Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 200 (2022) 699–709

Procedia Computer Science

www.elsevier.com/locate/procedia

3rd International Conference on Industry 4.0 and Smart Manufacturing

Assessment of the PLC Code generated with the GEMMA-GRAFCET Methodology

Alejandro Mejia[a], Andres Felipe Guarnizo[b], Giacomo Barbieri[a,*]

[a]*Department of Mechanical Engineering, Universidad de los Andes, Bogota D.C. (Colombia)*
[b]*Department of Mechatronics Engineering, Universidad Ean, Bogotá D.C. (Colombia)*

Abstract

Industry 4.0 has fostered the digital transformation of enterprises through the retrofitting or replacement of traditional manufacturing processes with Cyber Physical Systems (CPS). Even if interoperability is paramount to correctly operate CPS, it still remains a challenge due to lack of universal standards for the management of the operational modes of CPS through the PLC code. The GEMMA-GRAFCET Methodology (GG-Methodology) was introduced as a standard approach and vocabulary to face this issue. However, the PLC code obtained from its application has not been analyzed yet. Thus, the novelty of this research work consists in the assessment of the PLC code obtained from the application of the GG-Methodology using metrics resulting from static and dynamic analysis. In comparison to a benchmark methodology, the GG-Methodology results in PLC code more complex and with larger response time, but more maintainable and with similar execution time. Given the large number of methodologies for the development of PLC code presented in the literature, authors hope that the approach adopted within this work may be utilized for generating a quantitative and objective classification that may guide the control practitioner in the methodology selection process.

Keywords: Automation; PLC Code; GEMMA-GRAFCET; Software Metrics; Design Methodology.

1. Introduction

Industry 4.0 was officially announced in 2013 as a German strategic initiative to take a pioneering role in industries which are currently revolutionizing the manufacturing sector [14]. This initiative has fostered the digital transformation of enterprises through the retrofitting or replacement of traditional manufacturing processes with digitalized, interconnected and interoperated systems [17, 15]. Reconfigurable-, adaptive-, and evolving- factories constituted by Cyber Physical Systems have been developed to achieve the mass-customization and personalization required nowadays [9, 12].

* Corresponding author.
E-mail address: g.barbieri@uniandes.edu.co

1877-0509
10.1016/j.procs.2022.01.268

Cyber Physical Systems (CPS) are defined as physical and engineered systems whose operations are monitored, coordinated, controlled and integrated by a computing communication core generally hosted in the cyber space [19]; i.e. edge, fog or cloud. CPS exchange data: (i) horizontally: to coordinate the heterogeneous systems that carry out the manufacturing process's tasks; (ii) vertically: to achieve advanced data processing capabilities through the communication in between the physical and the cyber space. Therefore, interoperability is fundamental for the correct operation of CPS.

Interoperability is defined as the ability of systems to exchange data, and share information and knowledge [8]. Two components are necessary to achieve interoperability: (i) enabling technologies for the data exchange; (ii) standard syntax and semantics to correctly interpret the information content of the data. *Industrial Internet of Things* (IIoT) provides the enabling technologies to implement the horizontal and vertical communication needed to the realization of CPS [24]. However, interoperability still remains a challenge due to lack of universal standards [16].

In response to this issue, Barbieri and Gutierrez [3] proposed the *GEMMA-GRAFCET Methodology* (GG-Methodology): a standard approach and vocabulary for the management of the Operational Modes (OMs) of CPS through the automation software, thus facilitating the exchange of information between the PLCs (Programmable Logic Controllers) and the external environment. The methodology consists of: (i) GEMMA-GRAFCET Representation (GG-Representation): to completely specify the management of the OMs with a standard syntax and semantics; (ii) Hierarchical Design Pattern (HDP): to generate hierarchical, modular, readable and maintainable PLC code which consists in a one-to-one translation of the GG-Representation.

The GG-Methodology has been demonstrated to reach interoperability both within the horizontal and vertical integration. Concerning the *horizontal integration*, the methodology has been utilized for the implementation of a decentralized architecture of CPS supervised from a central coordinator [11]. Concerning the *vertical integration*, it has been applied for the communication in between a PLC (physical space) and a digital twin (cyber space) [3]. Furthermore, a Model-Driven Engineering online tool – referred to as *GG-Generator*[1] – has been implemented to specify and automatically generate PLC code compliant with the GG-Methodology.

In this work, the PLC code generated with the GG-Methodology is assessed using metrics that involve both the structural relations between the different objects of the code and the real-time behavior. Furthermore, these metrics are also computed for a methodology based on the Statechart diagram that is utilized as a benchmark for the evaluation of the GG-Methodology.

Given the above, the paper is structured as follows: Section 2 resumes the methodologies for PLC code generation that are compared within this work. The workflow and metrics utilized for the assessment are illustrated in Section 3, while Section 4 shows the case study implemented for the comparison of the two methodologies. Obtained results are discussed in Section 5 and finally, Section 6 presents the conclusions and sets the directions for future work.

2. Compared Methodologies

In this section, the compared methodologies are illustrated: GG-Methodology (Section 2.1) and Statechart-based Methodology (Section 2.2).

2.1. GEMMA-GRAFCET Methodology

GEMMA (Guide d'Etude des Modes de Marche et d'Arrêt) is a checklist which allows to graphically define all the start and stop modes of a system and their evolution [1]. However, GEMMA is partially specified with respect to the syntax and semantics causing misunderstandings and possible emergent behaviors. In the GG-Methodology, the GRAFCET international standard (IEC 60848 [13]) is utilized to semantically specify the GEMMA guideline and a design pattern is introduced to generate PLC Structured Text (ST) code which consists in a one-to-one translation of the GEMMA-GRAFCET specifications.

The GG-Methodology is depicted in Figure 1 and consists of [3]: (i) *GEMMA-GRAFCET Representation* (GG-Representation) for defining the OMs of CPS without the occurrence of emergent behaviors; (ii) *Hierarchical Design*

[1] https://giacomobarbieri1.github.io/MDE_GEMMA-GRAFCET/

Pattern (HDP) for the development of hierarchical PLC code from specifications expressed in accordance with the proposed GG-Representation. The HDP generates hierarchical PLC code for separating the management of the OMs (common for each CPS) from their nested behavior (specific to each CPS); see Figure 1.

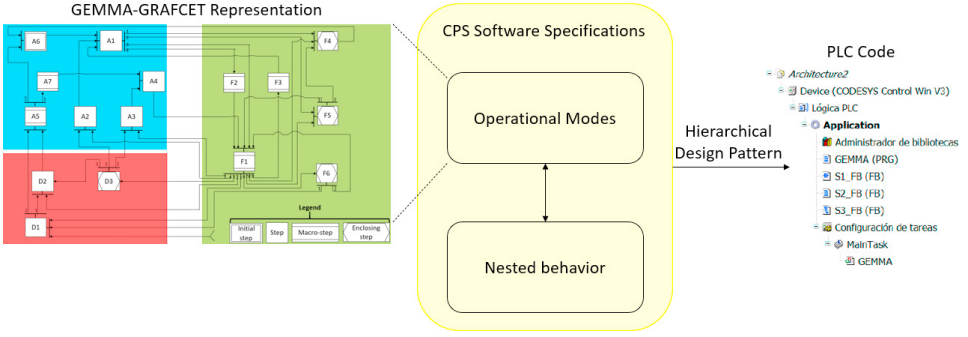

Fig. 1: GEMMA-GRAFCET Methodology: GG-Representation for the specification of the OMs of CPS, and HDP for generating PLC code in accordance with the software specifications.

2.2. *Statechart-based Methodology*

Statechart diagrams originated from the domain of reactive technical embedded systems and are utilized to model the behavior of systems that can be described by the principle of state machines or finite automatons [26]. In the *Statechart-based Methodology* (S-Methodology [4]), the software of CPS is specified without the use of OMs and is directly represented through Statechart diagrams. A planar representation is obtained and converted into PLC ST code as depicted in Figure 2.

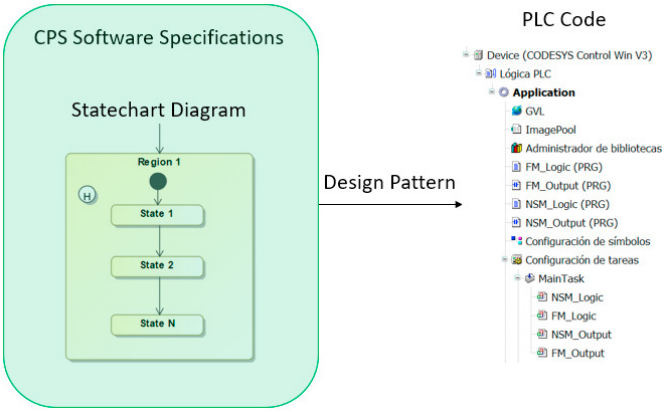

Fig. 2: Statechart-based Methodology: Statechart diagram for the specification of the software of CPS and design pattern for the generation of PLC code.

3. Research workflow

In this section, the utilized research workflow is illustrated. In particular, the protocol defined for the comparison of the two methodologies is shown in Section 3.1, while the selected comparison metrics are indicated in Section 3.2.

3.1. Comparison Protocol

The protocol defined for the comparison of the two methodologies consists of (Fig. 3):

- *Case Study*: a case study is selected with the objective to assess metrics that enable the comparison of the two methodologies. The case study must be complex enough to allow the comparison; i.e. with different OMs. The selection of the case study also implies the definition of the functional requirements that the PLC code must fulfill;
- *PLC Code*: once the case study has been selected, PLC code is written with the two methodologies. Before moving to the following step, both the PLC codes must be tested to verify the fulfilment of the functional requirements of the case study;
- *Software Metrics*: after the verification of the PLC code, metrics are assessed. In this work, metrics resulting from static and dynamic analysis are computed as indicated in Section 3.2;
- *Comparison*: after the assessment of the metrics, the two methodologies are compared, and pros and cons of the GG-Methodology are identified.

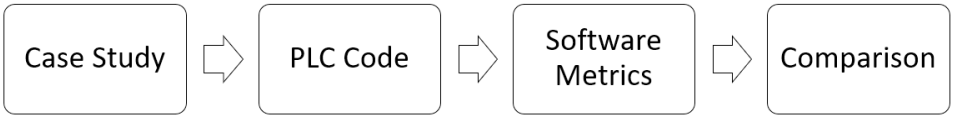

Fig. 3: Protocol utilized for the comparison of the two methodologies.

3.2. Software Metrics

In the PLC Code step of Figure 3, it is verified that the PLC code obtained with the two methodologies fulfills the functional requirements of the case study. Therefore, the comparison is performed using metrics related to non-functional requirements. In this work, metrics resulting from static and dynamic analysis are assessed and respectively illustrated in Section 3.2.1 and Section 3.2.2.

3.2.1. Static Analysis Metrics

Static analysis is a white-box testing that consists in the assessment of the source code in a non-runtime environment. This type of analysis looks at details such as the conformity with a coding standard or set of rules, the syntax of the code, and the code optimization amongst others; see [18, 6]. In other domains, such as embedded systems and computer science, static analysis tools are more common and used more intensively than the PLC one. Furthermore, their use in the PLC domain requires some adaptations. In response to this issue, Capitán and Vogel-Heuser [7] introduced McCabe, Halstead and Kafura metrics oriented to PLC code. Whereas, Zhabelova and Vyatkin [25] proposed a maintenance index for PLC code to quantify the facility to change a software POU (Program Organizational Unit).

Given the above, the metrics computed with static analysis and selected for the comparison are:

- *McCabe*: a Cyclomatic Complexity number (CC) is assessed through this metric. The CC analyzes the number of nodes (i.e. processing tasks), edges (i.e. flow between nodes) and connected components to measure the control flow of the code. The CC can be assessed for different granularity levels of the code; i.e. the whole PLC code or single POUs or parts of POUs. Given N_{edge} and N_{nodes} respectively the number of edges and nodes of the analyzed PLC code and d the number of decisions, CC is computed as:

$$CC = N_{edges} - N_{nodes} + 2d$$

Halstead Metric	Equation
Program Length	$N = N_1 + N_2$
Program Vocabulary	$n = n_1 + n_2$
Program Volume	$V = N \cdot log_2(n)$
Difficulty	$D = (n_1/2) \cdot (N_2/n_2)$
Program Effort	$E = D \cdot V$

Table 1: Operations necessary for the calculation of the Program Effort metric.

- *Halstead*: set of metrics calculated through the operands and operators within the considered POU. The main purpose of the Halstead metrics is 'to estimate the difficulty to understand or write a program and the effort to write the program' [10]. Within this work, the Program Effort metric is utilized for the assessment. However, its quantification implies the calculation of further Halstead metrics. Given n_1 the number of distinct operators, n_2 the number of distinct operands, N_1 the total number of operators and N_2 the total number of operands, the Program Effort metric can be computed as indicated in Table 1;
- *Kafura*: this metric is generally related with the code modularity since it is computed through the calls and data exchanged among POUs. It analyzes three factors [7]: (i) lack of functionality; (ii) lack of composability; (iii) lack of decomposability. Given Si the size of the program, Fan_{in} the number of variables into a POU plus the number of POUs which are called by the considered POU, Fan_{out} the number of variables exiting the POU plus the number of POUs which invoke the considered POU, Kafura Complexity (KC) is computed as:

$$KC = Si \cdot (Fan_{in} \cdot Fan_{out})^2$$

- *Maintenance Index*: maintainability is the degree to which a program is open to change [25]. Given V the Halstead's Program Volume, CC the Cyclomatic Complexity and LOC the Lines of Code, the Maintenance Index (MI) is computed as:

$$MI = 171 - 5.2 \cdot \ln V - 0.23 \cdot CC - 16.2 \cdot \ln LOC$$

3.2.2. Dynamic Analysis Metrics

Dynamic analysis is a black-box testing that consists in the assessment of the source code in a runtime environment. This analysis involves the monitoring of POUs with the goal of getting execution traces of selected software variables [23, 22]. Test-case scenarios are built to analyze different aspects of the verified PLC code [21]. In this work, two metrics are computed related to the real-time behavior of the PLC code:

- *Response time*: provides an indication of the code reactivity. It calculates the amount of time taken from the code to implement a defined test-case scenario;
- *Execution time*: CPU time taken to compute a defined POU.

4. Test-bench for Wick Soilless Cultivation

Next, the protocol indicated in Section 3 is applied to compare the two methodologies for the development of PLC code. Section 4.1 describes the selected case study, and Section 4.2 the PLC code generated with the two methodologies. Finally, in Section 4.3 static and dynamic analysis are implemented for the calculation of the metrics utilized in Section 5 for the comparison.

4.1. Case Study

The case study selected for the comparison is the Uniandes test-bench for wick soilless cultivation [2]. This system has the objective to test two lines of plants with different fertigation strategies and consists of two modules (Fig-

ure 4): (i) *Nutrient Solution Module*: responsible for preparing nutrient solutions with an established value of pH and electrical conductivity; (ii) *Fertigation Module*: responsible for the implementation of the fertigation strategy and can independently manage two samples ('lines') of plants. The number of sensors and actuators, and the required flexibility make the test-bench similar as features to industrial plants but still at a laboratory scale. For this reason, this test-bench has been selected as case study for the comparison of the two methodologies.

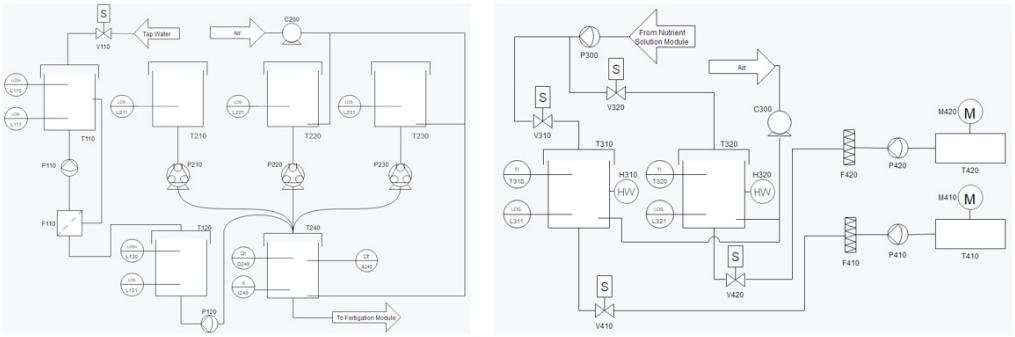

Fig. 4: P&ID diagram of the test-bench for wick soilless cultivation. The Nutrient Solution Module is represented on the left-hand side and the Fertigation Module on the right-hand side.

4.2. PLC Code

The CoDeSys programming environment[2] was selected for the PLC code writing. With the objective to compare the two methodologies, the same *software architecture* was defined. This consists of a unique task with three programs: (i) Line 1: for the control of the first line of plants; (ii) Line 2: for the control of the second line of plants; (iii) Output write: for the activation and deactivation of the system actuators. After that, the PLC code of the case study was written with the two methodologies.

As indicated in Section 2, the *GG-Methodology* consists in a GG-Representation and a HDP for the PLC code generation. The GG-Representation of the Line 1 program is depicted in the left-hand side of Figure 5 and was built using the GG-Generator. The GG-Representation of the case study was obtained through the instantiation of the default GEMMA representation presented in the GG-Generator. This was possible through the GUI of the GG-Generator, by changing the typology of the states, hiding unnecessary states and transitions, declaring variables, and specifying conditions of the transitions. Then, PLC code was automatically generated following the HDP.

Then, the *S-Methodology* was utilized for the generation of the PLC code of the test-bench for wick soilless cultivation. The Statechart diagram of the Line 1 program is depicted in the right-hand side of Figure 5. After the development of the Statecharts relatives to the Line 1 and Line 2 programs, the control software was converted into ST PLC code using the design pattern presented in [4].

Before moving to the assessment of the metrics, the PLC code fulfillment of the functional requirements relative to the selected case study was verified. A *Software-in-the-Loop* Virtual Commissioning (VC) simulation was implemented for this purpose. CoDeSys offers an integrated Soft-PLC, named CoDeSys Win Control V3, that emulates an industrial controller under Windows enabling the virtual execution of the control software. Therefore, the VC simulation was performed by interfacing a Simulink model of the test-bench and CoDeSys Win Control V3 using the OPC protocol (Open Platform Communications); see [4] for further details.

[2] CoDeSys PLC: https://www.codesys.com/

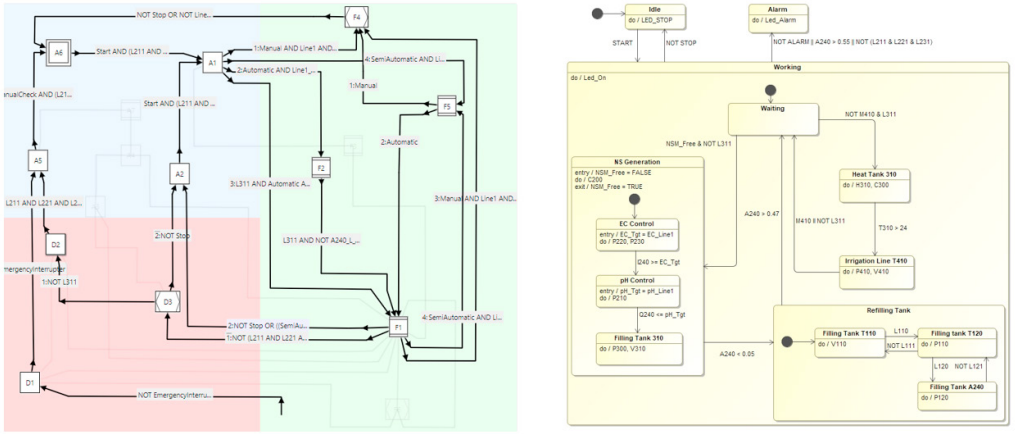

Fig. 5: Control software of the Line 1 program: GG-Representation on the left-hand side and Statechart diagram on the right-hand side.

4.3. Software Metrics

Considering that the CoDeSys programming environment was adopted to write the PLC code, different proprietary tools were utilized for the assessment of the metrics calculated with static and dynamic analysis.

Static Analysis Metrics. CoDeSys Static Analysis is a tool for checking the source code on the basis of defined rules. It was utilized for the calculation of the Cyclomatic Complexity number and the Halstead metrics. Whereas, the Kafura Complexity and the Maintenance Index were computed manually since CoDeSys Static Analysis does not assess them.

Dynamic Analysis Metrics. CoDeSys Test Manager enables the programming and execution of automated test-cases of applications and libraries developed within the CoDeSys programming environment. In dynamic software testing [20], the different actions that constitute a test case are verified by evaluating the values acquired by selected PLC software variables. Traditionally, the monitored variables change due to the forcing of variables that emulate physical sensors. In this case study, the VC simulation was utilized to enable the generation of a more realistic test environment in which the monitored variables changed because of a mathematical model; see [4]. The following test-case scenario was built to assess the *Response Time* of the two methodologies: the system starts working and after the generation of the nutrient solution, a fertigation of one minute per line is implemented. The time taken for completing the test-case scenario was defined as the response time metric.

Concerning the assessment of the execution time, the PLC code was enhanced with instructions that compute the actual time. By subtracting the actual time of different parts of the POUs, the execution time was calculated. In this work, the longest execution time was of interest since may define a lower bound that the task cycle should not pass to avoid jitter. Therefore, the *Worst Case Execution Time* (WCET) was computed.

Considering the importance of precisely computing the time within the assessment of Dynamic Analysis Metrics, the PLC code was run in a *real-time controller*; i.e. a Raspberry Pi 4 with 8GB of RAM. With respect to the use of the CoDeSys Soft-PLC within a Personal Computer, the utilization of a real-time controller enabled to accurately compute the Dynamic Analysis Metrics.

5. Results and Discussion

In this section, the assessed metrics are presented, along with the comparison of the two methodologies. The metrics resulting from the *static analysis* are depicted in Figure 6 through a radar chart. These metrics were computed considering the sum of the metrics of all the POUs of the PLC code obtained with each methodology. To generate a plot with the same direction of fulfillment, the inverse of the Program Effort, Cyclomatic Complexity and Kafura

Complexity was computed. Therefore, the higher are the numbers within the plot, the better are the metrics for the considered methodology. It can be noticed that the GG-Methodology is better for the Maintenance Index, while the S-Methodology for the Program Effort, Cyclomatic Complexity and Kafura Complexity.

Concerning the Program Effort and the Cyclomatic Complexity, the lower are these numbers, and the shorter and the simpler is the PLC code. On the one hand, the GG-Methodology implements the GEMMA guideline managing all its OMs. For this reason, more states are utilized with respect to the S-Methodology and a higher *Program Effort* is obtained. On the other hand, having more states implies the utilization of more paths, bringing to a higher *Cyclomatic Complexity* number. Therefore, the GG-Methodology results in a PLC code which is longer and more complex than the S-Methodology due to the overhead necessary for the implementation of the GEMMA guideline.

Concerning the Kafura Complexity, the higher is this number and the more interconnected are the POUs to their environment. Considering that the GG-Methodology generates hierarchical PLC code, different communicating POUs are obtained, resulting in a higher *Kafura Complexity* than the code developed with the S-Methodology. Even if the tendency may be the one to conclude that the PLC code obtained with the GG-Methodology is less modular, in the authors' opinion this metric just indicates that the code is more interconnected and not that the code is less modular. In fact, in the GG-Methodology the POUs communicate vertically following modular principles; see Figure 1.

Concerning the Maintenance Index, the higher is this number and the more maintainable is the PLC code. Given the higher complexity and coupling degree of the POUs, the higher *Maintenance Index* of the GG-Methodology with respect to the S-Methodology may appear counterintuitive. However, this result is justified considering the high fragmentation of the PLC code generated with the GG-Methodology, which consists in POUs and invoked function blocks. This fragmentation makes the code obtained with the GG-Methodology more maintainable.

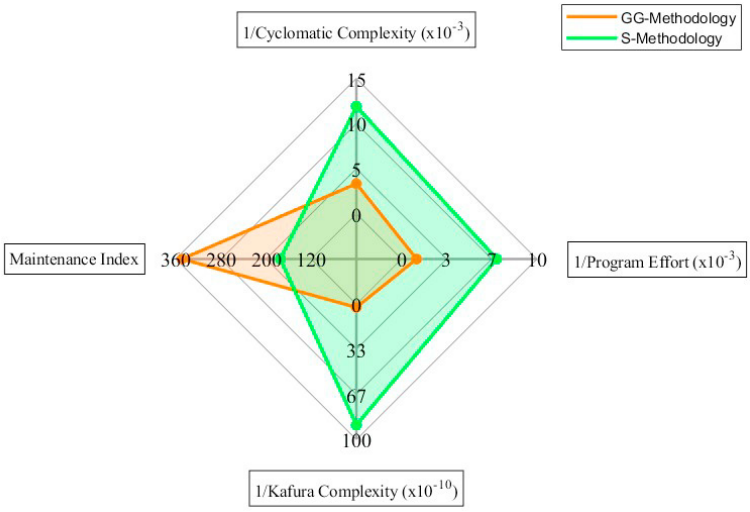

Fig. 6: Static Analysis Metrics computed for the PLC code obtained with the GG-Methodology and the S-Methodology.

After the static analysis, the metrics resulting from the *dynamic analysis* were calculated for the test-case scenario shown in Section 4.3, and are illustrated in Figure 7. The response time is reported in the left-hand side, while the WCET in the right-hand side. Given the stochasticity of the experiment, ten repetitions were implemented to evaluate the variability of the metrics through the random error formulation [5].

Looking at Figure 7, it can be noticed that the PLC code obtained with the GG-Methodology is slower than the S-Methodology; i.e. higher *Response Time*. This can be explained considering that the GG-Methodology implements the GEMMA guideline resulting in a PLC code with more states and paths that takes more time in completing the test-case scenario.

Concerning the *WCET*, it can be notice that the difference of the PLC code obtained with the two methodologies is statistically not significant, since the two error bars present an overlapping area. Therefore, the two methodologies present similar performance in terms of WCET.

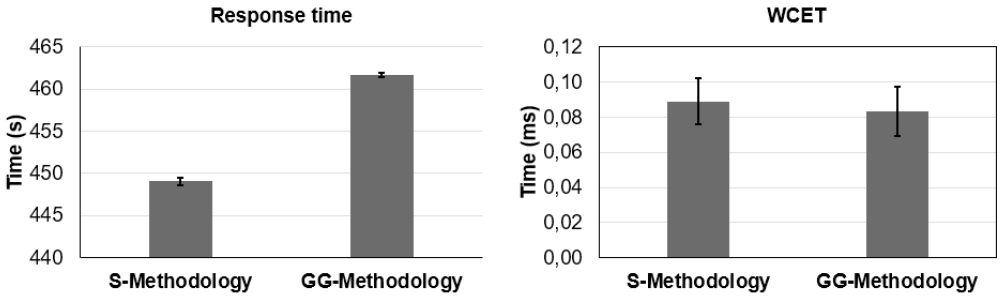

Fig. 7: Response time and WCET of the PLC code of the two methodologies with the corresponding error bars.

In summary, the GG-Methodology has the following characteristics with respect to the S-Methodology:

- *Pros*: more maintainable due to the fragmentation of the PLC code;
- *Cons*: more complex and with larger response time due to the interconnection of the POUs, and the higher number of states and paths;
- *Neutral*: similar WCET.

The higher complexity and response time were expected in the GG-Methodology due to the intrinsic overhead introduced from the implementation of the GEMMA guideline. Nevertheless, the complexity does not constitute an issue in the application of the methodology since the utilization of the GG-Generator automates the generation of the PLC code. The higher maintainability demonstrates that the HDP generates *fragmented PLC code* which results to be modular. Even if the obtained results seem reasonable, the two methodologies should be tested in other test-case scenarios and with further case studies to better validate the computed metrics.

Finally, it must be noticed that the case study utilized for the comparison has been completely virtual and the effects of the hardware behavior have not been included within the simulation model; e.g. response time of the actuators, etc. The influence of the hardware on the Dynamic Analysis Metrics should be quantified in future work.

6. Conclusions and Future Work

In the Industry 4.0 context, interoperability is paramount to correctly operate Cyber Physical Systems. However, interoperability still remains a challenge due to lack of universal standards. To face this issue, the *GG-Methodology* was introduced to facilitate the exchange of information between the PLCs and the external environment. Even if it has been demonstrated that the methodology provides a standard approach and vocabulary for the management of the OMs of CPS through the automation software, the PLC code obtained from its application has never been analyzed. Thus, the objective of this research work has been the assessment of the PLC code obtained with the GG-Methodology. The objective has been reached by assessing the PLC code through selected metrics that involve both the structural relations between the different objects of the code and the real-time behavior. Furthermore, these metrics have also been computed for the PLC code obtained with a methodology based on the Statechart diagram that has been utilized as a benchmark to better evaluate the GG-Methodology.

The results illustrated that a PLC code with higher complexity and response time is generated from the application of the GG-Methodology. Even if the higher complexity has already been mitigated from the introduction of a MDE tool, the higher response time remains an issue that the control practitioner needs to consider within the decision to develop PLC code compliant or not with the GEMMA guideline. Whereas, the higher maintainability demonstrates that the application of the GG-Methodology generates *fragmented PLC code* which results to be modular. Concerning the WCET, statistically significant differences have not been identified in the application of the two methodologies.

Even if the objective of this work has been to assess the PLC code generated with the GG-Methodology, authors have the opinion that the identified research workflow has the potential to become an approach for the comparison

of methodologies for the development of PLC code. Next, some *future works* are identified to better assess the GG-Methodology and to refine the research workflow utilized within this article:

- *Validation of metrics*: the GG-Methodology and the S-Methodology should be tested in other test-case scenarios and with further case studies to better validate the computed metrics;
- *Physical case study*: up to now, the GG-Methodology has been applied to a virtual lab case study (i.e. VC simulation) and validated based on the feedback of designated users. To further validate it, the GG-Methodology should be implemented in a physical system;
- *Protocol to compare methodologies for PLC code*: the research workflow should be enhanced with the inclusion of the hardware behaviour to become a protocol to compare methodologies for the development of PLC code. Then, it may be applied to the numerous methodologies proposed in the literature to provide a quantitative and objective classification.

References

[1] ADEPA, 1981. GEMMA (Guide d'Étude des Modes de Marches et d'Arrets). Technical Report. Agence nationale pour le Developpment de la Production Automatisée.
[2] Barbieri, G., 2019. A small-scale flexible test bench for the investigation of fertigation strategies in soilless culture. World Academy of Science, Engineering and Technology 13, 5–9.
[3] Barbieri, G., Gutierrez, D., 2021. A GEMMA-GRAFCET methodology to enable digital twin based on real-time coupling. Procedia Computer Science 180, 13–23.
[4] Barbieri, G., Quintero, G., Serrato, O., Otero, J., Zanger, D., Mejia, A., 2021. A mathematical model to enable the virtual commissioning simulation of wick soilless cultivations. Journal of Engineering Science and Technology 16, 3325–3342.
[5] Beckwith, T.G., Marangoni, R.D., Lienhard, J.H., 2009. Mechanical measurements. Pearson.
[6] Beller, M., Bholanath, R., McIntosh, S., Zaidman, A., 2016. Analyzing the state of static analysis: A large-scale evaluation in open source software, in: 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), pp. 470–481.
[7] Capitán, L., Vogel-Heuser, B., 2017. Metrics for software quality in automated production systems as an indicator for technical debt, in: 2017 13th IEEE Conference on Automation Science and Engineering (CASE), pp. 709–716.
[8] Chen, D., Doumeingts, G., Vernadat, F., 2008. Architectures for enterprise integration and interoperability: Past, present and future. Computers in industry 59, 647–659.
[9] Dotoli, M., Fay, A., Miśkowicz, M., Seatzu, C., 2019. An overview of current technologies and emerging trends in factory automation. International Journal of Production Research 57, 5047–5067.
[10] Gsellmann, P., Melik-Merkumians, M., Schitter, G., 2018. Comparison of code measures of iec 61131–3 and 61499 standards for typical automation applications, in: 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 1047–1050.
[11] Hernandez, J.D., Gutierrez, D., Barbieri, G., 2021. A GEMMA-based Decentralized Architecture for Smart Production Systems. Studies in Computational Intelligence 987, 17–29.
[12] Hu, S.J., 2013. Evolving paradigms of manufacturing: From mass production to mass customization and personalization. Procedia Cirp 7, 3–8.
[13] IEC, 2002. IEC 60848: Grafcet specification language for sequential function charts. Technical Report. International Electrotechnical Commission.
[14] Kagermann, H., Lukas, W.D., Wahlster, W., 2011. Industrie 4.0: Mit dem internet der dinge auf dem weg zur 4. industriellen revolution. VDI nachrichten 13, 2–3.
[15] Konur, S., Lan, Y., Thakker, D., Morkyani, G., Polovina, N., Sharp, J., 2021. Towards design and implementation of industry 4.0 for food manufacturing. Neural Computing and Applications , 1–13.
[16] Lu, Y., 2017. Industry 4.0: A survey on technologies, applications and open research issues. Journal of industrial information integration 6, 1–10.
[17] Meindl, B., Ayala, N.F., Mendonça, J., Frank, A.G., 2021. The four smarts of industry 4.0: Evolution of ten years of research and future perspectives. Technological Forecasting and Social Change 168, 120784.
[18] Nikolić, D., Stefanović, D., Dakić, D., Sladojević, S., Ristić, S., 2021. Analysis of the tools for static code analysis, in: 2021 20th International Symposium INFOTEH-JAHORINA (INFOTEH), pp. 1–6.
[19] Rajkumar, R., Lee, I., Sha, L., Stankovic, J., 2010. Cyber-physical systems: the next computing revolution, in: Design automation conference, IEEE. pp. 731–736.
[20] Rösch, S., Tikhonov, D., Schütz, D., Vogel-Heuser, B., 2014. Model-based testing of plc software: test of plants' reliability by using fault injection on component level. IFAC Proceedings Volumes 47, 3509–3515.
[21] Safyallah, H., Sartipi, K., 2006. Dynamic analysis of software systems using execution pattern mining, in: 14th IEEE International Conference on Program Comprehension (ICPC'06), pp. 84–88.

[22] Salah, M., Mancoridis, S., Antoniol, G., Di Penta, M., 2006. Scenario-driven dynamic analysis for comprehending large software systems, in: Conference on Software Maintenance and Reengineering (CSMR'06), pp. 10 pp.–80.
[23] Shijo, P., Salim, A., 2015. Integrated static and dynamic analysis for malware detection. Procedia Computer Science 46, 804–811. Proceedings of the International Conference on Information and Communication Technologies, ICICT 2014, 3-5 December 2014 at Bolgatty Palace Island Resort, Kochi, India.
[24] Sisinni, E., Saifullah, A., Han, S., Jennehag, U., Gidlund, M., 2018. Industrial internet of things: Challenges, opportunities, and directions. IEEE Transactions on Industrial Informatics 14, 4724–4734.
[25] Zhabelova, G., Vyatkin, V., 2015. Towards software metrics for evaluating quality of iec 61499 automation software, in: 2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA), pp. 1–8.
[26] Züllighoven, H., 2004. Object-oriented construction handbook: Developing application-oriented software with the tools & materials approach. Elsevier.