Model graph generation for naval cyber-physical systems

Nicolas Pelissero, Pedro Laso, John Puentes

Model graph generation for naval cyber-physical systems

Nicolas Pelissero
Chair of Naval Cyber Defense
Ecole navale
Brest, France
nicolas.pelissero@ecole-navale.fr

Pedro Merino Laso
Naval Academy Research Institute
French Maritime Academy (ENSM)
Nantes, France
pedro.merino-laso@supmaritime.fr

John Puentes
IMT Atlantique
Lab-STICC, UMR CNRS 6285
Brest, France
john.puentes@imt-atlantique.fr

***Abstract*—Naval vessels infrastructures are evolving towards increasingly connected and automatic systems. Such accelerated complexity boost to search for more adapted and useful navigation devices may be at odds with cybersecurity, making necessary to develop adapted analysis solutions for experts. This paper introduces a novel process to visualize and analyze naval Cyber-Physical Systems (CPS) using oriented graphs, considering operational constraints, to represent physical and functional connections between multiple components of CPS. Rapid prototyping of interconnected components is implemented in a semi-automatic manner by defining the CPS's digital and physical systems as nodes, along with system variables as edges, to form three layers of an oriented graph, using the open-source Neo4j software suit. The generated multi-layer graph can be used to support cybersecurity analysis, like attacks simulation, anomaly detection and propagation estimation, applying existing or new algorithms.**

***Index Terms*—Multi-layer graph generation, naval systems dependencies, cybersecurity, rapid prototyping, modeling.**

I. Introduction

Naval cyber-physical systems (CPS) are heterogeneous and highly interconnected components of vessels. Categorized as critic systems because of the multiple and costly consequences in case of dysfunction, naval CPS must be monitored and analyzed permanently to identify potential operational and cybersecurity risks. While specialized supervision instruments are commonly used for operational follow-up, cybersecurity risks are assessed by means of audits, vulnerability tests, and attack feasibility studies, among others. Given the complexity of implementing risk tests on already installed infrastructure, computational representation appears as an intermediate solution to simulate and evaluate the impact of anomalies and attacks. This requires nevertheless a convenient support to replicate the corresponding CPS interconnectivity and functionality. Graphs, mathematical abstractions, illustrated as structures that show relations between objects, helping to visualize, analyze, and understand complex problems, present the necessary characteristics to obtain the required computational representation.

Current trends in maritime technology, particularly digital twins [1] and autonomous ships [2], show that CPS monitoring and analysis will broaden regardless of crew reduction. Therefore, the development of computational representations like graphs, should follow the advances in the field to **model** complex CPS. Yet, despite the existence of graph representation software, adapted processes and tools to model **naval CPS** in a simple and flexible manner are not currently available. For instance, it is either very arduous or not possible to associate diverse user defined features to modeled nodes and links generated by an existing tool. Current graph generation applications are mainly domain-specific, e.g. content analysis for social networks, lacking of features that could enable adaptation to other domains. Moreover, system modeling is essential to test and evaluate the behavior of prototyped interdependent components under highly restrictive mission constraints [3], applying **simulation**, on which various aspects as performance, stability, anomaly detection, and cyber security are examined. Those components or subsystems cover all the aspects of ship operations, being part of critical infrastructure like platform management and navigation control in civilian ships, along with combat management in military ships.

Our main motivation to develop the proposed procedure is related to previous works focused on anomaly propagation analysis [4], [5] and the study of dependencies between maritime systems [6], which require such essential computational representation support. We propose in this paper a suitable **directed graph model generation process**, to make representations of naval CPS according to operational conditions, derived from use case scenarios.

The rest of the paper is organized as follows. In Section II, some relevant related works are reviewed. Section III describes the conceptualized graph model. Section IV illustrates the implementation to generate the graph model. Section V presents how this generated model can be applied to a naval supply water management system. Our preliminary results are discussed in Section VI.

II. Related Works

Graphs have been used in the maritime domain with varied objectives. This section summarizes some works on maritime applications, along with a synopsis about existing graph tools. Graphs were used to represent the paths followed by maritime merchandises, permitting to identify the connections between ports and study their vulnerabilities in the supply chain [7]. Maritime architectures were depicted with multi-layers graphs [8], representing data and information exchanges between the

main abstract systems, although without functional dependencies, which are very important.

Other works have examined maritime CPS cybersecurity aspects, like attack path analysis using graphs to estimate and quantify the studied paths [9]. Path attack analysis based on graphs is a challenging issue that depends on the calculation of pertinent edge's weights. To this end, the navigation system of a generic ship composed of 13 nodes was analyzed taking into account the criticality of system components and their dependencies [10]. Graphs were applied to identify vulnerabilities and understand potential cascading failures of water management systems in vessels [11], which are crucial for navigation. In a previous work [5], we proposed a graph-based analysis approach to study anomaly propagation in a water management system, calculating the edge's weights using a risk assessment approach to evaluate each specific subsystem.

Multiple software tools exist to create and manage graphs e.g., Gephi, Neo4j, GraphBuilder, Titan, GraphX, and Giraph. Among these, Gephi and Neo4j seem to be closely related to our objective, although are optimized for different needs. Developed to analyze data from a graph, Gephi [12] allows users to interact easily with the graph and associated data, to modify it or create new graphs from calculated results. The advantage of this tool consists on its possibilities to visualize a graph and its evolution in time. Contrarily, Neo4j has been conceived to manage data bases with a graph [13]. Differing from traditional data bases that organize data in relational tables formed by columns and rows, this tool uses a flexible structure that is defined by relations between the recorded data. Neo4j is an open-source software suite with libraries that include a wide variety of functionalities, from data integration to visualization, including analysis algorithms. In addition, numerous functions allow integrating created graphs to different applications. Both tools propose known graph processing algorithms. We have chosen Neo4j given the flexibility of its libraries and analysis tools, as well as it's strongly active community and diversified associated documentation. Above all, Neo4j provides a wide range of developer resources to support the development of customized applications. Also, Neo4j, Inc. proposes workshops and conferences regularly to highlight uses of their tool in different research areas.

III. Graph Model

Vessels' systems and subsystems can be represented as the nodes of a graph, while their connections, interactions, or dependencies are the edges. Accordingly, those connections must have a predefined direction to complete the definition of a directed graph. In our case, concerned CPS subsystems include pumps, sensors, and actuators that are physical devices, and controllers, Programmable Logic Controller (PLC), input/output modules, on-board workstations, and supervisory control and data acquisition systems (SCADA), which are part of various digital systems and networks. It is thus necessary to differentiate three graph components, namely, digital, physical, and system variables, which interact in complex manners and are assigned to different layers of the model.

A. Graph's layers

The graph is composed by three layers where each subsystem and associated variables are represented. The three layers of the graph model are defined as follows:

- Digital subsystem layer: Composed by subsystems that integrate networking and/or computing capacities, i.e. SCADA, PLC, or on-board workstation.

- Physical subsystem layer: Consisting of subsystems with physical processes interaction, data transmission of measures, or control commands reception, i.e. a pump activated to fill a tank, or a temperature sensor to control an engine heating.

- Subsystem variables layer: Intended to represent the CPS current state by means of measured values and control variables, associated to the corresponding digital or physical subsystems of the two previous layers, e.g. the pump activation variable, the tank level, or the engine temperature. This layer is crucial because it defines system variables dependencies relying on the structural and operational conditions of a use case scenario.

Each subsystem is placed in the digital or physical layer based on its primary role in the system. Afterwards, associated nodes that represent the respective internal variables are created. For example, a thermometer is represented by a node in the physical layer and the internal variable with the measured temperature is represented by a node in the variables layer.

B. Graph's edges

Every node of each layer has one or more dependencies to other nodes. We distinguish a non-exhaustive list of dependency types based on CPS inherent properties. For example, control commands dependency between a PLC and an actuator, or a sensor measurement dependency between a sensor and a PLC. Other types of dependencies may be modelled like digital or physical dependency, and functional dependency. The key feature of the proposed process is to generate a three layers graph model using a script that defines in a structured, intuitive, and simple manner, each layer's components, together with the control commands, sensor measurements, and additional directional dependencies, all defined by the user for each node as needed.

IV. Graph generation process

The main principle of the proposed rapid graph prototyping process is to specify in a simple manner the directed graph using a text description input file (Fig. 1), read by a Neo4j script that generates the graph. The resulting graph is created with the three pre-defined layers and dependencies presented in Section III. Analysis algorithms can then be applied to this computational representation of a graph. Basic details about these steps are described in the next subsections.

A. Description file

The input file includes all required information, i.e. every subsystem and variables that compose the represented system, categorized in one of the three layers, combined with all the necessary dependencies between subsystems to define the

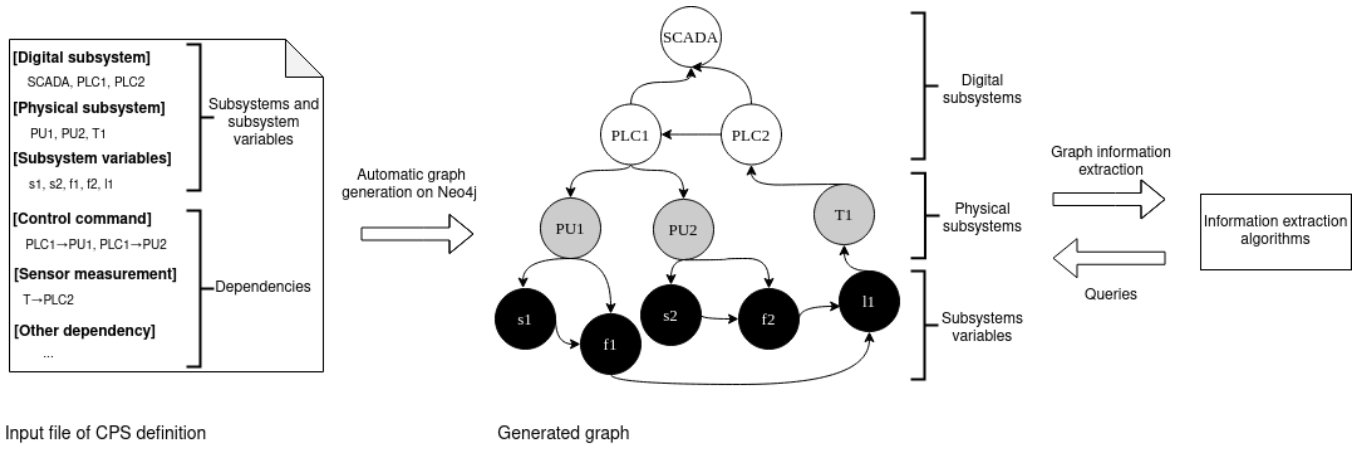

Fig. 1. Schematic description of the directed graph generation and algorithm interactions for a water supply management system.

edges. The file is formed by a header and four sections. In the file header, a system *name* is assigned first, followed by the *type* of ship's functional block identification, the system is part of. Then, each line of the *digital-systems* section, names according to the respective identification *id* code and *name* acronym, the corresponding subsystems. Equivalently, in the *physical-systems* section, each line designates the identification *id* code and *name* acronym of the concerned subsystems. Studied subsystem *variables* are listed next, providing their *id* code and *name* acronym. Finally, in the *dependencies* section each line defines the subsystems and variables connections using the previously defined *id* codes and a pre-defined identification value. If required, the file could be modified to be compatible with another format, for instance, JSON, XML, or YAML, used by the extraction and graph generation code.

B. Graph generation

Once the file has been created, it is read by a Neo4j script that extracts the information and generates the graph, making use of the required functions. In our case, graph generation and information extraction are characterized by interactions between the processing application and Neo4j. Those interactions are based on the use of a specific function of Neo4j's library, which allows sending a cypher (Neo4j's graph query language) query as text parameter from the processing application to store and extract information from the graph database. Each operation performed on the graph from the processing application, is defined by the same function structure, modifying the text parameter according to the seeked result. Some examples of command parameters are:

- Create a node by specifying a name parameter, e.g. *PLC1*, and a label type from one of the three defined graph layers.

- Obtain the *nearest neighbor* of a given node by using its name parameter.

- Create a *command control* relationship between one node of the digital layer and one node of the physical layer, associating an *exposure* feature to the created relationship.

- Use a Neo4j's embedded graph algorithm to obtain the *closeness centrality score* of each graph's nodes, and rank them by decreasing value.

C. Analysis

The generated structural and functional computational graph representation is therefore compatible with the numerous Neo4j's analysis tools [13], available as library functions. Specialized toolboxes include developed platform and user defined functions applicable to extract information from multi-layer network representations [14]. Seven categories of algorithms are available, specifically, path finding, centrality, similarity, link prediction, node embeddings, node classification, and community detection.

V. Use case: supply water management

As depicted in Fig. 2, maritime vessels use water management systems with different ends, for example supply water management. Another application can be the control of ballast devices. When studied in detail, these systems are equivalent to systems used in non-maritime infrastructure. Consequences of anomalies and attacks in both types of systems can have adverse large-scale effects [15].

To show how directed graphs can be modelled with the proposed process, a supply water management CPS infrastructure is defined step by step, according to the principles described in Section III. The main role of this system is to satisfy consumer demand ensuring the required service continuity. This is an essential on board system, because if the quality of distributed water is not optimal, it can constitute a transmission path for diseases, with dangerous consequences for the passengers and the crew [16]. As example of water management system, we have identified C-Town, a water network widely used in research works [17].

A. File description

C-town is a water distribution network that is composed by 388 nodes, 429 pipes, 7 tanks, 11 pumps, 4 valves (1

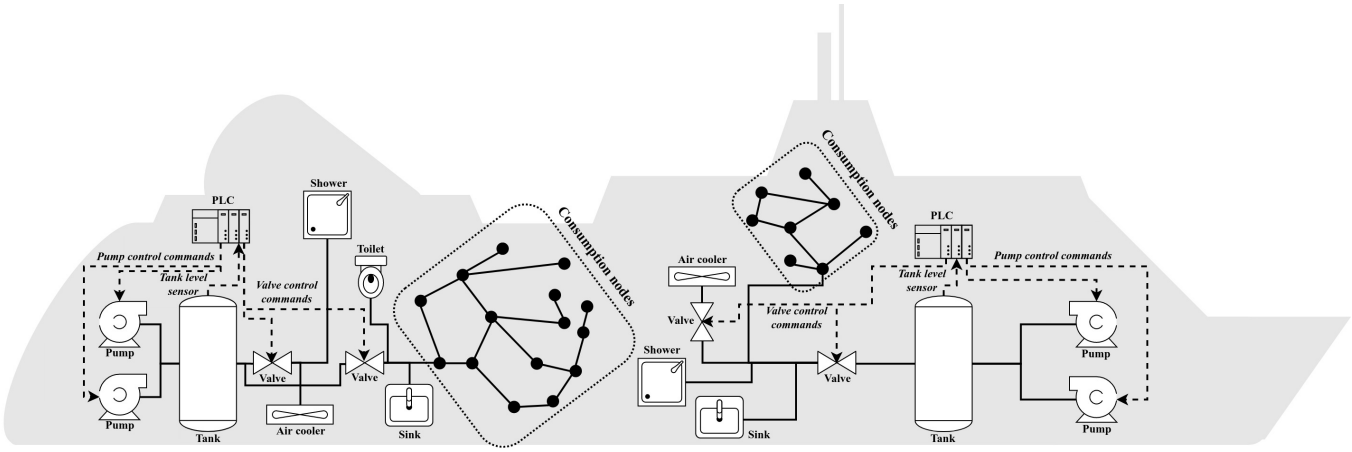

Fig. 2. Ship water management system

actionable), 9 PLC, and 1 SCADA. Water storage and distribution depend on the water levels of 7 tanks. The level of these tanks is controlled by 11 pumps, while 9 PLCs control or monitor each actuator, i.e. pumps and valves, and each sensor. A SCADA system collects the PLCs' readings and coordinates the process. Multiple processes are coded into the PLCs to control actuators, in accordance with sensor readings that can be provided by other PLCs.

In our use case, C-Town is described with a JSON format as presented below. This information is stored in a file that is the input of the defined process.

```
{
    "name": "water-system",
    "type": "Operational System",
    "digital-systems": {
        { "id": "d1", "name": "SCADA" },
        { "id": "d2", "name": "PLC1" },
        { "id": "d3", "name": "PLC2" }
    },
    "physical-systems": {
        { "id": "p1", "name": "PU1" },
        { "id": "p2", "name": "PU2" },
        { "id": "p3", "name": "T1" }
    },
    "variables": {
        { "id": "v1", "name": "s1" },
        ...
    },
    "dependencies" : {
        {"d2", "d1", 13},
        {"d3", "d2", 15},
        ...
    }
}
```

B. Graph generation

Having defined the graph's nodes, i.e. digital systems (SCADA and PLCs), physical systems (pumps and level sensor), and associated variables, the graph's edges are dependencies between pairs of nodes, with an assigned identification number. In order to simplify the large resulting graph, only part of it is displayed schematically in Fig. 1.

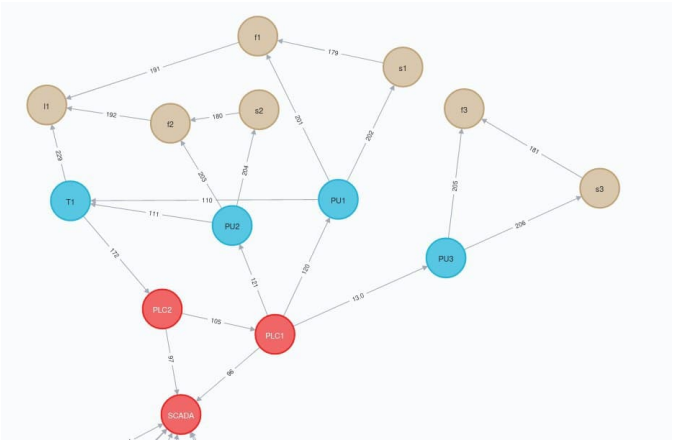

Fig. 3. Neo4j's graph representation of a part of C-town water management system.

The Neo4j's output (Fig. 3) differentiates the three graph layers with three colors: red for the digital layer, blue for the physical layer, and brown for the system variables. Note that the SCADA node, which concentrates all the information, is displayed on the bottom left side of the figure. Also pictured in the output, edges identification values are used internally by analysis applications to specify dependencies, associating particular features, like for instance weights that estimate cybersecurity risks.

C. Analysis possibilities

After the graph is generated and visualized, the resulting network can be analyzed with multiple algorithms. Out of

the large variety of Neo4j's classic graph analysis algorithms (Section IV-C) we have applied [18] the *closeness centrality* algorithm to detect nodes that are able to spread an anomaly very efficiently through the graph, and *path finding*, based on a depth-first search algorithm, to obtain the potential anomaly propagation paths from a start node, among others.

It is also possible to develop adapted analysis algorithms for particular applications using Neo4j. In our case, we have implemented the calculation of edges weights based on a preliminary risk analysis of dependencies. Additionally, a wide variety of drivers permits to integrate numerous external tools in a compatible manner, without disruption.

VI. Discussion

Being highly interconnected, naval CPS need to be studied with adapted tools and processes. One of the main interconnection characteristics is related to the possibility of using computational representations like a graph, to specify the set of systems, subsystems, and dependencies. We have defined a procedure that allows representing a naval CPS in the form of a three connected layers graph, illustrating how it works describing a use case.

Preliminary results indicate that further developments need to be done to improve the process implementation. Data manipulation may be a difficulty when the number of nodes and edges increases significantly, making necessary the use of a complementary visual interface to configure interactively the input file content. This interface could also include modules for the other steps of the procedure, i.e. graph generation and analysis. Finally, it is interesting to note that although an efficient JSON version of the input data format was implemented, currently there are not specific standardization recommendations about how this kind of graph representation should be defined computationally for CPS.

References

[1] Í. A. Fonseca and H. M. Gaspar, "Challenges when creating a cohesive digital twin ship: a data modelling perspective," *Ship Technology Research*, pp. 1–14, 2020.

[2] S. Thombre, Z. Zhao, H. Ramm-Schmidt, J. M. V. García, T. Malkamäki, S. Nikolskiy, T. Hammarberg, H. Nuortie, M. Z. H. Bhuiyan, S. Särkkä *et al.*, "Sensors and ai techniques for situational awareness in autonomous ships: A review," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[3] O. Jacq, X. Boudvin, D. Brosset, Y. Kermarrec, and J. Simonin, "Detecting and hunting cyberthreats in a maritime environment: Specification and experimentation of a maritime cybersecurity operations centre," in *2018 2nd Cyber Security in Networking Conference (CSNet)*, 2018, pp. 1–8.

[4] N. Pelissero, P. Merino Laso, and J. Puentes, "Naval cyber-physical anomaly propagation analysis based on a quality assessed graph," in *2020 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*. IEEE, 2020, pp. 1–8.

[5] N. Pelissero, P. Merino Laso, and J. Puentes, "Impact assessment of anomaly propagation in naval water distribution cyber-physical system (accepted)," in *2021 IEEE International Conference on Cyber Security and Resilience (IEEE CSR)*. IEEE, 2021, pp. 1–8.

[6] N. Pelissero, P. Merino Laso, O. Jacq, and J. Puentes, "Towards modeling of naval systems interdependencies for cybersecurity," in *2021 IEEE Oceans conference*. IEEE, 2021, pp. 1–7, (accepted).

[7] H. Liu, Z. Tian, A. Huang, and Z. Yang, "Analysis of vulnerabilities in maritime supply chains," *Reliability Engineering & System Safety*, vol. 169, pp. 475–484, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0951832016301776

[8] O. Rodseth, M. J. Christensen, and K. Lee, "Design challenges and decisions for a new ship data network," *ISIS*, pp. 15–16, 2011.

[9] G. Kavallieratos and S. Katsikas, "Attack path analysis for cyber physical systems," in *Computer Security*, S. Katsikas, F. Cuppens, N. Cuppens, C. Lambrinoudakis, C. Kalloniatis, J. Mylopoulos, A. Antón, S. Gritzalis, W. Meng, and S. Furnell, Eds. Cham: Springer International Publishing, 2020, pp. 19–33.

[10] A. Akbarzadeh and S. Katsikas, "Identifying critical components in large scale cyber physical systems," in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, ser. ICSEW'20. New York, NY, USA: Association for Computing Machinery, 2020, p. 230–236. [Online]. Available: https://doi.org/10.1145/3387940.3391473

[11] C. J. Goodrum, C. P. Shields, and D. J. Singer, "Understanding cascading failures through a vulnerability analysis of interdependent ship-centric distributed systems using networks," *Ocean Engineering*, vol. 150, pp. 36–47, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0029801817307679

[12] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: an open source software for exploring and manipulating networks," in *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 3, no. 1, 2009.

[13] J. J. Miller, "Graph database applications and concepts with neo4j," in *Proceedings of the Southern Association for Information Systems Conference, Atlanta, GA, USA*, vol. 2324, no. 36, 2013.

[14] S. Boccaletti, G. Bianconi, R. Criado, C. I. Del Genio, J. Gómez-Gardenes, M. Romance, I. Sendina-Nadal, Z. Wang, and M. Zanin, "The structure and dynamics of multilayer networks," *Physics Reports*, vol. 544, no. 1, pp. 1–122, 2014.

[15] S. Lee, "Managing water quality on board passenger vessels to ensure passenger and crew safety," *Perspectives in Public Health*, vol. 139, no. 2, pp. 70–74, mar 2019.

[16] D. B. Jernigan, J. Hofmann, M. S. Cetron, J. Nuorti, B. Fields, R. Benson, R. Breiman, H. Lipman, R. Carter, C. Genese *et al.*, "Outbreak of legionnaires' disease among cruise ship passengers exposed to a contaminated whirlpool spa," *The Lancet*, vol. 347, no. 9000, pp. 494–499, 1996.

[17] R. Taormina, S. Galelli, N. O. Tippenhauer, E. Salomons, and A. Ostfeld, "Characterizing cyber-physical attacks on water distribution systems," *Journal of Water Resources Planning and Management*, vol. 143, no. 5, p. 04017009, 2017.

[18] A. Hodler and M. Needham, *Graph Algorithms: Practical Examples in Apache Spark and Neo4j*. O'Reilly Media, 2019.