Model-Based Approach for Co-optimization of Safety and Security Objectives in Design of Critical Architectures

Kunal Suri, Gabriel Pedroza, Patrick Leserf

Model-based Approach for Co-optimization of Safety and Security Objectives in Design of Critical Architectures

Kunal Suri[1,⋆], Gabriel Pedroza[1], Patrick Leserf[2]

[1] Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France
[2] ESTACA, 12 Rue Paul Delouvrier, Montigny-le-Bretonneux 78180, France
{firstname.lastname}@cea.fr, patrick.leserf@estaca.fr

Abstract. During the development of Cyber-Physical Systems (CPS) safety and security are major concerns to be considered as it has been established by various literature. Moreover, these concerns must be included early on during the System Development Life Cycle (SDLC). In this work, we focus on the design-phase of the SDLC to assist the engineers in conducting design-space exploration of the system hardware architecture w.r.t to both safety and security concerns. In this way, the engineers may perform simulations to find a set of quasi-optimal solutions before developing an actual physical prototype. To achieve this, our tooled method builds on our previous work [11] and supports a multi-concern analysis by leveraging Model-Driven Engineering (MDE) techniques such as SysML modeling along with the transformation of SysML models into representations which are finally optimized via constraint solvers. Overall, the method and framework shall support the design of the system architecture from a repository of components based on possible configuration alternatives, which satisfy the system objectives such as reliability and cost. Such functions can help to evaluate the effects of integrating safety and security features thus showing their interplay. The overall approach is illustrated via an automotive CPS case study.

Keywords: MDE · HW Architecture · Optimization · Safety · Security

1 Introduction

Similar to information systems, the development of Cyber-Physical Systems (CPS) [10], such as autonomous vehicles [8] and industrial production systems [19] (Industry 4.0 [9]), involves a complex *System Development Life Cycle* (SDLC[3]) [7]. Various stakeholders, namely systems engineers, software engineers, safety engineers, security engineers, are involved during the SDLC, wherein each of them is concerned with a specific aspect (or viewpoint) of the system. As CPS involves human users, it is essential to make them fail-safe by building them in a

⋆ This work is the result of a collaborative project between CEA-LIST and ESTACA, a period during which Kunal was following a postdoctoral fellowship in ESTACA.

[3] https://www.nist.gov/publications/system-development-life-cycle-sdlc

safety-aware manner. Likewise, these CPS must also be developed in a security-aware manner. This is because in the past many of the safety-critical CPS were constructed and used as standalone systems without giving much consideration to the security aspect. However, today most of these systems are highly interconnected and software intensive, which exposes them to the possibility of cyber-attacks which in turn may lead to safety related problems for the users [3]. Even with the recent increase in the use of Artificial Intelligence (AI) and other automation technology for CPS development, designing these systems still involves a lot of human experts/engineers. Thus, it is important to provide the engineers with methods and automated frameworks that ease the integration of non-functional aspects like, for instance, co-optimization of both safety and security objectives, especially during the development of the system hardware architecture [12, 14]. For the same reason, the design phase is crucial within the SDLC, since errors or lacks introduced during the design time can propagate to other phases resulting in wastage of effort, time and resources [20]. This makes it critical to design a sound and well-formed system architecture, which satisfies all the functional and non-functional requirements (i.e., multiple objectives) and to analyze the system objectives via simulations for the desired outcomes. In this way, the engineers will be able to find a set of quasi-optimal solutions before developing a physical prototype of the hardware.

In general, different engineer teams participate during the design process and in particular when safety and security aspects need to be integrated. Thus, it is not expected a single stakeholder having expertise in both domains. Thus, in order to perform the aforementioned multi-objective system analysis, engineers need methods and automated frameworks that will support them to perform the analysis related to safety and security requirements along with the ability to discover or trace the link between these requirements and the lower-level criteria of the system. In the context of this work, our focus remains on supporting the engineers during the design phase following a model-driven approach wherein we build on our previous work [11] by extending it to handle the following: (1) multiple system components and (2) multiple objectives to be included in the optimization criteria for finding the quasi-optimal system architecture. Our method supports a multi-concern analysis by making use of Model-Driven Engineering (MDE) techniques such as SysML modeling along with the automatic transformation of SysML models into representations that can be optimized via relevant algorithms. We illustrate this method by reusing and extending the case study from [11] with the security perspective. Overall, this method shall support the design of the system architecture from a repository of components based on possible configuration alternatives, which satisfy the system objectives such as reliability and cost which can impact or be related to both safety and security requirements.

The remainder of this paper is organized as follows: in Section 2, we introduce some preliminary information necessary to better understand various concepts used in this work. In Section 3, we detail our method along with the various steps (and sub-steps) and in Section 4, we detail the experimentation results and the

automated framework developed for supporting this design-space exploration. In Section 5, we survey some related work and finally, in Section 6, we conclude the paper and provide some perspective on the future works.

2 Preliminaries

In this section, we briefly introduce various concepts and techniques necessary to understand the work detailed in this paper. These concepts are as follows:

- **MDE in systems engineering** has a large community of contributors and users along with the availability of mature tools (e.g. Eclipse Papyrus[4]). MDE uses models as the primary artifacts, which help to enhance the understandability of complex systems and aids in the reduction of the chances of error due to the use of principles, standards (e.g. work done by the Object Management Group (OMG[5])) and tools [11, 15, 16]. MDE supports creation of a coherent model of a system that may be augmented with relevant information for different stakeholders. This model when transformed into different formats allows representing various formalization relevant for different domains. In this work, we make extensive use of SysML 1.4 language along with model transformation technique, which is briefly detailed as follows:
  - **Systems Modeling Language[6] (SysML® 1.4)** is a general-purpose graphical modeling language developed by the OMG for systems engineering (SE). It reuses a subset of Unified Modeling Language (UML) and provides additional extensions to address the requirements in UML for SE. SysML is used to specify, analyze, design, and validate complex systems, which may include any type of system such as hardware, software and information. SysML comprises of the four essential diagrams that are referred to as the *Four Pillars of SysML*, which are: (1) Requirement, (2) Activity, (3) Block, and (4) Parametric diagrams. In this work, we mainly depend on the Requirements and the Block diagrams to model the system depicted in our case study.
  - **Model Transformation** plays a key role in MDE as it allows the generation, sometimes by refinement, of lower-level models from higher level (or abstract) models, eventually generating the executable codes (and vice-versa, i.e., reverse engineering). Model transformation allows the mapping and synchronization of models, which may be at the same or different levels of abstraction. As per Czarnecki et al. [6], there are several major categories of model transformation. However, in our work, we use the model-to-text approach to generate executable scripts from our SysML models (based on a template). This template-based approach consists of target text along with the specific code that is generated based on the values provided in the model.

[4] `https://www.eclipse.org/papyrus/`
[5] `https://www.omg.org/index.htm`
[6] `https://www.omg.org/spec/SysML/1.4`

- **Constraint Satisfaction Problems (CSPs)** are mathematical problems, which are defined as a set of variables that have specific conditions that should never be violated while solving these problems (i.e., constraints) [4]. In other words, a CSP consists of the following: (1) a finite set of variables $(V_1, V_2, ..., V_n)$; (2) a non-empty domain having some values to be assigned to each variable $(D_{V_1}, D_{V_2}, ..., D_{V_n})$; and (3) a finite set of constraints $(C_1, C_2, ..., C_n)$, wherein each of the constraint C_i puts a limit on the values permitted or excluded for the variables (e.g., $V2 \neq V3$). CSP is used by researchers in AI to solve problems such as scheduling. The solution set can include a unique solution, all the solutions in the space, or an optimal solution. In this work, we support the engineers to find optimal solutions based on the values of the objective functions provided as input by the user in the SysML model.
- **Multi-Objective Optimization and Pareto Front** is an optimization problem involving multiple objective functions. It broadly falls under the area of decision making involving multiple criteria. These problems are concerned with scenarios wherein it is crucial to simultaneously optimize more than one objective function such as either cost, reliability or performance. These metrics are applied in various domains such as economics, logistics and engineering, where optimal decisions are needed by managing the trade-offs between two or more conflicting objectives. For instance, minimizing the cost of an automobile while maximizing its safety. Likewise, in a multi-objective optimization problem involving several objectives, an optimal solution is called *Pareto optimal* if there exists no possibility to improve an objective function without degrading the others. Thus, a Pareto optimal solution is an optimal trade-off between various objectives. The set of all the Pareto optimal solutions is called as *Pareto Front* (also known as Pareto frontier or Pareto set), which is graphically visible in form of a distinct front of points. In this work, we provide the engineers with a set of Pareto optimal solutions.

3 Method: Model-based Co-optimization of Objectives

3.1 Overview

In this section, we present the overview of our method that consists of two main steps as detailed in Section 3.3 and Section 3.4. This method is illustrated based on the case study detailed in Section 3.2, which depicts an *Embedded Cognitive Safety System* (ECSS) (see Figure 2 (source [11])).

This method is an extension of our previous work [11], and is envisioned to illustrate the possibility of integrating both safety and security related objective functions as a way to support decision making so as to avoid unnecessary trade-offs. Furthermore, in this work, one of our major focus has been on the readiness and automation of the framework presented in [11], i.e., to scale it for handling a dynamic range of hardware (HW) components to support the engineering needs. Figure 1 represent the steps (and sub-steps) of our method, which starts with capturing the safety and security based requirements from different

stakeholders using SysML requirement diagram (see label 1 in Figure 1). The model-based requirement gathering allows to capture different requirements and supports traceability between them. Such traceability enables the engineers to see which requirements are related to each other and how they can be tackled together [9]. Next, a SysML model is created, that is annotated with all the relevant information related to the system components along with the values of objective functions from both safety and security aspects (see label 2). The SysML model also contains the information related to the type of variability needed to explore the design-space (detailed in Section 3.2). Next, the SysML model is transformed into text (i.e., executable python scripts) using a template-based approach that involves the generation of mathematical optimization models based on the information about variability choices added to the SysML model (see label 3) [2]. Concretely, this model transformation step relies on Eclipse based technologies such as Xtext[7]. The variability choices are transformed into a set of 0-1 variables (integer programming). Then, based on the constraints defined between components of the SysML model, the problem can be solved as a CSP via some standard solvers. In our case, the generated Python scripts are executed using python specific CSP solver[8] (see label 4 in Figure 1). The solver generates solutions that are visible as Pareto Front graphs. These solutions are based on the calculated value of the objective functions. In our case and to facilitate safety-security interplay, we rely upon basic objective functions which can be common to both safety and security, e.g., cost and reliability. An engineer can choose the best solutions among the trade-off solutions that fit their requirements. This method is implemented using the Eclipse Papyrus framework for both SysML modeling and model transformation.

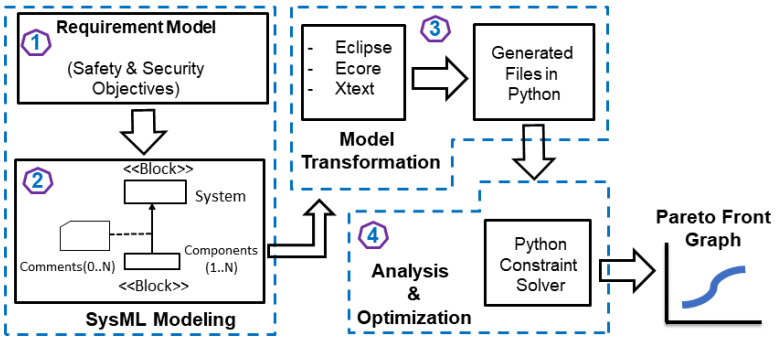


Fig. 1: Steps for finding optimal system architecture

3.2 Case Study

Figure 2 is sourced from our previous work [11] and depicts an Embedded Cognitive Safety System (ECSS). It is an integrated system on a chip (SoC) used

[7] https://www.eclipse.org/Xtext/documentation/index.html
[8] https://github.com/python-constraint/python-constraint

in various domains such as automotive or drones. It supports line detection, obstacle detection and distance measurement with a stereoscopic view. The ECSS embedded hardware platform comprises of CMOS (complementary metal-oxide semiconductor), image sensors, processing elements (CPU), and vehicle interface networks (FlexRay, CAN). The CMOS image sensors are connected to the CPU via Digital Video Port (DVP), a type of parallel bus interface. CPU such as Cortex A9, iMX35 support image processing. The vehicle interface is integrated into the ECSS with a transceiver component, connected to the processing element with a digital port (DP) which is a parallel bus interface.

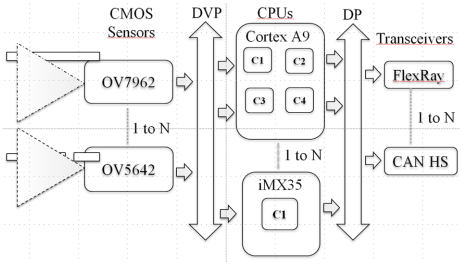


Fig. 2: Embedded Cognitive Safety System (ECSS)

The ECSS must be developed in both safety and cyber-security aware manner as malicious attacks such as CAN bus attacks may expose the ECSS to component misbehavior or failure [3]. In this context, we consider that the ECSS can be protected by integration of modules that include built-in security features like specific encryption mechanisms, Public key infrastructure (PKI), Trusted Platform Modules (TPM) or Hardware Security Modules (HSM) [23], [22]. As referred modules may also require to be redundant so as to satisfy safety requirements, an impact on the overall cost is expected due to security. Since built-in security features are often HW greedy, a non-negligible impact on performance and real-time safety constraints can occur, e.g., delayed braking CAN messages. Thus, a basic but relevant interplay between safety and security can be studied via the optimization of both cost and performance objective functions. In this work towards multi-objective simulation, we mainly focus on the cost function.

Explaining the Design-Space Exploration: To illustrate some of the complexity faced by engineers, we represent a HW setup wherein there are three slots available for each component of an ECSS (to allow redundancy). The Figure 3 represents the different types of variability involved in selecting a set of HW components for the ECSS. Hence, the number of slots plays an important role, as the solution space grows exponentially on its number. To better illustrate it, let us consider a HW type such as CPU, wherein three slots shall represent a variability matrix of M×3. Indeed, each row of this matrix represents a set of non-distinct CPU elements from one specific manufacturer. For example, we may assume that row 1 represents a Cortex-A9 from Samsung, while row 2 from STMicroelectronics (ST), row 3 from MediaTek (MT), and so on. This

is because there could be several possible configurations used to allow CPU redundancy in the chipset. This M×3 array can accept three main configuration categories which are (1) instance variability (IV), i.e., replicas of the same component of the same type, (2) component variability (CV), i.e., components from a different manufacturer, and (3) mixed variability (MV), i.e., IV + CV. Since for each column of size M, any subset of it is also a configuration, the solution space can grow up to $3.(2^M)$ configurations.

Fig. 3: Design-space definition for HW architecture

As each of the CPU incurs a specific cost (i.e., product cost plus the cost of security features) and provides specific reliability, an engineer must find the optimal redundancy values and variability type for designing an architecture that satisfies the system requirements. Furthermore, this type of variability issue shall exist for each HW component of the system to be designed, making it quite a complex process to find the optimal set of solutions considering all the components and all the different variability configurations.

3.3 Step 1: SysML Modeling

Our method starts with developing a model to capture requirements in textual form and link them to the various modeling elements via relationships such as verify or derive. Figure 4 represents a SysML based requirement model consisting of standard requirement elements. In [11] the authors introduced a new requirement type called *<<ArchRequirement>>* to model a specific type of architectural requirement by extending the standard requirement element using a stereotype. Likewise, in this work, we assist the engineers to model safety and security requirements by introducing *<<SafeReq>>* and *<<SecReq>>* respectively. The overall system requirements are composed of the aforementioned requirements, which shall be further refined based on the system needs. The *<<ArchRequirement>>* requirement is evaluated via an objective function *<<HWCostEvaluation>>*, which we introduce as a stereotype by extending the SysML Constraint Block. This objective function is related to the requirement via a dependency called *Evaluate*, depicting a design-time relationship between the elements. The architecture requirement called *MaxRedundancy* introduces a maximum limit to the redundancy of a component (e.g., max number of sensors $= M/2$). These types of restrictions are needed as the hardware redundancy incurs a cost that directly influences the overall system cost.

To better visualize and integrate the Multi-Objective Optimization (MOO) context into system modeling, i.e., a type of analysis context, a SysML Block Definition Diagram (BDD) is modeled (see Figure 5). This BDD consists of Constraint Blocks along with their relationships having a top-level block called *ECSS MOO*, which references the ECSS system block. This BDD also contains the objective functions that are a representation of the optimization model. The Pareto front is a result of the MOO context and provides various alternatives to the engineer. In this approach, the MOO context shall be passed to an external CSP solver. The results from the solver are provided to the engineer in form of the values of the Pareto front. The objective function extends the standard SysML Constraint Block and consists of the optimization goal (maximize or minimize). The ECSS MDO Context constraint block in Figure 5 represents the Pareto front via two value vectors, i.e., *BestCost* and *BestRel*, which are produced by the two objective functions. The BDD has constraint properties, i.e., *HWCostEvaluation* and *SystemReliability*, which are typed by the objective function. The *SystemReliability* function, represents the calculation of system reliability (R) based on the parameters received from the ECSS system (the component reliability) and the Zero-One model. Likewise, the *HWCostEvaluation* gets its values from the ECSS including both safety and security components. The Zero-One model represents the optimization model described in Section 3.4. It has a parameter and a set of constraints deduced from the ECSS and from the model itself.

As the main part of this step, a SysML BDD is developed for modeling the ECSS architecture (see Figure 6). Based on the requirement, it shall contain all the information about the underlying hardware resources and their composition.

In our method, the optimization problem to be solved involves finding the right balance between the optimal level for redundancy of each component and the cost of the component. During the first step, only the composition is known and not the redundancy level. We already detailed different types of variability configurations in Section 3.2. They help to explore the design space by permuting the redundant elements (including security features) to reach minimal global cost and the maximum redundancy allowed.

3.4 Step 2: Model-Transformation and Optimization

This section details the second step of the approach wherein a SysML BDD model (see Figure 6) is transformed into a mathematical formalism susceptible to optimization via a space search and constraint solver algorithm [4]. The representation is based upon zero-one variables that can be solved via a CSP solver. The ECSS is composed of subsystems S_i, wherein each S_i is associated with a given block. A subsystem S_i is nothing but a component slot set (vertical) to be configured by components of the same type selected from repository C_i (provided by the same or different manufacturers). C_{ij} represents the j^{th} component in the repository C_i: each selected component C_{ij} shall have a position j in the repository C_i and can be used at a position k in the subsystem S_i. We define the following sets and parameters:

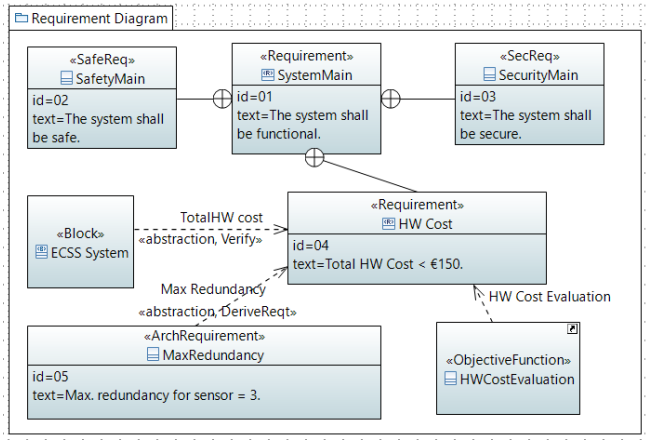


Fig. 4: Requirement model for ECSS architecture optimization

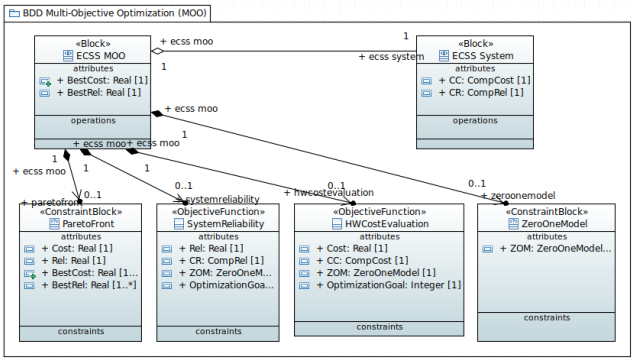


Fig. 5: BDD for MOO context modeling

- $cost_{ij}$ is the cost of the component C_{ij}, while θ_i is the interconnection cost for any component
- rel_{ij} is the reliability of component C_{ij}
- α_{ij} and β_{ij} is the number of input and output ports of component C_{ij}. For instance, in an ECSS, first the video sensor is activated and then the data is processed, thus the video sensor has no input port and only one output port, i.e., α_{ij}=0 and β_{ij}=1.

To find optimal solutions, we first find all the possible solutions and then evaluate each different solution relying upon the mathematical representation of the objective function. For achieving this, we make use of the Python-based CSP package called *Python-Constraint* (version 1.4.0) as it is simple to use and is widely used in various research works.

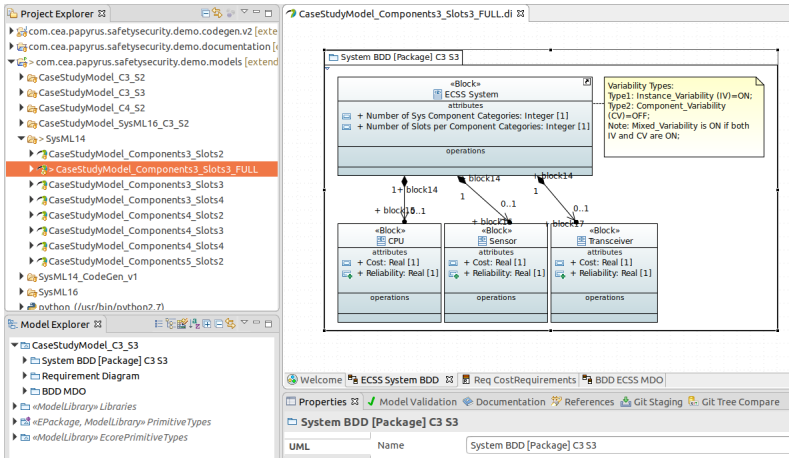


Fig. 6: BDD for ECSS composition

For the use of zero-one programming, we assume that the range of components C_{ij} will be a M×N matrix, where M is the number of slots for each component (vertical size), and N the number of components (horizontal size). Thus, the problem is defined as follow:

$\forall i \in S, j \in C_i, k \in S_i$

$$a_{ijk} = \begin{cases} 1, & \text{if } C_{ij} \text{ is used in system } S_i \text{ at position k} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

Constraints: Each system has some default constraints that are derived based on the decision variables. Even if the introduction of new constraints may not be straightforward, some of them have been previously prioritized in different application domains and can be reused, for instance, performance and energy constraints [17]. In our case study, the following constraints are modeled, such as at any position k in the final sub-system, there shall be only one component:

$$\forall i, j, \sum_{k} a_{ijk} \leqslant 1 \tag{2}$$

Based on the requirements, several other constraints can be included in the CSP program. This shall allow the solver to provides a better solution targeted to the specific objectives of the system design. For instance, the constraint on a digital connection (system BUS) that connects each sensor to a CPU and each CPU to a transceiver can be represented as:

$$\sum_{j,k} a_{1jk}\beta_{1j} \leqslant \sum_{j,k} a_{2jk}\alpha_{2j} \tag{3}$$

Objective Functions: In our case study, we have a system cost and system reliability which are based on component redundancy and extended from functions in [11]. As previously discussed, the total cost is an objective function that allows a safety-security interplay since components can include features impacting both. Thus, $Cost = Cost_{safety} + Cost_{security}$ and the total cost is given by:

$$TotalCost = \sum_{i,j,k} Cost_{ij} \left[a_{ijk} + exp\left(\theta_i \sum_k a_{ijk}\right) \right] \tag{4}$$

The system reliability (R) is calculated by using the serial-parallel interconnection model, which is given as:

$$R = \prod_i \left[1 - \prod_{j,k} \left[1 - a_{ijk} rel_{ij} \right] \right] \tag{5}$$

The goal of the optimization is to minimize the cost and maximize the reliability, i.e., $min(TotalCost)$ and $max(R)$ by using a different configuration of components based on variability types.

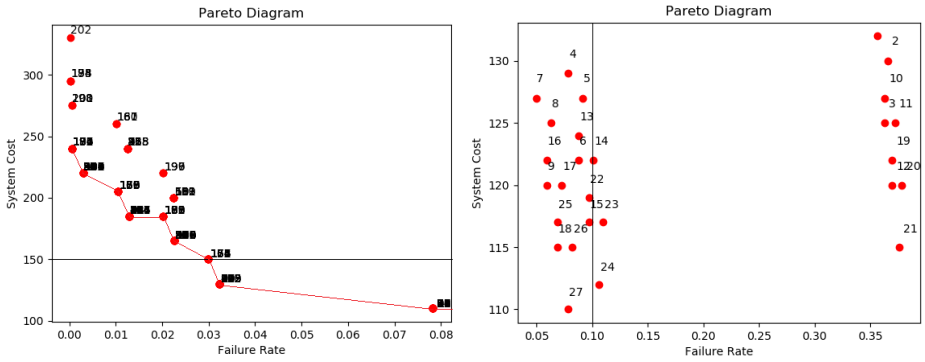
4 Experimentation Results

A proof of concept for our method is implemented using the Eclipse Papyrus framework for modeling SysML models and performing model-transformation (i.e., code generation). The code generation is based on the API's provided by *Papyrus Designer*[9] and the experimentation is performed using multiple SysML models (visible in top left-side of Figure 6). Each of the SysML models has a set of different input values w.r.t the number of slots available on the hardware, values of the cost and reliability, and the number of sub-components in the systems. The information in these SysML models is used to generate the Python script, which is then executed. It uses the python-constraint API for creating a CSP problem and the related variables, which along with the added constraints provide the set of solutions. Each of these solutions is then used to calculate the objective functions, which are represented as a point on the graph.

An output generated is shown in Figure 7. It depicts two images with two types of variability, i.e., CV and IV. The solutions that are generated represent a quasi-optimal solution for a hardware architecture having slots for component types such as a CPU, Sensor and Transceiver. In this example, the hardware shall have three slots for each component type, i.e., to accommodate up to 3 elements of each type (3×3 matrix). For one of such experiments, the total number of discrete output solutions associated with CV is 16 and IV is 27. The X-axis of the graph represents the failure rate, while the Y-axis represents the total system cost. The engineers can use the framework to perform simulations to find the best suitable choice for the given requirement. The framework assists them by providing the Pareto Front and visual support of the threshold area based on acceptable cost or acceptable failure rates.

[9] `https://wiki.eclipse.org/Papyrus_Software_Designer`

As seen in Figure 7, a visual assistance is provided in the form of the Pareto diagrams and acceptance threshold (e.g. system cost $\leqslant 150$) which support the engineers to find a (quasi) optimized system having the reliability and cost as per the requirements allocated.



(a) Pareto Front for Component Variability (b) Graph depicting Instance Variability

Fig. 7: Graph depicting different variability types

5 Related Work

In literature, various academic and industrial research works have addressed the problem of both safety and security for CPS [12] such as EU MERGE[10] project or the EU AQUAS[11] project. Paul [14] categorized such works into four main groups, that are: (1) *independent analysis of safety and security*, i.e., works analyzing either safety or security concerns during the SDLC without considering the other, (2) *augmenting safety engineering with security techniques*, i.e., works where various processes, methods and tools in the safety engineering domain are updated with concepts and features from the security domain, (3) *augmenting security engineering with safety techniques*, and (4) *addressing safety and security co-engineering together*, i.e., approaches considering a unification of processes, methods and tools to perform both the safety and security analysis in parallel. Various works have used the strengths of MDE for conduction a co-engineering analysis. Pedroza et al. [16] proposed MDE based framework using a SysML-based environment called AVATAR that captured both safety and security related elements in SysML models. They also gave forth the importance to develop a CPS in a co-engineering manner with Safety and Security perspectives rather than in a standalone manner [15]. Our work complements

[10] `http://www.merge-project.eu`
[11] `https://aquas-project.eu/`

these approaches by providing a way to solve architecture composition and optimization problems whereas still addressing safety and security concerns. Some existing approaches such as [21, 18] assist users to perform multiple analyses, however, they are mainly focused on optimizing the component parameters, such as CPU frequency or memory management, rather than in structural features as in our approach. Meyer et al. [13] proposed an optimization technique for their microwave module but a lack of redundancy constraints is observed. Other approaches such as Design-Space Exploration (DSE) [1] and redundancy allocation problem (RAP) [5] are similar to our approach which remains nonetheless quite generic and extensible, given the few SysML profile specializations introduced and the possibility to easily integrate new user-defined objective functions.

6 Conclusion and Perspectives

In this paper, we proposed a method and framework to perform a multi-objective optimization of system hardware architecture during the design-phase of the SDLC. The approach leverages MDE techniques, especially SysML modeling and model transformation, to assist engineers in the conception of the system architecture during design-space exploration. The tooled method starts with modeling the overall system requirements, including safety and security specific requirements, and then modeling the relevant information about the hardware components to be used and optimized via block diagrams. Such information includes values related to the objective functions, architectural constraints, number of slots, number of components and type of variability. Next, these SysML BDD models are automatically transformed into Python scripts based on the underlying mathematical representation, which includes integer variables, linear constraints and objective functions. The Python scripts are executable and are solved using Python-based CSP solvers. Based, on our case study, an engineer is presented with quasi-optimal solutions, i.e., specific configuration w.r.t hardware components that maximize the objective functions such as reliability while minimizing the global costs. Since the integration of safety features (like redundancy) and security features (like encryption modules) have in common certain objective functions (like cost), the optimization of the latter can help to evaluate the impact and interplay between both. As perspectives, we intend to extend our framework by including more common or specific objective functions related to security and safety to strengthen the approach and coverage of security engineering. Furthermore, we plan to test the approach's scalability with larger and more complex case studies from different CPS domains.

References

1. Apvrille, L.: Ttool for diplodocus: an environment for design space exploration. In: NOTERE. pp. 1–4. ACM (2008)
2. Bettini, L.: Implementing domain-specific languages with Xtext and Xtend. Packt Publishing Ltd (2016)

3. Bozdal, M., Samie, M., Jennions, I.: A survey on can bus protocol: Attacks, challenges, and potential solutions. In: ICCECE. pp. 201–205. IEEE (2018)
4. Brailsford, S.C., Potts, C.N., Smith, B.M.: Constraint satisfaction problems: Algorithms and applications. Euro. J. of Operat. Res. **119**(3), 557–581 (1999)
5. Coit, D.W., Smith, A.E.: Optimization approaches to the redundancy allocation problem for series-parallel systems. In: Fourth Indus. Eng. Research Conf. Proc. pp. 342–349 (1995)
6. Czarnecki, K., Helsen, S.: Feature-based survey of model transformation approaches. IBM Systems Journal **45**(3), 621–645 (2006)
7. Eigner, M., Dickopf, T., Apostolov, H., Schaefer, P., Faißt, K.G., Keßler, A.: System lifecycle management. In: IFIP Intl. Conf. on PLM. pp. 287–300. Springer (2014)
8. Fagnant, D.J., Kockelman, K.: Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations. Transportation Research Part A: Policy and Practice **77**, 167–181 (2015)
9. Kannan, S.M., Suri, K., Cadavid, J., et al.: Towards industry 4.0: Gap analysis between current automotive mes and industry standards using model-based requirement engineering. In: ICSAW 2017. pp. 29–35. IEEE (2017)
10. Lee, E.A.: Cyber physical systems: Design challenges. In: ISORC. pp. 363–369. IEEE (2008)
11. Leserf, P., de Saqui-Sannes, P., Hugues, J., Chaaban, K.: Sysml modeling for embedded systems design optimization: A case study. In: MODELSWARD 2015. pp. 449–457 (2015)
12. Lisova, E., Sljivo, I., Causevic, A.: Safety and security co-analyses: A systematic literature review. IEEE Systems Journal (2018)
13. Meyer, J., Ball, M., Baras, J., Chowdhury, A., Lin, E., Nau, D., Rajamani, R., Trichur, V.: Process planning in microwave module production. 1998 AI and Manuf.: State of the Art and State of Practice (1998)
14. Paul, S., Rioux, L.: Over 20 years of research into cybersecurity and safety engineering: a short bibliography. In: Safety and Security Engineering, vol. 5, pp. 335–349. WIT Press (2015)
15. Pedroza, G.: Towards safety and security co-engineering. In: Security and Safety Interplay of Intelligent Software Systems, pp. 3–16. Springer (2018)
16. Pedroza, G., Apvrille, L., Knorreck, D.: Avatar: A sysml environment for the formal verification of safety and security properties. In: NOTERE. pp. 1–10. IEEE (2011)
17. Roux, B., Gautier, M., Sentieys, O., Derrien, S.: Communication-based power modelling for heterogeneous multiprocessor architectures. In: MCSOC. pp. 209–216. IEEE (2016)
18. Spyropoulos, D., Baras, J.S.: Extending design capabilities of sysml with trade-off analysis: Electrical microgrid case study. In: CSER. pp. 108–117 (2013)
19. Suri, K., Cadavid, J., et al.: Modeling business motivation and underlying processes for rami 4.0-aligned cyber-physical production systems. In: ETFA. pp. 1–6. IEEE (2017)
20. Suri, K., Gaaloul, W., Cuccuru, A.: Configurable iot-aware allocation in business processes. In: Intl. Conf. on Serv. Comp. (SCC). pp. 119–136. Springer (2018)
21. Van Huong, P., Binh, N.N.: Embedded system architecture design and optimization at the model level. Intl. J. of Comp. and Comm. Eng. **1**(4), 345 (2012)
22. Wolf, M., Gendrullis, T.: Design, implementation, and evaluation of a vehicular hardware security module. In: Kim, H. (ed.) Information Security and Cryptology - ICISC 2011. pp. 302–318 (2012)
23. Wolf, M., Weimerskirch, A., Wollinger, T.: State of the art: Embedding security in vehicles. EURASIP J. Emb. Sys. **2007** (01 2007)