked above – or device alarm types.

Back to RQ 1, a third significant outcome is the introduction of an **anomaly detection framework**. The proposal relies on the aforementioned KG-based data platform and the implementation of three complementary families of detection techniques: data extraction from the graph using SPARQL queries and reasoning schemes established from business rules (Chapter 5, validating Hypotheses 4 & 5), event correlation using learned or declared procedural models following process mining and conformance checking approaches (Chapter 6, validating Hypotheses 3, 4 & 5), and subgraph classification using incident contexts learned through graph traversal and graph embedding techniques (Chapter 7, validating Hypotheses 2 & 3). Beyond highlighting each technique individually, a key implication of the proposal is that it combines intrinsic explainability and probabilistic reasoning capabilities through the application of the synergistic reasoning principle (cooperative decision-making). The principle emphasized here is that each technique, taken individually, allows for the reinjection of knowledge into the KG, which can then serve as an additional contextual element for a second technique. In line with this idea, a second implication is that the combination of detection techniques allows

for coverage of a theoretically broader range of abnormal situations than what a single model specialized in a specific data nature (e.g. time series, tables, graphs) or application domain (e.g. network traffic, computer viruses) would allow.

Finally, considering both RQ 1 and RQ 2, this thesis work resulted in the organization of a user evaluation of anomaly detection models through a **UI/UX study and design proposal** (Chapter 8). Beyond the technical and ergonomics details for exploring the KG and presenting inference results, a significant implication of this proposal is that it establishes a strong foundation for future work on enhancing operational efficiency in incident management. This is achieved by demonstrating the ability to capture and analyze a formal incident context through the comprehensive approach proposed.

## 9.3 Limitations and Further Perspectives

**Towards trustworthy synergical reasoning with knowledge graphs.** Summarizing the points discussed above, we are now able, thanks to the NORIA-O data model and the proposed KG-based architecture, to study incident management at the level of user actions (e.g. network administrators, customers, attackers) by contextualizing them within the network topology. Incident resolution methods become explicit and more easily shareable among practitioners and with clients seeking accountability.

However, although the results of this thesis have provided, through a set of coherent building blocks, an answer to how to define and leverage an anomaly model for a dynamic system, a new key question emerges to guide future work: *how to select and ideally order each anomaly detection approach to ensure trustworthy decision-making?*

Considering the limitations and future work identified in each chapter: **NORIA-O** (Chapter 3) showed that while we can now leverage an explicit representation of the network dynamics, complementary vocabularies (e.g. OTN or 5G mobile network specifications) and reconciliation techniques (e.g. log parsing [175, 325, 26] and NLP-related techniques [37, 156, 276]) should be implemented and used to fully enable cross-technical-domain anomaly correlation based on symbolic knowledge representation; the **KG-based platform** (Chapter 4) has demonstrated how to convert and link massive and heterogeneous data, but the pre-conditions and triggering mechanisms for opportunistic reasoning (e.g. SKOS reconciliation as a service, in-line graph clustering) and data/model management (e.g. KG pruning and summarization techniques) are yet to be designed and integrated; the **model-based design** (Chapter 5) and **process mining** (Chapter 6) anomaly detection approaches showed how expert knowledge could serve as a basis for implementing logic-related detection use cases, however, we now need to further develop the knowledge capture methods, particularly by identifying a criterion (e.g. algorithmic complexity, representation complexity) that would indicate which

combination of formalism to use and how to factorize/group detection models without losing detection capacity; the **statistical learning** approach (Chapter 7) has demonstrated its ability to capture an incident context and provide indications of probable causes and solutions, however, it remains to be established to what extent a procedural model can enhance the accuracy of this approach (e.g. using specific sampling and walk strategies [169] or including P/T nets as RDF graphs [97, 186] into the embeddings computation), or how a causal model could emerge from the approach and be used by NetOps/SecOps teams; finally, the **UI/UX design** study (Chapter 8) showed that while specialized ergonomics and tooling allow for efficient situation understanding based on knowledge graphs, the evaluation of the proposed approaches mentioned above needs to be complemented by identifying how to practically use them beyond the simple framework of anomaly detection (e.g. calculating similarity between incidents, comparing different scenarios, event/alarm clustering) and to what extent we can guarantee quick response (e.g. including "short cut" properties into the KG [187], monitoring the data model expressivity, monitoring the knowledge graph density[1] and refresh rate, adding indexes to the KG-DBMS or offloading processing to specific hardware platforms) and trust of detection models for users (e.g. collaborative filtering to reinforce confidence in ambiguous cases [42], explain the inference with natural language or in a graphical format [237]).

With these limitations in mind, to further elaborate on our new research objective, a strategy could be to consider the NORIA-O model, the KGC platform, and the statistical learning approach as viable baseline options. Subsequently, identifying the performance and explainability conditions in anomaly detection would consist in conducting a system sensitivity analysis. Indeed, the incident management process coupled with the continuous feeding of the graph database ensures the continuous learning of incident context based on the combined information of network status, incident categorization by NetOps/SecOps experts in trouble tickets, and tracking of remediation actions. Therefore, at a constant performance level of anomaly detection and incident classification, gradually substituting/complementing one approach with another or perturbing the KG and data model should help identify the point at which symbolic approaches definitively take over the more generalist nature of the statistical learning approach. From an analytical perspective, this exploratory process involves traversing the configuration space (phase space) of the inference system. Finding this phase transition point could ultimately be related to identifying the inference conditions where an autonomous network monitoring and control system would become no more accountable (i.e. explainable) and should interrupt its decision/control process, deeming that future ICT system states might be undesirable (e.g. highly complex for a return to normal operations, lethal for users).

[1] For directed graphs: $d = \frac{m}{n(n-1)}$, where $n$ is the number of nodes and $m$ the number of edges.

**Additional work streams and opportunities.** Taking a broader perspective on the incident management domain, we can see that we are ultimately also able to approach the study of network dynamics based on activities, interactions, and the configuration of the networks themselves. This allows us to strive for making recommendations for improving ICT systems engineering based on observations of their behavior. In line with User and Entity Behavior Analytics (UEBA), this is particularly evident in the idea of including activity traces semantically in the KG and performing embeddings on them. Conversely, this leads to the notion that the vector space resulting from graph embeddings is partly structured by a decision-making process, which opens up avenues for future research in the field of eXplainable AI (XAI). Leveraging the concept of moduli spaces [103] (a geometric space whose points represent algebro-geometric objects of some fixed kind), an option here could be to study the effect of continuous transformation of incident contexts on the corresponding vectors.

In terms of anomaly detection, future work will also focus on further contextualizing anomalies by adding additional data sources (e.g. network traffic from NetFlow [73] probes, cybersecurity vulnerabilities from the CVE[2] database and vulnerability scanner results), comparing with additional predictive models open to causal analysis (e.g. logical tensor networks [211, 111], geometric deep learning [64]), analyzing predictive architecture variants (e.g. neuro-symbolic architectures [14, 353, 361]) through model combination or stacking, and studying the impact of semanticization of unstructured data on model performance (e.g. utilizing reconciliation techniques [35, 276] and language models [157, 26]).

Although improved anomaly detection performance and understanding should result from this, it is noteworthy to mention that the framework introduced by this thesis work has the potential to change the way we perceive the organization of incident management. Indeed, it might concentrate administrative tasks on data model management, formal definition of use cases, and data linking. This highlights the need for providing training to network administrators, and developing and testing methods at an enterprise scale to manage the consistency and alignment of data models and anomaly detection models. It is therefore also important for future work to explore the impact of model modifications on reasoning capabilities, expressiveness, and performance. Furthermore, it is crucial to ensure that the primary form of presenting KG data to network practitioners is not limited to Semantic Web serialization formats such as RDF/XML, Turtle, or TriG. Emphasizing meaningful annotation of ontologies and SKOS vocabularies for natural language representation should be maintained to ensure a high level of adoption of this overall approach. Leveraging language models could be an option for generating information when it is lacking or hard to summarize, both when adding new knowledge to ontologies and KGs [373], or presenting KG data to users [282].

Finally, from the perspective of information processing IT architectures and observability,

[2] https://cve.mitre.org/

future work should ensure the interoperability of different data silos used for constructing a KG at an enterprise scale and provide differentiated access to subsets of it. Indeed, the framework proposed in this work showed high potential to operate at the scale of a large enterprise, and has highlighted the importance of using diverse data facets to learn and infer system behavior effectively. From an IT architecture perspective, to support this ambition, it typically involves deploying graph stores, interfacing with legacy DBMS using Virtual Knowledge Graph (VKG) systems (e.g. Ontop[3]), and querying the entire system through the SPARQL federated query[4] approach. However, several challenges still need to be addressed, such as establishing a correspondence between query language algebras that do not have an obvious bijection (e.g. SPARQL *vs* ArangoDB Query Language[5]). Algebraic analysis, potentially through the concept of database normalization or category theory [109], is a typical approach to make progress on this. Another significant challenge is enabling the handling of high-throughput and short-lived data in graph stores without degrading the read/write performance of this particular storage type. On-the-fly data aggregation [128, 313] or hybrid architectures that combine graph stores with Time Series DataBase (TSDB) are interesting options in this regard.

[3] https://ontop-vkg.org/
[4] https://www.w3.org/TR/sparql11-federated-query/
[5] https://arangodb.com/

# Appendices

# Chapter A

# Publications list

The research carried out during this doctoral work has lead to the publication of the following scientific papers, communications and artifacts:

## Peer-Reviewed Workshops and Conferences

1. Lionel Tailhardat, Raphaël Troncy, and Yoan Chabot. **Walks in Cyberspace: Improving Web Browsing and Network Activity Analysis with 3D Live Graph Rendering.** In *The Web Conference*, Developers Track, April 25–29, 2022, Lyon, France. https://doi.org/10.1145/3487553.3524230 [203]

2. Lionel Tailhardat, Yoan Chabot, and Raphaël Troncy. **Designing NORIA: a Knowledge Graph-based Platform for Anomaly Detection and Incident Management in ICT Systems**. In $4^{th}$ *International Workshop on Knowledge Graph Construction (KGCW)*, May 28, 2023, Crete. https://ceur-ws.org/Vol-3471/paper3.pdf [205]

3. Lionel Tailhardat, Raphaël Troncy, and Yoan Chabot. **Leveraging Knowledge Graphs For Classifying Incident Situations in ICT Systems.** In *The* $18^{th}$ *International Conference on Availability, Reliability and Security (ARES)*, GRASEC track, August 29–September 1, 2023, Benevento, Italy. https://doi.org/10.1145/3600160.3604991 [204]

4. Lionel Tailhardat, Benjamin Stach, Yoan Chabot, Raphaël Troncy. **Graphameleon: Relational Learning and Anomaly Detection on Web Navigation Traces Captured as Knowledge Graphs**. In *The Web Conference*, Resources Track, May 13–17, 2024, Singapore. [340]

5. Lionel Tailhardat, Raphaël Troncy, Yoan Chabot. **NORIA-O: An Ontology for Anomaly Detection and Incident Management in ICT Systems**. In $21^{st}$ *European Semantic Web Conference (ESWC)*, Resources Track, May 26–30, 2024, Hersonissos, Greece. **Best Paper Award nominee**. [339]

6. Youssra Rebboud, Lionel Tailhardat, Pasquale Lisena, Raphaël Troncy. **Can LLMs Generate Competency Questions?** In *21$^{st}$ European Semantic Web Conference (ESWC)*, LLMs for KE Track, May 26–30, 2024, Hersonissos, Greece. [300]

7. Lionel Tailhardat, Benjamin Stach, Yoan Chabot, Raphaël Troncy. **Graphaméléon : apprentissage des relations et détection d'anomalies sur les traces de navigation Web capturées sous forme de graphes de connaissances**. In *Plate-Forme Intelligence Artificielle (PFIA)*, IC Track, July 01–05, 2024, La Rochelle, France. **Best Paper Award**. [341]

8. Lionel Tailhardat, Yoan Chabot, Antoine Py, Perrine Guillemette. **NORIA UI: Efficient Incident Management on Large-Scale ICT Systems Represented as Knowledge Graphs**. In *The 19$^{th}$ International Conference on Availability, Reliability and Security (ARES)*, GRASEC track, July 30–August 02, 2024, Vienna, Austria. [338]

## Posters, Demos, Invited Talks and Blogs

1. Lionel Tailhardat, Yoan Chabot, and Raphaël Troncy. **NORIA - Machine LearNing, Ontology and Reasoning for the Identification of Anomalies.** Position poster presented at the *Institut d'Automne en Intelligence Artificielle (IA$^2$)*, Sorbonne Center for Artificial Intelligence (SCAI), September 2021, Paris, France. https://genears.github.io/pubs/IA2-2021-NORIA-POSTER.pdf

2. Lionel Tailhardat, Yoan Chabot, Perrine Guillemette, and Antoine Py. **Semantical anomaly sensing – Recommend remediation solutions using knowledge graphs.** Software platform prototype presented at the *Orange Open Tech Days (OOTD)*, November 2023, Châtillon, France. https://hellofuture.orange.com/app/uploads/2023/11/2023-OpenTechDays-book-demonstrations-conferences.pdf

3. Yoan Chabot, Lionel Tailhardat, Perrine Guillemette, and Antoine Py. **NORIA: Network anomaly detection using knowledge graphs.** Blog article in *Orange – Hello Future*, 2024. https://hellofuture.orange.com/en/noria-network-anomaly-detection-using-knowledge-graphs/.

4. Lionel Tailhardat. **Anomaly detection for telco companies: challenges and opportunities in knowledge graph construction.** Keynote Talk at the *5$^{th}$ International Workshop on Knowledge Graph Construction (KGCW)*, 2024. https://genears.github.io/pubs/KGCW-2024-keynote.pdf.

## Tutorial

1. Lionel Tailhardat. **Eléments d'Exploitation Des Réseaux Pour Une Conception Raisonnable.** Lecture presented at the LGI Safety & Risks chair, CentralSupélec, March 1, 2021. https://genears.github.io/pubs/lgi_orange_2020-2021_lecture.pdf

## Code and Dataset

The links to all resources & tools realized and published in the context of this research are summarized in Table A.1.

| **Resource or Tool** | **URL** |
|---|---|
| **TOOLS** | |
| NORIA-O | https://w3id.org/noria |
| grlc | https://github.com/Orange-OpenSource/grlc |
| SMASSIF-RML | https://github.com/Orange-OpenSource/SMASSIF-RML |
| ssb-consum-up | https://github.com/Orange-OpenSource/ssb-consum-up |
| SemNIDS | https://github.com/D2KLab/SemNIDS |
| Dynagraph | https://github.com/Orange-OpenSource/dynagraph |
| Graphameleon | https://github.com/Orange-OpenSource/graphameleon |
| **DATA** | |
| Graphameleon dataset | https://github.com/Orange-OpenSource/graphameleon-ds |

Table A.1: Links to resources and tools.

# Chapter B

# Additional materials

These appendices complement the information presented in the main body of this thesis manuscript. They consist of the following five independent parts: Appendix B.1 reviews the graph forms commonly used in DSSs, Appendix B.2 provides the full list of anomaly detection techniques studied in Chapter 2, Appendix B.3 presents the implementation of the dataset used for the experiments in Part II, Appendix B.4 introduces an algorithm to produce a graphical root cause analysis view in the absence of a procedural model, and Appendix B.5 discusses the relationships between data structures and computational efficiency.

## B.1 Formal Graph Forms

The aim of this section is to provide an overview of available formal graph forms that are commonly used within DSSs. Casual *graph notations* from graph theory are presented in Section B.1.1, followed by *data graph models* in Section B.1.2, and finally *graphical models* and *graph related forms* in Section B.1.3. Knowledge graphs are discussed in p. 184 of Section B.1.3.

An overwhelming literature about graphs and related topics already exists, therefore no deep analysis of each item will be given here unless necessary; for more details or a broader overview, the curious reader may refer to reference books and survey papers such as [6, 43] for the *graph theory* aspects, [119, 8] for the *knowledge graph* aspects, and [199, 118, 309] for the *knowledge representation* and *combinatorial optimization* aspects.

### B.1.1 Graph Notations

Given a set of abstract objects $V = \{v_1, v_2, \ldots, v_n\}$ and some notion of *relatedness* between these data points, let us define $\Gamma^+(v_i)$ the *successor function* giving the set of $v \in V$ directly related to $v_i$ when taking $v_i$ as a starting point. From the example of Figure B.1, $\Gamma^+(v_1) = \{v_2, v_3\}$, $\Gamma^+(v_2) = \{v_3, v_5\}$, $\Gamma^+(v_3) = \{v_3\}$, $\Gamma^+(v_4) = \{v_2, v_5\}$ and $\Gamma^+(v_5) = \{\varnothing\}$.

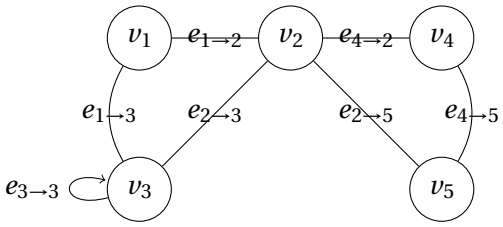



Figure B.1: Basic directed graph

**Directed graph.** Knowing $V$ and the full set of successors of $v \in V$, one can thus define a graph $G$ (Eq. B.1):

$$G = (V, A) \tag{B.1}$$

where $V$ is called the *vertex set* and $A = \{(v_i, v_j) \mid \forall v_i, \exists v_j \in \Gamma^+(v_i)\}$ is the *arc set*. Such graph is called a *directed graph*. Elements of $V$ may also be called *nodes*.

Assuming that the application of $\Gamma^-(v_i)$, the *predecessor function*, yields the exact same arc set as the successor function, or assuming that no particular meaning of *direction* is to be given to arcs, one can then define an *undirected graph*.

**Undirected graph.** We call *undirected graph* the graph defined by (Eq. B.2):

$$G = (V, E) \tag{B.2}$$

where $E$ is the *edge set*, i.e. the set $E = \{(v_i, v_j)\}$ where $(v_i, v_j)$ can interchangeably be read $(v_j, v_i)$. Based on these definitions, complementary functions and concepts may be defined to describe graphs:

- The *order* of the graph: $ord(G) := \|V\| = N$,
- The *size* of the graph: $size(G) := \|A\| = K$, with $0 \leq K \leq N \cdot (N-1)/2$ for a non-planar graph,
- The *density* of the graph: $density(G) := K/N$,
- The *out-going degree* of a vertex: $d_v^+ := \|\Gamma^+(v)\|$, the *in-going degree* of a vertex: $d_v^- := \|\Gamma^-(v)\|$ and the *degree* of a vertex: $d_v := \|\Gamma^+(v)\| + \|\Gamma^-(v)\|$.
- A *path* is a set of arcs where the end vertex of each arc, but for the last one, is the starting vertex of the next arc. A path where the starting node is also the ending node is called a *cycle*. The *length* of a path is the number of its arcs.
- A graph is said *connected* if there are paths containing each pair of vertices. More generally, the *connectivity* of a graph relates to the number of vertices or edges to be

removed to separate the initial graph into two or more subgraphs. When removing vertices, such separation is called a *vertex cut* and the number $k$ of vertices to remove leads to say that the graph is *$k$-vertex-connected* (or $k$-connected).

- A graph that is connected and has no cycles is called a *tree graph*. When working with a Directed Acyclic Graph (DAG), the graph is called *polytree* (or *arborescence*, *oriented tree*) and the starting vertex is called the *root*[1]. A *forest* is a disjoint union of trees. For applications where the use of a *metric distance* is needed on top of the graph structure, trees can be embedded in a hyperbolic plane [316, 31].

Moreover, looking at $A$ as a set of pairs, $A$ is a binary relation over the set $V$ (i.e. $A \subset V \times V$, where $\times$ is the *cartesian product* operator). In the undirected graph case, this relates to the mathematical concepts of *preordered set* (i.e. a binary relation that is reflexive and transitive) and *lattice* (i.e. a partially ordered set with unique least upper bounds and greatest lower bounds) [309, §1 & §3]. Also, a natural representation of $A$ is given by an *adjacency matrix*.

**Adjacency matrix.** An adjacency matrix $\mathbf{A}(G)$ is a square matrix of size $N \times N$ where the $a_{i,j}$ element of $\mathbf{A}$ takes the value:

$$a_{i,j} = \begin{cases} 1 & \text{if vertex } v_i \text{ is connected to vertex } v_j \\ 0 & \text{otherwise} \end{cases} \tag{B.3}$$

If $G$ is *simple* (i.e. no $v_i \rightarrow v_i$ arc or edge), then $diag(\mathbf{A}(G)) = 0$ (i.e. the diagonal of the adjacency matrix has only zero values). If $G$ is an undirected graph, then $\mathbf{A}(G)$ is a symmetrical matrix.

From the example of Figure B.1, the following $\mathbf{A_1}$ and $\mathbf{A_2}$ matrices are the adjacency matrices of the directed graph and equivalent undirected graph, respectively:

$$\mathbf{A_1} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad \mathbf{A_2} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Using the adjacency matrix definition and assuming that the arcs (or edges) should carry some quantitative information about the relationship between vertices (e.g. the intensity of the relationship), this leads us to the representation this information as a *weighted adjacency matrix*.

[1] From the Figure B.1, choosing the vertex $v_4$ as the root leads to the polytree where $V = \{v_2, v_3, v_4, v_5\}$ and $A = \{(v_2, v_3), (v_2, v_5), (v_4, v_2), (v_4, v_5)\}$.

**Weighted adjacency matrix.** Given some *weight* value $w_{ij}$ associated to the relationship between the vetices $v_i$ and $v_j$, then a weighted adjacency matrix $\mathbf{W}(G)$ is an adjacency matrix where the $w_{i,j}$ element of $\mathbf{W}$ takes the value:

$$w_{i,j} = \begin{cases} w_{ij} & \text{if vertex } v_i \text{ is connected to vertex } v_j \\ 0 & \text{otherwise} \end{cases}$$

### B.1.2 Data Graph Models

Based on the elementary directed graph (Eq. B.1) and undirected graph (Eq. B.2) structures, extended constructs can be designed to capture the relational structure of more complex data sets. These complex data sets may include labeled relationships, complex objects with numerous characteristics, and groups of data sets from various fields of applications. This ambition relates to the notion of *interpretation* from *descriptive logic*, which stands for the idea that an unambiguous reading of abstract constructs with respect to real world facts can be done (see [119, §2.2.1] and [8, B.5] for more information). This unambiguous reading can be used for deductive reasoning. A brief overview of these *data graph models* is provided below. Data graph models can also be referred to as *graphical models* (see below).

**Directed edge-labeled graph.** Given a directed graph $G$ such as in (B.1) and some need to tag arcs with a label from a predefined label list (e.g. the arc $(v_1, v_2)$ has the label $e_{1\to 2}$ in Figure B.1), then a *directed edge-labeled graph* is defined by (Eq. B.4):

$$G = (V, A, L) \tag{B.4}$$

where $V$ is a finite set of vertices, $L$ is a finite set of arc labels, and $E \subseteq V \times L \times V$ a set of arcs.

**Property graph.** Given a directed graph $G$ such as in (Eq. B.1) and some need to add named values (i.e. properties) to vertices and arcs, then a *property graph* is defined by the tuple (Eq. B.5):

$$G = (V, A, L, P, U, a, l, p) \tag{B.5}$$

where $V$ is a finite set of vertices ids, $A$ a finite set of arcs ids, $L$ a finite set of labels, $P$ a finite set of properties, $U$ a finite set of values, $a : A \to V \times V$ a mapping function of arc ids to a pair of vertices ids, $l : V \cup A \to 2^L$ a mapping function of vertex or arc id to a set of labels, and $p : V \cup A \to 2^{P\times U}$ a mapping function of vertex or arc id to a set of property-value pair.

**Graph dataset.** Given a directed edge-labeled graph $G$ as defined in (Eq. B.4) and $n$ an associated *graph name*, a *named graph* is defined as a pair $(n, G)$. Based on this, a *graph dataset* is defined by (Eq. B.6):

$$D = (G_D, N) \tag{B.6}$$

where $G_D$ is a directed edge-labeled graph called the *default graph*, and $N$ is either the empty set or a set of named graphs $\{(n_1, G_1), \ldots, (n_k, G_k)\}$. Graph names $n$ are used only once (i.e. $n_i = n_j$ if and only if $i = j$).

**Multigraph.** A *multigraph* is a graph where multiple edges (or arcs) can connect a given pair of vertices. Two cases of multigraph definitions occurs: *1)* multigraphs where edges are *without own identity* and, *2)* multigraphs where edges are *with their own identity*. Formal definitions are provided below for the undirected and unlabeled case:

- **Edges without own identity:** the multigraph $G$ follows the formal definition of (Eq. B.2) where $E$ is a *multiset* of unordered pairs of vertices (i.e. a set that allows for multiple instances of $(v_i, v_j)$ pairs).
- **Edges with own identity:** the multigraph $G$ is defined by (Eq. B.7):

  $$G = (V, E, r) \tag{B.7}$$

  where $r : E \rightarrow \{(v_i, v_j) \mid v_i, v_j \in V\}$ is a mapping function of edges to an unordered pair of ending vertices.

**Hypergraph.** From the binary relation between vertices provided by an adjacency matrix (Eq. B.3), only two vertices can be linked by an arc or an edge. In contrast, an *hypergraph* is a graph data model where arcs or edges can connect more than two vertices. A more complex construct called *generalized hypergraph* [46, §14.2.1] can be furthermore defined from there with the addition of two features: *1)* arcs (or edges) that connect to arcs (or edges); *2)* vertices that contain embedded hypergraphs[2]. Going a step further, weighted labeled hypergraph can also be defined. An hypergraph is not to be confused with a multigraph (see above): an *hypergraph* has edges that can connect multiple vertices, whereas a *multigraph* has multiple edges connecting a given pair of vertices.

**Multilevel graph.** When the vertices of a graph can be categorized into different "levels" (or "domains") and "ties" (i.e. edges) represent relationships within and across levels, then this

[2] Embedded hypergraphs: this follows the same modeling principles as *nested graphs* and *hypernodes* [303, 213].

graph structure is called a *multilevel graph* [369]. A $k$-level graph has $k$ different categories of vertices.

Multilevel graphs can also be defined through a *graph clustering* (or *graph coarsening*) task [121, 286]. Given an undirected weighted graph (Eq. B.8):

$$G = (V, E, c, w) \tag{B.8}$$

where $c : V \to \mathbb{R}_{\geq 0}$ are vertices weights and $w : E \to \mathbb{R}_{>0}$ are edges weights, and given a partition $N = N_1 \cup N_2 \cup \cdots \cup N_P$ of $G$ such that: *1)* the sum of the vertices weights in each $N_j$ is "about the same"; *2)* the sum of all edge weights of edges connecting all different pairs $N_j$ and $N_k$ is minimized. Then an *edge contraction* operation can be fulfilled over some partitions $N_j$ resulting in *quotient graph* $Q$ of $G$ that holds the same meaning as $G$ (i.e. provide a correct representation or estimate of $G$ with respect to the use case).

**Bipartite graph.** Given an undirected graph $G = (V, E)$, like in (Eq. B.2), this graph is called *bipartite* if $V$ can be partitioned in two sets $V_1$ and $V_2$ (i.e. two different groups) such that edges follow the constraint $E \subseteq V_1 \times V_2$ (i.e. relations are only between vertices of different groups). From there, a bipartite graph can be formalized as follows (Eq. B.9):

$$G = (V_1, V_2, E) \tag{B.9}$$

where $V_1$ and $V_2$ are disjoint and independent sets of vertices, and $E \subseteq V_1 \times V_2$ is the *edge set*.

When $E = V_1 \times V_2$ (i.e. each vertex of $V_1$ is connected to all vertices of $V_2$), the bipartite graph is said *complete* and is denoted $K_{m,n}$, where $m = \|V_1\|$ and $n = \|V_2\|$.

The notion of bipartite graph can be generalized to $n$-partites graphs by partitioning $V$ in $n$ groups such that the edge set still follows the different-group constraint (Eq. B.10):

$$E \subseteq \bigcup_{i \neq j} V_i \times V_j \tag{B.10}$$

### B.1.3 Graphical Models and Graph Related Forms

*Graphical models* refer to the use of a *graph structure* along with additional ad hoc axioms to represent and resolve a given problem instance that is not naturally represented in its initial form with a graph structure. This kind of form transposition provides additional means for resolving problems with the help of algorithms specifically designed for graph structures, problems that would have been hard to resolve otherwise (e.g. optimization and decision problems). The inherent universality of the graph structure for modeling purposes entails the

existence of a wide variety of derivatives and related forms. The following definitions provide a brief overview of the sub-families of *graphical models* and *graph related forms* that are present in decision problems. In each case, a semantic about vertices and edges must be explicitly defined to ensure a correct grounding with the application under consideration (i.e. the goal of the application, the associated algebra and/or algorithms).

**Temporal graph.** The will to use a *temporal graph* relates to the need for analysis of dynamical features in data graph models. The variety of object types to represent (e.g. asset status duration, event occurence time) and the multiple ways to manipulate the concept of time (e.g. reachability [182], causality [202], stability [242] or timing [379] analysis) entails that a practical form for the representation of dynamic systems may require more than a series of static graph snapshots set together (Figure B.2). Thus a single and general formal definition of temporal graphs may not apply (see [287] for a thorough analysis of the representation and manipulation challenges).

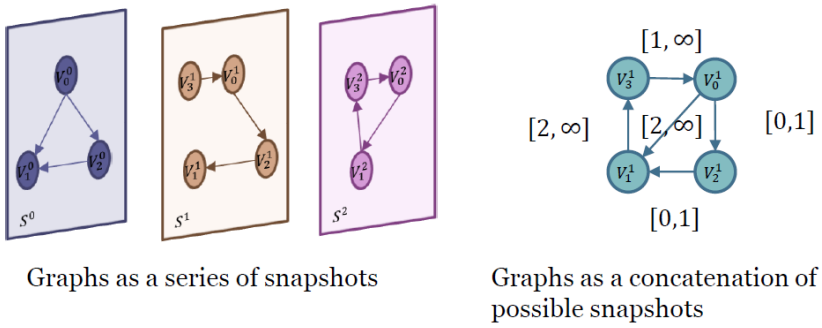
One possible approach to defining temporal graphs is through the concept of the "narrative of events" [171] (i.e. the temporally ordered set of actions), which relates to the *directedness* and the *timescale* of (emerging) patterns of relations.

The *directedness* is also known as the *time asymmetry property*, e.g. $v_i \rightarrow v_j$ should be distinguished from $v_j \rightarrow v_i$ if the time of occurence $t(v_j)$ of an event $v_j$ is after $t(v_i)$; that can be represented with the help of a *directed graph* and associated depending on the practical use case (e.g. a *directed property graph* where vertices have a "timestamp" property). The *timescale* of successive events leads to the *composition of events* (e.g. with the timestamped relations $v_i \rightarrow v_j$ and $v_j \rightarrow v_k$ one can infer $v_i \rightarrow v_k$) and the *parallelism of events* (e.g. two distinct interactions between $v_i$ and $v_j$ with similar temporal characteristics can be represented by two named arcs between $v_i$ and $v_j$); that, in turn, can be represented with the help of a *directed multigraph*.

In both the *directedness* and *timescale* perspectives, the temporal graph form should be chosen through the lenses of the *temporal graph algebra* [363] it conveys or the associated set of applicable analysis techniques (e.g. edge or node prediction through graph embedding [351, 68] or message passing [98]). Although this choice may seem complex, there exist two main models for dynamic graphs [98]: Discrete-Time Dynamic Graph (DTDG) (i.e. sequences of static graph snapshots taken at intervals in time) and Continuous-Time Dynamic Graph (CTDG) (i.e. timed list of events with a set of transformation operators on nodes/edges/features). Following on the CTDG case, a temporal graph can be formalized as follows (Eq. B.11):

$$G = (V, E, T, \rho) \tag{B.11}$$

where $V$ is the set of vertices, $E$ the set of edges, $T \in \Omega^T \times \Omega^T$ a time attribute (e.g. a period of observation) on the time domain $\Omega^T$, and $\rho : (V \cup E) \times \Omega^T \times \Omega^T \rightarrow \{0,1\}$ a presence function that indicates whether a given vertex or edge exists at the given time $T$.




Graphs as a series of snapshots

Graphs as a concatenation of all possible snapshots

Figure B.2: Approaches for temporal graph representation, from [222] and [107, §2].

**Knowledge graph & semantic graph.** Knowledge graphs basically designate a formal representation of Cyber-Physical Systems (CPSs) with help of a data graph model: vertices designate tangible/conceptual entities, and edges qualify the nature of their relationships. Figure B.3 provides a general example of this.

The "knowledge graph" phrase itself has a long definition drifting history, with first relevant usage in publications found in the 70's [8, Section A]. Ideas from *graph notations* (see above) and *description logic* [119] lead to the current situation where knowledge graphs can be found in two major flavors[3]. A common practice to distinguish these is to use the knowledge graph phrase for *property graph*-based models , and semantic graphs for logically-framed models.

In the first case (i.e. *property graph*-based), the graph semantics (i.e. the way knowledge should be interpreted) apply to the graph as a whole. Labeled edges provide directions about the nature of the relationship between vertices. Interesting characteristics about *property graph*-based graphs are that: *1)* vertices and edges can be augmented with valued properties for additional context definition (e.g. name, surname and date of birth for a Person vertex); *2)* vertices can relate multiple time to other vertices (e.g. various network sessions between host A and host B vertices); *3)* data schema can be expanded depending on use-cases needs by direct addition of edges and properties to the graph.

In the second case (i.e. *logically*-framed models), entities are connected with directed binary relations (Eq. B.12):

$$\langle s, p, o \rangle \tag{B.12}$$

[3] Knowledge graphs definitions can be more precisely staggered over four categories; see [8, Section A.3] on this.

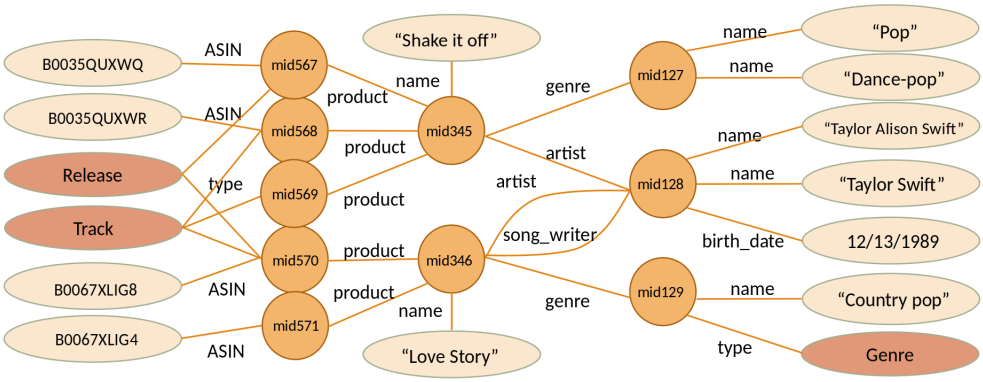



Figure B.3: Example of a knowledge graph.
Nodes represent *entities* (e.g. "mid345") or "entities types" (e.g. "Recording"). Edges are property-named "relationships" (e.g. "artist" for *(entity)-(entity)* relationship, "type" for *(entity)-(entity type)* relationships). The example is called "Product Graph - example for 2 songs" and is from [378].

where $s$ is a *subject vertex*, $p$ a *predicate edge* and $o$ an *object vertex*. Extensions to (Eq. B.12), such as $\langle s, p, o, t \rangle$ or $\langle s, p, o, [t_s, t_e] \rangle$, allow for time-stamped relations [68].

*Predicate edges* play a fundamental role in semantic graphs. Firstly, assuming that $s$ and $o$ vertices are called ressources [305], the graph semantics is per resource. This relates to the notion that predicates materialize a truth value[4] over given entities. For example, Socrates is a man is equivalent to the $(\texttt{Socrate}) \xrightarrow{\texttt{is a}} (\texttt{Man})$ graph. Secondly, predicate edges' directedness enforce a single possible interpretation of relations. For example, one cannot say that Man are Socrate, unless an explicit symmetry of above mentioned $\xrightarrow{\texttt{is a}}$ relation is stated.

Semantic graphs benefit from a set of implementation standards commonly referred to as Semantic Web [352] languages. Most important components of this framework are the Ressource Description Framework (RDF) [305] (a framework for representing information in the Web), the RDF Schema (RDFS) [82] (a data-modeling vocabulary for RDF data), and the Web Ontology Language (OWL) [54] (an ontology language for the Semantic Web). In short, RDFS gives structure (e.g. object class definition and domains/ranges constraints over relations) and OWL gives meaning (i.e. applicable logical entailments). Applicable relations are of the following kinds: *1) object properties* link individuals to individuals (e.g. `<hasMember>`); *2) datatype properties* link individuals to data values (e.g. `<releaseDate>`).

The need for formally defined meaning relates to the fact that various kinds of knowledge can be represented with knowledge graphs. Knowledge management typically distinguish the following organizations: *1) dictionary* as a collection of terms and their definitions; *2) taxonomy* used to create a hierarchy between terms; *3) thesaurus* for describing some basic relationships between terms (mostly by just specifying that two terms are related), and *4) ontology* for providing all that a dictionary, taxonomy and thesaurus do, but with better-qualified relationships

[4] General reflections on *predicate logic* and *syllogism* can be found in Wikipedia - Syllogism.

between concepts (e.g. `<<Actor wins Academy Award>>`).

Although property graphs and semantic graphs look the same but stand on their own in two separate application approaches, [126] argue that an hybrid situation is beneficial for representing CPSs. The authors notably recommend that "CPS graphs should [not systematically] be reduced to RDF graphs" as the description of complex systems with RDF may not be carried out without either a cumbersome reification process, or making an edge into a properly instantiated resource. Moreover, converting property graphs to semantic graphs would obfuscate, flatten and dissolve the relevant graph structure that is used by graph-theoretical algorithms. Meta-models like the NGIS-LD [105] provide a formal basis for representing property graphs using RDF/RDFS/OWL, and therefore allow for best of both world usage through back and forth conversion. Additional approaches, like RDF-star/SPARQL-star [96], work toward an hybridized solution through syntactic extensions for better reification. Requests usually devoted to property graphs become here applicable to semantic graphs (e.g. `<<Kubrik influencedBy Welles>> significance 0.8`). Experiments like Single-Triple Named Graphs, Singleton Properties, Nested Triples, and Nested Triple Patterns are analogous to RDF-star/SPARQL-star. Above examples indicate an on-going trend towards connectors or bridges that will enable hybridized property/semantic graph solutions.

**Similarity graphs.** Given a set of data points $X = \{x_1, x_2, \dots x_n\}$ and some notion of *similarity* measure $s_{ij} \geq 0$ between these data points, then one can define a mapping $X \rightarrow V$ (i.e. the vertex $v_i$ represents the data point $x_i$) and $S \rightarrow E$ (i.e. the arc or edge $(v_i, v_j)$ holds the similarity value $s_{ij}$) to build a graph representing weighted similarity relations between data points. Such construct is called a *similarity graph* and similarity values are commonly stored in a weighted adjacency matrix .

Usual techniques to model the local neighborhood relationships between the data points [359, §2.2] are to build *1)* an *$\epsilon$-neighborhood graph* (i.e. connect all points whose pairwise distances are smaller than $\epsilon$); *2)* a *k-nearest neighbor graph* (i.e. connect vertex $v_i$ with vertex $v_j$ if $v_j$ is among the k-nearest neighbors of $v_i$); or *3) similarity weighted fully connected graph* (i.e. connect all points with positive similarity and weight all edges by $s_{ij}$). For the fully connected graph technique, the use of a similarity function accounting for local neighborhood modeling is expected; the *Gaussian similarity function* (Eq. B.13) is part of it:

$$s(x_i, x_j) = exp(-\|x_i - x_j\|^2/(2\sigma^2)) \tag{B.13}$$

where the parameter $\sigma$ controls the width of the neighborhoods.

**Transition system.** A *transition system* specifies the allowed evolutions of a given abstract reactive system. It is commonly represented by a directed graph. It can be formalized, in its minimalist form, by the quadruple (Eq. B.14):

$$\mathscr{T} = (S, I, \mathscr{A}, \delta) \tag{B.14}$$

where $S$ is a set of states, $I \subseteq S$ a non-empty subset of initial states, $\mathscr{A}$ a set of actions, and $\delta$ a total transition relation $\delta \subseteq S \times \mathscr{A} \times S$ (i.e. for every state there must exist an action that leads to a subsequent state). A run of $\mathscr{T}$ is an infinite sequence $\rho = s_0 s_1 \ldots$ of states $s_i \in S$ such that $s_0 \in I$ and $\forall i \in \mathbb{N}, (s_i, A_i, s_{i+1}) \in \delta$ for some $A_i \in \mathscr{A}$.

When $S$ and $\mathscr{A}$ are finite, $\mathscr{T}$ is said *finite*; thus relating a transition system to a Finite State Machine (FSM), i.e. when it can be in exactly one of a finite number of states at any given time.

An extended form $\mathscr{T} = (S, I, \mathscr{A}, \delta, AP, L)$, where $AP$ is a set of atomic propositions and $L : S \rightarrow 2^{AP}$ a labeling function, allows for including propositional constraints and labeled states is given [70, Section 2.1]. This extended form allows for applications such as *model checking*, simulation through *Petri nets* and *Proclets* [374] or specific modeling constructs like BPMN[5] and Event-driven Process Chain (EPC). It also sets a link with other theoretical approaches like *Kripke models* and Abstract Rewriting Systems (ARSs) [333].

**Decision/condition/evolutionary trees.** A *decision tree* is a special instance of a *directed tree* used as a decision support tool or as a convenient representation of a sequential decision process. In general terms, vertices represent conditions and arcs represent decisions (e.g. a `if <condition> then <...> else <...>` clause).

In more formal terms, a decision tree is constructed using a directed graph from (Eq. B.1) with a set of vertices $V$ split into three disjoint sets $V = \mathscr{D} \cup \mathscr{C} \cup \mathscr{T}$ of decision (i.e. where the decision maker selects an action), chance (i.e. where the decision is taken randomly), and terminal (i.e. the end of the decision process) vertices, respectively [53]. For each arc $a \in A$ and its associated $(v_1, v_2)$ pair, the $v_1 \in V$ denote the *parent* vertex and the $v_2 \in V$ denote the *child* vertex. Terminal vertices can only be child vertices (i.e. terminal vertices have no children).

From this preliminary set up, a *decision tree* is defined as follows (Eq. B.15):

$$DT = (G, y, p) \tag{B.15}$$

where $y : A \rightarrow \mathbb{R}$ is a *payoff* mapping function associating a valued result (e.g. a gain) to the arc stemming from the $v \in \{\mathscr{D}, \mathscr{C}\}$ parent vertex, and $p : \{a \in A : a1 \in \mathscr{C}\} \rightarrow [0, 1]$ is a mapping function associating a probability of selection to the arcs, this probability being subject to a

[5] https://www.bpmn.org/

*correctly defined probability* constraint $\forall v \in \mathscr{C} : \sum_{a \in A, a_1 = v} p(a) = 1$.

In the industry, decision trees are widely used in the field of reliability engineering and can be found under the terms FTA (Fault Tree Analysis) and DFT (Decision Fault Tree). Such kind of trees can be produced with decision support and dependability modeling tools like KB3[6], TopEvent FTA[7] or OpenFTA[8].

**Some additional graph-related models, in short.** In the **dependency/influence/probabilistic diagrams and graphs** category: Data Flow Diagram (DFD) are a way of representing a flow of data through a process or a system, e.g. in threat modeling [5, 29]. In the **ensemblist diagrams and graphs** category: *Euler diagrams* and *Venn diagrams* (Figure B.4) are graphical representations of set-theoretic basic operations[9] over a set of points. They allow for visually exploring set intersections, through intersection formulas, or through trees (typically formulated in Newick syntax)[10]. *Spider diagrams*, a.k.a. *constraints diagrams*, are an extension of Euler and Venn diagrams with a logical tree. Relations within this tree are defined by a logical `OR` operator ($\vee$ logical symbol). A quantifier may be associated to the `OR` operators (i.e. expressions involving two place predicates). In the **lattices** category: lattices as an abstract structure in order theory, such as in [263] where they allow to define a formal context (Eq. B.16) using Formal Concept Analysis (FCA):

$$\mathscr{F} = (G, M, I) \tag{B.16}$$

where $G$ is a set of objects, $M$ a set of attributes, and $I \subseteq G \times M$ the description of objects through attributes via a binary relation. Lattices can be projected onto a simplicial complex [34], which establishes a connection with topology and graph theory for applications in analyzing the structure of the lattice. Lattices are also known as *lattice graphs*, *mesh graphs*, or *grid graphs*.

## B.2 Anomaly Detection: State of the Art

The following Table B.1 provides complete details on the shortlist of 55 references from the literature review on the anomaly detection techniques (Chapter 2), including a summary of each approach.

[6] https://www.edf.fr/en/the-edf-group/inventing-the-future-of-energy/r-d-global-expertise/our-offers/simulation-softwares/kb3

[7] https://www.fault-tree-analysis.com/free-fault-tree-analysis-software

[8] https://sourceforge.net/projects/openfta/

[9] Within set theory, fundamental operations for constructing new sets from given sets are: unions, intersections, complements and the cartesian product.

[10] https://icytree.org/

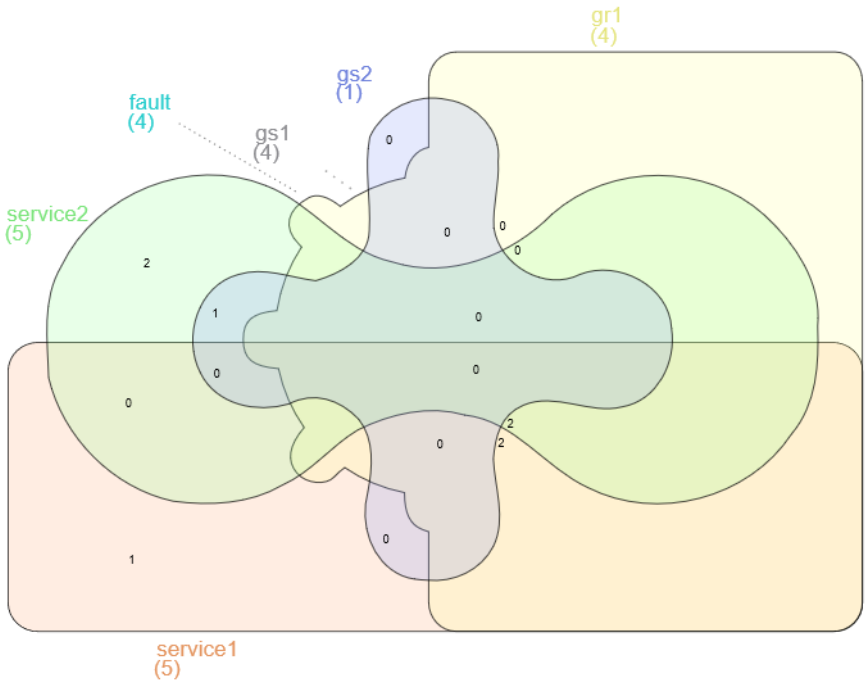



Figure B.4: Venn diagram example

Venn diagrams are part of the ensemblist diagrams family. The sets in this diagram represent a hierarchy of network function for service delivery. The assets participating in the system operation can simultaneously be part of a redundancy group ($gr1$), part of a service chain ($gs1$ and $gs2$), and be identified as being in a faulty state ($fault$). The intersection of sets corresponds to the application of a logical `AND` operator. The numbers within the sets indicate the count of declared assets that satisfy an intersection of sets. It allows to highlight common elements in the functioning and situation signatures (e.g. `[service1] and [gr1] and [service2] and [fault]: e2 e3`). Diagram built using [141].

Table B.1: Anomaly detection models comparison table, as per a subset of the evaluation criteria defined in §2.4.1. In the *Name or short title* column, we indicate the name of the method when provided by the authors, it is the abbreviated title of the paper otherwise. Abbreviations: *Ref.* = bibliographical reference, *SC-SH* = Smart-Cities & Smart-Homes, *n.a.* = non applicable.

| Theme | Ref. | Year | Name or short title | Data structures within method |
|---|---|---|---|---|
| | *Short description* | | | |
| **Design** | | **Graph-based** | | |
| Net-IT | [217] | 2019 | Measuring Network Reliability … | Mixed (seq. + graph) |
| | *Probabilistic measure based on network performance through iterative link removal.* | | | |
| **Design** | | **Knowledge-based** | | |
| Energy systems | [79] | 2018 | Ontology-driven multilevel sequential … | Mixed (seq. + graph) |
| | *Mining sequential patterns on a knowledge graph.* | | | |
| Net-IT | [19] | 2023 | Inferring Threatening IoT Dependencies … | Mixed (seq. + tab.) |
| | *Knowledge graph construction for dependency calculus between devices.* | | | |
| **Design** | | **Model checking** | | |
| Net-IT | [177] | 2006 | Perracotta | Sequential data |
| | *Association rule learning from software process logs, followed by checking for non-conforming event sequences using activity patterns expressed in rules in a form close to Computation Tree Logic (CTL).* | | | |
| **Design** | | **Rule-based** | | |
| Energy systems | [218] | 2008 | Gestion de la complexité dans … | Graph |
| | *Modeling systems logically and exploring state space to identify potential failure scenarios.* | | | |
| **Detect. & Class.** | | **Graph-based** | | |
| CyberSec | [375] | 2007 | GBAD-MDL/P/MPS | Graph |
| | *Detecting node/edge modifications, insertions, and deletions.* | | | |
| CyberSec | [387] | 2017 | Measuring Similarity between Cyber … | Tabular |
| | *Computing similarity between incident reports using graph-based clustering and assignment algorithm.* | | | |

| Theme | Ref. | Year | Name or short title | Data structures within method |
|---|---|---|---|---|
| | *Short description* | | | |
| CyberSec | [326] | 2020 | MIDAS | Graph streams |
| | *Detecting sudden bursts of activity with repeated nodes or edges by computing the evolution of event count.* | | | |
| CyberSec | [233] | 2020 | ADSAGE | Sequential data |
| | *Classifying the edge representing the predicted next event from audit events as attributed graph edges using a Feedforward Neural Network (FFNN) model for classification and a Recurrent Neural Network (RNN) model for predicting the next event.* | | | |
| CyberSec | [207] | 2021 | A Picture is Worth 1,000 Rows | Tabular |
| | *Graph-based clustering to identify related alerts, using centrality measures to prioritize clusters.* | | | |
| Fraud analytics | [210] | 2023 | PANG | Graph |
| | *Anomaly detection and fraud analytics in relational or graph-structured data, involving learning graph patterns, ranking and filtering them, graph embedding, and training a classifier using the bag-of-subgraphs concept.* | | | |
| Net-IT | [93] | 2015 | DNODA/CNA | Graph |
| | *Analyzing node attribute distribution in relation to global, local, and community perspectives using the Direct Neighbour Outlier Detection Algorithm (DNODA) and Community Neighbor Algorithm (CNA) techniques.* | | | |
| Net-IT | [320] | 2017 | Model-Based Probabilistic Reasoning … | Mixed (sequential data + graph) |
| | *Event classification using a probabilistic model (Bayesian Network) representing network topology and observables, accounting for alarm diffusion phenomena, provided by an expert or learned from incomplete data.* | | | |
| Net-IT | [69] | 2022 | Graph Neural Network Based Root … | Time series |
| | *RCA by constructing a device dependency graph structure using a neural network from performance time series, and performing node classification over the dependency graph using a Graph Convolutional Network (GCN).* | | | |
| SC-SH | [317] | 2020 | Anomaly Detection in Smart Homes … | Sequential data |
| | *Behavior classification on event logs using a probabilistic model (Bayesian Network).* | | | |
| **Detect. & Class.** | | | **Knowledge-based** | |
| CyberSec | [30] | 2004 | Rich Event Representation for … | Graph |

| Theme | Ref. | Year | Name or short title | Data structures within method |
|---|---|---|---|---|
| | *Short description* | | | |
| | *Classification of computer forensic scenarios by extracting facts from digital event logs and applying expert-made correlation rules using the RETE algorithm.* | | | |
| CyberSec | [324] | 2010 | An Ontology-Based Forensic … | Graph |
| | *Classification and risk analysis of attacks using subsumption-based and graph-traversal-based methods.* | | | |
| CyberSec | [249] | 2013 | An Ontology-Based Forensic … | Graph |
| | *SPARQL-based situation understanding by exploring evidence data from a knowledge graph.* | | | |
| CyberSec | [258] | 2015 | PACO | Sequential data (network) |
| | *Binary classification of network traffic using an Instance-Based Learning agent and the PACO OWL-DL ontology.* | | | |
| Generic | [292] | 2020 | StreamingMASSIF | Sequential data |
| | *Processing of data streams using Semantic Web techniques and Complex Event Processing (CEP).* | | | |
| SC-SH | [263] | 2018 | MRGS/MFCSoFCA+symACS | Sequential data |
| | *Trace classification comparing the similarity of traces to behavior patterns derived from Multi-Resolution Grid-based Segmentation (MRGS) and Multi-Frequency Co-occurrence Similarity (MFCS) analysis, in conjunction with Formal Concept Analysis (FCA).* | | | |
| **Detect. & Class.** | | | **Markov model** | |
| CyberSec | [259] | 2000 | A Markov Chain Mo… | Sequential data |
| | *Threshold-based event classification using a transition probability matrix. The anomaly detection threshold is determined based on the low probability of a temporal behavior observed in the recent past.* | | | |
| **Detect. & Class.** | | | **ML-based** | |
| CyberSec | [137] | 2018 | RL4AD | Sequential data (network) |
| | *Anomaly detection on sequential data using a hybrid reinforcement learning / deep neural network model.* | | | |
| Energy systems | [66] | 2015 | Isolation Forest… | Data points |
| | *Benchmarking the Isolation Forest method over 30+ datasets.* | | | |
| Generic | [113] | 2008 | iForest | Data points |

| Theme | Ref. | Year | Name or short title | Data structures within method |
|---|---|---|---|---|
| | *Short description* | | | |
| | *Anomaly detection (outliers) by scoring data points based on their distance from the root of a binary search tree.* | | | |
| Generic | [272] | 2016 | EncDec-AD | Time series |
| | *Anomaly likelihood based on the reconstruction error of a time series with respect to a learned normal model, which is a single-layer Long Short Term Memory (LSTM).* | | | |
| Generic | [344] | 2020 | Introduction aux auto-encodeurs | Time series |
| | *Anomaly detection using the reconstruction error obtained from an EncDec model. The anomaly detection threshold is defined as one standard deviation above the mean reconstruction error.* | | | |
| Generic | [278] | 2020 | Timeseries Anomaly Detection… | Time series |
| | *Anomaly detection time series data using the reconstruction error obtained from a convolutional autoencoder. The anomaly detection threshold is defined as the maximum mean absolute error loss value observed during training.* | | | |
| Industry 4.0 | [289] | 2020 | How Airbus Detects Anomalies … | Sequential data |
| | *Anomaly detection on sequential data using the reconstruction error obtained from an LSTM autoencoder.* | | | |
| Industry 4.0 | [238] | 2020 | IForestASD | Sequential data |
| | *Anomaly detection on sequential data using an extended version of the Isolation Forest technique.* | | | |
| Industry 4.0 | [251] | 2021 | Approche de traitement des … | Sequential data |
| | *Model-based classification (Bayes, decision tree, deep learning) of data over a sliding-window bag of features.* | | | |
| Industry 4.0 | [67] | 2021 | SM-iForest | Time series |
| | *Anomalous data segments detected using Isolation Forest and a similarity measure.* | | | |
| Net-IT | [342] | 2013 | Network Failure Detection … | Sequential data |
| | *SysLog logs clustered using mined patterns and non-negative matrix factorization.* | | | |
| Net-IT | [342] | 2013 | Network Failure Detection … | Sequential data |
| | *RCA using user messages from social networks, where the messages are first classified using a Support Vector Machine (SVM) model.* | | | |
| Net-IT | [136] | 2020 | Single Model Based Anomaly … | Time series |

| Theme | Ref. | Year | Name or short title | Data structures within method |
|---|---|---|---|---|
| | *Short description* | | | |
| | *Anomalous data segments detected across multiple time series using an Isolation Forest algorithm applied to z-Scores, which represent deviations of each key performance indicator value from its own patterns.* | | | |
| Net-IT | [174] | 2021 | FastTrack/AugSplicing | Graph streams |
| | *Detection of synchronous events in multidimensional data streams based on the identification of dense blocks of data (sub-tensors) and iterative partitioning for near-real-time analysis.* | | | |
| **Detect. & Class.** | | | **Model checking** | |
| Business process | [3] | 2011 | Cost-Based Conformance Chec… | Sequential data |
| | *Event logs are replayed on a process model to identify skipped activities and inserted activities.* | | | |
| CyberSec | [267] | 2004 | Scenario Graphs and Attack … | Graph |
| | *Definition of rules in the form of graphs, followed by rule execution for activity detection.* | | | |
| **Detect. & Class.** | | | **Rule-based** | |
| CyberSec | [364] | 1999 | BRO | Sequential data (network) |
| | *Rule-based alerting on live network traffic using a multi-agent software system.* | | | |
| CyberSec | [281] | 2003 | Using CLIPS to Detect Net… | Sequential data (network) |
| | *Rule-based alerting on network traffic by combining the SNORT and CLIPS tools and including string pattern matching, certainty factors and time-stamp operators.* | | | |
| CyberSec | [322] | 2014 | Co-FQL | Sequential data (network) |
| | *Fuzzy rule-based classification of network traffic packets, where the appropriate combination of rules is learnt and decided with help of Q-Learning (reinforcement learning algorithm to learn the value of an action in a particular state).* | | | |
| CyberSec | [315] | 2018 | Cognitive CyberSecurity (CCS) | Graph |
| | *Anomaly detection using a SWRL-based rule set on RDF data, with expert knowledge from a complementary knowledge graph (UCO), and additional analytics (statistical, graph) to enrich the knowledge graph.* | | | |
| Net-IT | [280] | 2006 | Automatic Detection of SLS … | Sequential data |

| Theme | Ref. | Year | Name or short title | Data structures within method |
|---|---|---|---|---|
| | *Short description* | | | |
| | *Rule-based situation classification on event logs using the CLIPS tool.* | | | |
| **Diagnostic Aid** | | **Graph-based** | | |
| CyberSec | [130] | 2016 | TerminAPTor | Mixed (seq. + graph) |
| | *Relational structure discovery between attack alerts using information flows and tag propagation approaches.* | | | |
| Net-IT | [329] | 2019 | SAKURA | Mixed (seq. + graph) |
| | *RCA by solving a logical dependency graph with observational values using a Satisfiability Modulo Theory (SMT) solver.* | | | |
| Net-IT | [372] | 2020 | AIOps Series V : RCA Based … | Mixed (seq. + graph) |
| | *Unsupervised out-of-trend alerting using LSTM and probability density on transaction volume data stored in a KG.* | | | |
| Net-IT | [328] | 2020 | Fault Management of… | Mixed (seq. + graph) |
| | *RCA by constructing a dependency graph based on prior knowledge of network topology, and then calculating constraints (SAT) based on observed events and their testability.* | | | |
| Net-IT | [92] | 2023 | Clustering of Liv… | Mixed (seq. + graph) |
| | *Alarm clustering by pre-filtering events based on network topology, followed by clustering based on inter-arrival times.* | | | |
| **Diagnostic Aid** | | **Knowledge-based** | | |
| CyberSec | [302] | 2013 | Automated Classification of Computer … | Mixed (seq. + graph) |
| | *Situation classification is performed using tableau calculus, leveraging an ontology designed for network attacks.* | | | |
| CyberSec | [301] | 2014 | A Formalised Ontology for Network … | Mixed (seq. + graph) |
| | *Situation classification is performed using tableau calculus, leveraging an ontology designed for network attacks.* | | | |
| CyberSec | [15] | 2016 | A Semantic-Web-Technology-… | Graph |
| | *Situation understanding with both SWRL-based and SPARQL-based approaches, using an ad hoc ontology and an RDF-based representation of facts and events.* | | | |
| Energy systems | [184] | 2022 | Root Cause Analysis in the Industrial … | Graph |
| | *RCA using both SWRL-based and SPARQL-based approaches on data stored in a KG.* | | | |

| Theme | Ref. | Year | Name or short title | Data structures within method |
|---|---|---|---|---|
| | *Short description* | | | |
| Industry 4.0 | [62] | 2018 | FOLIO | Tabular |
| | *RCA by annotating and processing event streams using rules, leveraging FMEA mapped onto the FOLIO ontology, and utilizing SWRL rules from Fault Tree Analysis (FTA).* | | | |
| Net-IT | [390] | 2023 | KTeleBERT | Mixed (seq. + unstr.) |
| | *RCA using a Graph Convolutional Network (GCN) on network events and recorded network documentation elements stored in a knowledge graph.* | | | |
| **Diagnostic Aid** | | | **Model checking** | |
| Software engineering | [357] | 2014 | Perfume | Sequential data |
| | *Association rule learning over logs using Timed Propositional Temporal Logic (TPTL).* | | | |

## B.3 Anomaly Detection: Dataset for Experiments

The following listing (Listing B.1) depicts a fictitious network infrastructure (Figure B.5) that was developed and used for anomaly detection experiments. It was specifically created for the Orange Open Tech Days 2023 "*Semantical anomaly sensing - Recommend remediation solutions using knowledge graphs*" demo (Chapter A).

```
# === PREFIXES ===================================================================
# --- Basic ---
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix prov: <http://www.w3.org/ns/prov#> .

# --- Domain specific ---
@prefix seas: <https://w3id.org/seas/> .
@prefix org: <http://www.w3.org/ns/org#> .

# --- NORIA-O ---
@prefix noria: <https://w3id.org/noria/ontology/> .
@base <https://w3id.org/noria/> .

# ================================================================================
# --- Application ---
<object/APP_99995_TST>
  a noria:Application;
  prov:wasDerivedFrom "noria-sdlri2023-toy-example";
  noria:applicationBusinessImportance "Not defined";
  noria:applicationFunctionalDomain "Enterprise Management";
  noria:applicationFunctionalSubDomain "Accounting Management";
  noria:applicationModelIdentifier "AM099995";
  noria:applicationNumericalIdentifier "99995";
  noria:applicationShortIdentifier "OOTD_TST";
  noria:applicationType <kos/application/type/application>;
  noria:elementManagedBy <agent/TEAM_SDLRI>;
  rdfs:comment "NORIA-SDLRI2023 application";
  rdfs:label "OOTD_TST" .

<object/APP_99996_TSI>
  a noria:Application;
  prov:wasDerivedFrom "noria-sdlri2023-toy-example";
  noria:applicationBusinessImportance "Not defined";
  noria:applicationFunctionalDomain "Enterprise Management";
  noria:applicationFunctionalSubDomain "Accounting Management";
  noria:applicationModelIdentifier "AM099996";
  noria:applicationNumericalIdentifier "99996";
  noria:applicationShortIdentifier "OOTD_TSI";
  noria:applicationType <kos/application/type/infrastructure>;
  noria:elementManagedBy <agent/TEAM_SDLRI>;
  rdfs:comment "NORIA-SDLRI2023 infrastructure";
  rdfs:label "OOTD_TSI" .

# --- Network elements ---
<object/RES_TOY_srv_tst_1>
  a noria:Resource;
  prov:wasDerivedFrom "noria-sdlri2023-toy-example";
  noria:resourceForApplication <object/APP_99995_TST>;
  noria:resourceHostName "srv_tst_1";
  noria:resourceLogisticId "srv_tst_1";
  noria:resourceInstallationDate "2020-07-09";
  noria:resourceType <kos/Resource/type/server>;
  noria:resourceManagedBy <agent/TEAM_SDLRI>;
  noria:resourceProductModel <object/PRODUCT_Orange_SDLRI_SERVER>;
  seas:subSystemOf <object/RES_TOY_rack_01> .

```

```
<object/RES_TOY_srv_tst_2>
  a noria:Resource;
  prov:wasDerivedFrom "noria-sdlri2023-toy-example";
  noria:resourceForApplication <object/APP_99995_TST>;
  noria:resourceHostName "srv_tst_2";
  noria:resourceLogisticId "srv_tst_2";
  noria:resourceInstallationDate "2020-07-09";
  noria:resourceType <kos/Resource/type/server>;
  noria:resourceManagedBy <agent/TEAM_SDLRI>;
  noria:resourceProductModel <object/PRODUCT_Orange_SDLRI_SERVER>;
  seas:subSystemOf <object/RES_TOY_rack_02> .

<object/RES_TOY_rt_tsi_1>
  a noria:Resource;
  prov:wasDerivedFrom "noria-sdlri2023-toy-example";
  noria:resourceForApplication <object/APP_99996_TSI>;
  noria:resourceHostName "rt_tsi_1";
  noria:resourceLogisticId "rt_tsi_1";
  noria:resourceInstallationDate "2020-07-01";
  noria:resourceType <kos/Resource/type/network-element>;
  noria:resourceManagedBy <agent/TEAM_SDLRI>;
  noria:resourceProductModel <object/PRODUCT_Orange_SDLRI_ROUTER>;
  seas:subSystemOf <object/RES_TOY_rack_01> .

<object/RES_TOY_rt_tsi_2>
  a noria:Resource;
  prov:wasDerivedFrom "noria-sdlri2023-toy-example";
  noria:resourceForApplication <object/APP_99996_TSI>;
  noria:resourceHostName "rt_tsi_2";
  noria:resourceLogisticId "rt_tsi_2";
  noria:resourceInstallationDate "2020-07-01";
  noria:resourceType <kos/Resource/type/network-element>;
  noria:resourceManagedBy <agent/TEAM_SDLRI>;
  noria:resourceProductModel <object/PRODUCT_Orange_SDLRI_ROUTER>;
  seas:subSystemOf <object/RES_TOY_rack_02> .

# --- Organization ---
<agent/ORG_orange-france>
  a org:Organization;
  org:hasUnit <agent/OU_ORANGE_SDLRI> .

<agent/OU_ORANGE_SDLRI>
  a org:OrganizationalUnit;
  prov:wasDerivedFrom "noria-sdlri2023-toy-example";
  rdfs:label "Orange SDLRI department";
  org:hasUnit <agent/TEAM_SDLRI>;
  org:classification <kos/org/org-classification/manufacturer> .

<agent/TEAM_SDLRI>
  a org:OrganizationalUnit;
  prov:wasDerivedFrom "noria-sdlri2023-toy-example";
  noria:teamInstructions "Enjoy the demo!";
  rdfs:label "SDLRI demo team";
  org:classification <kos/org/ou-classification/tsg>;
  org:identifier "SDLRI" .

<object/PRODUCT_Orange_SDLRI_ROUTER>
  a noria:ProductModel;
  prov:wasDerivedFrom "noria-sdlri2023-toy-example";
  noria:productManufacturedBy <agent/OU_ORANGE_SDLRI>;
  rdfs:label "Orange_SDLRI_ROUTER" .

<object/PRODUCT_Orange_SDLRI_SERVER>
  a noria:ProductModel;
  prov:wasDerivedFrom "noria-sdlri2023-toy-example";
  noria:productManufacturedBy <agent/OU_ORANGE_SDLRI>;
  rdfs:label "Orange_SDLRI_SERVER" .

# --- Network interfaces ---
<object/NI_TOY_rt_tsi_2_rt_tsi_1>
  a noria:NetworkInterface;
  prov:wasDerivedFrom "noria-sdlri2023-toy-example";
  rdfs:label "Te-1-0";
  rdfs:comment "Back-to-back RES_TOY_rt_tsi_2 <=> RES_TOY_rt_tsi_1";
  noria:networkInterfaceOf <object/RES_TOY_rt_tsi_2> ;
  noria:networkInterfaceConnects <object/NL_TOY_rt_tsi_1_rt_tsi_2> ;
```

```
  noria:networkInterfaceAdministrativeStatus <kos/Resource/AdministrativeState/unlocked>;
  noria:networkInterfaceOperationalStatus <kos/Resource/OperationalState/disabled>.

# --- Network links ---
<object/NL_TOY_srv_tst_1_rt_tsi_1>
  a noria:NetworkLink;
  prov:wasDerivedFrom "noria-sdlri2023-toy-example";
  noria:networkLinkId "NL_TOY_srv_tst_1_rt_tsi_1";
  noria:networkLinkTerminationResource
    <object/RES_TOY_srv_tst_1>,
    <object/RES_TOY_rt_tsi_1>.

<object/NL_TOY_srv_tst_2_rt_tsi_2>
  a noria:NetworkLink;
  prov:wasDerivedFrom "noria-sdlri2023-toy-example";
  noria:networkLinkId "NL_TOY_srv_tst_2_rt_tsi_2";
  noria:networkLinkTerminationResource
    <object/RES_TOY_srv_tst_2>,
    <object/RES_TOY_rt_tsi_2>.

<object/NL_TOY_rt_tsi_1_rt_tsi_2>
  a noria:NetworkLink;
  prov:wasDerivedFrom "noria-sdlri2023-toy-example";
  noria:networkLinkId "NL_TOY_rt_tsi_1_rt_tsi_2";
  noria:networkLinkTerminationResource
    <object/RES_TOY_rt_tsi_1>,
    <object/RES_TOY_rt_tsi_2>.

<object/NL_TOY_rt_tsi_1_mid_1>
  a noria:NetworkLink;
  prov:wasDerivedFrom "noria-sdlri2023-toy-example";
  noria:networkLinkId "NL_TOY_rt_tsi_1_mid_1";
  noria:networkLinkTerminationResource
    <object/RES_TOY_rt_tsi_1>,
    <object/RES_TOY_rt_bkb_1>.  # Host in the backbone network

<object/NL_TOY_rt_tsi_2_mid_2>
  a noria:NetworkLink;
  prov:wasDerivedFrom "noria-sdlri2023-toy-example";
  noria:networkLinkId "NL_TOY_rt_tsi_2_mid_2";
  noria:networkLinkTerminationResource
    <object/RES_TOY_rt_tsi_2>,
    <object/RES_TOY_rt_bkb_2>.  # Host in the backbone network

# --- Alarm ---
<event/AIS_TOY_srv_tst_1_ifDown>
  a noria:EventRecord;
  prov:wasDerivedFrom "noria-sdlri2023-toy-example";
  noria:alarmMonitoredAttribute "InterfaceOperStatus";
  noria:alarmSeverity <kos/Notification/Severity/PerceivedSeverity/major>;
  noria:logOriginatingManagedObject <object/RES_TOY_srv_tst_1>;
  noria:logOriginatingManagementSystem <object/APP_AM008897>;  # Application for alarm management
  noria:loggingTime "2023-09-08 03:31:17.0";
  dcterms:description "Interface OperStatus down, link 'srv_tst_1_rt_tsi_1'";
  dcterms:identifier "AIS_TOY_srv_tst_1_ifDown";
  dcterms:type <kos/Notification/EventType/communicationsAlarm>.

<event/AIS_TOY_rt_tsi_1_TimeOut>
  a noria:EventRecord;
  prov:wasDerivedFrom "noria-sdlri2023-toy-example";
  noria:alarmMonitoredAttribute "ResourcePollingStatus";
  noria:alarmSeverity <kos/Notification/Severity/PerceivedSeverity/warning>;
  noria:logOriginatingManagedObject <object/RES_TOY_rt_tsi_1>;
  noria:logOriginatingManagementSystem <object/APP_AM008897>;  # Application for alarm management
  noria:loggingTime "2023-09-08 03:38:00.0";
  dcterms:description "ResourcePollingStatus exceeded timeout threshold, resource 'rt_tsi_1'";
  dcterms:identifier "AIS_TOY_rt_tsi_1_TimeOut";
  dcterms:type <kos/Notification/EventType/equipmentAlarm>.

<event/AIS_TOY_rt_tsi_2_ifDown>
  a noria:EventRecord;
  prov:wasDerivedFrom "noria-sdlri2023-toy-example";
  noria:alarmMonitoredAttribute "InterfaceOperStatus";
  noria:alarmSeverity <kos/Notification/Severity/PerceivedSeverity/major>;
  noria:logOriginatingManagedObject <object/RES_TOY_rt_tsi_2>;
  noria:logOriginatingManagementSystem <object/APP_AM001997>;  # Application for alarm management
```

```
  noria:loggingTime "2023-09-08 03:31:17.0";
  dcterms:description "Interface OperStatus down, link 'rt_tsi2_rt_tsi_1'";
  dcterms:identifier "AIS_TOY_rt_tsi_2_ifDown";
  dcterms:type <kos/Notification/EventType/communicationsAlarm>.

# --- Logs ---
<event/LOG_TOY_srv_tst_2_sshd_n1_01>
  a noria:EventRecord;
  prov:wasDerivedFrom "noria-sdlri2023-toy-example";
  noria:alarmMonitoredAttribute "Secure remote-login and remote-execution";
  noria:alarmSeverity <kos/Notification/Severity/PerceivedSeverity/indeterminate>;
  noria:logOriginatingManagedObject <object/RES_TOY_srv_tst_2>;
  noria:logOriginatingManagementSystem <object/APP_AM001997>;  # Application for alarm management
  noria:loggingTime "2023-09-08 02:20:00.0";
  noria:logText "sshd:ConnectionEstablished";
  noria:logOriginatingAgent <agent/USER_legitimate>;
  dcterms:identifier "AIS_TOY_srv_tst_2_sshd_n1_01".

<event/LOG_TOY_srv_tst_2_sshd_n1_02>
  a noria:EventRecord;
  prov:wasDerivedFrom "noria-sdlri2023-toy-example";
  noria:alarmMonitoredAttribute "Secure remote-login and remote-execution";
  noria:alarmSeverity <kos/Notification/Severity/PerceivedSeverity/indeterminate>;
  noria:logOriginatingManagedObject <object/RES_TOY_srv_tst_2>;
  noria:logOriginatingManagementSystem <object/APP_AM001997>;  # Application for alarm management
  noria:loggingTime "2023-09-08 02:20:15.0";
  noria:logText "sshd:AcceptedPassword";
  noria:logOriginatingAgent <agent/USER_legitimate>;
  dcterms:identifier "AIS_TOY_srv_tst_2_sshd_n1_02".

<event/LOG_TOY_srv_tst_2_sshd_n2_01>
  a noria:EventRecord;
  prov:wasDerivedFrom "noria-sdlri2023-toy-example";
  noria:alarmMonitoredAttribute "Secure remote-login and remote-execution";
  noria:alarmSeverity <kos/Notification/Severity/PerceivedSeverity/indeterminate>;
  noria:logOriginatingManagedObject <object/RES_TOY_srv_tst_2>;
  noria:logOriginatingManagementSystem <object/APP_AM001997>;  # Application for alarm management
  noria:loggingTime "2023-09-08 02:25:00.0";
  noria:logText "sshd:ConnectionEstablished";
  noria:logOriginatingAgent <agent/USER_rogue>;
  dcterms:identifier "AIS_TOY_srv_tst_2_sshd_n2_01".

<event/AIS_TOY_srv_tst_2_sshd_n2_02>
  a noria:EventRecord;
  prov:wasDerivedFrom "noria-sdlri2023-toy-example";
  noria:alarmMonitoredAttribute "Secure remote-login and remote-execution";
  noria:alarmSeverity <kos/Notification/Severity/PerceivedSeverity/warning>;
  noria:logOriginatingManagedObject <object/RES_TOY_srv_tst_2>;
  noria:logOriginatingManagementSystem <object/APP_AM001997>;  # Application for alarm management
  noria:loggingTime "2023-09-08 02:25:10.0";
  noria:logText "sshd:FailedPassword";
  noria:logOriginatingAgent <agent/USER_rogue>;
  dcterms:identifier "AIS_TOY_srv_tst_2_sshd_n2_02".

<event/AIS_TOY_srv_tst_2_sshd_n2_03>
  a noria:EventRecord;
  prov:wasDerivedFrom "noria-sdlri2023-toy-example";
  noria:alarmMonitoredAttribute "Secure remote-login and remote-execution";
  noria:alarmSeverity <kos/Notification/Severity/PerceivedSeverity/indeterminate>;
  noria:logOriginatingManagedObject <object/RES_TOY_srv_tst_2>;
  noria:logOriginatingManagementSystem <object/APP_AM001997>;  # Application for alarm management
  noria:loggingTime "2023-09-08 02:25:20.0";
  noria:logText "sshd:ConnectionClosedPreAuth";
  noria:logOriginatingAgent <agent/USER_rogue>;
  dcterms:identifier "AIS_TOY_srv_tst_2_sshd_n2_03".

<event/LOG_TOY_srv_tst_2_sshd_n3_01>
  a noria:EventRecord;
  prov:wasDerivedFrom "noria-sdlri2023-toy-example";
  noria:alarmMonitoredAttribute "Secure remote-login and remote-execution";
  noria:alarmSeverity <kos/Notification/Severity/PerceivedSeverity/indeterminate>;
  noria:logOriginatingManagedObject <object/RES_TOY_srv_tst_2>;
  noria:logOriginatingManagementSystem <object/APP_AM001997>;  # Application for alarm management
  noria:loggingTime "2023-09-08 02:26:00.0";
  noria:logText "sshd:ConnectionEstablished";
  noria:logOriginatingAgent <agent/USER_rogue>;
```

```
  dcterms:identifier "AIS_TOY_srv_tst_2_sshd_n3_01".

<event/AIS_TOY_srv_tst_2_sshd_n3_02>
  a noria:EventRecord;
  prov:wasDerivedFrom "noria-sdlri2023-toy-example";
  noria:alarmMonitoredAttribute "Secure remote-login and remote-execution";
  noria:alarmSeverity <kos/Notification/Severity/PerceivedSeverity/warning>;
  noria:logOriginatingManagedObject <object/RES_TOY_srv_tst_2>;
  noria:logOriginatingManagementSystem <object/APP_AM001997>;  # Application for alarm management
  noria:loggingTime "2023-09-08 02:26:10.0";
  noria:logText "sshd:FailedPassword";
  noria:logOriginatingAgent <agent/USER_rogue>;
  dcterms:identifier "AIS_TOY_srv_tst_2_sshd_n3_02".

<event/AIS_TOY_srv_tst_2_sshd_n3_03>
  a noria:EventRecord;
  prov:wasDerivedFrom "noria-sdlri2023-toy-example";
  noria:alarmMonitoredAttribute "Secure remote-login and remote-execution";
  noria:alarmSeverity <kos/Notification/Severity/PerceivedSeverity/warning>;
  noria:logOriginatingManagedObject <object/RES_TOY_srv_tst_2>;
  noria:logOriginatingManagementSystem <object/APP_AM001997>;  # Application for alarm management
  noria:loggingTime "2023-09-08 02:26:20.0";
  noria:logText "sshd:ConnectionClosedPreAuth";
  noria:logOriginatingAgent <agent/USER_rogue>;
  dcterms:identifier "AIS_TOY_srv_tst_2_sshd_n3_03".

<event/AIS_TOY_srv_tst_2_sshd_n3_04>
  a noria:EventRecord;
  prov:wasDerivedFrom "noria-sdlri2023-toy-example";
  noria:alarmMonitoredAttribute "Secure remote-login and remote-execution";
  noria:alarmSeverity <kos/Notification/Severity/PerceivedSeverity/indeterminate>;
  noria:logOriginatingManagedObject <object/RES_TOY_srv_tst_2>;
  noria:logOriginatingManagementSystem <object/APP_AM001997>;  # Application for alarm management
  noria:loggingTime "2023-09-08 02:26:30.0";
  noria:logText "sshd:ConnectionClosedPreAuth";
  noria:logOriginatingAgent <agent/USER_rogue>;
  dcterms:identifier "AIS_TOY_srv_tst_2_sshd_n3_04".

<event/LOG_TOY_srv_tst_2_sshd_01>
  a noria:EventRecord;
  prov:wasDerivedFrom "noria-sdlri2023-toy-example";
  noria:alarmMonitoredAttribute "Secure remote-login and remote-execution";
  noria:alarmSeverity <kos/Notification/Severity/PerceivedSeverity/indeterminate>;
  noria:logOriginatingManagedObject <object/RES_TOY_srv_tst_2>;
  noria:logOriginatingManagementSystem <object/APP_AM001997>;  # Application for alarm management
  noria:loggingTime "2023-09-08 03:20:00.0";
  noria:logText "sshd:ConnectionEstablished";
  noria:logOriginatingAgent <agent/USER_rogue>;
  dcterms:identifier "AIS_TOY_srv_tst_2_sshd_01".

<event/AIS_TOY_srv_tst_2_sshd_02>
  a noria:EventRecord;
  prov:wasDerivedFrom "noria-sdlri2023-toy-example";
  noria:alarmMonitoredAttribute "Secure remote-login and remote-execution";
  noria:alarmSeverity <kos/Notification/Severity/PerceivedSeverity/warning>;
  noria:logOriginatingManagedObject <object/RES_TOY_srv_tst_2>;
  noria:logOriginatingManagementSystem <object/APP_AM001997>;  # Application for alarm management
  noria:loggingTime "2023-09-08 03:20:10.0";
  noria:logText "sshd:FailedPassword";
  noria:logOriginatingAgent <agent/USER_rogue>;
  dcterms:identifier "AIS_TOY_srv_tst_2_sshd_02".

<event/AIS_TOY_srv_tst_2_sshd_03>
  a noria:EventRecord;
  prov:wasDerivedFrom "noria-sdlri2023-toy-example";
  noria:alarmMonitoredAttribute "Secure remote-login and remote-execution";
  noria:alarmSeverity <kos/Notification/Severity/PerceivedSeverity/warning>;
  noria:logOriginatingManagedObject <object/RES_TOY_srv_tst_2>;
  noria:logOriginatingManagementSystem <object/APP_AM001997>;  # Application for alarm management
  noria:loggingTime "2023-09-08 03:20:20.0";
  noria:logText "sshd:ConnectionEstablished";
  noria:logOriginatingAgent <agent/USER_rogue>;
  dcterms:identifier "AIS_TOY_srv_tst_2_sshd_03".

<event/AIS_TOY_srv_tst_2_sshd_04>
  a noria:EventRecord;
```

```
  prov:wasDerivedFrom "noria-sdlri2023-toy-example";
  noria:alarmMonitoredAttribute "Secure remote-login and remote-execution";
  noria:alarmSeverity <kos/Notification/Severity/PerceivedSeverity/warning>;
  noria:logOriginatingManagedObject <object/RES_TOY_srv_tst_2>;
  noria:logOriginatingManagementSystem <object/APP_AM001997>;  # Application for alarm management
  noria:loggingTime "2023-09-08 03:20:30.0";
  noria:logText "sshd:FailedPassword";
  noria:logOriginatingAgent <agent/USER_rogue>;
  dcterms:identifier "AIS_TOY_srv_tst_2_sshd_04".

# --- Trouble ticket ---
<document/TT_TOY_2023SDLRI01>
  a noria:TroubleTicket;
  prov:wasDerivedFrom "noria-sdlri2023-toy-example";
  noria:troubleTicketRelatedResource <object/APP_99995_TST>;
  noria:troubleTicketCategory <kos/TroubleTicket/trouble-category/service-impaired>;
  noria:troubleTicketOrigin <kos/TroubleTicket/origin/clt>;
  noria:troubleTicketPriority <kos/TroubleTicket/priority/2>;
  noria:troubleTicketStatusCurrent <kos/TroubleTicket/status/initialised>;
  noria:troubleTicketType <kos/TroubleTicket/type/failure>;
  noria:troubleTicketUrgency <kos/TroubleTicket/urgency/immediate-response>;
  dcterms:description "Customer call reporting slow downs in data processing by the APP_99995_TST application.";
  dcterms:identifier "TOY_2023SDLRI01";
  dcterms:created "2023-09-08T03:50:00.0Z"^^xsd:dateTime.

# === EOF ===
```

Listing B.1: OOTD 2023 dataset: implementation, in Turtle syntax, of a knowledge graph describing a fictitious network infrastructure with events and organization details, for anomaly detection experiments.

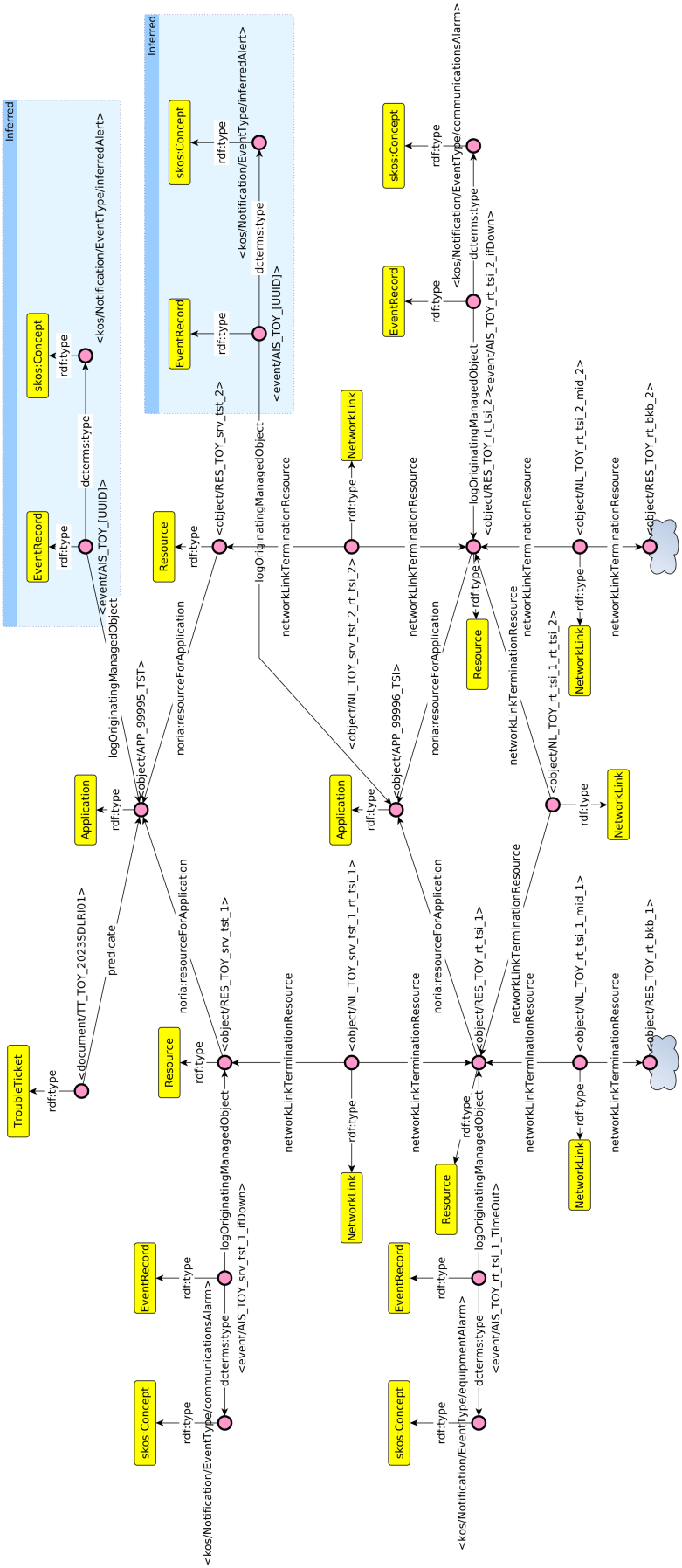





Figure B.5: OOTD 2023 knowledge graph
Overview of the OOTD 2023 dataset (Listing B.1). `noria` is the default namespace for concepts and relationships. The diagram conforms to the Graffoo graphical notation [330].

## B.4 Time-Ordered Contact Map for Statistical Learning

In this appendix, we outline an algorithm for calculating causal relationships between events on a network topology graph. This is an extension of the "graphical root cause analysis" representation idea presented in Figure 8.10, but this time without prior knowledge of event sequences, as it was the case in Section 8.5 with a procedural model projected onto the graph. Here, we consider that we only have a subgraph of network topology with timestamped event nodes (Figure B.6), as could result from an incident context calculation using the graph embeddings approach from Chapter 7. The approach presented below notably solves the problem of disambiguating events for which the occurrence time is close or identical; for this, we assume that the mechanism of fault propagation on the network is a function of the distance to be traveled in terms of the number of network hops. This proposal represents a direction for future work.

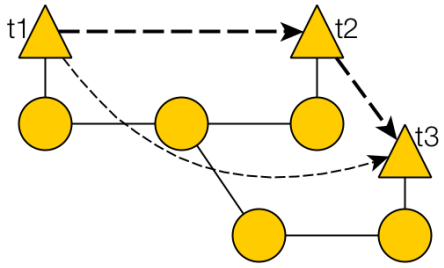



Figure B.6: Time-ordered contact map for statistical learning
A toy example of a network topology with three events (triangular shapes with $t_1 \leq t_2 \leq t_3$). The heavy dashed arcs represent "followed by" relationships (bold numbers in Eq. B.17). The light dashed arc represents the transitive cause-effect relationship of the $t_1$ event to the $t_2$ event, based on the composition $(t_2 \rightarrow t_3) \circ (t_1 \rightarrow t_2)$.

**Goal.** Computing event sequence relationships candidates on a network topology sub-graph, with controlled number of candidates w.r.t. $\|\text{event edges}\| = \left(\sum_{n=1}^{N} n\right) - N$.

**Principle.** The approach is two-fold. *Contact map:* place each of the events in the subgraph along the axes of the subgraph's adjacency matrix (Eq. B.17), with a score at each pairing used to represent how close they are in the topology space. *Sequence:* relate events based on their order in time and closeness in term of network topology space.

$$\begin{pmatrix} 0_{1\rightarrow 1} & \mathbf{2}_{1\rightarrow 2} & 3_{1\rightarrow 3} \\ 2_{2\rightarrow 1} & 0_{2\rightarrow 2} & \mathbf{3}_{2\rightarrow 3} \\ 3_{3\rightarrow 1} & 3_{3\rightarrow 2} & 0_{3\rightarrow 3} \end{pmatrix} \tag{B.17}$$

**Algorithm.** The following steps sketch the algorithm to compute the time-ordered contact map. The Figure B.6 and Equation (B.17) are illustrations for this algorithm.

1. Retrieve the network topology graph with event records,
2. Compute all shortest path length between event nodes (caution: this is expensive for large graphs),
3. Set shortest path lengths in an adjacency matrix,
4. Arrange the matrix columns & rows with continuous time ordering,
5. Get the upper-right triangular part of the matrix (i.e. getting back to a directed graph),
6. Prune the matrix rows towards the lowest distance,
7. Assert time relationship between event nodes from the remaining adjacencies, but for null distances.

## B.5 Data Structure and Computation Efficiency

As mentioned in Section 2.2.2, ICT systems event data processed by NMS/SIEM DSSs fall into the *big data* category with high volume, variety, and velocity characteristics [308]. This classification has potential consequences on anomaly detection, particularly in terms of data ingestion, processing performance, and data retention, which need to be explored. The analysis of the field is driven by the inherent link between Knowledge Representation and Reasoning (KRR), leading to understand the relationships and trade off between *how data is conceptually stored and managed* and *how hardware handle data*. In the following paragraphs, we discuss these two perspectives to highlight the challenges and opportunities associated with knowledge representation as graphs and Graph-Based Anomaly Detection (GBAD).

**Table *vs* graph data management.** The design and administration of DataBase Management System (DBMS) is a long-standing and constantly evolving field of research. Six research axes have emerged over time and structure the domain [296]: query optimization and execution; relational operators; files and access methods; buffer management; disk space management; database design. Assuming a correct database design (i.e. a conceptual data model meeting the use case requirements), the next classical quality evaluation criteria are the *access performance* (e.g. read-write I/O) and *database capacity* (e.g. maximum number of stored records).

The *access performance* itself has multiple performance factors (Figure B.7). A first and leading factor is how data records are interconnected. This typically draws a distinction between Relational DBMS (RDBMS) and Graph DBMS (GDBMS).

GDBMS are well-suited for highly connected data (graph-like data) as their internal data structures allow them to avoid the execution of multiple `JOIN` operations [360], which are often required in the case of RDBMS where connected data is implemented through *foreign keys* as *logical pointers* (using these increases computational complexity for retrieving graph data for as multiple lookup sequences are needed to resolve references). To avoid the `JOIN` complexity, GDBMS implement *arcs* as *physical pointers* to the filesystem/memory address of the neighboring entities. Hence, given a data node from where to start a search query, data fetching is enhanced by direct knowledge of where to read data from the neighboring; thus providing *1)* fast response time for local search and graph traversal queries, *2)* the ability to avoid the use of index.

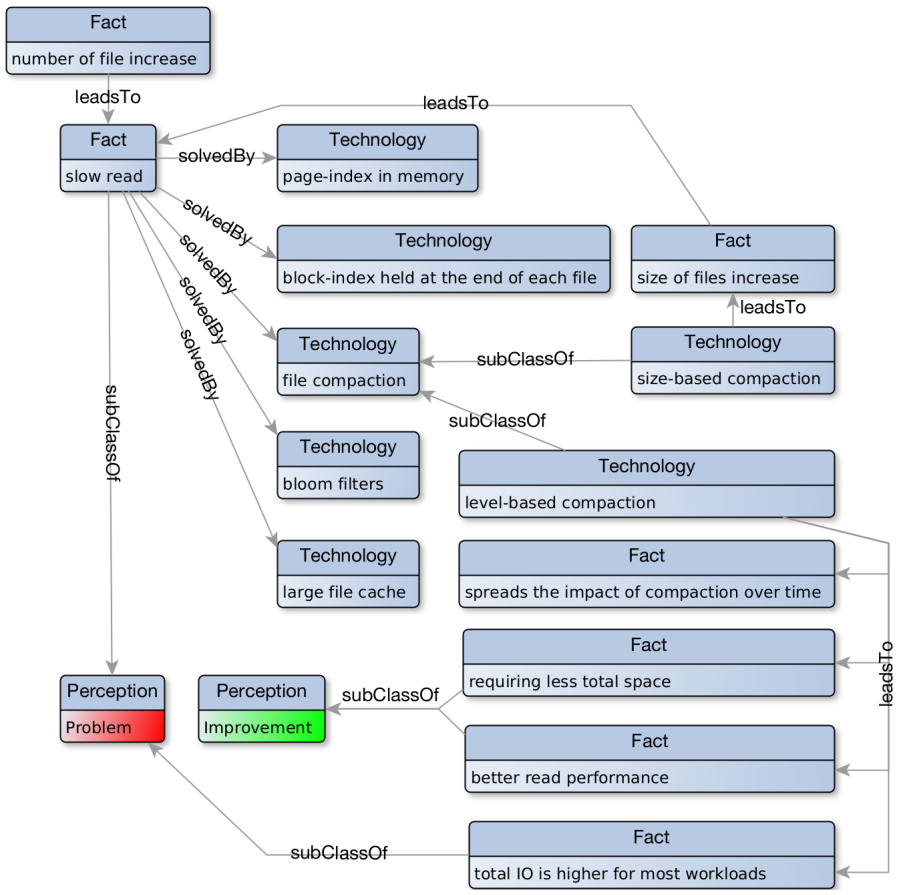



Figure B.7: DBMS technological and performance factors relationships.
This cause/consequence/solution diagram illustrates technological and performance factors entanglement for an increase in the number of files situation.

Conversely, RDBMS have an intrinsic *data alignement* characteristic linked to the use of *data types and size* for table design. This allows for fast response time to queries like "find all entities of type $X$" (e.g. aggregation functions to be computed over all the rows of a table) as the computation of row pointers relies on a fixed offset: $pointer_{row+1} = pointer_{row} + row_{size}$, depicting a sequential access pattern.

In addition, the use of an *index* can help to improve data lookup, filtering, join and aggregation operations speed [269]. In short, an index is an ordered data structure that focus on data values and maps these values to physical/logical pointers. Index data structures implementations are typically of two kind: *1)* tree-like [47, 230]; *2)* hash tables.

The use of the index technology[11] is not exclusively for RDBMS: "find all entities of type *X*" queries execution can also be optimized for GDBMS with help of an index. The notion of "indexing node" is an in-graph example of such. The addition of a search engine to GDBMS is another one for an out-of-graph approach (e.g. like Apache Lucene in [349]). Selecting a specialized index w.r.t. the main request argument is also a useful strategy. For example, the Neo4J GDBMS[12] maintains varied indexed views on its data and focus its lookup operation on the most appropriated one as needed (i.e. `nodestore`, `relstore`, `propostore`, `stringstore` and `arraystore`).

**Transactions for data integrity.** The second data access performance factor is defined within a multi-user or multi-process context (i.e. users/process may try to write/update a same record at a same time) through the concept of *transaction* (TXN): a sequence of one or more operations (reads or writes) which reflects a single real-world transition. This concept is critical for both integrity and performance concerns, namely: *1) recovery & durability* (i.e. the capability to keep the DBMS data consistent and persistent in case of crashes, aborts or system shutdowns); *2) concurrency* (i.e. the capability to achieve better performance by parallelizing transactions without creating anomalies).

When understood at information system design level, these two points of view show that they are not limited to DBMS query execution considerations. Event-driven architectures [63, 146] are an example of such paradigm change about "what data semantic is stored in database". Figure B.8 illustrates this with a CQRS-ES architecture.

**Data continuity, locality, and alignment for high-perf at the hardware level.** General performance expectations, as mentioned above, implicitly rely on performance factors that are highly relevant in hardware design considerations. For example, the von Neumann computer architecture highlights the role of memory and Instruction Set Architecture (ISA) in design considerations [65]. On one side, memory is a limited table of fixed-size data words. On the other hand, the ISA reflects what can be requested from a computer, in a more or less complex fashion depending on the case. Therefore, insights into processing architectures are of prime

[11] Let us remark that the *index* technology is an active research domain that is also present in the Operating System (OS) design domain (e.g. see [124] for the filesystem sub-domain).

[12] Neo4J insights given here are deduced from the analysis of code available at github.com/neo4j/neo4j. Additional optimization process are also implemented, such as data caching though a software prefetcher and the use of the Java Non-blocking I/O subsystem.

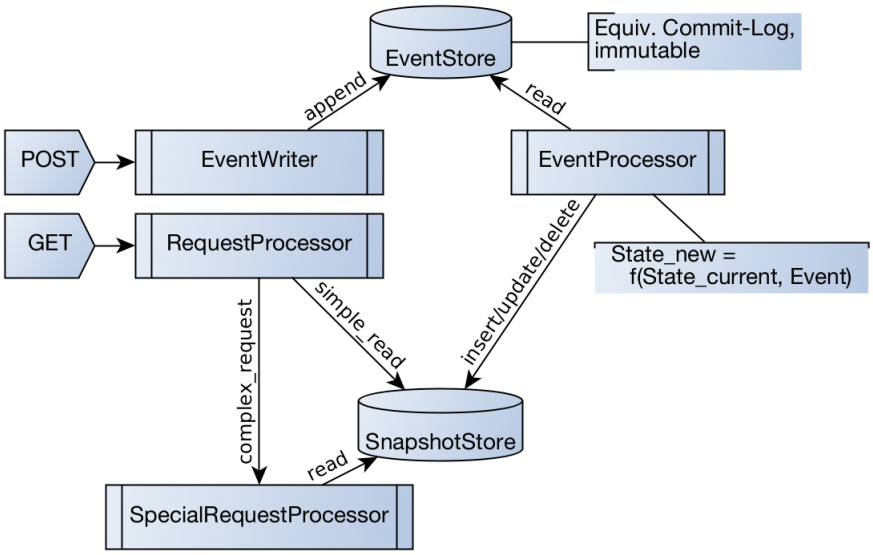



Figure B.8: Event-Stream architecture.
This architecture follows the Command Query Responsibility Segregation (CQRS) and Event Sourcing (ES) principles. *POST* command queries forward events to the *EventStore*, *GET* command queries fetch (intermediary) states through a *RequestProcessor*. *POST*/*GET* command queries act on different process and stores. *EventProcessor* is a function taking a current state and an event as arguments, and returning the next state. Inspired by [63].

interest for evaluating the potential of hardware acceleration.

At *memory module* level, one example of a hardware acceleration strategy is the use of a Content-Addressable Memory (CAM). CAMs allow for fast indirection to associated entities when given a specific value (or value pattern) to search for. The topic of DBMS data traversal mentioned above can be seen as a practical use case for CAMs. As discussed in [271], the performance of CAMs in terms of data fetching time is linked to: *1)* the number of transistors used in memory cells; *2)* the *search line* and *match line* utilization scheme; *3)* firmware optimizations such as *memory bank activation*, *pre-calculus*, and *dense encoding*.

At the *processing unit* level, hardware acceleration involves choosing a specific architecture from the options encompassed within the "xPU" acronym. This includes Central Processing Units (CPUs), Graphical Processing Units (GPUs), Vector Processing Units (VPUs), and Field Programmable Logic Arrays (FPGAs). At a more detailed level, each of these architectures refers to a set of sub-components and implementation strategies that best match specific business needs (e.g. general-purpose applications, real-time video encoding, speech recognition, etc.). In addition to memory architecture, *data processing architectures*[13] and *instruction execution optimization* – such as the *Out-of-Order Execution (OoOE)* and *speculative* families of techniques – are useful guides for comparing designs. Table B.2 provides a first level of

[13]The comparison of data processing architectures is commonly made through the Flynn's taxonomy: Single Instruction Single Data (SISD); Single Instruction Multiple Data (SIMD); Multiple Instruction Single Data (MISD); Multiple Instruction Multiple Data (MIMD).

analysis of generally acknowledged pros and cons about xPUs.

| | Pros | Cons |
|---|---|---|
| **CPU** | OoOE.<br>Branch prediction. | SISD.<br>CISC. |
| **GPU** | Matrices processing. | Memory transfer cost.<br>Poor efficiency for sparse matrix. |
| **VPU** | SIMD.<br>Small power consumption. | Small number of operators ($x$, $*$, $supeq$).<br>No data sharing between processes. |
| **FPGA** | Fast computation capabilities and small power consumption inherent to PLD. | Need special design skills.<br>Functional blocks subject to intellectual property. |

Table B.2: Generally acknowledged pros and cons about xPUs.
The following abbreviations apply: Complex Instruction Set Computer (CISC), Programmable Logic Device (PLD), Central Processing Unit (CPU), Graphical Processing Unit (GPU), Vector Processing Unit (VPU), Field Programmable Logic Array (FPGA), Out-of-Order Execution (OoOE), Single Instruction Multiple Data (SIMD), Single Instruction Single Data (SISD).

However, when using a data processing application such as anomaly detection techniques or a more specific GBAD algorithm, it is important to analyze potential pitfalls related to processing type and data representation (Table B.3). Therefore, selecting the appropriate hardware is crucial, while also considering how the software interacts with the hardware. The research and applied domain of high-performance programming focuses on studying optimal code execution [176]. This domain provides general concepts and strategies for data handling. Hints from this domain refer to general concepts about data handling strategies: *1)* **data continuity** (i.e. to which degree subsequent data elements can be accessed sequentially); *2)* **data locality** (i.e. *spatial* locality refers to data that is stored near in proximity to recently accessed data; *temporal* locality refers to the principle that data most recently accessed is likely to be accessed in the near future [391]); *3)* **data alignment** (i.e. processed data share a common bit-encoding size and endianess).

Several lines of research have explored these concepts. Firstly, [391] investigates the impact of prefetch schemes commonly used in von Neumann architectures on Breadth-First Search (BFS) graph traversal algorithms. The study reveals that these prefetchers, which rely on spatial locality principles, introduce I/O bottlenecks and are not suitable for the "random pointer hopping" memory access pattern [232, 262]. To address this issue, the author proposes and tests a new hardware prefetcher called Graph-Locality Prefetcher (GLP) that mitigates the performance degradation caused by traditional prefetchers. Secondly, focusing on *data alignment* and *data feed*, an abstraction layer between the xPUs and the programming languages is proposed in [36]. This abstraction layer monitors calculus done at the xPU level and dynamically adapt the data feed (e.g. data alignment and endianess) for optimal execution.

Data feed is also hinted in [176] where, for example, the *MKL* library[14] is said to be optimized for *in-memory* processing whereas *DAAL*[15] is optimized for data handled through *distributed computing* and arriving in *stream-like* fashion [176, Chap. 13, p. 203].

Additional lines of research are discussed hereafter by focusing on higher layers of the data processing components (i.e. data structure, data encoding and software architecture).

| **Application type** | **Execution characteristic(s)** | **Architecture best choice(s)** |
|---|---|---|
| **Traditional** | Highly branching execution graph (e.g. decision tree). | CPU |
| **ML/DL** | Loop a same operation over a high volume of data. | GPU and VPU |
| **Graph processing** | Random pointer hoping access. Sparse matrix data. | PIUMA [4] VPU At the filesystem level, a data continuity/locality management strategy can also hold [176]. |

Table B.3: xPU best choice w.r.t. *processing type* and *data representation* potential pitfalls.

**Data structure & encoding *vs* memory footprint & computational efficiency.** Typical storage structures for graphs include the *adjacency matrix* and *adjacency list*. In [321], the memory footprint of these structures is analyzed and compared to the needs of graph processing algorithms. The study demonstrates that graph processing algorithms, typically of the "traversal" kind, are more computationally efficient with adjacency lists, as they directly include a pointer to adjacent nodes. Additionally, real-world graphs, which typically have a high order and low density, result in sparse adjacency matrices, considered a greedy and inefficient use of memory. Therefore, the general conclusion drawn is that adjacency lists are the best choice in terms of the memory footprint-computational efficiency pair.

Another aspect related to data structures is the consideration of *queries*. As mentioned earlier, data alignment and data locality are crucial factors for performance in data processing. In the general case of graph traversal, where any direction within the graph needs to be considered unless there is prior knowledge about usual traversal queries associated with a specific graph [40], the locality criterion is equivalent to or even more important than the alignment criterion.

[14] The Intel Math Kernel Library (MKL) is a library of optimized, general-purpose math software. It is available at www.intel.com - oneMKL as part of the Intel oneAPI Base Toolkit.

[15] Intel Data Analytics Acceleration Library (DAAL) is a library of optimized algorithmic building blocks for data analysis. It is available at www.intel.com - oneDAL as part of Intel oneAPI Base Toolkit.

The relationship between data locality and graph data structure is further explored in applied research papers. For example, [195] proposes identifying principal components of graph-based data by analyzing the increase in graph complexity with depth-traversal. In [362], attenuated Bloom filters are discussed as a means to accelerate the decision-making process of a depth-traversal procedure through the prediction of "far features". Additionally, [168] opportunistically selects the most efficient data storage structure (row-oriented, cluster-oriented, or column-oriented) based on the data compression ratio as a decision criterion.

**On long duration storage.** Regarding the *data storage period* criterion, the driving forces in system design can be categorized into two potentially competing categories: modeling, and data retention.

In terms of *modeling*, it is a well-established fact in Machine Learning (ML) and Description Logic (DL) that having more data available can make a model more powerful. This leads to a natural trend towards big data, which involves high data volume and long storage periods, especially in AI-driven businesses. This trend is rooted in the principles of ML/DL techniques, such as overparametrization and regularization effects [253, §2]. There is also a promise of greater business opportunities associated with the use of "dark data".[16] However, pursuing infinite data is not a panacea, as ML/DL techniques can suffer from issues like anomaly detection model starting up time, re-training delay/cost, or poor accuracy when long-term data is considered. For example, in [247] the effect of detection depth on time-series with an Long Short Term Memory (LSTM) model has been studied, highlighting the potential drawbacks of considering long-term data. Additionally, improving the performance of a DL model is not solely dependent on having more parameters, as deep neural networks often generalize better than shallow networks with the same number of parameters [383].

In terms of *data retention*, legal and business requirements dictate the collection of logs and set limits on both data content and duration. Telecommunications service providers, for example, typically deal with Call Detail Records (CDRs) [120] and IP Detail Records (IPDRs) [179]. The storage period for these records can range from a minimum of three months to a maximum of seven years, depending on the specific business domain and the regulations in the hosting country (e.g. [77] for French regulatory issues). Compliance standards such as PCI DSS[17], HIPAA[18], NERC[19], and SOX[20] outline the duration requirements for data retention. It is important to note that logs may contain sensitive information about individual users, which raises privacy concerns. Regulations like General Data Protection Regulation (GDPR) impose

[16] https://www.gartner.com/en/information-technology/glossary/dark-data

[17] Payment Card Industry Data Security Standard, see https://www.pcisecuritystandards.org/document_library

[18] Health Insurance Portability and Accountability Act, see https://www.hhs.gov/hipaa/for-professionals/index.html

[19] North American Electric Reliability Corporation, see https://www.nerc.com/Pages/default.aspx

[20] Sarbanes-Oxley Act (SOX), see https://www.soxlaw.com/

specific content and duration policies for managing logs. For example, a minimum storage period of 3 months is required, followed by anonymization and eventual erasure of logs after one year [13]. *Record management* is the implementation of these policies at the organizational level, ensuring compliance with privacy regulations and data retention requirements.

Record management is defined as the "field of management responsible for the efficient and systematic control of the creation, receipt, maintenance, use and disposition of records, including the processes for capturing and maintaining evidence of and information about business activities and transactions in the form of records" [158]. Similarly, it is defined by the set of activities including "the planning, controlling, directing, organizing, training, promoting, and other managerial activities involving the life cycle of information, including creation, maintenance (use, storage, retrieval), and disposal, regardless of media" [41, DL1.105].

The driving forces mentioned above provide some guidance for time-related aspects of database capacity planning. The data retention period, ranging from three months to potentially over seven years, can be considered "long" in terms of data volume and velocity when viewed from a day-to-day perspective. This introduces a tradeoff between persistent data storage and access speed. In the field of data processing, *persistent data* refers to information that is infrequently accessed and not likely to be modified. On the other hand, *dynamic data* or *transactional data* refers to information that is periodically updated, meaning it changes asynchronously over time as new information becomes available. Considering this distinction between persistent data and dynamic data is important when making decisions about database design, storage, and access speed. It allows for a more efficient allocation of resources based on the specific characteristics and requirements of the data.

Various research lines and strategies address the challenge of long-term and short-term data access. Key principles used to tackle this problem include *data compression*, *data abstraction*, *subsampling*, *resolution adaptation*, and *data forget*. *Data compression* is an important principle that can be applied in different ways [247], for example: *1)* basic data compression is applied by DBMS; *2)* old time-series data are "consolidated" (i.e. subsampled and averaged).

**Summary and concluding remarks.** In this section, we have discussed computational efficiency from four perspectives: DBMS, hardware, data structure/encoding, and data retention. These angles have been shown to be entangled or stacked, and, in summary, the goodness of fit of all the technological and abstract components encompassed in these perspectives determines I/O perfomances.

A more detailed analysis has highlighted the concept of *data continuity* as a key factor for I/O performance in general. At the OS level, studies such as [47, 124, 230] have demonstrated that this continuity characteristic is influenced by related concepts and hypotheses such as

mutable/immutable variables, reference locality, and programming models. Therefore, unless there is complete control over the implementation of the hardware-OS-software stack, graph data processing should prioritize understanding the technical and algorithmic principles that can ensure data is stored, transferred, and processed in a continuous and localized manner.

In addition, the analysis of storage engines' implementation suggests that using *indirection tables* with as many tables as there are ways (views) to query the data (i.e. queries/views with a look-up component that is principally node-based, edge-based, attribute-based, value-based, etc.) is the best approach for data traversal. Therefore, *traversal performance* should be considered as a primary analysis axis, if not as important as the *data access performance* criterion. Similarly, *data access performance* is shown to be determined by the fit between both data structures and hardware architectures depending on "where data is managed" and "how data is presented". Thus any processing modeling attempt that follows the pattern of (algorithm with data) $\xrightarrow{\text{xPU}}$ (processed data) will be superficial and will not generalize. Finally, the concept of *data site* (location) can be useful for modeling structure mappings and analyzing the performance of the processing pipelines. Different data structures perform differently depending on the site considered (e.g. file system, Python dictionaries, CPU registers). This concept is related to the knowledge compilation [140] and Abstract Rewriting Systems (ARSs) [333] research topics.

To conclude on the richness and complexity of the data structure and computation efficiency topic, it is worth noting that it is an established and active technical/research domain with well-organized communities. In the 80s, the Fifth Generation Computer Systems (FGCS) initiative[21] aimed at efficient reasoning with hardware implementation but failed to achieve its goals due to the lack of expressiveness in the usage of tree-like structures [201]. More recently, the OpenCypher project[22] maintains various sub-topics of research in data structure and computation efficiency, including data models, expressiveness, scalability, and performance. Relevant research papers, such as a survey of graph database models [303] and approaches for high-performance exploration on large graphs [260], can be found there. Additionally, bridges with other scientific domains, such as biology, can enrich the technological approaches discussed above. For example, Rent's rule in biology provides an empirical law for evaluating the compactness of technical/biological systems [209], which relates to concerns about locality and compression.

[21] See https://en.wikipedia.org/wiki/Fifth_Generation_Computer_Systems for an overview.
[22] https://www.opencypher.org/

# Chapter C

# Résumé en français

## C.1 Introduction

**Problématiques et motivations.** L'exploitation de réseaux informatiques et télécoms de grande taille (backbones internationaux haut débit, réseaux d'entreprises, réseaux d'accès à Internet) amène, tôt ou tard, à être confronté à la gestion de situations d'incident complexes, telles que des perturbations globales de services du fait de défaillances en cascade ou de cyber attaques. Pour établir un diagnostic et gérer les incidents, les équipes de support technique utilisent typiquement des informations provenant d'outils de supervision et d'aide à la décision de la famille des NMS (Network Monitoring System) et des SIEM (Security Information and Event Management). Ces outils utilisent généralement une représentation élémentaire des infrastructures et des services réseau.

De façon simplifiée, les réseaux informatiques et télécoms sont un ensemble d'ordinateurs, de routeurs et autres dispositifs connectés et configurés pour le traitement et la transmission de données. De même, un service réseau est l'utilisation de cette capacité de traitement et de partage à des fins spécifiques, des plus triviales (divertissement, réservation de billets, domotique) aux plus exigeantes (marchés financiers, gestion des feux de circulation, gestion de centrales nucléaires).

Bien qu'évident au premier abord, ce niveau de description n'est pas suffisant pour garantir une qualité de service élevée pour des réseaux de grande taille. Cela est dû à l'hétérogénéité des TIC (Techniques de l'Information et de la Communication) qui les composent, rendant difficile le diagnostique et la résolution des incidents : afin d'assurer le bon fonctionnement des services, les équipes de supervision doivent prendre en compte et interpréter rapidement un grand nombre d'informations provenant de systèmes techniques divers et dynamiques. On peut prendre pour exemple une architecture de service qui combine des machines virtuelles réparties dans divers centres de calcul, ces centres étant interconnectés par un réseau mul-

ticouche IPoDWDM[1]. Être efficace dans ce contexte nécessite d'intégrer et de corréler les données de sources et de nature variées, telles que celles provenant des outils de gestion des machines virtuelles, des outils de gestion du réseau de transport optique (qui sont potentiellement gérées par un opérateur tiers), des informations sur les opérations planifiées et les informations de contact des équipes de maintenance locales. Compte tenu d'interdépendances complexes entre les services et l'infrastructure réseau, l'importance d'une représentation complète et standardisée des connaissances sur les actifs et les événements des réseaux devient évidente pour quiconque souhaite développer un système d'aide à la décision capable de capturer et d'analyser un contexte d'incident dans son ensemble.

Dans le même temps, nous pourrions être tentés de résoudre ces problèmes de complexité et d'efficacité opérationnelle en ajoutant des techniques d'IA (Intelligence Artificielle/Augmentée) aux outils de supervision. Cela s'observe déjà dans divers produits commerciaux et open source, notamment pour le regroupement et la priorisation d'alarmes, les alertes en cas de rupture de tendance (p.ex. une augmentation soudaine du trafic réseau) ou les alertes en cas de comportements à risque d'utilisateurs (p.ex. des tentatives d'authentification inhabituellement fréquentes à partir de diverses localisations). En pratique, cela correspond à mettre en œuvre un ensemble de règles métier combiné à de l'analyse de corrélations. Cette approche est efficace dans la mesure où les règles métier donnent, grâce à leur caractère logique, une certaine forme d'explicabilité aux alertes et recommandations générées par l'IA. Malgré cela, la charge opérationnelle pour les équipes de supervision reste importante car les systèmes à base de règles sont compliqués à maintenir en raison d'un grand nombre de règles ad hoc et qui réagissent sur des états discrets des systèmes, ce qui nuit à la généralisation et entraîne de nombreux faux négatifs. Une approche alternative consiste à utiliser des modèles probabilistes issus de l'apprentissage automatique. C'est également efficace car ce type d'approche permet une généralisation des modèles de détection à différents types de signaux, mais perd en explicabilité en raison de l'opacité des modèles (p.ex. un tenseur de probabilités dans un réseau de neurones) et induit, là encore, une charge opérationnelle due aux faux positifs. Étant donné que les deux approches présentent des avantages et des inconvénients complémentaires, la question se pose d'identifier des principes qui peuvent être partagés pour répondre aux exigences d'explicabilité et de généralisation, en ligne avec le besoin mentionné juste avant de représentation standardisée.

**Approche générale et résultats.** Dans cette thèse, nous abordons les enjeux de la représentation standardisée des données et de la corrélation d'informations à travers la pratique de l'ingénierie des connaissances. Plus précisément, nous étudions l'intérêt de structurer les données des réseaux dans un graphe de connaissances [8, 119], et explorons en quoi cette structure permet de maîtriser la complexité des réseaux, notamment pour des applications en

[1] Internet Protocol (IP) over Dense Wavelength-Division Multiplexing (DWDM).

détection d'anomalies sur des réseaux dynamiques et de grande taille. Quatre ensembles de résultats et de contributions ont émergé de ce travail de recherche, qui peut être résumé ainsi :

- **NORIA-O.** Nous avons proposé l'ontologie NORIA-O, un modèle de données reposant sur les techniques du Web sémantique [352] et développé en collaboration avec des experts en réseaux et en cybersécurité. L'ontologie permet aux équipes d'exploitation et aux algorithmes d'analyse d'avoir une vue complète et homogène des réseaux, y compris pour des données provenant de sources différentes. Les modèles de données existants se concentrent sur la représentation des ressources informatiques et de leur utilisation, mais il n'existe actuellement aucun modèle qui décrive les interdépendances entre les aspects structurels, dynamiques et fonctionnels d'une infrastructure réseau. NORIA-O comble cette lacune.

- **Plateforme de données basée sur les graphes de connaissances.** Nous avons développé une architecture de traitement de données qui combine des principes de conception du domaine des NMS/SIEM avec des outils du Web sémantique pour construire un graphe de connaissances. Cette architecture permet l'interconnexion de données hétérogènes en utilisant des vocabulaires contrôlés, ce qui est primordial pour une interprétation efficace des événements et des incidents. Les graphes de connaissances et les techniques du Web sémantique jouent un rôle essentiel dans la résolution des défis liés à l'hétérogénéité des données, au partage des connaissances et au raisonnement logique/probabiliste. La plateforme inclut des fonctions telles que le traitement par lots/en flux, la transformation déclarative des données, le patching et la réconciliation des données, la gestion centralisée de la configuration de la plateforme, l'audit de la provenance des données, et le transfert des données sémantiques par un bus de messages.

- **Système de techniques de détection d'anomalies.** Nous avons défini trois familles de techniques de détection d'anomalies s'appuyant sur les graphes de connaissances et le modèle de données NORIA-O : *model-based design*, *process mining* et *statistical learning*. Nous avons démontré que, à travers une vue homogénéisée des données des réseaux, les capacités d'inférence de chaque famille de techniques sont étendues à un plus large éventail de situations anormales. De même, le couplage entre les techniques et les sources de données est réduit. Nous avons également démontré que l'utilisation conjointe des techniques selon le principe de raisonnement synergique (prise de décision coopérative) [46] amène le système à présenter simultanément une propriété d'explicabilité intrinsèque et de capacité de raisonnement probabiliste. Ce principe consiste à réinjecter les connaissances résultantes de l'application d'une technique dans le graphe de connaissances, ce qui fournit des informations supplémentaires de contexte lors de l'application ultérieure des autres techniques. Par rapport à un modèle

unique de détection spécialisé sur un type de donnée ou un domaine d'application, cette approche permet de couvrir globalement un plus nombre de situations anormales.

- **Conception et ergonomie d'une interface homme-machine.** Nous avons proposé une architecture logicielle client-serveur utilisant les techniques Web [310] et un graphe de connaissances pour accompagner les équipes support dans leur prise de décision lors d'anomalies réseau complexes, à la fois pour le périmètre de la gestion des infrastructures et celui de la cybersécurité. À travers cela et une série d'entretiens avec un panel d'experts des opérations réseaux, nous avons identifié les attentes essentielles des équipes en matière d'ergonomie et de fonctions. Nous avons notamment démontré la nécessité de concevoir ce type de solution en allant au-delà de la seule récupération et présentation des données du graphe de connaissances. Le raisonnement synergique et l'exploration interactive des systèmes techniques apparaissent en effet comme des principes essentiels à mettre en œuvre. L'évaluation de l'interface par un panel d'utilisateurs fournit par ailleurs une base de travail importante pour de futur travaux de conception d'outils de type NMS/SIEM qui utiliseraient des graphes de connaissances et des mécanismes de raisonnement hybride logique/probabiliste.

Cette recherche a été développée dans le cadre du projet "NORIA" de la direction de la recherche d'Orange[2], un opérateur international d'infrastructures et de services réseaux. Les résultats de cette thèse de doctorat ont mené à plusieurs publications et présentations (Chapitre A), dont huit papiers de recherches acceptés dans des conférences scientifiques internationales et un en cours de relecture. Ces résultats ont par ailleurs eu un impact industriel significatif, menant notamment au déploiement actif d'un système de jumeau numérique [163] basé sur les graphes de connaissances au niveau du groupe Orange. Ils ont de même donné l'opportunité d'encadrer et de soutenir quatre étudiants dans leurs stages et projets finaux, leur fournissant ainsi une formation en techniques du Web sémantique et en architecture des systèmes d'aide à la décision. Enfin, les résultats de cette thèse ont également contribué à l'établissement de la communauté "RML join", un groupe de travail au sein du Knowledge Graph Construction Community Group du W3C[3], dans le cadre d'un effort de standardisation de l'ontologie RML [23] afin d'en faire une recommandation W3C.

Dans la suite de ce chapitre, nous présentons les détails essentiels des travaux de cette thèse. Dans la Section C.2 nous abordons le développement de NORIA-O et de l'architecture de construction de graphes de connaissances. Dans la Section C.3 nous présentons les trois familles de techniques de détection d'anomalies et leur utilisation à travers l'interface homme-machine développée. Enfin, nous concluons ce résumé en Section C.4.

[2] https://hellofuture.orange.com
[3] https://www.w3.org/community/kg-construct/

## C.2 Ingénierie des connaissances et données massives

**Une ontologie pour la détection d'anomalies et la gestion des incidents des TIC.** Afin de développer NORIA-O, nous avons réuni un panel d'experts en provenance de diverses entités du groupe Orange en rapport aux domaines de l'ingénierie et de l'exploitation des réseaux. Ce panel se compose de 16 experts qui représentent collectivement 150 membres d'équipes des opérations. Nous avons suivi l'approche dite des "Questions de Compétence" [385] pour établir le modèle conceptuel du domaine de discours. Cette approche consiste à identifier les concepts et relations clés du domaine de discours par le découpage, selon un ensemble de motifs sémantiques pré-établis, des requêtes d'utilisateurs formulées en langage naturel. À l'issue de plusieurs réunions de travail, les équipes ont validé 26 questions de compétences. Ces questions sont disponibles dans la Table 3.1 et en open source à l'adresse https://w3id.org/noria/cqs/.

Pour l'étape d'implémentation, nous définissons l'espace de nommage `noria`. Nous avons choisi d'utiliser une axiomatisation reposant principalement sur un raisonnement par subsomption en utilisant le langage de représentation RDFS [82], et le langage OWL [319] lorsqu'une structuration logique plus poussée est nécessaire (p.ex. disjonction de classes, relations qualifiées). Nous avons de même suivi les principes de la méthode "Linked Open Terms" (LOT) [224] qui vise, entre autres, la réutilisation ou l'alignement à des vocabulaires existants. Des ontologies existantes utilisant le modèle de graphe RDF, nous interconnectons et/ou étendons les vocabulaires suivants : **BBO** [22] pour décrire les activités du point de vue de la modélisation des processus métier conformément au standard BPMN ; **BOT** [215] pour décrire la localisation des ressources techniques ; **DCTERMS** pour décrire les instances de NORIA-O en tant qu'éléments d'un catalogue ; **DevOpsInfra** [270] pour permettre des interactions potentielles de NORIA-O avec la pratique DevOps ; **FOAF** [81] pour représenter les personnes et groupes sociaux ; **FOLIO** [62] pour permettre l'analyse des causes racines selon l'approche FMEA (analyse des modes de défaillance) ; **ORG** [86] pour décrire les parties prenantes et les organisations associées ; **SEAS & PEP** [240, 241] pour décrire les systèmes techniques et l'exécution de procédures ; **UCO** [388] pour permettre une évaluation des risques de cybersécurité sur les instances de NORIA-O ; **SLOGERT** [26] pour décrire les journaux système et envisager l'utilisation de la suite d'outils de SLOGERT de traitement des journaux. Des modèles ne reposant pas sur RDF, nous réutilisons une partie du vocabulaire et de la taxonomie du **TM Forum**[4] concernant l'interopérabilité des systèmes de gestion des opérateurs télécoms, de l'**ITU-T** [166, 164] sur les messages des systèmes de supervision dans l'industrie des télécommunications, et de l'**IETF** pour une utilisation précise de la terminologie des techniques d'Internet telles que définies dans les "Requests for comments" (RFC)[5].

[4] https://github.com/tmforum-apis
[5] https://datatracker.ietf.org/

L'implémentation de NORIA-O est disponible en open source à l'adresse https://w3id.org/noria/. Le modèle est composé de 59 classes, 107 propriétés d'objet et 71 propriétés de données. La Figure C.1 en donne un aperçu. Son expressivité est $ALCHOI(D)$ selon l'outil Protégé 5.1 [227].

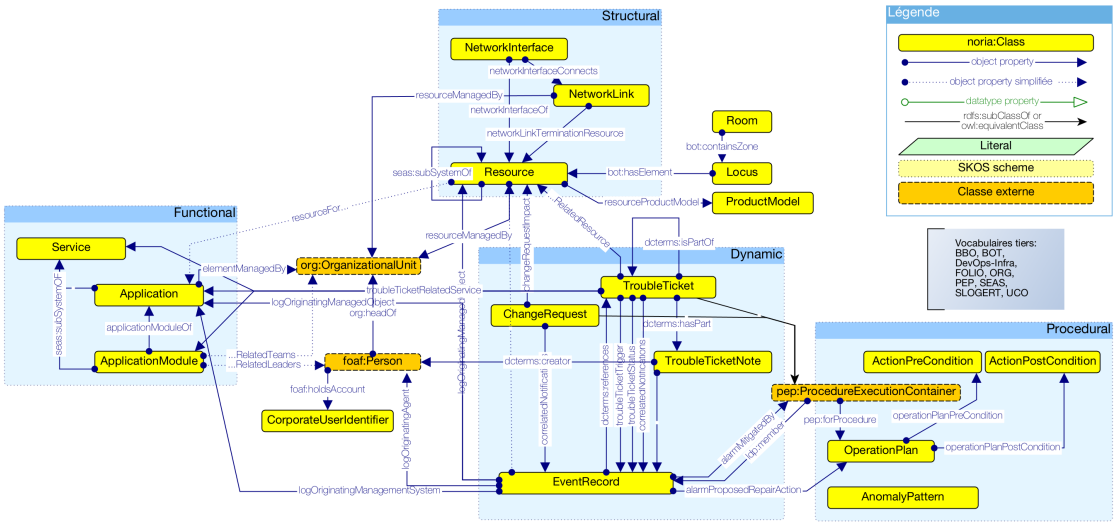



Figure C.1: Aperçu du modèle NORIA-O
Ce diagramme montre les classes et les propriétés les plus importantes du modèle de données, ainsi que les facettes du domaine de discours et les relations avec des modèles tiers. Les facettes sont définies ainsi : *structural* = décrit les éléments physiques et logiques constituant les réseaux (p.ex. serveurs, routeurs, liaisons Ethernet), *functional* = décrit les services et zones d'échanges, *dynamic* = décrit les séquences d'événements, *procedural* = décrit comment les choses fonctionnent ou comment une séquence d'événements doit être interprétée. `noria` est l'espace de nommage par défaut. Certaines propriétés sont regroupées du modèle sout regroupées dans ce visuel (voir "object property simplifiée") afin de simplifier la représentation. Le diagramme suit partiellement la spécification Graffoo [330].

**Une plateforme basée sur les graphes de connaissances.** En partant de l'hypothèse que la mise en correspondance de sources de données multiples à travers leur représentation sémantique permettrait d'améliorer les capacités de détection des outils de supervision, nous avons cherché à comprendre comment les processus type de traitement des données classiques (p.ex. "Extract, Transform, Load", CRISP-DM [323]) pourraient s'appliquer aux graphes de connaissances. Ces processus classiques ne prennent pas directement en compte les capacités d'abstraction et de raisonnement offertes par les techniques du Web sémantique et les graphes de connaissances. De plus, l'intégration des données et les fonctions d'aide à la décision y sont généralement considérées comme deux aspects distincts. Une approche cognitive de l'analyse des données par itérations (i.e. des données brutes → des informations → des connaissances) ou par raisonnement synergique [46] est ainsi écartée.

Sur la base de ces considérations, nous proposons un modèle de chaîne de traitement (Figure C.2) pour guider les étapes de conception d'une plateforme basée sur les graphes de connaissances : les *données non structurées* (p.ex. les journaux d'événements) entrent dans la chaîne et sont transformées en *données structurées* grâce à l'application d'un *modèle de structure*

pré-défini/appris. Un modèle sémantique est appliqué pour créer des *données annotées*. Celles-ci peuvent être associées à des connaissances supplémentaires provenant d'un *service d'enrichissement* (p.ex. la mise en correspondance des ressources techniques avec des informations sur l'organisation de l'entreprise ou sur les vulnérabilités des systèmes). Le *service de raisonnement* (p.ex. règles métier, propagation des convictions, prédiction de liens/entités) traite les *données annotées* pour produire des connaissances supplémentaires (i.e. des *données interprétées*). Des agents en aval (p.ex. les équipes opérationnelles, des outils du système d'information) sont informés (p.ex. diagnostic d'un incident) en interrogeant les données interprétées. Ce modèle de chaîne de traitement est ouvert à des processus complémentaires, tels que l'intégration de flux direct de données structurées ou l'exécution récursives de l'étape d'inférence.

Nous proposons les principes de conception suivants pour rationaliser le développement d'une plateforme basée sur les graphes de connaissances et en faciliter l'adoption par les utilisateurs : *1) Minimiser les besoins de transformation à l'entrée de la chaîne* : l'encodage des données (sérialisation et structuration) doit être rétro-compatible le plus tôt possible dans la chaîne de traitement afin de limiter le nombre de technologies utilisées ; *2) Indépendance à l'utilisation en aval* : les applications en aval de la chaîne fonctionnent potentiellement en paralèle les unes des autres et en s'intéressant à différents aspects des données, la sérialisation/structure doit donc permettre une séparation facile des données des métadonnées sans imposer d'appels de procédures distantes spécifiques ; *3) Indépendance à l'implémentation* : une approche déclarative (i.e. description abstraite) des règles de transformation et mécanismes de traitement permet de décrire et de transposer le comportement du système indépendamment des implémentations et socles technologiques ; *4) Intégrer, personnaliser ou développer* : donner la priorité à l'intégration de solutions existantes qui répondent aux exigences, étendre/adapter les solutions partiellement conformes ou développer des solutions spécifiques si aucune des options précédentes ne s'applique.

En suivant ces principes, nous avons développé et déployé une architecture de traitement des données (Figure C.2) de type Lambda [95] en combinant des solutions open source bien connues (Apache Kafka[6], Apache Airflow[7], OpenLink Virtuoso[8]), des projets académiques (RMLMapper [24], StreamingMASSIF [292], string2vocabulary [276], grlc [9], RDFUnit[9]) et du code ad hoc publié en open source (Table 4.1). L'architecture proposée a été instanciée et testée dans un environnement industriel, produisant un graphe de connaissances RDF qui présente un fort potentiel pour traiter les anomalies impliquant plusieurs systèmes à partir de données hétérogènes. Elle permet typiquement de :

[6] https://kafka.apache.org/
[7] https://airflow.apache.org/
[8] https://virtuoso.openlinksw.com/
[9] https://github.com/AKSW/RDFUnit

1. Construire un graphe de connaissances RDF à partir de données statiques (listes de ressources informatiques, organisation) et de données en flux (tickets d'incident, journaux système, alarmes) ;
2. Annoter les données par des informations de provenance et de niveau de confiance ;
3. Modifier/réconcilier/lier les données sémantiques (patching) au moment de la construction du graphe ou a posteriori ;
4. Mettre en œuvre des algorithmes de raisonnement en cascade et synergiques.

La solution minimise notamment les efforts en matière de qualité et de traçabilité des données grâce à l'utilisation généralisée du vocabulaire RML [24] pour décrire les transformations, ainsi que grâce au stockage centralisé des données et des règles de transformation dans le graphe de connaissances. En outre, la solution ouvre à la possibilité d'un traitement distribué ou déclenché par des événements grâce à la généralisation du transfert de données RDF par un bus de messages. Cependant, en l'état, nous observons que l'annotation de la provenance des données à l'échelle du jeu de données, plutôt qu'au niveau de chaque triplet du graphe généré, entraîne une perte de granularité de l'information après les étapes de patching. De même, une étude de charge approfondie est nécessaire pour envisager le déploiement des outils de traitement de flux sur des données massives (p.ex. les données de télémétrie provenant de routeurs de réseau large bande ou d'une flotte d'objets IoT).

## C.3 Détection d'anomalies et aide au diagnostique

Pour aborder efficacement le sujet de la détection d'anomalies sur les réseaux de grande taille, nous avons commencé par mener des entretiens avec des experts de l'exploitation des réseaux en vue d'identifier les cas de détection et de diagnostic clés dans leur activité quotidienne. Ces cas d'utilisation sont analysés en vue d'une représentation en logiques descriptives [119] afin de fournir des garanties d'explicabilité et de transposabilité lors d'éventuelles implémentations dans des solutions algorithmiques.

À travers cette analyse, nous démontrons que le diagnostique d'incidents nécessite l'utilisation de la logique implicative, ce qui conditionne la façon dont le graphe de connaissances est construit, ainsi que les possibilités de transcription des cas de détection en SPARQL (le langage de requête standard des graphes de connaissances RDF) [368]. Pour premier exemple, l'hypothèse du monde ouvert[10] – inhérente à l'utilisation des logiques descriptives et des graphes de connaissances – peut être aisément contredite par le fait qu'une mesure de l'état du système (un observable tel que la présence de signaux sur une interface d'un équipement de réseau) puisse temporairement échapper aux outils de supervision (perte de paquet dans le

[10] Open World Assumption [8] : ne pas matérialiser un fait ne veut pas dire qu'il n'est pas avéré.

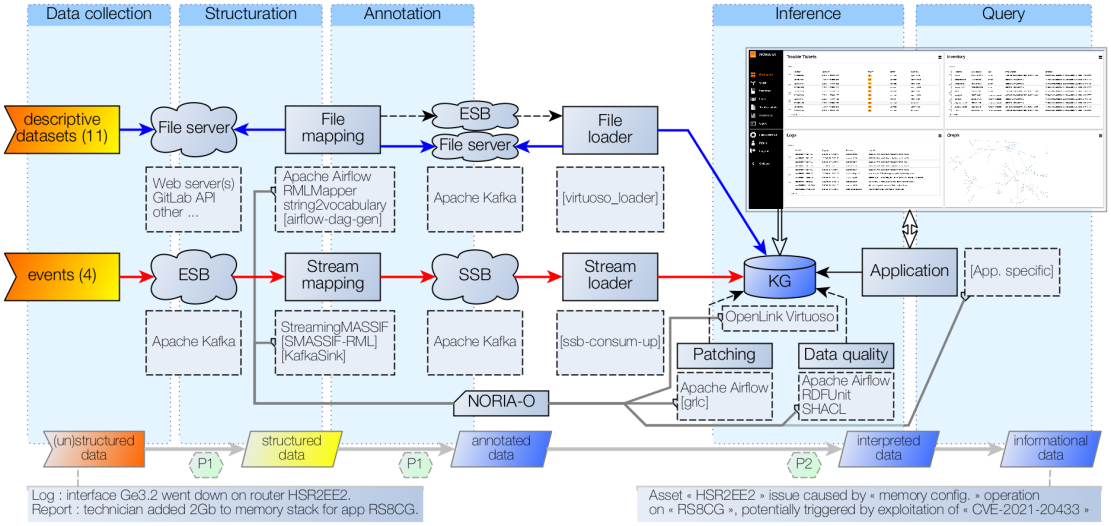



Figure C.2: Aperçu de la chaîne d'intégration de données.
L'architecture présentée est de type Lambda [95]: une chaîne transforme les données statiques par lots à partir de règles de transformation en langage RML, une chaîne traite les données en flux à partir de règles elles aussi en RML, le graphe de connaissances est construit au fur et à mesure en enregistrant les données sémantiques issues des deux chaînes précédentes dans une base de données spécialisée (KG). Les données du graphe sont structurées selon l'ontologie NORIA-O. Acronymes : ESB = Enterprise Service Bus, SSB = Semantic Service Bus, KG = Knowledge Graph. Les flèches pleines représentent les flux de données, les flèches en pointillés représentent les flux de contrôle et de requête. Les flèches commencent à partir du composant initiant le flux/la transaction. Les chiffres pour les blocs "descriptive datasets" et "events" font référence au nombre de sources utilisées dans nos expérimentations pour construir le graphe de connaissance. Les noms des composants entre parenthèses sont liés aux outils de la Table 4.1. $P1$ et $P2$ représentent les groupes de composants de traitement dans une lecture de la chaîne d'outils $ET[P1]L[P2][L]$ (i.e. en correspondance avec le modèle de processus Extract-Transform-Load).

réseau, défaut dans le traitement, etc.), ce qui induirait des faux positifs/négatifs selon la façon d'implémenter la règle de détection. Le second exemple vient du fait que le processus de diagnostic peut amener à utiliser des opérateurs qui ne sont pas aisément exprimables en logique descriptive ou qui sont absents de SPARQL (p.ex. calcul de voisinage, plus court chemin), ce qui implique de sortir d'un cadre strictement logique en définissant des opérateurs additionels retournant des ensembles ("set returning functions"). Ce second exemple se retrouve notamment dans le besoin d'être en capacité de "*circonscrire l'espace d'investigation des ressources réseaux et de recherche des causes pour les situations d'incident multi-applications*" tel que formulé par les experts lors des entretiens, et qui définit un idéal à atteindre en matière de solution algorithmique à développer.

Sur la base de ces analyses, nous avons défini trois familles d'approches pour modéliser et détecter les anomalies (Table C.1) à l'aide de graphes de connaissances, chaque famille couvrant des attentes spécifiques en rapport aux cas de détection mentionnés par les experts (p.ex. analyse de séquence, apprentissage de contexte) et des propriétés différentes (p.ex. apprentissage automatique *vs* connaissances métier, explicabilité intrinsèque *vs* post-hoc).

En utilisant NORIA-O et le graphe de connaissances produit dans les étapes précédentes de cette thèse (Section C.2), nous montrons comment chaque approche peut être mise en

Table C.1: Familles de techniques de modélisation des anomalies

| Principe | Forces | Faiblesses |
|---|---|---|
| | **Model-Based Design** | |
| Interroger le graphe pour récupérer les anomalies et leur contexte. | Détection d'anomalies "enregistrées" de quelque manière que ce soit dans le graphe grâce aux systèmes de supervision ; traduction simple des règles de détection d'anomalies ; plusieurs niveaux d'abstraction (subsumption). | Repose sur les connaissances d'experts ; absence de raisonnement probabiliste ; difficile de représenter des décisions séquentielles ; peut nécessiter d'inférer plus d'informations préalables sur l'anomalie, par exemple son type à l'aide d'une classification. |
| | **Process Mining** | |
| Aligner une séquence d'entités sur des modèles d'activité, puis utiliser cette similarité pour guider la réparation. | Détection d'anomalies avec plusieurs signaux d'alerte et décisions séquentielles ; modèles rejouables. | Repose sur les connaissances d'experts ; peut nécessiter de débruiter les modèles ; similarité probabiliste. |
| | **Statistical Learning** | |
| Relier des entités en fonction de similarités de contexte, puis utiliser cette similarité pour alerter et guider la réparation. | Détection d'anomalies avec plusieurs signaux d'alerte. | Nécessite un réglage fin de la définition du contexte en fonction du cas d'utilisation et des exigences sur la profondeur temporelle à capturer; correspondances probabilistes. |

œuvre et utilisée individuellement ou en combinaison pour gérer une large gamme de situations d'incidents. Les trois paragraphes qui suivent présentent chacune des trois familles d'approches, et le quatrième paragraphe montre comment utiliser les approches à travers une application Web développée spécifiquement.

**Des connaissances métier et des modèles formels pour la détection d'anomalies.** Nous explorons la famille de techniques *model-based design* à travers trois expériences reposant sur l'idée que les données présentes dans le graphe de connaissances sont nécessaires et suffisantes pour détecter et qualifier une anomalie. Pour cela, nous utilisons directement les techniques du Web sémantique, à savoir l'exécution de requêtes SPARQL pour interroger le graphe, et les règles de raisonnement implicites au modèle de données qui structure le graphe.

La première expérience correspond au cas décrit en Figure C.3, représentatif d'une situation de dégradation de la redondance d'un système. L'approche *model-based* équivaut à considérer qu'une règle métier fourni par les experts sert de référence à l'implémentation du mécanisme de détection. Dans le cas présent, l'objectif "*Quelles applications ont k sur n (50%) ressources en alarme ? Lever une alerte pour celles-ci.*" est rempli en transcrivant la règle en requête SPARQL (Listing C.1).

```
# === PREFIXES ============================================================
PREFIX dcterms: <http://purl.org/dc/terms/>
```

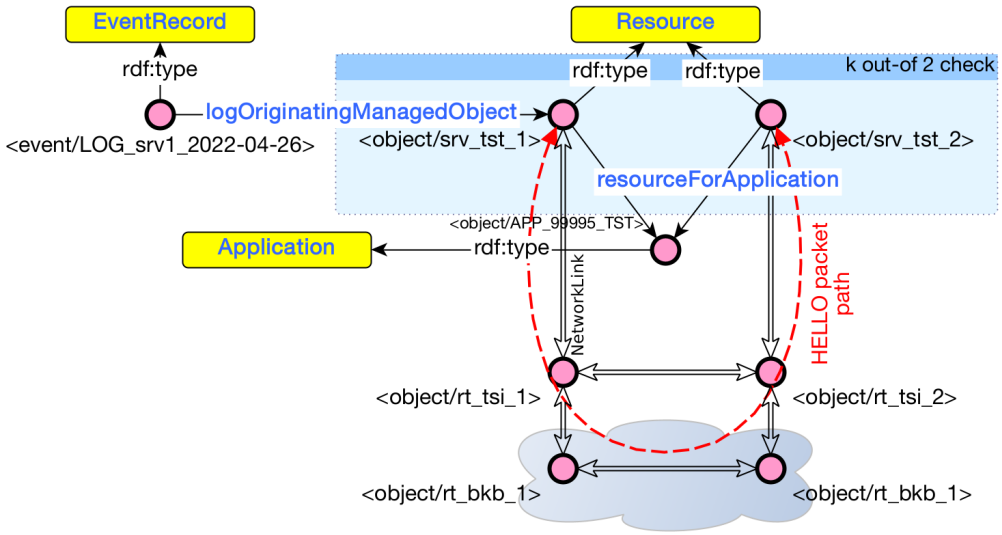



Figure C.3: Motif de graphe pour le cas de détection "k sur n"
Exemple de motif de graphe pour le cas de détection "Quelles applications ont k sur n (50%) ressources en alarme ?". Le diagramme inclut une vue simplifiée de la topologie du réseau (réseau d'accès et réseau cœur) à travers lequel les serveurs communiquent pour échanger des messages Hello. `noria` est l'espace de nommage par défaut pour les concepts et les relations. Le diagramme suit partiellement la spécification Graffoo [330].

```
PREFIX prov: <http://www.w3.org/ns/prov#>
PREFIX noria: <https://w3id.org/noria/ontology/>
BASE <https://w3id.org/noria/>

# === REQUETE ================================================================
CONSTRUCT {
  ?EventURI a noria:EventRecord ;
    prov:wasDerivedFrom "noria-sdlri2023-toy-example" ;
    noria:alarmMonitoredAttribute "Resilience" ;
    noria:alarmSeverity <kos/Notification/Severity/PerceivedSeverity/major> ;
    noria:logOriginatingManagedObject ?App ;
    noria:logOriginatingManagementSystem <object/APP_NORIA_AD> ;
    noria:loggingTime ?NotificationConstructTime ;
    dcterms:description "Application at risk: K out-of N (50%)" ;
    dcterms:type <kos/Notification/EventType/inferredAlert> .
} WHERE {
  SELECT
  ?App (COUNT(DISTINCT ?Res) AS ?ResTotal)
  (COUNT(DISTINCT ?ResImp) AS ?ResWithImpact)
  (NOW() as ?NotificationConstructTime)
  (IRI(CONCAT("event/AIS_TOY_", strUUID())) AS ?EventURI)
  WHERE {
    ?Res a noria:Resource ;
      noria:resourceForApplication ?App .
    OPTIONAL {
      ?Event a noria:EventRecord ;
        noria:logOriginatingManagedObject ?Res .
      BIND (?Res AS ?ResImp) }
  } GROUP BY ?App HAVING ( (?ResWithImpact / ?ResTotal) >= 0.5)
}
```

Listing C.1: Requête SPARQL pour le problème "k sur n" de dégradation de redondance, avec génération d'une entité EventRecord pour alerter.

Les deux expériences suivantes (non détaillées ici, voir la Section 5.4 pour cela) complètent l'exécution d'une requête par du raisonnement automatique. Nous utilisons l'ontologie FO-

LIO [62] pour l'analyse de cause racine, une étude préalable des modes de défaillance[11] servant de règles métier pour le système de déduction. Enfin, nous utilisons le générateur d'ontologies CFGOwl [255] pour la classification de séquences d'événements, une spécification préalable des motifs de séquences ayant été effectuée en syntaxe Lark[12] (une syntaxe et un analyseur pour définir une grammaire non contextuelle) et servant de règles métier pour le système de déduction.

Dans l'ensemble, les trois expérimentations montrent la faisabilité de transcrire directement des connaissances d'experts dans un mécanisme formel de détection. Ces mécanismes permettent de détecter une anomalie à partir des données présentes dans un graphe de connaissances, mais ne permettent cependant pas de prédire un état futur du système. Pour contourner cette limitation, l'expert doit inclure de lui-même un indice des potentialités dans le modèle formel, indice qui ne sera pas utilisé en tant que tel dans le mécanisme de raisonnement mais simplement comme une information complémentaire rapportée aux utilisateurs.

**Apprentissage relationnel et détection d'anomalies sur les traces de navigation Web.** Une part importante des interactions des utilisateurs avec les applications se fait désormais via une interface Web. Un scénario relativement simple pourrait être le suivant : après une phase de reconnaissance du système ciblé, l'utilisateur malveillant accède directement à la page d'accueil d'une plateforme de service, trompe le système d'authentification (p.ex en utilisant une technique d'injection SQL[13]), extrait des données, puis quitte le service en naviguant directement vers une autre page. La détection de ce type de situation nécessite l'analyse à la fois des journaux d'application et du contenu du trafic réseau. Malheureusement, les journaux d'application peuvent être inaccessibles ou inutilisables en raison de problèmes de confidentialité ou de formatage incorrect. De même, le trafic réseau peut être chiffré ou inaccessible à la collecte, ce qui entraîne la perte d'informations sur l'interaction de l'utilisateur avec la plateforme et le scénario d'attaque [151].

Pour aborder ces difficultés, nous faisons l'hypothèse qu'une contextualisation des traces d'activité des utilisateurs et des processus informatiques par rapport aux informations de topologie de réseau devrait permettre d'identifier les comportements anormaux. Pour apprendre des modèles d'activité interprétables sous forme de données liées, nous développons une extension de navigateur Web et l'utilisons dans une chaîne de traitement de données (Figure C.4) mettant en pratique la famille de techniques dénommée plus haut *process mining*. L'extension, baptisée Graphaméléon, collecte les traces d'activité des utilisateurs (trafic réseau, interactions avec le navigateur Web) lors d'une session de navigation Web et sérialise ces

[11] Aussi connu sous le terme "Failure Mode and Effect Analysis" dans le monde Anglo-saxon.
[12] https://www.lark-parser.org/ide/
[13] https://en.wikipedia.org/wiki/SQL_injection

données en RDF en utilisant le vocabulaire UCO [388]. Les données résultantes sont ensuite intégrées dans un graphe de connaissances pour interpréter les traces d'activité à un niveau sémantique et dériver des motifs, notamment sous forme de réseaux de Petri [374]. Ces modèles d'activité peuvent ensuite être utilisés pour identifier des situations similaires en les projetant sur le graphe de connaissances et, sur la base de cette projection, obtenir des informations contextuelles en parcourant le graphe.

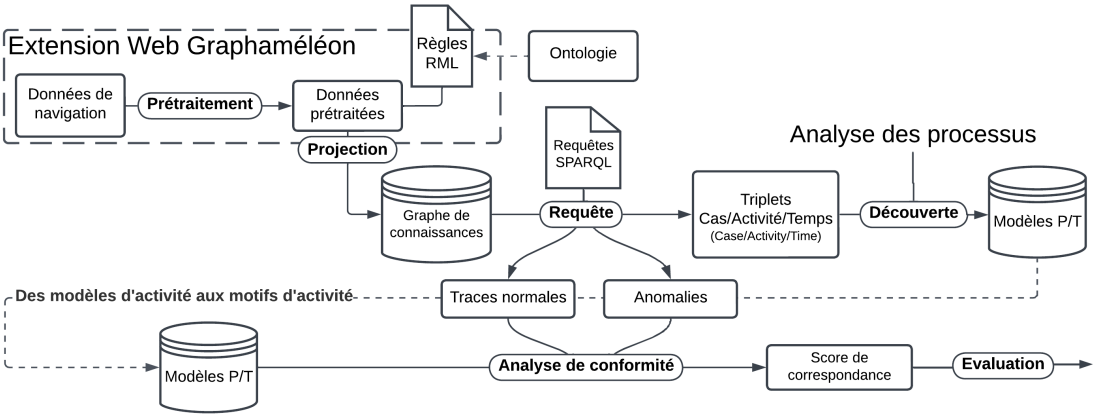



Figure C.4: Aperçu du pipeline de traitement des données de Graphaméléon.
L'extension Web Graphaméléon capture et annote l'activité de l'utilisateur au niveau du navigateur Web. Un composant de découverte des processus (process mining) dérive des modèles d'activité à partir du graphe de connaissances RDF résultant. Ces modèles peuvent être utilisés pour construire une bibliothèque de modèles d'activité, qui sont ensuite utilisés par un composant de vérification de conformité (conformance checking) pour classer de nouvelles traces d'activité en tant qu'activités normales ou anormales.

Nous évaluons notre approche à travers deux séries d'expérimentations. Tout d'abord, nous analysons la corrélation entre le volume de triplets RDF générés par Graphaméléon et la nature/complexité d'un ensemble de sites Web de référence accessibles sur l'Internet. Ensuite, nous modélisons et identifions trois scénarios de navigation Web en utilisant Graphaméléon et les réseaux de Petri dans un environnement contrôlé. L'un des scénario correspond à une attaque par injection SQL, comme évoqué au début de ce paragraphe. Les expériences sont menées en utilisant Graphaméléon v2.1.0, disponible en open source sur https://github.com/Orange-OpenSource/graphameleon. Nous mettons de même à disposition les données associées à ces expériences sur https://github.com/Orange-OpenSource/graphameleon-ds.

L'expérience d'analyse de la complexité des sites Web montre que l'augmentation des échanges réseau (visible notamment à travers la structure du graphe résultant) est fonction de la politique d'anti-traçage appliquée au niveau du navigateur Web, ce qui fournit une base de travail intéressante pour de futurs travaux de catégorisation de sites Web par les stratégies de traçage en ligne mises en œuvre par les éditeurs de contenu Web et la topologie du réseau qui y correspond. Avec la seconde expérience, portant sur la classification des traces de navigation, nous montrons la façon de procéder pour utiliser les outils de découverte de processus (librairie

PM4PY Process Mining [11]) sur des graphes de connaissances. Notre approche permet de dériver des motifs d'activité sous forme de modèles procéduraux utiles pour constituer une base de données du comportement du couple utilisateur-système. L'utilisation de ces modèles d'activité révèle toutefois les limites en l'état de la technique dite de "conformance checking" pour la détection d'anomalies lorsque de micro-écarts se produisent entre l'activité d'un utilisateur/système par rapport à un modèle. Nous avons également souligné le défi d'aligner les modèles d'activité en raison de l'absence d'une méthode fiable pour identifier de façon univoque les éléments des pages HTML du Web à travers l'utilisation de différents navigateurs, ou selon les sessions de navigation.

**Plongement de graphe pour la capture et la catégorisation d'un contexte d'incident.** Nous explorons la troisième famille de techniques dénommée *statistical learning* en proposant d'accompagner l'exploitant réseaux dans une compréhension rapide d'une situation décrite dans un ticket d'incident. Pour cela, nous développons un classifieur qui permettrait de catégoriser une dynamique de réseau donnée à partir d'une référence de ticket d'incident. Du point de vue du vocabulaire NORIA-O, la construction du classifieur implique d'apprendre le modèle relationnel sur les événements proches de la ressource signalée dans un ticket d'incident (i.e. parcourir le graphe et en calculer un plongement [312, 153]), puis de relier le modèle relationnel à une catégorie (Eq. C.1 et Figure C.5) :

$$\begin{aligned} &\Big\{EventRecord.logOriginatingAgent\big(Resource(i)\big) \\ &\qquad .logOriginatingAgent\big(Resource(i)_{neighbors}\big)\Big\} \\ \sim TroubleTicket.&\Big(relatedResource\big(Resource(i)\big) \\ &\qquad \sqcap problemCategory\Big) \end{aligned} \tag{C.1}$$

où `noria:problemCategory` est un attribut décrivant la nature finale ou l'impact technique d'une entité `noria:TroubleTicket`. Cet attribut fait partie des champs clés utilisés dans le système de gestion des incidents pour qualifier pleinement la résolution de l'incident lors de sa clôture.[14] Il s'agit d'une `owl:ObjectProperty` avec des valeurs issues du vocabulaire contrôlé `kos/TroubleTicket/trouble-category`, un schéma de concepts SKOS définissant neuf concepts dans NORIA-O.

Pour les expérimentations, nous utilisons la bibliothèque pyRDF2Vec [61] et la bibliothèque scikit-learn [279]. pyRDF2Vec[15] est une implémentation en Python de l'algorithme RDF2Vec [283], qui capture le contexte des nœuds du graphe RDF (propriétés et nœuds voisins) sous forme de vecteurs de caractéristiques latentes (plongements). Les étapes de traitement des données pour notre approche sont les suivantes :

[14] Autres propriétés possibles : `noria:problemResponsibility`, `noria:troubleTicketCause` et `pep:forProcedure`.

[15] Voir INK [59] pour une alternative à pyRDF2Vec.

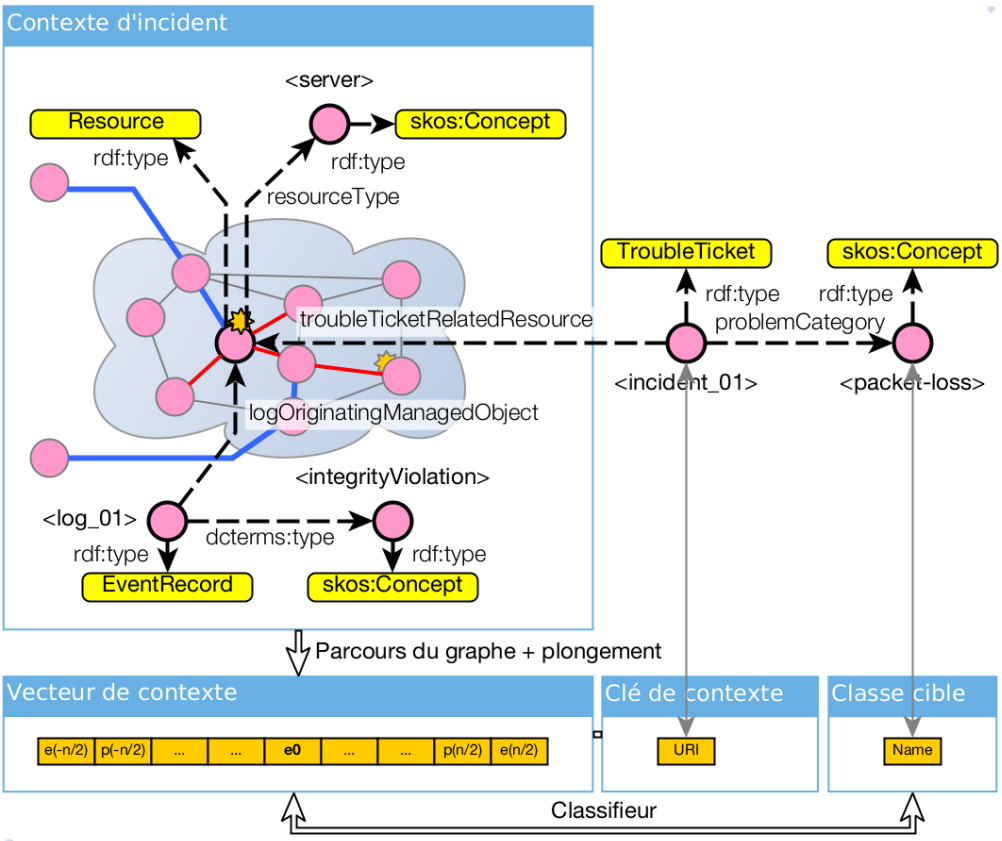



Figure C.5: Apprentissage d'un contexte d'incident par plongement de graphe
Ce diagramme illustre le processus de construction d'un classifieur multi-classes à partir du contexte d'un ticket d'incident en mettant en évidence la relation entre un contexte d'incident (i.e. le sous-graphe centré sur une entité `Resource` concernée par un `TroubleTicket` donné), le vecteur de contexte (i.e. le plongement du sous-graphe dans un espace vectoriel), et le modèle de classification qui est entraîné sur la relation du vecteur de contexte à la classe de problème définie au niveau de l'entité `TroubleTicket`. L'URI de l'entité `TroubleTicket` est utilisée comme clé de contexte (i.e. l'identifiant du vecteur dans la base de données des vecteurs) afin de récupérer le vecteur de contexte à partir de l'identifiant de l'entité `TroubleTicket`, ainsi que pour rechercher une ou plusieurs entités `TroubleTicket` à partir d'un vecteur de contexte. `noria` est l'espace de nommage par défaut pour les concepts et les relations. Autres espaces de nommage : `rdf` = <https://www.w3.org/1999/02/22-rdf-syntax-ns#>, `skos` = <https://www.w3.org/2004/02/skos/core#>. Le diagramme suit partiellement la spécification Graffoo [330].

1. Nous créons une liste de parcours du graphe (i.e. des séquences de sommets) en traversant l'ensemble de données RDF avec des entités `noria:TroubleTicket` comme points de départ, tout en filtrant la propriété `noria:problemCategory` car elle est utilisée pour la classification ;
2. Nous calculons les plongements pour les entités `noria:TroubleTicket` en entraînant un modèle Word2Vec sur les parcours de graphe ;
3. Nous entraînons un classifieur de type Random Forest[16] en prenant les plongements comme échantillons d'entraînement, et les entités `noria:problemCategory` comme valeurs cibles.

[16]Les classifieurs de type Random Forest utilisant des arbres de décision en font un choix privilégié pour l'explicabilité intrinsèque qu'ils offrent au modèle.

Le jeu de données utilisé comprend environ 400'000 entités (Table 7.1) avec une profondeur temporelle de 111 jours pour les événements. Dans l'ensemble, nous concluons que le classifieur fonctionne relativement bien, mais que le jeu de données est trop petit (en particulier pour certaines classes) et incohérent pour généraliser et répondre immédiatement au besoin de "circonscrire l'espace d'investigation" évoqué plus haut. L'approche typique pour surmonter ce problème consiste à améliorer l'étape de construction du graphe de connaissances en utilisant des sources de données plus larges et en renforçant les processus de gestion de la qualité des données. En plus d'analyser les performances de notre approche, nous comparons les résultats d'inférence du classifieur avec un ensemble de requêtes SPARQL similaires aux cas de détection des tickets d'incidents du jeu de données. Cette comparaison montre que l'espace vectoriel des plongements du graphe de connaissances reflète une structuration logique. Cela ouvre une opportunité de recherche pour identifier des modèles causaux à partir de l'espace latent en utilisant la similarité sémantique, comme discuté dans [144].

**Une interface pour l'analyse basée sur les graphes de connaissances.** Les graphes de connaissances RDF ont fait leur preuves depuis de nombreuses années pour résoudre des problèmes d'intégration de données et de raisonnement logique sur des données hétérogènes, notamment par l'entremise des ontologies et de vocabulaires partagés [8]. Malgré cela et l'existence de diverses ontologies liées aux domaines NetOps/SecOps (Table 2.3), nous constatons que l'utilisation de ce mode de représentation n'est pas encore répandue dans les outils de type NMS et SIEM. Intégrer les graphes de connaissances dans ces outils tout en facilitant l'accès aux algorithmes de détection d'anomalies et aux outils de recommandation de solutions semble donc la voie naturelle à suivre pour réduire la surcharge cognitive des experts en supervision de réseaux et améliorer l'efficacité opérationnelle dans la gestion des incidents. Les multiples facettes de connaissances qui doivent être représentées pour la compréhension d'une situation d'incident posent cependant une limite à l'exploration intuitive et efficace des graphes de connaissances (i.e. un accès rapide et limité aux informations pertinentes), surtout lorsque des délais de réponse courts sont imposés par les accords de niveau de service (SLA). Une compréhension approfondie des ontologies en jeu peut en effet s'avérer essentielle. En ce qui concerne l'utilisation de l'IA, diverses approches démontrent un intérêt pratique pour la détection d'anomalies ou l'assistance au diagnostic (Table B.1). Cependant, comme suggéré plus haut, il est important que les experts en supervision aient la capacité de combiner et d'appeler différentes approches et modèles afin de couvrir un large éventail de comportements des systèmes et de s'assurer d'une prise de décision bien étayée.

Pour relever ces défis, nous faisons l'hypothèse que l'intégration des graphes de connaissances dans les NMS/SIEM est la voie à suivre à condition que l'ergonomie des outils d'aide à la décision réponde aux exigences métier des équipes NetOps/SecOps en termes d'accessibilité de l'information (p.ex. en brisant les silos techniques), de contextualisation de la situation

des incidents (p.ex. en réduisant la charge cognitive grâce au regroupement des alarmes) et de continuité des tâches opérationnelles (i.e. éviter une ergonomie disruptive par rapport aux usages courants). Sur la base de cette hypothèse, nous développons et présentons NORIA UI, une architecture logicielle client-serveur Web [310] pour la gestion des anomalies réseau, basée sur les données stockées dans un graphe de connaissances, et dont l'ergonomie (UI/UX) est le résultat d'une collaboration avec un panel d'experts NetOps/SecOps d'Orange. La Figure C.6 donne un aperçu de l'interface de NORIA UI.

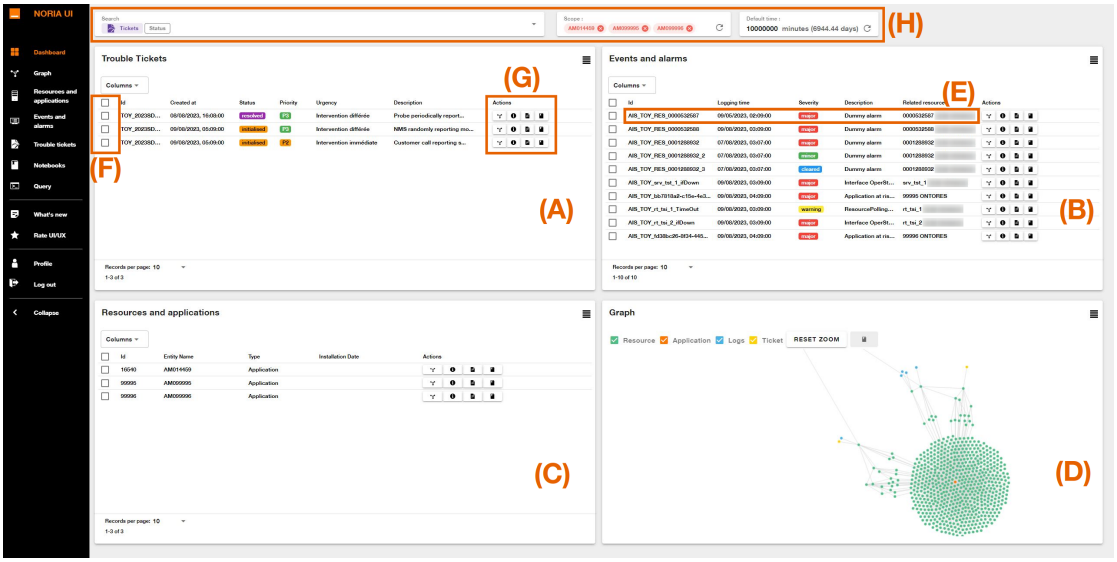



Figure C.6: NORIA UI : la page du tableau de bord
La page du tableau de bord se compose de quatre panneaux offrant un accès à des informations sur la vie du réseau issues du graphe de connaissances selon quatre facettes complémentaires : (*A*) les tickets d'incidents, (*B*) les événements et les alarmes, (*C*) les ressources et les applications, et (*D*) la topologie réseau enrichie. Les entités du graphe sont affichées avec leurs principales propriétés (*E*). Des cases à cocher (*F*) permettent un pivot d'affichage qui s'applique à tous les panneaux. Des commandes contextuelles sont associées à chaque entité représentée (*G*). Un champ de saisie (*H*) permet de définir le périmètre technique affiché et de rechercher des entités en fonction de leurs propriétés.

À travers ces travaux, nos principales contributions sont les suivantes. Tout d'abord, nous détaillons les exigences métier en matière d'ergonomie et de fonctions pour l'élaboration d'un outil d'aide à la décision de nouvelle génération. Ensuite, nous présentons les détails techniques de l'architecture technique et logicielle mise en œuvre pour répondre à ces exigences. Les fonctions développées correspondent à la mise en œuvre du principe du raisonnement synergique pour la combinaison de différentes techniques d'IA (Figure C.7), ainsi qu'à la mise en œuvre de mécanismes d'interaction pour l'analyse exploratoire de réseaux multicouches. Ces fonctions peuvent être résumées ainsi: *1)* consultation croisée d'informations sur la topologie du réseau, les événements et les alarmes ; *2)* affichage de la topologie du réseau en 2D/3D enrichie d'indicateurs ; *3)* appel périodique ou à la demande d'outils d'analyse et de détection d'anomalies accessibles en tant que services Web ; *4)* agrégation et analyse d'informations dans un espace d'analyse dédié ("notebook") ; *5)* accès à des fonctions com-

munautaires pour le partage d'informations entre collaborateurs. Enfin, nous fournissons un retour d'expérience sur la solution proposée et ses perspectives, sur la base d'une analyse des performances de la plateforme technique, ainsi que d'une évaluation de l'UI/UX par les utilisateurs selon la méthode SUS [181, 180, 131] dans des situations opérationnelles. L'analyse du verbatim du panel d'utilisateur a globalement fait ressortir une forte acceptabilité de la proposition NORIA UI, ainsi que des marques d'intérêt spécifique selon les profils métier (p.ex. gestionnaire d'incident, administrateurs réseaux, ingénieur système) pour des développements complémentaires futurs (p.ex. résumé d'une situation, algorithmes supplémentaires de détection, commande du réseau depuis l'interface).

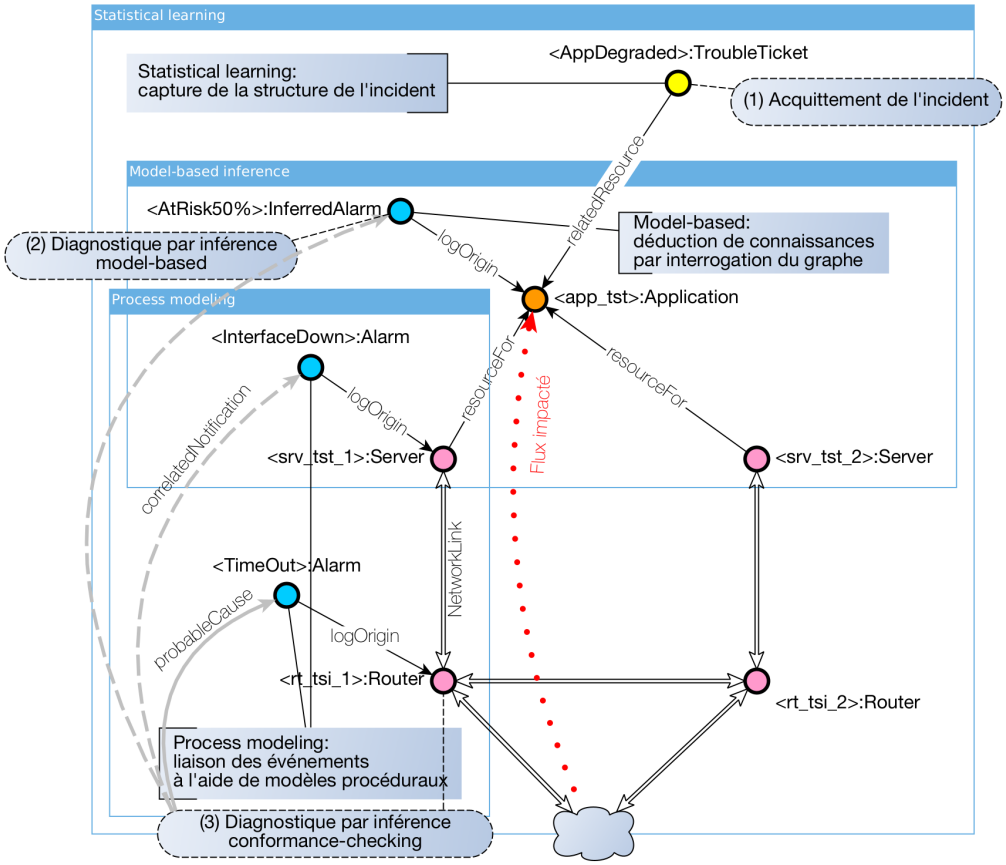



Figure C.7: Détection d'anomalies et raisonnement synergique

Ce diagramme montre comment les approches *model-based*, *process mining* et *statistical learning* se combinent pour la détection d'anomalies et l'aide au diagnostique, chaque approche étant liée à un domaine d'inférence spécifique. Le graphe est une version simplifiée d'un graphe de connaissances structuré par le modèle de données NORIA-O, représentant une topologie réseau avec des alarmes et un ticket d'incident. La situation peut être résumée comme suit : une alarme de délai d'attente dépassé (time out) est émise par le système de supervision en raison d'une défaillance matérielle sur `rt_tsi_1` le rendant inaccessible ; en conséquence, une alarme de perte de signal sur une interface est émise par `srv_tst_1` car son lien vers `rt_tsi_1` est en panne ; une alarme calculée (inferred alarm) grâce à une requête SPARQL est insérée dans le graphe et associée à `app_tst` pour indiquer un problème de résilience dû à une défaillance sur `srv_tst_1` ; un ticket d'incident est associé à `app_tst` suite à une plainte d'un utilisateur concernant les performances de l'application. Les zones en pointillés avec numérotation indiquent les étapes de diagnostic via l'interface utilisateur NORIA.

## C.4 Conclusion

Dans ce travail de recherche, nous avons étudié l'intérêt de structurer les données d'exploitation des réseaux informatiques et télécoms dans un graphe de connaissances, et dans quelle mesure cette structure peut simplifier la détection et le diagnostic d'anomalies sur des réseaux complexes. Pour orienter cette recherche et maximiser l'interopérabilité de nos propositions, nous avons utilisé l'ingénierie des connaissances et les techniques du Web sémantique.

Pour commencer, nous avons développé une nouvelle ontologie, NORIA-O, pour décrire les infrastructures réseaux, les incidents, et les opérations d'exploitation. Une architecture de construction de graphes de connaissances a été conçue pour transformer les données des réseaux provenant de sources diverses. Le graphe de connaissances résultant, structuré par NORIA-O, permet d'analyser le comportement des réseaux de manière homogène.

Nous avons de même défini et évalué trois familles de techniques algorithmiques utilisant les graphes de connaissances. Ces techniques peuvent être utilisées pour détecter des comportements anormaux des systèmes et aider les équipes d'exploitation dans le diagnostic d'incidents.

Enfin, nous avons proposé une architecture logicielle pour simplifier les interactions des exploitants avec le graphe de connaissances et les algorithmes d'aide au diagnostic via une interface graphique spécialisée.

Toutes les propositions ont été testées de manière indépendante par des expérimentations et démonstrations, ainsi que par un panel d'utilisateurs experts via l'interface graphique spécialisée dans le cadre d'une solution intégrée. Les résultats obtenus, tant sur le plan scientifique que sur la base des retours des utilisateurs et de l'impact industriel, ont démontré la faisabilité et l'intérêt d'utiliser les graphes de connaissances couplés à un ensemble de techniques d'intelligence artificielle. Cette approche permet de maîtriser la complexité des réseaux et ouvre la voie à la conception de systèmes plus résilients à l'avenir, notamment en capitalisant sur la compréhension de la dynamique des systèmes dans une forme utile à la fois pour les humains et les outils de calcul.

# Bibliography


[1] ISO/IEC JTC 1/SC 40. Information technology – Service management – Part 1: Service management system requirements. Technical Report 20000-1:2018, International Organization for Standardization/International Electrotechnical Commission, 2018.

[2] ISO/IEC JTC 1/SC 7. Systems and software engineering – Life cycle processes – Requirements engineering. Technical Report 29148:2018(E), ISO/IEC/IEEE, 2018.

[3] A. Adriansyah, B. F. van Dongen, and W.M.P. van der Aalst. Cost-Based Conformance Checking Using the A∗ Algorithm. Technical Report BPM-11-11, BPMcenter.org, 2011.

[4] Sriram Aananthakrishnan, Shamsul Abedin, Vincent Cavé, Fabio Checconi, Kristof Du Bois, Stijn Eyerman, Joshua B. Fryman, Wim Heirman, Jason Howard, Ibrahim Hur, Samkit Jain, Marek M. Landowski, Kevin Ma, Jarrod A. Nelson, Robert Pawlowski, Fabrizio Petrini, Sebastian Szkoda, Sanjaya Tayal, Jesmin Jahan Tithi, and Yves Vandriessche. The Intel Programmable and Integrated Unified Memory Architecture Graph Analytics Processor. *IEEE Micro*, 2023.

[5] Adam Shostack. Experiences Threat Modeling at Microsoft. In *Proceedings of the Workshop on Modeling Security (MODSEC08) Held as Part of the 2008 International Conference on Model Driven Engineering Languages and Systems (MODELS)*, Toulouse, France, 2008.

[6] Adrian Bondy and U. S. R. Murty. *Graph Theory*. Graduate Texts in Mathematics. Springer-Verlag, 2008.

[7] Agustinus Kristiadi, Mohammad Asif Khan, Denis Lukovnikov, Jens Lehmann, and Asja Fischer. Incorporating Literals into Knowledge Graph Embeddings. In *International Semantic Web Conference (ISWC)*, 2019.

[8] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, Roberto Navigli, Axel-Cyrille Ngonga Ngomo, Sabbir M. Rashid, Anisa Rula,

Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. Knowledge Graphs, 2020.

[9] Albert Meroño-Peñuela and Rinke Hoekstra. grlc Makes GitHub Taste Like Linked Data APIs. In *The Semantic Web: ESWC 2016 Satellite Events, Heraklion, Crete, Greece, May 29 – June 2, 2016*. Springer, 2016.

[10] Aldo Gangemi and Silvio Peroni. DiTTO: Diagrams Transformation inTo OWL. In *Proceedings of the ISWC 2013 Posters & Demonstrations Track, a Track within the 12$^{th}$ International Semantic Web Conference (ISWC)*, Sydney, Australia, 2013.

[11] Alessandro Berti, Sebastiaan van Zelst, and Wil van der Aalst. Process Mining for Python (pm4py): Bridging the Gap between Process-and Data Science. In *Proceedings of the ICPM Demo Track 2019, co-located with 1$^{st}$ International Conference on Process Mining (ICPM)*, 2019.

[12] Alessandro Oltramari, Loria Cranor, Robert Walls, and Patrick McDaniel. Building an Ontology of Cyber Security. In *9$^{th}$ Conference on Semantic Technologies for Intelligence, Defense, and Security (STIDS)*, 2014.

[13] Alexandre Joly. 7 points d'éclaircissement sur le RGPD appliqué aux sites Internet, 2017. Blog sur la sécurité informatique et la sensibilisation des entreprises et particuliers.

[14] Alexiei Dingli and David Farrugia. *Neuro-Symbolic AI*. Packt Publishing, 2023.

[15] Alfredo Cuzzocrea and Giuseppe Pirrò. A Semantic-Web-Technology-Based Framework for Supporting Knowledge-Driven Digital Forensics. In *Proceedings of the 8$^{th}$ International Conference on Management of Digital EcoSystems*, Biarritz, France, 2016. Association for Computing Machinery.

[16] Ali Sadeghian, Miguel Rodriguez, Daisy Zhe Wang, and Anthony Colas. Temporal Reasoning Over Event Knowledge Graphs. In *1$^{st}$ Workshop on Knowledge Base Construction, Reasoning and Mining (KBCOM)*, Los Angeles, California, USA, 2018.

[17] Alison Cartlidge, Ashley Hanna, Colin Rudd, Ivor Macfarlane, John Windebank, and Stuart Rance. *An Introductory Overview of ITIL V3*. The UK Chapter of the itSMF, 2007.

[18] Alistair Cockburn. *Writing Effective Use Cases*. The Agile Software Development Series. Addison-Wesley, 2012.

[19] Amal Guittoum, Francois Aïssaoui, Sébastien Bolle, Fabienne Boyer, and Noel De Palma. Inferring Threatening IoT Dependencies Using Semantic Digital Twins Toward Collaborative IoT Device Management. In *Proceedings of the 38$^{th}$ ACM/SIGAPP Symposium on Applied Computing*, New York, NY, USA, 2023. Association for Computing Machinery.

[20] Emna Amdouni, Arkopaul Sarkar, Clement Jonquet, and Mohamed Hedi Karray. IndustryPortal: a Common Repository for FAIR Ontologies in Industry 4.0. In *ISWC 2023 – 22$^{nd}$ International Semantic Web Conference, demo & poster session*, Athens, Greece, 2023.

[21] Amélie Cordier, Marie Lefevre, Pierre-Antoine Champin, Olivier Georgeon, and Alain Mille. Trace-Based Reasoning – Modeling Interaction Traces for Reasoning on Experiences. In *The 26$^{th}$ International FLAIRS Conference*, 2013.

[22] Amina Annane, Nathalie Aussenac-Gilles, and Mouna Kamel. BBO: BPMN 2.0 Based Ontology for Business Process Representation. In *20$^{th}$ European Conference on Knowledge Management (ECKM)*, Lisbon, Portugal, 2019. Academic Conferences and publishing limited.

[23] Ana Iglesias-Molina, Dylan Van Assche, Julián Arenas-Guerrero, Ben De Meester, Christophe Debruyne, Samaneh Jozashoori, Pano Maria, Franck Michel, David Chaves-Fraga, and Anastasia Dimou. The RML Ontology: A Community-Driven Modular Redesign After a Decade of Experience in Mapping Heterogeneous Data to RDF. In *The Semantic Web – ISWC 2023*. Springer Nature Switzerland, 2023.

[24] Anastasia Dimou, Miel Vander Sande, Pieter Colpaert, Ruben Verborgh, Erik Mannens, and Rik Van de Walle. RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data. In *Proceedings of the Workshop on Linked Data on the Web, LDOW 2014, co-located with the 23$^{rd}$ International World Wide Web Conference (WWW 2014)*. CEUR-WS.org, 2014.

[25] Anastasia Dimou, Tom De Nies, and Ruben Verborgh. Automated Metadata Generation for Linked Data Generation and Publishing Workflows. In *Proceedings of the Workshop on Linked Data on the Web, LDOW 2016, co-located with 25$^{th}$ International World Wide Web Conference (WWW 2016)*. CEUR-WS.org, 2016.

[26] Andreas Ekelhart, Fajar J. Ekaputra, and Elmar Kiesling. The SLOGERT Framework for Automated Log Knowledge Graph Construction. In *The Semantic Web*. Springer International Publishing, 2021.

[27] Andreas M. Hein and Stephen Baxter. Artificial Intelligence for Interstellar Travel. *Journal of the British Interplanetary Society (JBIS)*, 2019.

[28] Andrei Brazhuk. Security Patterns Based Approach to Automatically Select Mitigations in Ontology-Driven Threat Modelling. In *Open Semantic Technologies for Intelligent Systems (OSTIS)*, 2020.

[29] Andrei Brazhuk. Threat Modeling of Cloud Systems with Ontological Security Pattern Catalog. *International Journal of Open Information Technologies*, 2021.

[30] Andrew Clark, George Mohay, and Bradley Schatz. Rich Event Representation for Computer Forensics. In *Proceedings of the Fifth Asia Pacific Industrial Engineering and Management Systems Conference*, Gold Coast, Australia, 2004.

[31] Anna Klimovskaia, David Lopez-Paz, Léon Bottou, and Maximilian Nickel. Poincaré Maps for Analyzing Complex Hierarchies in Single-Cell Data. *Nature Communications*, 2020.

[32] Anouk Barberousse. Observation et Calcul – Les données traitées informatiquement sont-elles encore des données d'observation ?, 2011.

[33] ANSSI. EBIOS Risk Manager. Technical Report ANSSI-PA-048, Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI), 2018.

[34] Anton Freund. Analyse de concepts formels et complexes simpliciaux : comparaison de deux approches pour l'analyse des relations binaires. https://medias.ircam.fr/x32347f_analyse-de-concepts-formels-et-complexes-s, 2013.

[35] Antonin Delpeuch, Adrian Pohl, Fabian Steeg, Thad Guidry Sr., and Osma Suominen. Reconciliation Service API – A Protocol for Data Matching on the Web. Final Community Group Report reconciliation/CG-FINAL-specs-0.2-20230410, W3C, 2023.

[36] Areg Melik-Adamyan. Achieve High-Performance Scaling for E2E Machine Learning and Data Analytics Workflows (2704813), 2020.

[37] Aritran Piplai, Sudip Mittal, Anupam Joshi, Tim Finin, James Holt, and Richard Zak. Creating Cybersecurity Knowledge Graphs From Malware After Action Reports. *IEEE Access*, 2020.

[38] Armin Haller, Krzysztof Janowicz, Simon Cox, Danh Le Phuoc, Kerry Taylor, and Maxime Lefrançois. Semantic Sensor Network Ontology. W3C Recommendation OGC 16-079, W3C, 2017.

[39] Arnaud Vandecasteele and Aldo Napoli. An Enhanced Spatial Reasoning Ontology for Maritime Anomaly Detection. In $7^{th}$ *International Conference on System of Systems Engineering (SoSE)*. IEEE, 2012.

[40] Diego Arroyuelo, Aidan Hogan, Gonzalo Navarro, Juan Reutter, and Domagoj Vrgoč. Tackling Challenges in Implementing Large-Scale Graph Databases. *Commun. ACM*, 2024.

[41] Assistant Secretary of Defence for Networks and Information Integration, Department of Defence Chief Information Officer. Electronic Records Management Software Applications Design Criteria Standard. Technical Report DoD 5015.02-STD, United States Department of Defense, 2007.

[42] Aviad Elitzur, Rami Puzis, and Polina Zilberman. Attack Hypothesis Generation. In *European Intelligence and Security Informatics Conference (EISIC)*, 2019.

[43] Jørgen Bang-Jensen and Gregory Z. Gutin. *Digraphs: Theory, Algorithms and Applications*. Springer Monographs in Mathematics. Springer-Verlag, 2009.

[44] Aaron Bangor, Philip T. Kortum, and James T. Miller. Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale. *Journal of Usability Studies archive*, 2009.

[45] Jānis Bārzdiņš, Guntis Bārzdiņš, Kārlis Čerāns, Renārs Liepiņš, and Artūrs Sproģis. UML Style Graphical Notation and Editor for OWL 2. In *Perspectives in Business Informatics Research*. Springer Berlin Heidelberg, 2010.

[46] Ben Goertzel, Cassio Pennachin, and Nil Geisweiller. *Engineering General Intelligence, Part 1: A Path to Advanced AGI via Embodied Learning and Cognitive Synergy*. Atlantis Press, 2014.

[47] Ben Stopford. Log-Structured Merge Tree. http://www.benstopford.com/2015/02/14/log-structured-merge-trees/, 2015.

[48] Bernard Berthomieu, Pierre-Olivier Ribet, and François Vernadat. The tool TINA – Construction of abstract state spaces for petri nets and time petri nets. *International Journal of Production Research*, 2004.

[49] Bernard Chabot. Le dicho-scope et les dichotomies génériques. https://www.linkedin.com/pulse/le-dicho-scope-et-les-dichotomies-g%C3%A9n%C3%A9riques-bernard-chabot/, 2017.

[50] Tim Berners-Lee, Roy T. Fielding, and Larry M Masinter. Uniform Resource Identifier (URI): Generic Syntax. RFC 3986, 2005.

[51] Tarek R. Besold, Artur d'Avila Garcez, Sebastian Bader, Howard Bowman, Pedro Domingos, Pascal Hitzler, Kai-Uwe Kuehnberger, Luis C. Lamb, Daniel Lowd, Priscila Machado Vieira Lima, Leo de Penning, Gadi Pinkas, Hoifung Poon, and Gerson Zaverucha. Neural-Symbolic Learning and Reasoning: A Survey and Interpretation, 2017.

[52] Martin Björklund. The YANG 1.1 Data Modeling Language. RFC 7950, 2016.

[53] Bogumił Kamiński, Michał Jakubczyk, and Przemysław Szufel. A Framework for Sensitivity Analysis of Decision Trees. *Central European Journal of Operations Research*, 2018.

[54] Boris Motik, Peter F. Patel-Schneider, Bijan Parsia, Conrad Bock, Achille Fokoue, Peter Haase, Rinke Hoekstra, Ian Horrocks, Alan Ruttenberg, Uli Sattler, and Michael Smith.

OWL 2 Web Ontology Language – Structural Specification and Functional-Style Syntax (Second Edition). W3C Recommendation, W3C, 2012.

[55] Belkacem Ould Bouamama. Conception intégrée pour la surveillance robuste des systèmes. *Techniques de l'Ingénieur*, 2013.

[56] Tai-Danae Bradley. At the Interface of Algebra and Statistics, 2020.

[57] Bram Steenwinckel. IBCNServices/Folio-Ontology. https://github.com/IBCNServices/Folio-Ontology, 2019.

[58] Bram Steenwinckel, Dieter De Paepe, Sander Vanden Hautte, Pieter Heyvaert, Mohamed Bentefrit, Pieter Moens, Anastasia Dimou, Bruno Van Den Bossche, Filip De Turck, Sofie Van Hoecke, and Femke Ongenae. FLAGS: A Methodology for Adaptive Anomaly Detection and Root Cause Analysis on Sensor Data Streams by Fusing Expert Knowledge with Machine Learning. *Future Generation Computer Systems*, 2021.

[59] Bram Steenwinckel, Gilles Vandewiele, Michael Weyns, Terencio Agozzino, Filip De Turck, and Femke Ongenae. INK: Knowledge Graph Embeddings for Node Classification. *Data Mining and Knowledge Discovery*, 2022.

[60] Bram Steenwinckel, Gilles Vandewiele, Pieter Bonte, Michael Weyns, Heiko Paulheim, Petar Ristoski, Filip De Turck, and Femke Ongenae. Walk Extraction Strategies for Node Embeddings with RDF2Vec in Knowledge Graphs. In *Database and Expert Systems Applications (DEXA) Workshops*, 2021.

[61] Bram Steenwinckel, Gilles Vandewiele, Terencio Agozzino, and Femke Ongenae. pyRDF2Vec: A Python Implementation and Extension of RDF2Vec. In *Extended Semantic Web Conference (ESWC)*, 2023.

[62] Bram Steenwinckel, Pieter Heyvaert, Dieter De Paepe, Olivier Janssens, Sander Vanden Hautte, Anastasia Dimou, Filip De Turck, Sofie Van Hoecke, and Femke Ongenae. Towards Adaptive Anomaly Detection and Root Cause Analysis by Automated Extraction of Knowledge from Risk Analyses. In $9^{th}$ *International Semantic Sensor Networks Workshop (SSN)*, 2018.

[63] Brice Leporini. Event-Driven Architecture: State Is a Side Effect, 2021.

[64] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges, 2021.

[65] Jean-Christophe Buisson. *Concevoir son microprocesseur: structure des systèmes logiques*. Ellipses, 2006.

[66] Carol Stanton, Gilad Katz, and Dawn Song. Isolation Forest for Anomaly Detection. https://e3s-center.berkeley.edu/wp-content/uploads/2017/08/RET_CStanton-2015.pdf, 2015.

[67] Changgen Li, Liang Guo, Hongli Gao, and Yi Li. Similarity-Measured Isolation Forest: Anomaly Detection Method for Machine Monitoring Data. *IEEE Transactions on Instrumentation and Measurement*, 2021.

[68] Chengjin Xu, Mojtaba Nayyeri, Fouad Alkhoury, Hamed Shariat Yazdi, and Jens Lehmann. Temporal Knowledge Graph Embedding Model Based on Additive Time Series Decomposition. In *$19^{th}$ International Semantic Web Conference (ISWC)*, 2020.

[69] Chia-Cheng Yen, Wenting Sun, Hakimeh Purmehdi, Won Park, Kunal Rajan Deshmukh, Nishank Thakrar, Omar Nassef, and Adam Jacobs. Graph Neural Network Based Root Cause Analysis Using Multivariate Time-Series KPIs for Wireless Networks. In *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, Budapest, Hungary, 2022. IEEE.

[70] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.

[71] Philipp Cimiano and Johanna Völker. Text2Onto: a framework for ontology learning and data-driven change discovery. In *Proceedings of the $10^{th}$ International Conference on Natural Language Processing and Information Systems (NLDB)*. Springer-Verlag, 2005.

[72] CISCO Systems. Snort – Network Intrusion Detection & Prevention System. https://www.snort.org/.

[73] Benoît Claise. Cisco Systems NetFlow Services Export Version 9. RFC 3954, 2004.

[74] Claudia Villalonga, Muhammad Razzaq, Wajahat Khan, Hector Pomares, Ignacio Rojas, Sungyoung Lee, and Oresti Banos. Ontology-Based High-Level Context Inference for Human Behavior Identification. *Sensors*, 2016.

[75] CNSS Glossary Working Group. CNSSI 4009. Technical Report CNSSI No. 4009, Committee on National Security Systems (CNSS), 2015.

[76] Taco Cohen. Towards a Grounded Theory of Causation for Embodied AI. In *Workshop on Causal Representation Learning*, 2022.

[77] Commission Nationale Informatique & Libertés (CNIL). Guide pratique : Les durées de conservation. https://www.cnil.fr/sites/cnil/files/atoms/files/guide_durees_de_conservation.pdf, 2020.

[78] Manuel Costa, Jon Crowcroft, Miguel Castro, Antony Rowstron, Lidong Zhou, Lintao Zhang, Paul Barham, and Ant Rowstron. Vigilante: End-to-End Containment of Internet Worm Epidemics. *ACM Transactions on Computer Systems*, 2008.

[79] Pierre Dagnely, Tom Ruette, Tom Tourwé, and Elena Tsiporkova. Ontology-Driven Multilevel Sequential Pattern Mining: Mining for Gold in Event Logs of Photovoltaic Plants. In *2018 International Conference on Intelligent Systems (IS)*, 2018.

[80] Dan Brickley, Leigh Dodds, and Libby Miller. Term-Centric Semantic Web Vocabulary Annotations. W3c interest group note, W3C, 2009.

[81] Dan Brickley and Libby Miller. Friend of a Friend (FOAF) Vocabulary Specification. http://xmlns.com/foaf/spec/, 2004.

[82] Dan Brickley and Ramanathan V. Guha. RDF Schema. W3C Recommendation, W3C, 2014.

[83] Dan Geiger and Judea Pearl. Logical and Algorithmic Properties of Independence and Their Application to Bayesian Networks. *Annals of Mathematics and Artificial Intelligence*, 1990.

[84] Dario Bonino and Fulvio Corno. DogOnt – Ontology Modeling for Intelligent Domotic Environments. In *The Semantic Web – ISWC 2008*. Springer, 2008.

[85] Geneviève Dauphin-Tanguy. Les bond graphs et leur application en mécatronique, 1999.

[86] Dave Reynolds. The Organization Ontology. W3C Recommendation, W3C, 2014.

[87] David Allison, Paul Smith, and Kieran Mclaughlin. Digital Twin-Enhanced Incident Response for Cyber-Physical Systems. In *Proceedings of the $18^{th}$ International Conference on Availability, Reliability and Security*, ARES '23. Association for Computing Machinery, 2023.

[88] David Huynh. Freebase Parallax: A New Way to Browse and Explore Data. https://vimeo.com/1513562, 2008.

[89] David Martin, Mark Burstein, Jerry Hobbs, Ora Lassila, Drew McDermott, Sheila McIlraith, Srini Narayanan, Massimo Paolucci, Bijan Parsia, Terry Payne, Evren Sirin, Naveen Srinivasan, and Katia Sycara. OWL-S: Semantic Markup for Web Services. Technical Report OWL-S 1.1, DARPA Agent Markup Language (DAML) Program, 2004.

[90] David Swift. A Practical Application of SIM/SEM/SIEM Automating Threat Identification. White Paper, SANS Institute, 2007.

[91] Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, and Michael Grossniklaus. C-SPARQL: SPARQL for Continuous Querying. In *Proceedings of the $18^{th}$ International Conference on World Wide Web*, New York, NY, USA, 2009. Association for Computing Machinery.

[92] Diane Maillot-Tchofo, Ahmed Triki, Maxime Laye, and John Puentes. Clustering of Live Network Alarms Using Unsupervised Statistical Models. In *49$^{th}$ European Conference on Optical Communications (ECOC)*, Glasgow, Scotland, 2023.

[93] Dmitry Vengertsev and Hemal Thakkar. Anomaly Detection in Graph: Unsupervised Learning, Graph-based Features and Deep Architecture. Technical Report, Department of Computer Science, Stanford University, 2015.

[94] Jens Domke, Torsten Hoefler, and Satoshi Matsuoka. Fail-in-Place Network Design: Interaction Between Topology, Routing Algorithm and Failures. In *SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2014.

[95] Dorota Owczarek. Lambda vs. Kappa Architecture. A Guide to Choosing the Right Data Processing Architecture for Your Needs. https://nexocode.com/blog/posts/lambda-vs-kappa-architecture/, 2022.

[96] Dörthe Arndt, Jeen Broekstra, Bob DuCharme, Ora Lassila, Peter F. Patel-Schneider, Eric Prud'hommeaux, Ted Thibodeau, Jr., and Bryan Thompson. RDF-Star and SPARQL-Star. Draft community group report, W3C, 2021.

[97] Dragan Gašević and Vladan Devedžić. Petri Net Ontology. *Knowledge-Based Systems*, 2006.

[98] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal Graph Networks for Deep Learning on Dynamic Graphs. In *Graph Representation Learning and Beyond (GRL+), ICML Workshop*, 2020.

[99] Emna Amdouni and Clement Jonquet. FAIR or FAIRer? An Integrated Quantitative Fairness Assessment Grid for Semantic Resources and Ontologies. In *Metadata and Semantic Research*. Springer International Publishing, 2022.

[100] Eoghan Casey, Sean Barnum, Ryan Griffith, Jonathan Snyder, Harm van Beek, and Alex Nelson. Advancing Coordinated Cyber-Investigations and Tool Interoperability Using a Community Developed Specification Language. *Digital Investigation*, 2017.

[101] Eric Freeman, Elisabeth Robson, Kathy Sierra, and Bert Bates, editors. *Head First Design Patterns*. O'Reilly, Sebastopol, CA, 2004.

[102] Erkuden Rios, Eider Iturbe, Angel Rego, Nicolas Ferry, Jean-Yves Tigli, Stéphane Lavirotte, Gerald Rocher, Phu Nguyen, Hui Song, Rustem Dautov, Wissam Mallouli, and Ana Rosa Cavalli. The DYNABIC Approach to Resilience of Critical Infrastructures. In *Proceedings of the 18$^{th}$ International Conference on Availability, Reliability and Security*, ARES '23. Association for Computing Machinery, 2023.

[103] Alex Eskin and Maryam Mirzakhani. Invariant and Stationary Measures for the $SL(2,\mathbb{R})$ Action on Moduli Space. *Publications mathématiques de l'IHÉS*, 2018.

[104] ETSI. Method and pro Forma for Threat, Vulnerability, Risk Analysis (TVRA). Technical Specification ETSI TS 102 165-1 V5.2.3 (2017-10), ETSI, 2017.

[105] ETSI. Context Information Management (CIM); Information Model (MOD0). Group Specification DGS/CIM-006-MOD0, ETSI, 2019.

[106] European Commission. Directive on measures for a high common level of cybersecurity across the Union (NIS2 Directive). http://data.europa.eu/eli/dir/2022/2555/2022-12-27, 2022.

[107] Evaggelia Pitoura. Historical Graphs: Models, Storage, Processing. In *Business Intelligence and Big Data*, Lecture Notes in Business Information Processing. Springer International Publishing, 2018.

[108] Eyke Hüllermeier. *Case-Based Approximate Reasoning*. Theory and Decision Library B. Springer Netherlands, 2007.

[109] F. William Lawvere and Stephen H. Schanuel. *Conceptual Mathematics: A First Introduction to Categories*. Cambridge University Press, Cambridge, UK, $2^{nd}$ edition, 2009.

[110] Faranak Sobhani and Umberto Straccia. Towards a Forensic Event Ontology to Assist Video Surveillance-based Vandalism Detection. In *Proceedings of the $34^{th}$ Italian Conference on Computational Logic*, Trieste, Italy, 2019.

[111] Federico Bianchi and Pascal Hitzler. On the Capabilities of Logic Tensor Networks for Deductive Reasoning. In *AAAI 2019 Spring Symposium on Combining Machine Learning with Knowledge Engineering (AAAI-MAKE)*, Stanford University, Palo Alto, California, USA, 2019.

[112] Mark Fedor, Martin Lee Schoffstall, James R. Davin, and Dr. Jeff D. Case. Simple Network Management Protocol (SNMP). RFC 1157, 1990.

[113] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation Forest. In *2008 Eighth IEEE International Conference on Data Mining*, 2008.

[114] Ferdinand Wagner. *Modeling Software with Finite State Machines: A Practical Approach*. Auerbach Publications, 2006.

[115] Abraham D. Flaxman. Finding shortest path with SPARQL query. https://stackoverflow.com/questions/14388864/finding-shortest-path-with-sparql-query#14458932, 2013.

[116] Igor Nai Fovino, Ricardo Neisse, Alessandro Lazari, Gianpaolo Ruzzante, Nineta Polemi, and Malgorzata Figwer. European Cybersecurity Centres of Expertise Map: Definitions and Taxonomy. Technical Report JRC 111441, European Union, 2018.

[117] Franck Berthelon and Peter Sander. Emotion Ontology for Context Awareness. In *2013 IEEE 4$^{th}$ International Conference on Cognitive Infocommunications (CogInfoCom)*, 2013.

[118] Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter. *Handbook of Knowledge Representation*. Elsevier Science, 2007.

[119] Franz Baader, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.

[120] République Française. Code des postes et des communications électroniques – Article R10-13. https://www.legifrance.gouv.fr/codes/article_lc/LEGIARTI000025622766/, 2012.

[121] George Karypis, Vipin Kumar, and Yijiang Huang. Multilevel Graph Partitioning. https://people.csail.mit.edu/jshun/6886-s18/lectures/lecture13-1.pdf, 2018.

[122] Gerald H, Sitt Min Oo, Gertjan De Mulder, Michiel Derveeuw, Pieter Heyvaert, Wouter Maroy, Vincent Emonet, kmhaeren, Ben De Meester, Dylan Van Assche, Thomas, and ajuvercr. RMLio/RMLStreamer, 2022.

[123] Rainer Gerhards. The Syslog Protocol. RFC 5424, 2009.

[124] Dominic Giampaolo. *Practical File System Design with the Be File System*. Morgan Kaufmann Publishers, Inc., 1999.

[125] Gilbert Laycoc. *The Theory and Practice of Specification Based Software Testing*. PhD thesis, University of Sheffield, Department of Computer Science, 1992.

[126] Gilles Privat and Abdullah Abbas. "Cyber-Physical Graphs" vs. RDF Graphs. In *W3C Workshop on Web Standardization for Graph Data*, 2019.

[127] Birte Glimm, Ian Horrocks, Boris Motik, and Giorgos Stoilos. Optimising Ontology Classification. In *Proc. of the 9$^{th}$ Int. Semantic Web Conf. (ISWC)*. Springer, 2010.

[128] Graham Cormode and S. Muthukrishnan. An Improved Data Stream Summary: The Count-Min Sketch and Its Applications. *Journal of Algorithms*, 2005.

[129] Nicola Guarino and Christopher A. Welty. *An Overview of OntoClean*, pages 201–220. Springer Berlin Heidelberg, 2009.

[130] Guillaume Brogi and Valerie Viet Triem Tong. TerminAPTor: Highlighting Advanced Persistent Threats through Information Flow Tracking. In *8$^{th}$ IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 2016.

[131] Guillaume Gronier. F-SUS : la traduction Française du System Usability Scale. http://www.guillaumegronier.com/cv/blog/files/6545bc93a9d0952c2afac2581129ae7c-0.html, 2021.

[132] Guohui Xiao, Davide Lanti, Roman Kontchakov, Sarah Komla-Ebri, Elem Güzel-Kalaycı, Linfang Ding, Julien Corman, Benjamin Cogrel, Diego Calvanese, and Elena Botoeva. The Virtual Knowledge Graph System Ontop. In *The Semantic Web – ISWC 2020*, 2020.

[133] Gustave Guillaume, Olivier Soutet, and Roch Valin. *Temps et verbe suivi de l'architectonique du temps dans les langues classiques: théorie des aspects, des modes et des temps*. Honoré Champion éditeur, Paris, reproduction en fac-similé edition, 2021.

[134] Gustavo González-Granadillo, Susana González-Zarzosa, and Rodrigo Diaz. Security Information and Event Management (SIEM): Analysis, Trends, and Usage in Critical Infrastructures. *Sensors*, 2021.

[135] Hajo Rijgersberg, Mark van Assem, and Jan Top. Ontology of Units of Measure and Related Concepts. *Semantic Web*, 2013.

[136] Halil Ertan. Single Model Based Anomaly Detection for Multi-Item Datasets. https://towardsdatascience.com/single-model-based-anomaly-detection-for-multi-item-datasets-e4dded5eeeb1, 2020.

[137] Hari Koduvely. Anomaly Detection through Reinforcement Learning, 2018.

[138] Harpreet Kaur and Raman Maini. Identification of Recurring Patterns of Code to Detect Structural Clones. In *6$^{th}$ International Conference on Advanced Computing (IACC)*, 2016.

[139] Harry Chen and Dan Brickley. Geonames ontology in OWL. https://web.archive.org/web/20080212144050/http://www.geospatialsemanticweb.com/2006/10/14/geonames-ontology-in-owl, 2006.

[140] Hélène Fargier, Pierre Marquis, Alexandre Niveau, and Nicolas Schmidt. Carte de compilation des diagrammes de décision ordonnés a valeurs réelles. In *Huitièmes Journées de l'Intelligence Artificielle Fondamentale (JIAF)*, Angers, France, 2014.

[141] Henry Heberle, Gabriela Vaz Meirelles, Felipe R. da Silva, Guilherme P. Telles, and Rosane Minghim. InteractiVenn: A Web-Based Tool for the Analysis of Sets through Venn Diagrams. *BMC Bioinformatics*, 2015.

[142] Holger Knublauch and Dimitris Kontokostas. Shapes Constraint Language (SHACL). W3C Recommendation, W3C, 2017.

[143] Holger Knublauch, James A. Hendler, and Kingsley Idehen. SPIN – Overview and Motivation. W3C Member Submission, W3C, 2011.

[144] Hongyu Ren, Mikhail Galkin, Michael Cochez, Zhaocheng Zhu, and Jure Leskovec. Neural Graph Reasoning: Complex Logical Query Answering Meets Graph Databases, 2023.

[145] Brian K. Houghton. Terrorism Knowledge Base: A Eulogy (2004-2008). *Perspectives on terrorism*, 2010.

[146] Hristiyan Pehlivanov. CQRS Is an Anti-Pattern for DDD. https://dzone.com/articles/cqrs-is-an-anti-pattern-for-ddd, 2020.

[147] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission, W3C, 2004.

[148] IBM. IBM Security I2 Enterprise Insight Analysis – Overview. https://www.ibm.com/products/i2-enterprise-insight-analysis, 2020.

[149] IEEE. IEEE Standard for Ethernet. Technical report, IEEE, 2018.

[150] Ilaria Maresi. A Data Engineer's Guide to Semantic Modelling. Technical report, The Hyve, 2020.

[151] Iman Akbari, Mohammad A. Salahuddin, Leni Ven, Noura Limam, Raouf Boutaba, Bertrand Mathieu, Stephanie Moteau, and Stephane Tuffin. Traffic Classification in an Increasingly Encrypted Web. *Communications of the ACM*, 2022.

[152] HappyFox Inc. HappyFox Help Desk. https://www.happyfox.com/.

[153] Ines Chami, Sami Abu-El-Haija, Bryan Perozzi, Christopher Ré, and Kevin Murphy. Machine Learning on Graphs: A Model and Comprehensive Taxonomy, 2020.

[154] International Atomic Energy Agency. Design Basis Threat (DBT). https://www.iaea.org/topics/security-of-nuclear-and-other-radioactive-material/design-basis-threat, 2019.

[155] Irlan Grangel-Gonzalez, Lavdim Halilaj, Gokhan Coskun, Soren Auer, Diego Collarana, and Michael Hoffmeister. Towards a Semantic Administrative Shell for Industry 4.0 Components. In *2016 IEEE Tenth International Conference on Semantic Computing (ICSC)*. IEEE, 2016.

[156] Ismail Harrando, Pasquale Lisena, and Raphael Troncy. Apples to Apples: A Systematic Evaluation of Topic Models. In *Recent Advances in Natural Language Processing (RANLP)*, 2021.

[157] Ismail Harrando and Raphaël Troncy. Explainable Zero-Shot Topic Extraction Using a Common-Sense Knowledge Graph. In *3rd Conference on Language, Data and Knowledge (LDK 2021)*, Open Access Series in Informatics (OASIcs), Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[158] ISO. Information and Documentation – Records Management – Part 1: General. Technical Report 15489-1:2001, ISO, 2001.

[159] ISO/IEC. Information security, cybersecurity and privacy protection – Information security management systems – Requirements. Technical Report 27001:2022, ISO/IEC, 2022.

[160] ITEMO. Standards for Lightweight IT Service Management (FitSM). https://www.fitsm.eu/, 2022.

[161] ITU-T. X.200: Information Technology – Open Systems Interconnection – Basic Reference Model: The Basic Model. Recommendation X.200 (07/94), International Telecommunication Union (ITU), 1994.

[162] ITU-T. G.7710: Common Equipment Management Function Requirements. Recommendation G.7710/Y.1701, International Telecommunication Union (ITU), 2020.

[163] ITU-T. Y.3090: Digital twin network – Requirements and architecture. Recommendation Y.3090, International Telecommunication Union (ITU), 2022.

[164] ITU-T/CCITT. ITU-T Rec. X.721 (02/92) Information Technology – Open Systems Interconnection – Structure of Management Information: Definition of Management Information. Recommendation, International Telecommunication Union (ITU), 1992.

[165] ITU-T/CCITT. ITU-T Rec. X.733 (02/92) Information technology – Open Systems Interconnection – Systems Management: Alarm reporting function. Recommendation, International Telecommunication Union (ITU), 1992.

[166] ITU-T/CCITT. ITU-T Rec. X.735 (09/92) Information Technology – Open Systems Interconnection – Systems Management: Log Control Function. Recommendation, International Telecommunication Union (ITU), 1992.

[167] J. Halpern and C. Pignataro. Service Function Chaining (SFC) Architecture. RFC 7665, 2015.

[168] Jacopo Urbani and Ceriel Jacobs. Adaptive Low-level Storage of Very Large Knowledge Graphs. In *Proceedings of The Web Conference 2020*. Association for Computing Machinery, 2020.

[169] Jan Portisch and Heiko Paulheim. Walk This Way! Entity Walks and Property Walks for RDF2vec, 2022.

[170] Jan Portisch, Nicolas Heist, and Heiko Paulheim. Knowledge Graph Embedding for Data Mining vs. Knowledge Graph Embedding for Link Prediction – Two Sides of the Same Coin? *Semantic Web*, 2022.

[171] Jan Struyf, Jan Ramon, and Hendrik Blockeel. Compact Representation of Knowledge Bases in ILP. In *Inductive logic programming*, 2003.

[172] Jean-François Nogier, Thierry Bouillot, and Jules Leclerc. *Ergonomie des interfaces : guide pratique pour la conception des applications Web, logicielles, mobiles et tactiles*. Dunod, Paris, 2011.

[173] Jedrzej Potoniec, Dawid Wiśniewski, Agnieszka Ławrynowicz, and C. Maria Keet. Dataset of Ontology Competency Questions to SPARQL-OWL Queries Translations. *Data in Brief*, 2020.

[174] Jiabao Zhang, Shenghua Liu, Wenting Hou, Siddharth Bhatia, Huawei Shen, Wenjian Yu, and Xueqi Cheng. AugSplicing: Synchronized Behavior Detection in Streaming Tensors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Virtual Conference, 2021.

[175] Jieming Zhu, Shilin He, Jinyang Liu, Pinjia He, Qi Xie, Zibin Zheng, and Michael R. Lyu. Tools and Benchmarks for Automated Log Parsing. In *$41^{st}$ International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 2019.

[176] Jim Jeffers, James Reinders, and Avinash Sodani. *Intel Xeon Phi Processor High Performance Programming*. Morgan Kaufmann is an imprint of Elsevier, Cambridge, MA, 2016.

[177] Jinlin Yang, David Evans, Deepali Bhardwaj, Thirumalesh Bhat, and Manuvir Das. Perracotta: Mining Temporal API Rules from Imperfect Traces. In *Proceedings of the $28^{th}$ International Conference on Software Engineering*. Association for Computing Machinery, 2006.

[178] Johannes Koch, Carlos A. Velasco, and Philip Ackermann. HTTP Vocabulary in RDF 1.0. W3c working group note, W3C, 2017.

[179] John Blackford. Bulk Data Collection. Technical Report TR-232, The Broadband Forum, 2012.

[180] John Brooke. SUS: A Quick and Dirty Usability Scale. *Usability Eval. Ind.*, 1995.

[181] John Brooke. SUS: A Retrospective. *Journal of Usability Studies*, 2013.

[182] John Whitbeck, Marcelo Dias de Amorim, Vania Conan, and Jean-Loup Guillaume. Temporal Reachability Graphs. In *Proceedings of the 18$^{th}$ Annual International Conference on Mobile Computing and Networking*. ACM, 2012.

[183] Joint Task Force. Security and Privacy Controls for Information Systems and Organizations. Technical Report NIST SP 800-53r5, National Institute of Standards and Technology, 2020.

[184] Jorge Martinez-Gil, Georg Buchgeher, David Gabauer, Bernhard Freudenthaler, Dominik Filipiak, and Anna Fensel. Root Cause Analysis in the Industrial Domain Using Knowledge Graphs: A Case Study on Power Transformers. *Procedia Computer Science*, 2022.

[185] Josh Chessman. Magic Quadrant for Network Performance Monitoring and Diagnostics. Technical Report G00463582, Gartner, 2020.

[186] Juan C. Vidal, Manuel Lama, and Alberto Bugarın. A High-level Petri Net Ontology Compatible with PNML. *Petri Net Newsletter*, 2006.

[187] Julian Dibbelt, Ben Strasser, and Dorothea Wagner. Customizable Contraction Hierarchies. *ACM Journal of Experimental Algorithmics*, 2016.

[188] Kathy Haan. Top Website Statistics For 2023. https://www.forbes.com/advisor/business/software/website-statistics/, 2023.

[189] Kelly Kavanagh, Toby Bussa, and Gorka Sadowski. Magic Quadrant for Security Information and Event Management. Technical Report G00348811, Gartner, 2018.

[190] Kent Campbell. Seven Free Wikipedia Alternatives. https://blog.reputationx.com/wikipedia-alternatives, 2023. Accessed: 2023-08-10.

[191] Kent Erickson. Layer 2 Network Connection Awareness Improves Root Cause Analysis. https://www.zenoss.com/blog/layer-2-network-connection-awareness-improves-root-cause-analysis, 2015.

[192] Kingsley Uyi Idehen. Virtuoso 8.0: Creating a Custom Inference Rules Using SPIN Vocabulary. https://medium.com/virtuoso-blog/virtuoso-8-0-creating-a-custom-inference-rules-using-spin-vocabulary-d7a060f859ef, 2019.

[193] Adila Krisnadhi and Pascal Hitzler. A Core Pattern for Events. In *Advances in Ontology Design and Patterns [revised and extended versions of the papers presented at the $7^{th}$ edition of the Workshop on Ontology and Semantic Web Patterns, WOP@ISWC 2016, Kobe, Japan, $18^{th}$ October 2016]*, Studies on the Semantic Web. IOS Press, 2016.

[194] Krzysztof Janowicz, Armin Haller, Simon Cox, Danh Phuoc, and Maxime Lefrançois. SOSA: A Lightweight Ontology for Sensors, Observations, Samples, and Actuators. *SSRN Electronic Journal*, 2018.

[195] L. Bai, E. R. Hancock, L. Han, and P. Ren. Graph Clustering Using Graph Entropy Complexity Traces. In *Proceedings of the $21^{st}$ International Conference on Pattern Recognition (ICPR)*, 2012.

[196] Pierre Laperdrix, Nataliia Bielova, Benoit Baudry, and Gildas Avoine. Browser Fingerprinting: A survey. *ACM Trans. Web*, 2020.

[197] Laura Held. Examples of Content Heavy Editorial Website Designs. https://www.newmediacampaigns.com/blog/best-examples-of-content-heavy-editorial-website-designs, 2021. Accessed: 2023-08-10.

[198] Laurens De Vocht, Sam Coppens, Ruben Verborgh, Miel Vander Sande, Erik Mannens, and Rik Van de Walle. Discovering Meaningful Connections between Resources in the Web of Data. In *Proceedings of the WWW2013 Workshop on Linked Data on the Web*, Rio de Janeiro, Brazil, 2013.

[199] Eugene L. Lawler. *Combinatorial Optimization: Networks and Matroids*. Dover Publications, 2001.

[200] Leigh Dodds and Ian Davis. Linked Data Patterns: A Pattern Catalogue for Modelling, Publishing, and Consuming Linked Data. https://patterns.dataincubator.org, 2012.

[201] Lex Fridman. Douglas Lenat: Cyc and the Quest to Solve Common Sense Reasoning in AI. https://youtu.be/3wMKoSRbGVs, 2021.

[202] Lionel Tabourier, Alina Stoica, and Fernando Peruani. How to Detect Causality Effects on Large Dynamical Communication Networks: A Case Study. In *Communication Systems and Networks (COMSNETS)*, 2012.

[203] Lionel Tailhardat, Raphaël Troncy, and Yoan Chabot. Walks in Cyberspace: Towards Better Web Browsing and Network Activity Analysis with 3D Live Graph Rendering. In *WWW '22: Companion Proceedings of the Web Conference 2022*. Association for Computing Machinery, 2022.

[204] Lionel Tailhardat, Raphaël Troncy, and Yoan Chabot. Leveraging Knowledge Graphs For Classifying Incident Situations in ICT Systems. In *18$^{th}$ International Conference on Availability, Reliability and Security (ARES)*, 2023.

[205] Lionel Tailhardat, Yoan Chabot, and Raphaël Troncy. Designing NORIA: a Knowledge Graph-based Platform for Anomaly Detection and Incident Management in ICT Systems. In *4$^{th}$ International Workshop on Knowledge Graph Construction (KGCW)*, 2023.

[206] Pasquale Lisena, Albert Meroño-Peñuela, Tobias Kuhn, and Raphaël Troncy. Easy Web API Development with SPARQL Transformer. In *18$^{th}$ International Semantic Web Conference (ISWC), In-Use Track*, Auckland, New Zealand, 2019.

[207] Liz Maida. A Picture Is Worth 1,000 Rows: Visualizing Security Data with Graph Algorithms. https://www.slideshare.net/neo4j/a-picture-is-worth-1000-rows, 2021.

[208] Chris M. Lonvick and Tatu Ylonen. The Secure Shell (SSH) Connection Protocol. RFC 4254, 2006.

[209] Louis K Scheffer, C Shan Xu, Michal Januszewski, and al. A Connectome and Analysis of the Adult Drosophila Central Brain. *eLife*, 2020.

[210] Lucas Potin, Rosa Figueiredo, Vincent Labatut, and Christine Largeron. Pattern Mining for Anomaly Detection in Graphs: Application to Fraud in Public Procurement. In *Machine Learning and Knowledge Discovery in Databases: Applied Data Science and Demo Track*. Springer Nature Switzerland, 2023.

[211] Luciano Serafini, Ivan Donadello, and Artur d'Avila Garcez. Learning and Reasoning in Logic Tensor Networks: Theory and Application to Semantic Image Interpretation. In *Proceedings of the Symposium on Applied Computing (SAC '17)*, New York, NY, USA, 2017. Association for Computing Machinery.

[212] M. Ghijsen, J. van der Ham, P. Grosso, C. Dumitru, H. Zhu, Z. Zhao, and C. de Laat. A Semantic-Web Approach for Modeling Computing Infrastructures. *Computers & Electrical Engineering*, 2013.

[213] M. Levene and G. Loizou. A Graph-Based Data Model and Its Ramifications. *IEEE Transactions on Knowledge and Data Engineering*, 1995.

[214] Madhu Murali. 11 Examples of One-Page Websites to Inspire You. https://blog.hubspot.com/website/11-examples-of-one-page-websites-for-inspiration, 2023. Accessed: 2023-08-10.

[215] Mads Holten Rasmussen, Maxime Lefrançois, Georg Ferdinand Schneider, and Pieter Pauwels. BOT: The Building Topology Ontology of the W3C Linked Building Data Group. *Semantic Web Journal*, 2020.

[216] Malek Ben Salem and Chris Wacek. Enabling New Technologies for Cyber Security Defense with the ICAS Cyber Security Ontology. In *Proceedings of the Tenth Conference on Semantic Technology for Intelligence, Defense, and Security*, Fairfax VA, USA, 2015.

[217] Manish Thapa, Jose Espejo-Uribe, and Evangelos Pournaras. Measuring Network Reliability and Repairability Against Cascading Failures. *Journal of Intelligent Information Systems*, 2019.

[218] Marc Bouissou. *Gestion de la complexité dans les études quantitatives de sûreté de fonctionnement de systèmes*. Éditions Tec & Doc, Paris, 2008.

[219] Marco Rospocher, Chiara Ghidini, and Luciano Serafini. An Ontology for the Business Process Modelling Notation. In *Formal Ontology in Information Systems – Proceedings of the Eighth International Conference*. IOS Press, 2014.

[220] Maria Bermudez-Edo, Tarek Elsaleh, Payam Barnaghi, and Kerry Taylor. IoT-Lite: A Lightweight Semantic Model for the Internet of Things and Its Use with Dynamic Semantics. *Personal and Ubiquitous Computing*, 2017.

[221] Maria De Marsico and Stefano Levialdi. Evaluating web sites: exploiting user's expectations. *International Journal of Human-Computer Studies*, 2004.

[222] Maria Massri. *Designing a Temporal Graph Management System for IoT Application Domains*. PhD thesis, Université Rennes 1, Rennes, France, 2022.

[223] Maria Massri, Zoltan Miklos, Philippe Raipin, Pierre Meye, Amaury Bouchra Pilet, and Thomas Hassan. RTGEN++: A Relative Temporal Graph GENerator. *Future Generation Computer Systems*, 2023.

[224] María Poveda-Villalón, Alba Fernández-Izquierdo, Mariano Fernández-López, and Raúl García-Castro. LOT: An industrial oriented ontology engineering framework. Engineering Applications of Artificial Intelligence. *Engineering Applications of Artificial Intelligence*, 2022.

[225] 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)*, 2014.

[226] María Poveda-Villalón, Mari Carmen Suárez-Figueroa, Raúl García-Castro, and Asunción Gómez-Pérez. A Context Ontology for Mobile Environments. In *Proceedings of the Second Workshop on Context, Information and Ontologies*, 2010.

[227] Mark A. Musen. The Protégé Project: A Look Back and a Look Forward. *AI matters*, 2015.

[228] Mark Proctor. Drools: A Rule Engine for Complex Event Processing. In *Applications of Graph Transformations with Industrial Relevance*. Springer Berlin Heidelberg, 2012.

[229] Markus Triska. The Power of Prolog – Introduction to Modern Prolog. https://www.metalevel.at/prolog, 2005.

[230] MartinKL. Turning the Database Inside-out with Apache Samza. https://news.ycombinator.com/item?id=9145197, 2015.

[231] Marwan Ghanem, Clémence Magnien, and Fabien Tarissan. How to Exploit Structural Properties of Dynamic Networks to Detect Nodes with High Temporal Closeness. In *Cologne-Twente Workshop on Graphs and Combinatorial Optimization 2018 (CTW'18)*, Paris, France, 2018.

[232] Mary Brown, Roy M. Jenevein, and Nasr Ullah. Memory Access Pattern Analysis. In *Proceedings of the Workload Characterization: Methodology and Case Studies*, USA, 1998. IEEE Computer Society.

[233] Mathieu Garchery and Michael Granitzer. ADSAGE: Anomaly Detection in Sequences of Attributed Graph Edges Applied to Insider Threat Detection at Fine-Grained Level, 2020.

[234] Mathieu Lirzin and Béatrice Markhoff. Vers Une Ontologie Des Interactions HTTP. In *31$^{èmes}$ Journées Francophones d'Ingénierie Des Connaissances*, Angers, France, 2020.

[235] Mattijs Ghijsen, Jeroen Van Der Ham, Paola Grosso, Cosmin Dumitru, Hao Zhu, Zhiming Zhao, and Cees De Laat. A Semantic-Web Approach for Modeling Computing Infrastructures. *Computers & Electrical Engineering*, 2013.

[236] Mauro Dragoni, Tania Bailoni, Rosa Maimone, and Claudio Eccher. HeLiS: An Ontology for Supporting Healthy Lifestyles. In *The Semantic Web – ISWC 2018: 17$^{th}$ International Semantic Web Conference, Monterey, CA, USA, October 8–12, 2018, Proceedings, Part II*. Springer-Verlag, 2018.

[237] Mauro Dragoni, Tania Bailoni, Rosa Maimone, Michele Marchesoni, and Claudio Eccher. HORUS.AI – A Knowledge-Based Solution Supporting Health Persuasive Self-Monitoring. In *Proceedings of the ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks, co-located with 17$^{th}$ International Semantic Web Conference (ISWC)*, Monterey, USA, 2018.

[238] Maurras Ulbricht Togbe, Mariam Barry, Aliou Boly, Yousra Chabchoub, Raja Chiky, Jacob Montiel, and Vinh-Thuy Tran. Anomaly Detection for Data Streams Based on Isolation Forest Using Scikit-Multiflow. In *Computational Science and Its Applications – ICCSA 2020*, Cagliari, Italy, 2020. Springer International Publishing.

[239] Maxime Lefrançois. Smart Applications REFerence Ontology (SAREF). https://saref.etsi.org/, 2020.

[240] Maxime Lefrançois. Planned ETSI SAREF Extensions Based on the W3C&OGC SOSA/SSN-compatible SEAS Ontology Patterns. In *Workshop on Semantic Interoperability and Standardization in the IoT (SIS-IoT)*, 2017.

[241] Maxime Lefrançois, Jarmo Kalaoja, Takoua Ghariani, and Antoine Zimmermann. SEAS Knowledge Model. Deliverable 2.2, ITEA2 12004 Smart Energy Aware Systems, 2016.

[242] Joseph A. McNitt. Stability in Graph Dynamical Systems. Master's thesis, Faculty of the Virginia Polytechnic Institute and State University, 2018.

[243] Megan Katsumi and Mark Fox. Defining Activity Specifications in OWL. In *Proceedings of the $8^{th}$ Workshop on Ontology Design and Patterns (WOP 2017), co-located with the $16^{th}$ International Semantic Web Conference (ISWC 2017)*, Vienna, Austria, 2017.

[244] Megan Katsumi and Mark Fox. iCity Transportation Planning Suite of Ontologies. Technical report, University of Toronto, 2020.

[245] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling Relational Data with Graph Convolutional Networks. In *The Semantic Web*. Springer International Publishing, 2018.

[246] Miguel Hernan and Jamie Robins. *Causal Inference: What If*. Harvard College, 2020.

[247] Ming-Chang Lee, Jia-Chun Lin, and Ernst Gunnar Gran. How Far Should We Look Back to Achieve Effective Real-Time Time-Series Anomaly Detection? In *Advanced Information Networking and Applications*. Springer International Publishing, 2021.

[248] MISP project. MISP – Open Source Threat Intelligence Platform & Open Standards For Threat Information Sharing. https://www.misp-project.org/. Formely known as Malware Information Sharing Platform.

[249] Mohammed Alzaabi, Andy Jones, and Thomas A. Martin. An Ontology-Based Forensic Analysis Tool. In *Annual ADFSL Conference on Digital Forensics, Security and Law*, Richmond, Virginia, USA, 2013.

[250] Mozilla. Recommended Web Performance Timings: How Long Is Too Long? https://developer.mozilla.org/en-US/docs/Web/Performance/How_long_is_too_long, 2023.

[251] Myriam Lopez, Marie Beurton-Aimar, Gayo Diallo, and Sofian Maabout. Approche de traitement des logs pour la prédiction d'erreurs critiques. *Revue des Nouvelles Technologies de l'Information*, Extraction et Gestion des Connaissances, RNTI-E-37, 2021.

[252] Peter Naur. Programming as Theory Building. *Microprocessing and Microprogramming*, 1985.

[253] Neil C. Thompson, Kristjan Greenewald, Keeheon Lee, and Gabriel F. Manso. The Computational Limits of Deep Learning. In *Ninth Computing within Limits 2023 (LIMITS)*, 2023.

[254] Nicholas J. Car, Timo Homburg, Matthew Perry, John Herring, Frans Knibbe, Simon J.D. Cox, Joseph Abhayaratna, and Mathias Bonduel. OGC GeoSPARQL – A Geographic Query Language for RDF Data. OGC Implementation Standard, Open Geospatial Consortium, 2022.

[255] Nicolas Lazzari, Andrea Poltronieri, and Valentina Presutti. Classifying Sequences by Combining Context-Free Grammars and OWL Ontologies. In *The Semantic Web*. Springer Nature, 2023.

[256] Nidhi Rastogi, Sharmishtha Dutta, Mohammed J. Zaki, Alex Gittens, and Charu Aggarwal. MALOnt: An Ontology for Malware Threat Intelligence. In *Deployable Machine Learning for Security Defense*. Springer International Publishing, 2020.

[257] NIST. National Vulnerability Database (NVD). https://nvd.nist.gov/.

[258] Noam Ben-Asher, A. Oltramari, R. Erbacher, and Cleotilde González. Ontology-Based Adaptive Systems of Cyber Defense. In *$10^{th}$ Conference on Semantic Technology for Intelligence, Defense, and Security (STIDS)*, 2015.

[259] Nong Ye. A Markov Chain Model of Temporal Behavior for Anomaly Detection. In *Proceedings of the 2000 IEEE Workshop on Information Assurance and Security*, United States Military Academy, West Point, NY, USA, 2000.

[260] Norbert Martínez-Bazan, Victor Muntés-Mulero, Sergio Gómez-Villamor, Jordi Nin, Mario-A. Sánchez-Martínez, and Josep-L. Larriba-Pey. Dex: High-Performance Exploration on Large Graphs for Information Retrieval. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07. Association for Computing Machinery, 2007.

[261] Norbert Wiener, Ronan Le Roux, Robert Vallée, and Nicole Vallée. *La Cybernétique: information et régulation dans le vivant et la machine*. Sources Du Savoir. Éd. du Seuil, Paris, 2014.

[262] nponeccop. Graph Databases Utilizing Locality. https://stackoverflow.com/questions/24106195/graph-databases-utilizing-locality, 2014.

[263] Nyoman Juniarta, Miguel Couceiro, Amedeo Napoli, and Chedy Raıssi. Sequential Pattern Mining Using FCA and Pattern Structures for Analyzing Visitor Trajectories in a Museum. In *Proceedings of the Fourteenth International Conference on Concept Lattices and Their Applications*, Olomouc, Czech Republic, 2018.

[264] OASIS Open. Introduction to STIX – Structured Threat Information eXpression. https://oasis-open.github.io/cti-documentation/stix/intro.

[265] Marek Obitko, Václav Snásel, and Jan Smid. Ontology Design with Formal Concept Analysis. In *International Conference on Concept Lattices and their Applications (CLA)*, 2004.

[266] ODP. Ontology Design Patterns . org (ODP). http://ontologydesignpatterns.org.

[267] Oleg Mikhail Sheyner. *Scenario Graphs and Attack Graphs*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 2004.

[268] Open Networking Foundation. YANG, OpenConfig, and gNMI. https://opennetworking.org/wp-content/uploads/2019/10/NG-SDN-Tutorial-Session-2.pdf, 2019.

[269] Oracle. How MySQL Uses Indexes. https://dev.mysql.com/doc/refman/5.7/en/mysql-indexes.html, 2013.

[270] Oscar Corcho, David Chaves-Fraga, Jhon Toledo, Julián Arenas-Guerrero, Carlos Badenes-Olmedo, Mingxue Wang, Hu Peng, Nicholas Burrett, José Mora, and Puchao Zhang. A High-Level Ontology Network for ICT Infrastructures. In *$20^{th}$ International Semantic Web Conference (ISWC)*, 2021.

[271] K. Pagiamtzis and A. Sheikholeslami. Content-Addressable Memory (CAM) Circuits and Architectures: A Tutorial and Survey. *IEEE Journal of Solid-State Circuits*, 2006.

[272] Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection, 2016.

[273] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. Long Short Term Memory Networks for Anomaly Detection in Time Series. In *Computational Intelligence and Machine Learning*, Bruges, Belgium, 2015.

[274] Pankaj Prasad and Josh Chessman. Market Guide for IT Infrastructure Monitoring Tools. Technical Report G00450400, Gartner, 2019.

[275] James Parsons. Alexa.com Is Dead – Here Are 20 of the Best Alternatives. https://www.contentpowered.com/blog/alexa-com-dead-alternatives/, 2023.

[276] Pasquale Lisena, Konstantin Todorov, Cécile Cecconi, Françoise Leresche, Isabelle Canno, Frédéric Puyrenier, Martine Voisin, Thierry Le Meur, and Raphaël Troncy. Controlled Vocabularies for Music Metadata. In *$19^{th}$ International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

[277] Paul Cichonski, Tom Millar, Tim Grance, and Karen Scarfone. Computer Security Incident Handling Guide: Recommendations of the National Institute of Standards and Technology. Technical Report NIST SP 800-61r2, National Institute of Standards and Technology, 2012.

[278] Pavithra Vijay. Timeseries Anomaly Detection Using an Autoencoder. https://keras.io/examples/timeseries/timeseries_anomaly_detection/, 2020.

[279] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 2011.

[280] Pedro Alípio, José Neves, and Paulo Carvalho. Automatic Detection of SLS Violation Using Knowledge Based Systems. In *Knowledge-Based Intelligent Information and Engineering Systems*. Springer, Berlin, Heidelberg, 2006.

[281] Pedro Alipio, Paulo Carvalho, and José Neves. Using CLIPS to Detect Network Intrusions. In *Progress in Artificial Intelligence*. Springer, 2003.

[282] Yun Peng, Sen Lin, Qian Chen, Lyu Xu, Xiaojun Ren, Yafei Li, and Jianliang Xu. ChatGraph: Chat with Your Graphs, 2024.

[283] Petar Ristoski, Jessica Rosati, Tommaso Di Noia, Renato De Leone, and Heiko Paulheim. RDF2Vec: RDF Graph Embeddings and Their Applications. *Semantic Web*, 2019.

[284] Peter E. Kaloroumakis and Michael J. Smith. Toward a Knowledge Graph of Cybersecurity Countermeasures. Technical report, The MITRE Corporation, 2021.

[285] Peter Mell, Karen Scarfone, and Sasha Romanosky. Common Vulnerability Scoring System (CVSS). *IEEE Security Privacy*, 2006.

[286] Peter Sanders and Christian Schulz. Engineering Multilevel Graph Partitioning Algorithms. In *Algorithms – ESA 2011*. Springer Berlin Heidelberg, 2011.

[287] Petter Holme. Modern Temporal Network Theory: A Colloquium. *The European Physical Journal B*, 2015.

[288] Phil Blackwood. Gist Jumpstart. https://www.semanticarts.com/gist-jumpstart/, 2023.

[289] Philipp Grashorn, Jonas Hansen, and Marcel Rummens. How Airbus Detects Anomalies in ISS Telemetry Data Using TFX. https://blog.tensorflow.org/2020/04/how-airbus-detects-anomalies-iss-telemetry-data-tfx.html, 2020.

[290] Pierre-Antoine Champin, Alain Mille, and Yannick Prié. Vers des traces numériques comme objets informatiques de premier niveau. *Intellectica – La revue de l'Association pour la Recherche sur les sciences de la Cognition (ARCo)*, 2013.

[291] Pierre-Yves Vandenbussche, Ghislain A. Atemezing, María Poveda-Villalón, and Bernard Vatant. Linked Open Vocabularies (LOV): A Gateway to Reusable Semantic Vocabularies on the Web. *Semantic Web*, 2017.

[292] Pieter Bonte, Riccardo Tommasini, Emanuele Della Valle, Filip De Turck, and Femke Ongenae. Streaming MASSIF: Cascading Reasoning for Efficient Processing of IoT Data Streams. *Sensors*, 2018.

[293] Marcin Pietrasik and Marek Reformat. A Simple Method for Inducing Class Taxonomies in Knowledge Graphs. In *The Semantic Web*. Springer International Publishing, 2020.

[294] Qianru Zhou, Alasdair J. G. Gray, and Stephen McLaughlin. ToCo: An Ontology for Representing Hybrid Telecommunication Networks. In *$16^{th}$ European Semantic Web Conference (ESWC)*. Springer, 2019.

[295] Quentin Rousseau. Incident Management vs. Incident Response – What's the Difference? https://rootly.com/blog/incident-management-vs-incident-response-what-s-the-difference, 2021.

[296] Raghu Ramakrishnan and Johannes Gehrke. *Database Management Systems*. McGraw-Hill Higher Education. McGraw-Hill, $3^{rd}$, internat. edition, 2003.

[297] Ralph Hodgson. QUDT: Quantities, Units, Dimensions and Data Types Ontologies. https://doi.org/10.25504/FAIRsharing.d3pqw7, 2011. FAIRsharing.org.

[298] Randall Davis. Reasoning from First Principles in Electronic Troubleshooting. *International Journal of Man-Machine Studies*, 1983.

[299] Raphaël Troncy and Antoine Isaac. DOE : une mise en oeuvre d'une méthode de structuration différentielle pour les ontologies. In *Actes $13^{e}$ journées francophones sur Ingénierie des Connaissances (IC), Rouen (FR)*, 2002.

[300] Youssra Rebboud, Lionel Tailhardat, Pasquale Lisena, and Raphaël Troncy. Can LLMs Generate Competency Questions? In *Semantic Web – $21^{st}$ International Conference, ESWC 2024, LLMs for KE track, Hersonissos, Crete, Greece, May 26 – 30, 2024*, 2024.

[301] Renier Pelser van Heerden. *A Formalised Ontology for Network Attack Classification*. PhD thesis, Rhodes University, Grahamstown, South Africa, 2014.

[302] Renier van Heerden, Louise Leenen, and Barry Irwin. Automated Classification of Computer Network Attacks. In *2013 International Conference on Adaptive Science and Technology*, Pretoria, South Africa, 2013.

[303] Renzo Angles and Claudio Gutierrez. Survey of Graph Database Models. *ACM Computing Surveys*, 2008.

[304] Richard Caralli, James F. Stevens, Lisa R. Young, and William R. Wilson. Introducing OCTAVE Allegro: Improving the Information Security Risk Assessment Process. Technical report, Carnegie Mellon University, 2007.

[305] Richard Cyganiak, David Wood, and Markus Lanthaler. RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation, W3C, 2014.

[306] Riverbed Technology. Application Performance Monitoring for Microservices-Based Applications. https://www.aternity.com/blogs/monitoring-a-microservices-based-application/, 2017.

[307] Rob Bolton. Anomaly Detection and Root Cause Analysis with AppDynamics Cognition Engine. https://www.appdynamics.com/blog/product/anomaly-detection-root-cause-analysis-appdynamics-cognition-engine/, 2019.

[308] Rob Kitchin and Gavin McArdle. What Makes Big Data, Big Data? Exploring the Ontological Characteristics of 26 Datasets. *Big Data & Society*, 2016.

[309] Robert Faure, Bernard Lemaire, and Christophe Picouleau. *Précis de recherche opérationnelle: méthodes et exercices d'application*. Dunod, 2014.

[310] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, 2000.

[311] RSA-IDD-Legacy. Global NWP 11.5 Architecture Diagram. https://community.rsa.com/t5/image/serverpage/image-id/232564i79F0F53CE31E79A8, 2021.

[312] Russa Biswas, Lucie-Aimée Kaffee, Michael Cochez, Stefania Dumbrava, Theis E. Jendal, Matteo Lissandrini, Vanessa Lopez, Eneldo Loza Mencía, Heiko Paulheim, Harald Sack, Edlira Kalemi Vakaj, and Gerard de Melo. Knowledge Graph Embeddings: Open Challenges and Opportunities. *DROPS-IDN/v2/document/10.4230/TGDK.1.1.4*, 2023.

[313] Ryan McConville, Weiru Liu, and Paul Miller. Vertex Clustering of Augmented Graph Streams. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, 2015.

[314] Ryan Riegel, Alexander Gray, Francois Luus, Naweed Khan, Ndivhuwo Makondo, Ismail Yunus Akhalwaya, Haifeng Qian, Ronald Fagin, Francisco Barahona, Udit Sharma, Shajith Ikbal, Hima Karanam, Sumit Neelam, Ankita Likhyani, and Santosh Srivastava. Logical Neural Networks, 2020.

[315] S. N. Narayanan, A. Ganesan, K. Joshi, T. Oates, A. Joshi, and T. Finin. Early Detection of Cybersecurity Threats Using Collaborative Cognition. In *2018 IEEE $4^{th}$ International Conference on Collaboration and Internet Computing (CIC)*, 2018.

[316] Rik Sarkar. Low Distortion Delaunay Embedding of Trees in Hyperbolic Plane. In *Graph Drawing*, Lecture Notes in Computer Science. Springer, 2012.

[317] Sasan Saqaeeyan, Hamid Haj Seyyed Javadi, and Hossein Amirkhani. Anomaly Detection in Smart Homes Using Bayesian Networks. *KSII Transactions on Internet and Information Systems*, 2020.

[318] Scaled Agile, Inc. SAFe – Story. https://scaledagileframework.com/story/, 2022.

[319] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. Web Ontology Language (OWL). W3C Recommendation, W3C, 2004.

[320] Serge Romaric Mouafo Tembo, Sandrine Vaton, Jean-Luc Courant, Stephane Gosselin, and Michel Beuvelot. Model-Based Probabilistic Reasoning for Self-Diagnosis of Telecommunication Networks: Application to a GPON-FTTH Access Network. *Journal of Network and Systems Management*, 2017.

[321] Clifford A. Shaffer. *Data Structures & Algorithm Analysis in C++*. Dover Books on Mathematics. Dover Publications, $3^{rd}$ edition, 2011.

[322] Shahaboddin Shamshirband, Nor Anuar, Babak Daghighi, Miss Laiha Mat Kiah, Ahmed Patel, and Ajith Abraham. Co-FQL: Anomaly Detection Using Cooperative Fuzzy Q-learning in Network. *Journal of Intelligent and Fuzzy Systems*, 2014.

[323] Colin Shearer. The CRISP-DM Model: The New Blueprint for Data Mining. *Journal of data warehousing*, 2000.

[324] Sherif Saad and Issa Traore. Ontology-Based Intelligent Network Forensics Investigation. In *$19^{th}$ International Conference on Software Engineering and Data Engineering (SEDE 2010)*, San Francisco, California, USA, 2010.

[325] Shilin He, Pinjia He, Zhuangbin Chen, Tianyi Yang, Yuxin Su, and Michael R. Lyu. A Survey on Automated Log Analysis for Reliability Engineering. *ACM Computing Surveys*, 2021.

[326] Siddharth Bhatia, Bryan Hooi, Minji Yoon, Kijung Shin, and Christos Faloutsos. MIDAS: Microcluster-Based Detector of Anomalies in Edge Streams. In *Proceedings of the AAAI Conference on Artificial Intelligence*, New York, NY, USA, 2020. AAAI Press.

[327] Olivier Sigaud and Olivier Buffet. *Processus décisionnels de Markov en intelligence artificielle*, volume 1 - principes généraux et applications of *IC2 - informatique et systèmes d'information*. Lavoisier – Hermes Science Publications, 2008.

[328] Sihem Cherrared. *Fault Management of Programmable Multi-Tenant Networks*. PhD thesis, Université Rennes 1, 2020.

[329] Sihem Cherrared, Sofiane Imadali, Eric Fabre, and Gregor Gössler. SAKURA a Model Based Root Cause Analysis Framework for vIMS (Poster). In *Proceedings of the $17^{th}$ Annual International Conference on Mobile Systems, Applications, and Services*, Seoul Republic of Korea, 2019. ACM.

[330] Silvio Peroni. Graffoo: Graphical Framework for OWL Ontologies. https://essepuntato.it/graffoo/, 2013.

[331] Simon Cox and Chris Little. Time Ontology in OWL. Candidate Recommendation OGC 16-071r3, W3C, 2020.

[332] Simon Gottschalk and Elena Demidova. EventKG: A Multilingual Event-Centric Temporal Knowledge Graph. In *The Semantic Web*. Springer International Publishing, 2018.

[333] Joseph Sinyor. The Problem as a String Rewriting System. *International Journal of Mathematics and Mathematical Sciences*, 2010.

[334] Stefan Kempter. IT Process Maps – Incident Management. https://wiki.en.it-processmaps.com/index.php/Incident_Management, 2007.

[335] Stefan Kempter. IT Process Maps – Problem Management. https://wiki.en.it-processmaps.com/index.php/Problem_Management, 2007.

[336] Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez, and Mariano Fernández-López. The NeOn Methodology for Ontology Engineering. In *Ontology Engineering in a Networked World*. Springer, 2012.

[337] Osma Suominen, Henri Ylikotila, Sini Pessala, Mikko Lappalainen, Matias Frosterus, Jouni Tuominen, Thomas Baker, Caterina Caracciolo, and Armin Retterath. Publishing SKOS Vocabularies with Skosmos. http://www.skosmos.org/publishing-skos-vocabularies-with-skosmos.pdf, 2015.

[338] Lionel Tailhardat, Yoan Chabot, Antoine Py, and Perrine Guillemette. NORIA UI: Efficient Incident Management on Large-Scale ICT Systems Represented as Knowledge Graphs. In *$19^{th}$ International Conference on Availability, Reliability and Security (ARES)*, 2024.

[339] Lionel Tailhardat, Yoan Chabot, and Raphaël Troncy. NORIA-O: an Ontology for Anomaly Detection and Incident Management in ICT Systems. In *Semantic Web – 21$^{st}$ International Conference, ESWC 2024, Hersonissos, Crete, Greece, May 26 – 30, 2024, Proceedings*, 2024.

[340] Lionel Tailhardat, Benjamin Stach, Yoan Chabot, and Raphaël Troncy. Graphameleon: Relational Learning and Anomaly Detection on Web Navigation Traces Captured as Knowledge Graphs. In *The Web Conf, May 13 – 17, 2024, Singapore*, 2024.

[341] Lionel Tailhardat, Benjamin Stach, Yoan Chabot, and Raphaël Troncy. Graphaméléon : apprentissage des relations et détection d'anomalies sur les traces de navigation Web capturées sous forme de graphes de connaissances. In *Plate-Forme Intelligence Artificielle (PFIA), IC track, July 01 – 05, 2024, La Rochelle, France*, 2024.

[342] Tatsuaki Kimura, Kei Takeshita, Tsuyoshi Toyono, Masahiro Yokota, Ken Nishimatsu, and Tatsuya Mori. Network Failure Detection and Diagnosis by Analyzing Syslog and SNS Data: Applying Big Data Analysis to Network Operations. *NTT Technical Review*, 2013.

[343] Maltego Technologies. Maltego. https://www.maltego.com/.

[344] TensorFlow. Introduction aux auto-encodeurs. https://www.tensorflow.org/tutorials/generative/autoencoder, 2020.

[345] Thanos G. Stavropoulos, Dimitris Vrakas, Danai Vlachava, and Nick Bassiliades. BOnSAI: A Smart Building Ontology for Ambient Intelligence. In *Proceedings of the 2$^{nd}$ International Conference on Web Intelligence, Mining and Semantics*. Association for Computing Machinery, 2012.

[346] The MITRE Corporation. Cyber Security Measurement & Management Architecture. https://measurablesecurity.mitre.org/docs/Cyber_Security_Measurement_and_Management_Poster.pdf, 2013.

[347] The Zeek Project. The Zeek Network Security Monitor. https://zeek.org/. Formerly Bro IDS.

[348] Thematix Partners LLC. Visual Ontology Modeler. https://thematix.com/tools/vom/, 2013.

[349] Thibault Ehrhart, Pasquale Lisena, and Raphaël Troncy. KG Explorer: A Customisable Exploration Tool for Knowledge Graphs. In *6$^{th}$ International Workshop on the Visualization and Interaction for Ontologies and Linked Data, co-located with the 20$^{th}$ International Semantic Web Conference (ISWC)*, 2021.

[350] thinkwithgoogle.com. Find Out How You Stack Up to New Industry Benchmarks for Mobile Page Speed. https://think.storage.googleapis.com/docs/mobile-page-speed-new-industry-benchmarks.pdf, 2017. Accessed: 2023-08-10.

[351] Tianxing Wu, Arijit Khan, Melvin Yong, Guilin Qi, and Meng Wang. Efficiently Embedding Dynamic Knowledge Graphs. *Knowledge-Based Systems*, 2022.

[352] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web – A New Form of Web Content That Is Meaningful to Computers Will Unleash a Revolution of New Possibilities. *Scientific American*, 2001.

[353] Tim Bohne, Anne-Kathrin Patricia Windler, and Martin Atzmueller. A Neuro-Symbolic Approach for Anomaly Detection and Complex Fault Diagnosis Exemplified in the Automotive Domain. In *Proceedings of the 12$^{th}$ Knowledge Capture Conference 2023 (K-CAP '23)*, New York, NY, USA, 2023. Association for Computing Machinery.

[354] Timothy Lebo, Satya Sahoo, and Deborah McGuinness. PROV-O: The PROV Ontology. W3C Recommendation, W3C, 2013.

[355] TM Forum. TM Forum Open APIs. https://github.com/tmforum-apis, 2018.

[356] Tomas Fredberg, Gen Xu, Zhaojuan Bian, Illia Cremer, and Mike Riess. Intel CoFluent Technology Optimizes Ericsson Cloud Solutions. https://www.intel.com/content/www/us/en/cofluent/cofluent-ericsson-big-data-paper.html, 2017.

[357] Tony Ohmann, Michael Herzberg, Sebastian Fiss, Armand Halbert, Marc Palyart, Ivan Beschastnikh, and Yuriy Brun. Behavioral Resource-Aware Model Inference. In *Proceedings of the 29$^{th}$ ACM/IEEE International Conference on Automated Software Engineering – ASE '14*, Vasteras, Sweden, 2014. ACM Press.

[358] Masato Uchida. Recent Trends and Some Lessons for Serious Network Failures in Japan. In *International Conference on Intelligent Networking and Collaborative Systems (INCOS)*, 2015.

[359] Ulrike von Luxburg. A Tutorial on Spectral Clustering. *Statistics and Computing*, 2007.

[360] Yuval Filmus UTF-8. Time Complexity of Sort-Merge Join. https://cs.stackexchange.com/questions/68113/time-complexity-of-sort-merge-join, 2016.

[361] Emile van Krieken, Thiviyan Thanapalasingam, Jakub M. Tomczak, Frank van Harmelen, and Annette ten Teije. A-NeSI: A Scalable Approximate Method for Probabilistic Neurosymbolic Inference, 2023.

[362] Vasilios Siris, Torsten Braun, Francisco Barcelo-Arroyo, Dirk Staehle, Giovanni Giambene, and Yevgeni Koucheryavy, editors. *Traffic and QoS Management in Wireless*

*Multimedia Networks: COST 290 Final Report*. Lecture Notes in Electrical Engineering. Springer US, 2009.

[363] Vera Zaychik Moffitt and Julia Stoyanovich. Temporal Graph Algebra. In *Proceedings of The 16$^{th}$ International Symposium on Database Programming Languages*, DBPL '17. Association for Computing Machinery, 2017.

[364] Vern Paxson. Bro: A System for Detecting Network Intruders in Real-Time. *Computer Networks*, 1999.

[365] VirusTotal. YARA. https://github.com/VirusTotal/yara, 2021.

[366] W3C. Level 1 Document Object Mode Specificationl. Working draft, W3C, 1998.

[367] W3C. Fetch Metadata Request Headers. Working draft, W3C, 2021.

[368] W3C SPARQL Working Group. SPARQL Protocol and RDF Query Language 1.1 (SPARQL). W3C Recommendation, W3C, 2013.

[369] Peng Wang, Garry Robins, Philippa Pattison, and Emmanuel Lazega. Exponential Random Graph Models for Multilevel Networks. *Social Networks*, 2013.

[370] Xiaobo Wang, Junde Song, and Xianwei Zhou. Hypergraph Based Network Model and Architechture for Deep Space Exploration. In *Information Computing and Applications*, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[371] Wassim Sellil Atoui. *Toward Auto-configuration in Software Networks*. PhD thesis, Institut Polytechnique de Paris, 2020.

[372] WeBank FinTech. AIOps Series V : RCA Based on Knowledge Graph. https://fintech-12982.medium.com/aiops-series-v-|-rca-based-on-knowledge-graph-b18a56aef515, 2020.

[373] Yanbin Wei, Qiushi Huang, Yu Zhang, and James Kwok. KICGPT: Large Language Model with Knowledge in Context for Knowledge Graph Completion. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, Singapore, 2023. Association for Computational Linguistics.

[374] Wil M. P. van der Aalst, Paulo Barthelmess, Clarence A. Ellis, and Jacques Wainer. Proclets: A Framework for Lightweight Interacting Workflow Processes. *Int. J. Cooperative Inf. Syst.*, 2001.

[375] William Eberle and Lawrence Holder. Anomaly Detection in Data Represented as Graphs. *Intelligent Data Analysis*, 2007.

[376] William R. Hogan. Geographical Entity Ontology. https://github.com/ufbmi/geographical-entity-ontology/wiki, 2016.

[377] Xiangnan Ren, Olivier Curé, Li Ke, Jeremy Lhez, Badre Belabbess, Tendry Randriamalala, Yufan Zheng, and Gabriel Kepeklian. Strider: An Adaptive, Inference-Enabled Distributed RDF Stream Processing Engine. *Proceedings of the VLDB Endowment*, 2017.

[378] Xin Luna Dong. Knowledge Graph and Machine Learning: A Natural Synergy. http://lunadong.com/talks/KG_for_ML.pptx, 2020.

[379] Yan Li, Tingjian Ge, and Cindy Chen. Data Stream Event Prediction Based on Timing Knowledge and State Transitions. *VDLB Endowment*, 2020.

[380] Yassine Naghmouchi, Nancy Perrot, Nizar Kheir, Ali Ridha Mahjoub, and Jean-Philippe Wary. A New Risk Assessment Framework Using Graph Theory for Complex ICT Systems. In *Proceedings of the $8^{th}$ ACM CCS International Workshop on Managing Insider Security Threats*, 2016.

[381] Yechiel Levin, Batami Gold, Dan Mabee, Atik Mapari, Kent Sharkey, Robert Lyon, and David Coulter. Advanced Threat Detection with User and Entity Behavior Analytics (UEBA) in Microsoft Sentinel. https://learn.microsoft.com/en-us/azure/sentinel/identify-threats-with-entity-behavior-analytics, 2024.

[382] Yoan Chabot. *Construction, Enrichment and Semantic Analysis of Timelines : Application to Digital Forensics*. PhD thesis, University of Burgundy, 2015.

[383] Yoshua Bengio, Yann Lecun, and Geoffrey Hinton. Deep Learning for AI. *Communications of the ACM*, 2021.

[384] Youssra Rebboud, Pasquale Lisena, and Raphael Troncy. Beyond Causality: Representing Event Relations in Knowledge Graphs. In *Knowledge Engineering and Knowledge Management*. Springer International, 2022.

[385] Yuan Ren, Artemis Parvizi, Chris Mellish, Jeff Z. Pan, Kees van Deemter, and Robert Stevens. Towards Competency Question-Driven Ontology Authoring. In *$11^{th}$ European Semantic Web Conference (ESWC)*, 2014.

[386] Yves Raimond and Samer Abdallah. The Event Ontology. http://motools.sourceforge.net/event/event.html, 2007.

[387] Zach Kurtz and Samuel J. Perl. Measuring Similarity between Cyber Security Incident Reports. In *Forum of Incident Response and Security Teams (FIRST Conference)*, 2017.

[388] Zareen Syed, Ankur Padia, M. Lisa Mathews, Tim Finin, and Anupam Joshi. UCO: A Unified Cybersecurity Ontology. In *AAAI Workshop on Artificial Intelligence for Cyber Security*. AAAI Press, 2016.

[389] Zenoss. ZenOSS Logical Model. http://wiki.zenoss.org/ZenOSS_Logical_Model, 2014.

[390] Zhuo Chen, Wen Zhang, Yufeng Huang, Mingyang Chen, Yuxia Geng, Hongtao Yu, Zhen Bi, Yichi Zhang, Zhen Yao, Wenting Song, Xinliang Wu, Yi Yang, Mingyi Chen, Zhaoyang Lian, Yingying Li, Lei Cheng, and Huajun Chen. Tele-Knowledge Pre-training for Fault Analysis. In *$39^{th}$ International Conference on Data Engineering (ICDE)*, 2023.

[391] Zhuoran Yin. *Graph Locality Prefetcher for Graph Database*. PhD thesis, University of Waterloo, Waterloo, Ontario, Canada, 2015.

[392] Tanja Zseby, Benoît Claise, Juergen Quittek, and Sebastian Zander. Requirements for IP Flow Information Export (IPFIX). RFC 3917, 2004.