Cyber-Physical Systems Product Lines: Variability Analysis and Challenges

Aitor Arrieta[1] and Goiuria Sagardui[2] and Leire Etxeberria[3]

Abstract— **Cyber-Physical Systems (CPSs) are part of our daily lives since several years ago. These systems combine digital technologies (e.g., embedded systems, software, etc.) with physical processes. Currently, users demand customized products, what is leading these systems to incorporate variability. As a result, many CPSs are becoming product lines, where variability appears in software, sensors, actuators or in the physical layer. Depending on the user's needs or the context in which the system has to operate, features are selected and the CPS product line gets configured. This paper analyses the different variability points that a CPS product line can have with a taxonomy. The taxonomy can help future CPSs product lines developers to consider variability in several points of the CPS. In addition, some challenges in the field of CPSs product line engineering are highlighted.**

Keywords— **Cyber-Physical Systems, Product Lines, Configurable Systems, Variability**

I. Introduction

Cyber-Physical Systems (CPSs) integrate digital technologies (e.g., embedded systems and information technologies) with physical processes [1]. These systems are structured into two main layers (Fig. 1): the cyber and physical layer. On the one hand, the cyber layer addresses the digital technology in charge of computing the algorithms that control the physical processes. On the other hand, the physical layer addresses the physical units and the elements to interact with the cyber layer, i.e., sensors and actuators. In addition, the CPS works inside an environment (named outside environment in Fig. 1), that directly affects to the physical process (e.g., in the case of a mobile robot, the slopes). Sensors might also obtain data from the environment in which they operate (e.g., the outside temperature or obstacle detection) in order to inform about the conditions to the cyber layer.

Fig. 1 depicts the typical structure of a CPS. The cyber layer is composed by an embedded system that allocates embedded software in charge of computing the control algorithms. The cyber layer might also be connected to the services, also named as the "cloud". The physical layer addresses the physical process that typically is exposed to the user's needs. In addition, to interact with the cyber layer, data of the process is measured by sensors and instructions are performed via actuators.

These systems appear in our daily lives in several sectors: automotive, railway, elevation, etc. Currently, users are demanding different needs, such as different functionalities, cost, quality, etc. Thus, companies consider customer-specific products, ending up with several significantly similar product versions [2].

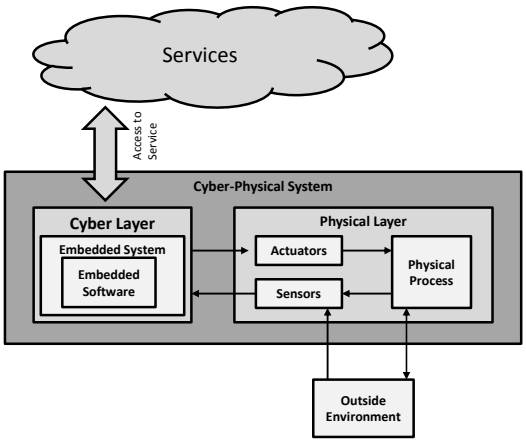

Fig. 1. Overview of the main elements of a cyber-physical system

In this process of giving response to the users' needs, variability (i.e., the ability to change or customize a product [3]) plays an important role. Variability can be understood as configurability or modifiability [4]. Configurability is referred to variation in the product space whereas modifiability refers to variation in the time space. Variability has to be managed from the beginning of the product development, until the final process of verification and validation stages. This strategy is typically followed in a product line engineering process. The idea of product line engineering is to develop a single variable product that can be derived into different ones [5]. CPSs that follow this paradigm are named CPSs product lines.

CPSs product lines are exposed to a large variability in both layers, the cyber as well as the physical layer. The main contribution of this paper is the analysis of the variability that a CPS product line can be exposed to. The analysed variability is reflected into a taxonomy that can be used by CPSs product lines developers to consider the different variability points that their system can have.

The paper is organized in the following structure: Section II summarizes the related work in the field of CPSs product line engineering, and positions this paper with the current state of the art. Section III explains two motivating examples of CPSs product lines: a tank system and a quad-copter. These ex-

[1]Electronics and Computing Department, Mondragon Uniberstitatea, e-mail: aarrieta@mondragon.edu
[2]Electronics and Computing Department, Mondragon Uniberstitatea, e-mail: gsagardui@mondragon.edu
[3]Electronics and Computing Department, Mondragon Uniberstitatea, e-mail: letxeberria@mondragon.edu

amples will be later used in Section IV, where a taxonomy is provided analysing the variability of CPSs product lines.

II. Related Work

Despite its importance, there is not much work in the field of CPSs product line engineering. To the best of our knowledge, the first time this concept was introduced was in 2013 by Yue et al., [6]. This research group has several works in the field of CPS product line engineering (e.g., [7]). The work performed by Behjati et al., [8] provides a taxonomy of variability types in the domain of Industrial Control Systems (ICSs) families, which are systems that also combine digital and physical technologies. They identify four variability types:

- Cardinality variability: Number of instances of a certain type, e.g., two sensor instances. UML properties and their multiplicities are used to model this type of variability in SimPL [8].
- Attribute variability: It is referred to the value of a configurable attribute
- Topology variability: It is the structural topology of a system, i.e., the connection between instances.
- Type variability: It is referred to the variability of a concrete type instance, where UML generalizations are employed to model this type of variability.

The taxonomy of [8] is more focused on ICTs, and the type of variability that SimPL allows to model. Although this taxonomy can also be applied to CPSs product lines, we propose other variability points. In addition, we classify the variability of both, the cyber and the physical layer.

Our previous work [9], analyses some of the variability points of the plant model of highly configurable CPSs. However, the analysis performed in [9] is just related to the physical layer and it is not as extent as this one. The main focus of our previous study was the automatic generation of plant models to reduce simulation time and complexity of CPSs product lines using MATLAB/Simulink models.

A taxonomy of product line representations is provided in [10], where a taxonomy of variability concepts for different development artifacts in the context of product line engineering is provided. The taxonomy provided in [10] classifies concepts related to the development of software product lines (SPLs) whereas our taxonomy is focused on the variability points of a CPS product line.

MATLAB/Simulink is one of the most used modelling and simulation tools for CPSs. Weiland and Manhart classified different variability modelling concepts for this tool [11] and our previous work [12] compared three techniques for variability modelling and management in Simulink. Nevertheless, the classification of these papers was not oriented to the variability of CPSs, but to the different strategies for modelling variability in Simulink.

III. Motivating Examples

This section presents two CPSs product lines as a motivating examples. The first example is an industrial tank system with different functionalities and equipment. The second example is a quad-copter product line with features that permit the system to get configured to work in different contexts. These examples will be later used in the taxonomy to better explain which are the variability that can be identified.

A. Tank System Product Line

This is a simple example of an industrial CPS product line. The case study involves the control of the liquid level of a configurable tank system. The tank can manipulate two kind of liquids, either a chemical product or water. Depending on the selected liquid there will be some constraints (e.g., in case the liquid is chemical, a sensor that measures the acidity will be mandatory). Constraints between features and requirements are also included in this system, as appreciated in Table I. The variability of the system is summarized bellow:

- **Variability in Sensors:** The system can have three types of sensors. While the sensor for measuring the level of the liquid is mandatory, a sensor for measuring the temperature will be optional and the inclusion of a sensor for measuring the pH will depend on the selected product. In addition, each of the sensor types might use sensors from different vendors. To make the example simple, we have chosen just two vendors for each type of sensor.
- **Variability in Actuators:** The actuators in charge of draining and filling the tank with the specified liquid are mandatory. Nevertheless, an alarm that warns about the system's dangerous states is optional; moreover, if one the users or the context demands an alarm, two kind of alarms may be chosen: either an alarm with sound, an alarm with light or both.
- **Variability in the Physical System:** The physical system has two main variability points. The first one refers to the previously mentioned liquid to be manipulated, which can be either chemical or water. The second variability point corresponds to the shape of the tank; the tank can be either cylindrical or conical. As a consequence, the dynamics of the system might differ.
- **Variability in Software**: The software of the proposed system has two types of variability points. On the one hand, the variability in the functions, which in charge of controlling the level of the liquid. On the other hand, the variability in the software of the selected sensors and actuators. The former, refers mainly to functionality, e.g., depending on the liquid to be manipulated, the maximum level will change. The latter refers to the drivers of the selected components, e.g., depending on the selected sensor,

TABLE I
REQUIREMENTS OF THE ILLUSTRATIVE EXAMPLE

REQ_ID	Requirements description
REQ1	The level of the liquid will be established by a manual reference, with an error of 2%
REQ2	The needed time for achieving the established level are as much 5000 seconds
REQ3.1	If the liquid is water, the maximum level of the liquid will be the 90 % of the capacity of the tank
REQ3.2	If the liquid is chemical, the maximum level of the liquid will be 75 % of the capacity of the tank
REQ4	If the acidity is more than X, the liquid will be drained
REQ5.1	If the temperature of the water is less than 2 C, the liquid of the tank will be drained
REQ5.2	If the temperature of the water is less than 2 C and more than 50 C, the liquid of the tank will be drained
REQ6.1	If the level of the liquid is 85 %, the alarm will be turned on, until an operator turns it off
REQ6.2	If the level of the liquid is 70 %, the alarm will be turned on, until an operator turns it off

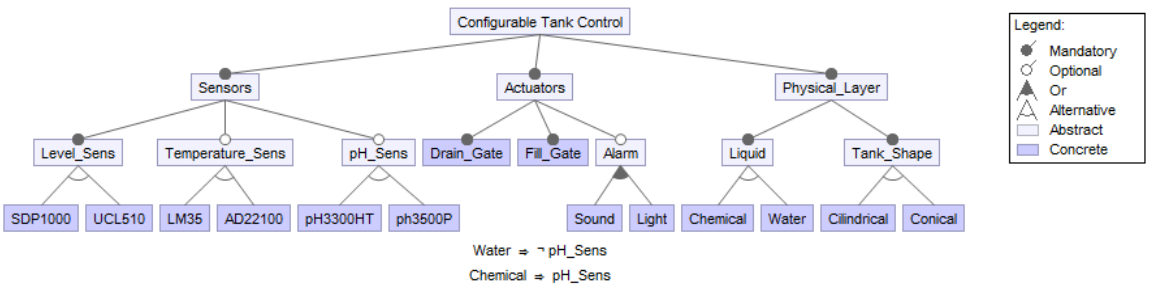

Fig. 2. Feature model of the tank control system

the driver of this sensor will need to be configured, otherwise, corrupted data might be sent to the software in charge of controlling the liquid level.

- **Variability in Requirements:** Based on the selected features, the requirements might differ from product to product. Table III-A describes the selected requirements for this motivating example. As it can be appreciated, requirement 3 is divided into REQ 3.1 and REQ 3.2 based on the chosen liquid. In addition, there are other requirements, e.g., REQ 4, which are product specific and might appear just if a feature is selected. In this case, the configurations manipulating chemical products will have to comply with REQ 4.

The variability of the tank product line can be depicted in Figure 2.

B. Quad-Copter Product Line

This example has been motivated by the huge range of applications and the big interest of the industry in the Unmanned Aerial Vehicles (UAVs). Although the quad-copters have several applications (e.g., aerial photography and video, rescue, cargo, inspections [1]), the features of each application are implemented independently, and to the best of our knowledge, there is no a quad-copter product line. In our case, we have taken a Quad-Copter named AR.Drone [13] and added variability to this system. We have managed the variability using the tool FeatureIDE [14] and modelled the system in Simulink taking as a base the model developed by Mosterman et al. [15]. In this case, the variability is more extent than in the example presented in Section III-A.

- **Variability in the Physical Layer:** With regard to the physical layer, there might also be several options. The system can have two kind of actuators: one of them is mandatory (the rotors) and the other is optional, a flying LED that turns on when the system is in the air; in the case of the rotors, the user might also choose high speed or low speed rotors. Regarding the sensors, four type of sensors can be used, two mandatory and two optional: the mandatory sensors include gyroscope and GPS; the optional sensors include obstacle sensor and sensor for the battery monitoring. In addition, the user will be able to choose among three sensor vendors when choosing the features of their drone. Finally, the user will be able to choose between two kind of batteries: a long or a short duration battery.
- **Variability in Software:** There are several variability points in the software of the system. For instance, the tracking system might differ from product to product, based on the user's needs: the UAV can go from coordinates to coordinates and wait there the time specified by the

[1] http://www.microdrones.com/en/applications/

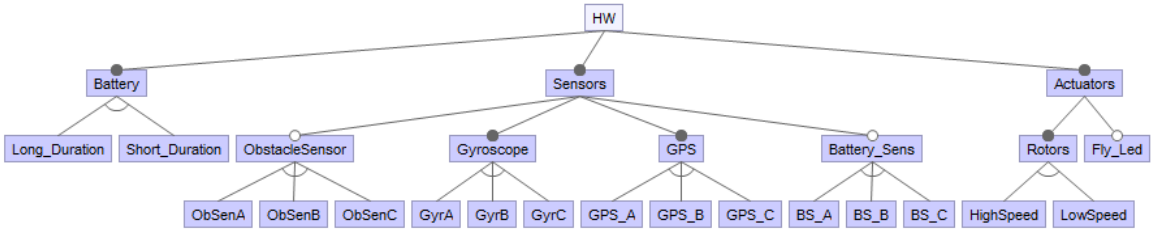

Fig. 3. Feature model of the hardware part of the UAV product line

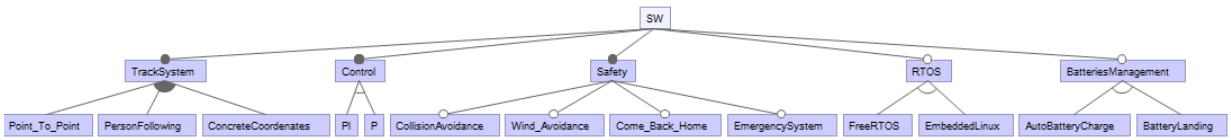

Fig. 4. Feature model of the software part of the UAV product line

user, the UAV can follow a person, or it can go to an specific point suggested by the user. The control position and control strategy can also differ, the user can choose a proportional control (P) or a proportional-integral control (PI)[2]. The user can also choose between two Real-Time Operating Systems (RTOS), in case the user or the application requires to use a RTOS. The software also integrates four optional safety functions: collision avoidance, wind avoidance, automatic return to home or an emergency system for landing in case of a detected anomaly. From the battery management perspective, the UAV can either find the closest battery station or in case there is no battery, it can land.

- **Variability in Requirements:** In this case, the system also will have variability in requirements. For instance, the requirements related to the speed of the UAV will be different depending on the type of selected rotors, i.e., higher speed will be allowed in high speed rotors than in low speed rotors. In addition there will be some requirements that will be removed from system variant to system variant. For instance, a configuration with the flying LED will have to comply with requirements related to this feature, whereas a configuration without the flying LED will not.

The variability of the UAV system product line is depicted in Figure 3 and 4.

[2]From the control engineering perspective, different kind of controllers might be used to control, for instance, the speed or the position of a system. In this kind of controllers, a reference system state value (e.g., speed of 1 m/s) is assigned and compared to the actual value of the system (e.g., 0 m/s), and depending on the difference between them, the controller will send a command value to the system. In this case two strategies have been proposed to control the different states of the system (proportional and proportional-integral)

IV. Taxonomy of Variability in Cyber-Physical Systems Product Lines

This section maps the identified variability in CPSs product lines into an taxonomy. Figure 5 depicts the highest level of the proposed taxonomy, and the parts where variability can be present in CPSs. The taxonomy classifies the variability corresponding to a CPS into the variability of the physical and the variability of the cyber layer. Regarding the physical layer, the analysed variability has been limited to the interaction with the cyber layer part (i.e., sensors and actuators) and to the physical process (e.g., mechanical elements). Regarding the cyber layer, the analysed variability has been limited to hardware related to the cyber part and the variability related to the software.

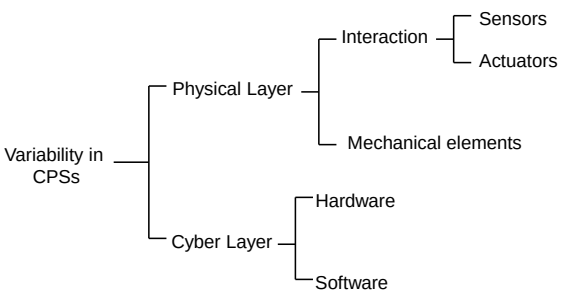

Fig. 5. High level overview of the developed taxonomy for CPSs product lines variability

A. Variability of the Physical Layer

In CPSs product lines, the variability of the physical layer is a critical part to give response to the needs that the different users demand. For instance, taking the motivating example explained in Section III-A, the shape of the tank might depend on the room in which it has to operate. In addition, there might be dependencies between the variation points related to the physical layer and the ones related to the software [2]. This means that the changes related to the physical layer of a CPSs could directly affect

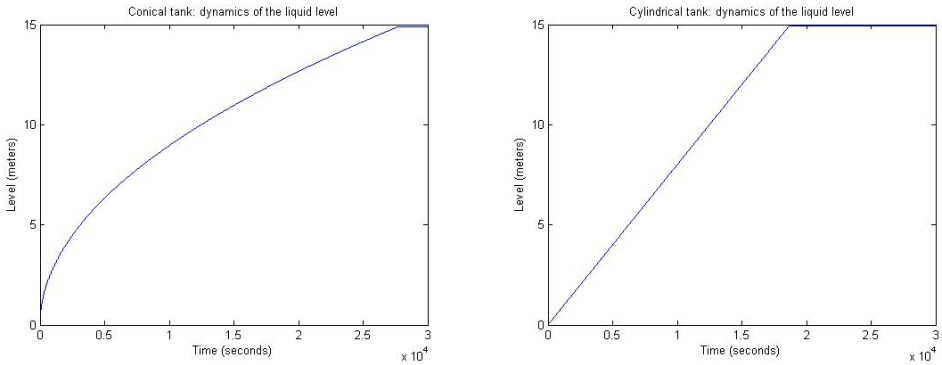

Fig. 6. Comparison of the variability in the physical layer and how it affects to the dynamics of the system in the case of the liquid level control of the tank system. It can be appreciated that the dynamic of the level increase in the conical tank is logarithmic whereas the dynamics of the level increase in the cylindrical tank is linear. In this case, the limit of the 5 meters is first crossed by the conical tank, but the reference level, i.e., 15 meters is first achieved by the cylindrical

to the software, where it might be configured to control specific physical properties; for instance, taking the example presented in Section III-B, the software that controls the flight LED should be configured, in case the customer selects having the flight LED in his product. The analysis of the physical layer of a CPS has been divided into the physical process and the interaction with the cyber layer, which is performed via sensors and actuators, as depicted in Figure 7.

- Mechanical elements: The mechanical elements are subjected to variability in terms of shape of the elements, size, material or weight. These variability points directly influence to the dynamics and kinematics of the system. For instance, taking the example presented in Section III-A, the tank might be conical or cylindrical; this has a direct influence on the dynamic of the liquid level, as depicted in Figure 6. The same can happen with the example of the quad-copter product line III-B, depending on the weight, the UAV will have more or less capabilities to achieve higher speeds.
- Interaction - Sensors: The sensors might also be exposed to variabilities. From configuration to configuration, sensors from different vendors might be used. Depending on the chosen sensor, the quality of the sensor might vary in terms of sensitivity, ranges or response time; the quality of a sensor often has a direct influence on the performance of the system. In addition, there are many complex sensors in the industry that are exposed to parametric values that are manually set up by the users. A sensor might also have cardinality variability, i.e., more than a sensor instance might be used [3]; this kind of variability might appear in safety-critical systems, where redundancy might be needed. The communication with the cyber layer of a sensor might also become a point influenced by variability; the sensor can send digital or analog data to the embedded system, or it might used field buses or serial interfaces. The variability in this field is also important from the performance point of view; depending on the communication system used, there might be delays that might lead the system not to meet performance related requirements.
- Interaction - Actuators: The actuators are also exposed to variability. The variability in the communication system is similar to the sensors, as depicted in Figure 7. The quality and characteristics of an actuator might also be exposed to variability; for instance, ranges and response time can vary depending on vendor and type of the actuator. Actuators might dissipate high energy, which might be a constraint in many configurations; as a result, the power consumption properties can also be exposed to variability and changes. As in the case of the sensors, actuators might also be exposed to cardinality variability; for instance, in the case of the tanks, there could be one or more drain gates, depending on the speed that is wanted the tank to be emptied.
- Cardinality: The whole physical layer can be exposed to cardinality. For instance, taking the tank product line example explained in Section III-A, there might be more than one tank. As a result, all the components for controlling the level of the liquid of each tank have to be replicated.

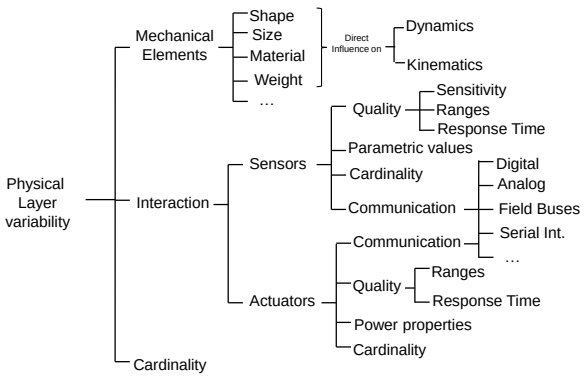

Fig. 7. Taxonomy of the variability in the Physical Layer for CPSs product lines

[3] for more information about this type of variability refer to [8]

B. Variability of the Cyber Layer

Variability of the cyber layer is also a critical part of a CPS product line. In case the variability is not considered in this layer, or a bad configuration of it, it might lead the system to be exposed to critical situations (e.g., the bad initialization of a driver to control a sensor from a different vendor might lead the sensor to send unreliable data [16]). In embedded and cyber-physical systems, variability can be handled in the software or the hardware part [17], thus the performed taxonomy classifies variability of the hardware and the variability of the software.

B.1 Variability of the Hardware

The hardware part of the cyber layer in a CPS can be considered the Electronic Control Unit (ECU) in charge of controlling the physical layer. When an engineer chooses the embedded system for a specific product, a wide range of ECUs properties might be chosen. In a CPS product line, hardware part can have many variability points in order to get adapted to the physical layer under control. Thus, variability in the hardware can be found in different places. Figure 8 classifies the different variability points we identified in the hardware related to the cyber layer of a CPS product line.

- Processor: Variability in the processor might be found in several places such as the architecture (the architecture of the processor can be Harvard or Von Neuman), number of bits (8, 16, 32 or 64), etc.
- Processor distribution: The ECU can have a single processor (i.e., monocore), or more than one processor (i.e., multi-core, many-core, multiprocessor, etc.). In case the hardware has a multicore architecture, variability can be found in the type of processor (i.e., hardcore or softcore), number of cores/processor that the ECU has, allocation of SW [4], communication system among cores, hypervisor, etc.
- Memories: The different type of memories of an embedded system can also be exposed to large variability. From configuration to configuration, the size of the memory, architecture or technology might vary. This type of variability should be taken into account as the memories of embedded systems form a critical part of the system.
- Electrical variability: The electrical properties of the hardware might vary from configuration to configuration. For instance, taking the example presented in Section III-B, the high speed rotors might need more power than the low speed rotors. As a consequence, there might be variabilities in the power supply, in the power that the inputs of the rotors or the outputs of the ECU can dissipate, etc.
- Interface variability: This variability refers to the ECU's connection with the physical world. In this case, variability might be found in the General Purpose Inputs and Outputs (GPIOs) in terms of number of inputs/outputs, electrical properties of the GPIOs; the Analog to Digital Converters (ADCs) or Digital to Analog Converters (DACs) might also be exposed to variability in terms of numbers of ports, electrical properties, sensitivity, etc. Other interfaces exposed to variability might be, for instance, serial interfaces or field-buses.

[4]This type of variability refers to the core or processor in which each piece of software is executed

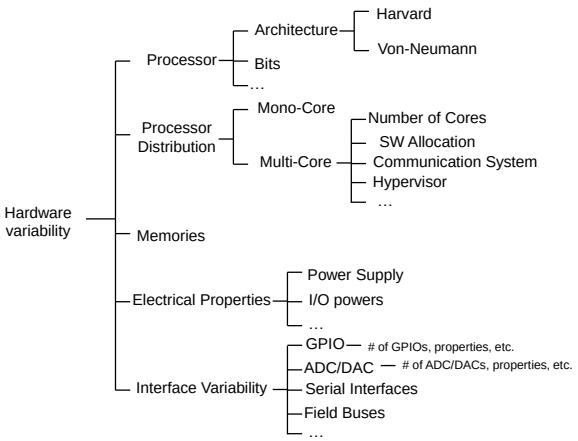

Fig. 8. Taxonomy of the variability in the Hardware for CPSs product lines

B.2 Variability of the Software

Variability in the software of CPSs product lines can be critical from different points of view: variability in software can directly affect in the functionality, safety, security or performance of the system. In addition, some parts of the software might have a direct dependency with the physical layer of the CPS [2], as well as with the hardware related to the cyber layer. As depicted in Figure 9, the developed taxonomy related to the software of a CPS product line, is divided into the following parts:

- Software Architecture: Variability is a relevant characteristic of the software architectures of CPSs and is explicitly reflected in and facilitated through the software architecture [18]. In these systems, the software architecture must consider the resource constraints when implementing variability [17]; for example, memory constraint might lead not to be feasible to implement all possible variants in the embedded software [17]. Bachmann et al., differentiates six sources of variation (which are later explained) in software architectures [19]: variation in function, variation in data, variation in control flow, variation in technology, variation in quality goals and variation in environment.
- Variation in Data: In an embedded software, there are many parametric values that can either be customized by the users or automatically configured based on the environment or components related to the CPS product line. The

parametrized values in a product line may exponentially grow the number of configurations that the system can be set to, which increments the complexity of the system.

- Variation in Functions: The software can vary from configuration to configuration in terms of functionality. For instance, in the example explained in Section III-A, depending on the type of liquid, i.e., water or chemical liquid, the functionality related to the control of the level varies. This kind of functionality must be managed by the software control system.
- Drivers: The drivers are also exposed to variability, specially when the different elements of the hardware related to the cyber layer are changed (e.g., communication buses, processors, etc.). The drivers in charge of controlling the data exchange between the sensors and the ECU and the ECU and actuators are also exposed to variability. In this sense, it is important to correctly initialize the drivers, otherwise corrupted data can be exchanged and the system might be exposed to dangerous situations.
- RTOS: A Real-Time Operating System (RTOS) that manages the execution of the different tasks is often mandatory in many systems. Nevertheless, the RTOS of a system might be also exposed to variability. Depending on the chosen RTOS, the scheduling strategy can be preemptive or non pre-emptive. In addition, there might be different parametrized values that might change the functionality of a RTOS.
- Tasks: The software tasks are also exposed to variability, and might differ from configuration to configuration. For instance, the number of tasks of a configuration might vary depending on the selected features. With regard to the deadlines, period and priorities, in a configuration a task can be critical, and thus, the deadline might be lower than in another configuration and the priority higher than in other configurations. The same might happen with their period and priorities.
- Communication with services: The software might vary depending on the type of communication that the CPS has with the cloud. For instance, depending on the chosen configuration, the communication protocol will vary, and as a consequence, the software will need to be adapted to these changes.
- Cardinality: The same piece of software might be instantiated once or more than once, for instance to control several physical processes or to obtain data from different sensors. Let's take as an example that for measuring the temperature of the liquid in the example presented in Section III-A two identical sensors are used; in this case, the driver in charge of obtaining the data from the sensors might be instantiated twice.

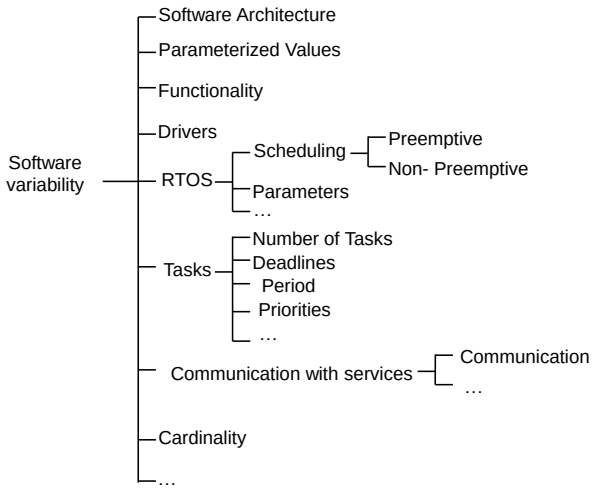

Fig. 9. Taxonomy of the variability in the software for CPSs product lines

V. Challenges in Cyber-Physical Systems Product Line Engineering

There are many challenges in the field of CPSs product line engineering. This section introduces some of them, which are mainly related to requirements engineering, variability management of CPSs product lines, dynamic CPS product lines, safety-critical CPSs and cooperative CPSs product lines.

One challenge in these kind of systems, as well as in Software Product Lines, is related to requirements engineering. The requirements also address variability, what means that from configuration to configuration, there are changes. Current requirement engineering tools might consider variability; however, traceability between these tools and the tools in charge of managing variability could be needed.

Variability management is the activity that analyses the variability of a system and classifies it in a tool. Feature modelling has been used as an standard when managing variability. According to Berger et al., feature models are used by more than 70 % of practitioners in industry [20]. Nevertheless, these tools are mainly oriented to software variability, and may have some limitations when managing variability of CPSs. Currently, there are other options for managing variability; for instance, SimPL [8] is a methodology to model product lines of families of ICSs, which have many similarities with CPSs product lines. An extension of feature models for CPSs product lines could help to manage several variability points in CPSs that in software are not considered. CPSs product lines might have several software parameters that can be changed from configuration to configuration, depending on the selected components (e.g., it might be needed to specified the number of sensors for measuring a certain physical parameter). There is a need to trace these software parameters with the selected components.

CPSs product lines usually bind their variability before getting configured. However, several domains might need their systems to be re-configured in runtime, making the systems more configurable,

as in the case of Dynamic Software Product Lines (DSPLs) [21]. The main differences among product lines and dynamic product lines are highlighted by Hinchey et al. in [22]. Dynamic CPSs product lines are systems in charge of adapting automatically their-self based on the user needs or the context changes in runtime; for instance, in the example presented in Section III-B, the UAV could be working in an indoor environment, and as a result, the functionalities related to wind avoidance would be deactivated. Nevertheless, if this system moves from the indoor environment to an outdoor environment, the wind avoidance functionality would be needed to be activated. These kind of systems provide several benefits, e.g., they can work in different environments, they can be automatically adapted to the different user needs, etc. Nevertheless, they address some challenges; for instance, there are many open issues related to the efficient management of variability across the different environments. In many cases, efficiently detecting the environment in which the dynamic CPS product line is working is a challenge, i.e., how can the system autonomously know the environment in which it is working? Among the differences between DSPLs and Dynamic CPSs is that whereas the former has to consider runtime variability just in the software, the latter has to consider runtime variability of software, hardware and physical elements.

Another interesting research area consists on the verification and validation stages of these systems. As these systems can get configured into millions of configurations, it is infeasible to test all of them. As a consequence, engineers cannot ensure that these systems will work in all the configuration. Efficiently choosing the configurations to test is one of the challenges. Many current practices (e.g., [23][24][25][26][27]) in product line engineering involve Combinatorial Interaction Testing (CIT), such as t-way testing, which are techniques to choose configurations where different features interact among them at least once. This way, it is possible to ensure that a configuration does not fail due to the interaction of two elements. However, this is not enough to ensure that the system will work in all the configurations.

CPSs combine different technologies from different domains. As a consequence, in the verification stages, the use of co-simulation tools is often needed. The use of different simulation tools address many challenges: clocks have to be synchronized, problems among the different data-types, and the increase of the time. In addition, CPSs product lines address an additional problem: all the models from the different simulation tools have to be efficiently configured to specific system variants. When there are several configurations to test, automation is needed in the configuration process, as manual configuration is non-systematic and error-prone.

VI. Conclusion

The variability of CPSs is becoming highly important when giving solution to different customers. As a result, CPSs product lines are increasing in different domains and industries. This paper presents a taxonomy of the variability that can be found in CPSs product lines. The paper might help to CPSs product line developers to consider the different variability points that the system they are designing can be exposed to. The developed taxonomy has been classified into the variability of the physical and cyber layer, which helps on the localization of the variability points. Nevertheless, we have identified some threads to validity of the developed taxonomy: as these systems are exposed to constant research, we might have been passing some variability points that should be also considered. In addition, we have extracted the variability of CPSs product line based on two examples from the state of the art, and thus, the analysis has not been exhaustive.

The research opportunities in the field of CPS product line engineering are large. The paper also highlights some challenges in this field. Among others, we have discussed issues related to requirements engineering, variability management or verification and validation stages.

VII. Acknowledgement

This work has been developed by the embedded systems group from Mondragon Goi Eskola Politeknikoa, supported by the Department of Education, Universities and Research of the Basque Government.

Referencias

[1] P. Derler, E. A. Lee, and A. Sangiovanni-Vincentelli, "Modeling cyber-physical systems," *Proceedings of the IEEE (special issue on CPS)*, vol. 100, no. 1, pp. 13 – 28, January 2011.

[2] J. Bosch, R. Capilla, and R. Hilliard, "Trends in systems and software variability," *IEEE Software*, vol. 32, no. 3, pp. 44–51, 2015. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/MS.2015.74

[3] J. V. Gurp, J. Bosch, and M. Svahnberg, "On the notion of variability in software product lines," in *Proceedings of the Working IEEE/IFIP Conference on Software Architecture*, ser. WICSA '01. Washington, DC, USA: IEEE Computer Society, 2001, pp. 45–54.

[4] S. Thiel and A. Hein, "Systematic integration of variability into product line architecture design," in *SPLC*, 2002, pp. 130–153.

[5] K. Mehner, M.-O. Reiser, and M. Weber, "Applying aspect-orientation techniques in automotive software product-line engineering," in *International Automotive Requirements Engineering Workshop*, Minneapolis, MN, United states, 2006. [Online]. Available: http://dx.doi.org/10.1109/AURE.2006.1

[6] T. Yue, S. Ali, and K. Nie, "Towards a search-based interactive configuration of cyber physical system product lines," in *ACM/IEEE 16th International Conference on Model Driven Engineering Languages and Systems, Poster*, 2013, pp. 71–75.

[7] K. Nie, T. Yue, S. Ali, L. Zhang, and Z. Fan, "Constraints: The core of supporting automated product configuration of cyber-physical systems," in *ACM/IEEE 16th International Conference on Model Driven Engineering Languages and Systems*, 2013, pp. 370–387.

[8] R. Behjati, T. Yue, L. Briand, and B. Selic, "Simpl: A product-line modeling methodology for families of integrated control systems," *Information and Software Tech-*

nology, vol. 55, no. 3, pp. 607 – 629, 2013, special Issue on Software Reuse and Product Lines.

[9] A. Arrieta, G. Sagardui, and L. Etxeberria, "Towards the automatic generation and management of plant models for the validation of highly configurable cyber-physical systems," in *Proceedings of 2014 IEEE 19th Conference on Emerging Technologies & Factory Automation (ETFA)*, 2014, pp. 1–8.

[10] F. Benduhn, "Representing variability in product lines: A survey of modeling and specification techniques," Master's thesis, University of Magdeburg - School of Computer Science, 2014.

[11] J. Weiland and P. Manhart, "A classification of modeling variability in simulink," in *Proceedings of the Eighth International Workshop on Variability Modelling of Software-Intensive Systems*, ser. VaMoS '14. New York, NY, USA: ACM, 2014, pp. 7:1–7:8.

[12] A. Arrieta, G. Sagardui, and L. Etxeberria, "A comparative on variability modelling and management approaches in simulink for embedded systems," in *V Jornadas de Computación Empotrada*, ser. JCE 2014, no. 26-33, 2014.

[13] P. jean Bristeau, F. Callou, D. Vissière, and N. Petit, "The navigation and control technology inside the ar.drone micro uav," 2011.

[14] T. Thuem, C. Kastner, F. Benduhn, J. Meinicke, G. Saake, and T. Leich, "Featureide: An extensible framework for feature-oriented software development," *Science of Computer Programming*, vol. 79, pp. 70 – 85, 2014.

[15] P. J. Mosterman, D. E. Sanabria, E. Bilgin, K. Zhang, and J. Zander, "Automating humanitarian missions with a heterogeneous fleet of vehicles," *Annual Reviews in Control*, vol. 38, no. 2, pp. 259–270, 2014.

[16] A. Heuer and K. Pohl, "Structuring variability in the context of embedded systems during software engineering," in *Proceedings of the Eighth International Workshop on Variability Modelling of Software-Intensive Systems*, ser. VaMoS '14. New York, NY, USA: ACM, 2014, pp. 21:1–21:8.

[17] M. Galster, P. Avgeriou, T. Männistö, and D. Weyns, "Variability in software architecture - state of the art," *Journal of Systems and Software*, vol. 91, no. 0, pp. 1 – 2, 2014.

[18] M. Galster and P. Avgeriou, "Handling variability in software architecture: Problems and implications," in *Proceedings of the 2011 Ninth Working IEEE/IFIP Conference on Software Architecture*, ser. WICSA '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 171–180. [Online]. Available: http://dx.doi.org/10.1109/WICSA.2011.30

[19] F. Bachmann and L. Bass, "Managing variability in software architectures," in *Proceedings of the 2001 Symposium on Software Reusability: Putting Software Reuse in Context*, ser. SSR '01. New York, NY, USA: ACM, 2001, pp. 126–132. [Online]. Available: http://doi.acm.org/10.1145/375212.375274

[20] T. Berger, R. Rublack, D. Nair, J. M. Atlee, M. Becker, K. Czarnecki, and A. Wasowski, "A survey of variability modeling in industrial practice," in *Variability Modelling of Software-intensive Systems (VaMoS)*, 2013, pp. 7:1–7:8.

[21] J. Bosch and R. Capilla, "Dynamic variability in software-intensive embedded system families," *Computer*, vol. 45, no. 10, pp. 28 – 35, 2012. [Online]. Available: http://dx.doi.org/10.1109/MC.2012.287

[22] M. Hinchey, S. Park, and K. Schmid, "Building dynamic software product lines," *Computer*, vol. 45, no. 10, pp. 22–26, 2012.

[23] R. Kuhn, R. Kacker, Y. Lei, and J. Hunter, "Combinatorial software testing," *Computer*, vol. 42, pp. 94–96, 2009.

[24] J. Petke, S. Yoo, M. B. Cohen, and M. Harman, "Efficiency and early fault detection with lower and higher strength combinatorial interaction testing," in *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, ser. ESEC/FSE 2013. New York, NY, USA: ACM, 2013, pp. 26–36. [Online]. Available: http://doi.acm.org/10.1145/2491411.2491436

[25] D. Marijan, A. Gotlieb, S. Sen, and A. Hervieu, "Practical pairwise testing for software product lines," in *Proceedings of the 17th International Software Product Line Conference*, ser. SPLC '13. New York, NY, USA: ACM, 2013, pp. 227–235. [Online]. Available: http://doi.acm.org/10.1145/2491627.2491646

[26] J. Ferrer, P. M. Kruse, F. Chicano, and E. Alba, "Evolutionary algorithm for prioritized pairwise test data generation," in *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '12. New York, NY, USA: ACM, 2012, pp. 1213–1220. [Online]. Available: http://doi.acm.org/10.1145/2330163.2330331

[27] M. B. Cohen, M. B. Dwyer, and J. Shi, "Constructing interaction test suites for highly-configurable systems in the presence of constraints: A greedy approach," *IEEE Trans. Softw. Eng.*, vol. 34, no. 5, pp. 633–650, 2008. [Online]. Available: http://dx.doi.org/10.1109/TSE.2008.50