Product Line Architecture Design of Software-Intensive Physical Protection Systems

ISSE 2020 - 6th IEEE International Symposium on Systems Engineering, Proceedings

Tekinerdogan, Bedir; Yakin, Iskender; Yagiz, Sevil; Ozcan, Kaan

https://doi.org/10.1109/ISSE49799.2020.9272239

Product Line Architecture Design of Software-Intensive Physical Protection Systems

Bedir Tekinerdoğan
Information Technology
Wageningen University & Research
Wageningen, The Netherlands
bedir.tekinerdogan@wur.nl

İskender Yakın
ASELSAN
Ankara, Turkey
iyakin@aselsan.com.tr

Sevil Yağız
ASELSAN
Ankara, Turkey
syagiz@aselsan.com.tr

Kaan Özcan
ASELSAN
Ankara, Turkey
mkozcan@aselsan.com.tr

***Abstract*— A physical protection system (PPS) integrates people, procedures, and equipment for the protection of assets or facilities against theft, sabotage, or other malevolent intruder attacks. Since PPSs are not radically different and share lots of commonalities, there is an important potential for reuse and herewith an opportunity to substantially reduce the cost and development time, and enhance the quality of the developed PPSs. In this paper, we report on the design of a product line architecture for a family of software-intensive PPSs. With this, we adopt a model-based systems engineering (MBSE) approach in which we focus on the architecture design of PPSs. We model the corresponding architecture view models for the PPS product line architecture and discuss the development of specific PPSs.**

Keywords— Physical Protection Systems, Systems Engineering, Product Line Engineering, Architecture Design

I. INTRODUCTION

Currently, many physical systems such as airports, rail transport, highways, hospitals, bridges, the electricity grid, dams, power plants, seaports, oil refineries, and water systems, require protection. In this context, a physical protection system (PPS) integrates people, procedures, and equipment for the protection of assets or facilities against theft, sabotage, or other malevolent intruder attacks [8]. A PPS provides *deterrence*, *detection*, *delay,* and *response* measures to protect the corresponding facility against an adversary's attempt to complete a malicious act. Designing effective PPSs requires a thorough analysis of the requirements and the resources to provide the protection that is needed. To guide the analysis and development process and properly realize these concerns, dedicated PPS methods have been proposed. A PPS method provides the step to assess the vulnerabilities of a facility together with the corresponding insider or outsider threats, and likewise provide adequate protection at critical points of the facility, by also effectively using the available resources.

This paper considers the context of an industrial company that is developing a broad range of PPSs for different kinds of facilities ranging over different business domains. Although each facility that needs to be protected is unique, the PPS share a broad range of features and the same gross-level structure, or architecture. Developing each PPS from scratch is a timely and costly activity. To reduce the time-to-market, reduce the cost of development, and increase the quality of the PPSs, a reuse-based development approach is an important and valuable alternative. In fact, reuse has already been an important goal in many industrial practices and has also been broadly addressed in the literature. Reuse can be done at a small scale for reusing small components, and in an ad hoc manner. Yet, it is now widely recognized that a large-scale and systematic reuse approach can achieve the most substantial benefit. This idea has culminated in the *product line engineering* (PLE) approach that is comprehensive and systematic and indeed focuses on exploiting reuse over the whole lifecycle process. A product line is defined as a set of systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way[3].

One of the critical assets in PLE is the product line architecture. The products in the product line typically share a common product line architecture, and it is thus essential to model and document the product line architecture properly. The product line architecture is needed to support the communication among stakeholders, guide the design decisions and the development of the artifacts in the life cycle. Explicitly modeling the artifacts in the lifecycle process is a recommended practice of the model-based systems engineering (MBSE), which focuses indeed on creating and exploiting domain models as the primary means of information exchange between engineers, rather than on document-based information exchange.

The architecture design, together with the rationale of the design decision, is described in the architecture documentation. Typically, architecture needs to be modeled for multiple stakeholders that have different concerns. Hence, a common practice for modeling the architecture is by modeling different architectural views, each of which is a representation of a set of system elements and relations associated with them to support a particular concern. Architectural views conform to well-defined viewpoints that represent the conventions for constructing and using a view. Having multiple views helps to separate the concerns and, as such, support the modeling, understanding, communication, and analysis of the software architecture for different stakeholders. In this paper, we focus on the design of a product line architecture of PPSs using multiple architecture views.

The remainder of the paper is organized as follows. In section 2, we present the background. Section 3 describes the product line scope for product line architecture. Section 4 describes the adopted architecture framework, the selected viewpoints, and the product line architecture views. Section 5 presents the application engineering process of the

product line architecture. Section 6 presents the related work, and finally, section 7 concludes the paper.

II. PRELIMINARIES

A. Physical Protection Systems

PPS design is a systematic approach that employs, in particular, a systems engineering approach [8][6]. Systems engineering is an interdisciplinary approach to translating users' needs into the realization of a system, its architecture, and design through an iterative process that results in an effective operational system [9][11].

Systems engineering can be applied for developing different systems, and as such, the method is agnostic to the domain of particular physical systems. Yet, to carefully address the specific concerns of PPSs dedicated PPS methods have been proposed [6][7][8]10][26]. Based on the identified PPS methods, we can state that the design of each PPS includes a predefined set of activities, including the determination of PPS objectives, the design and implementation of a PPS, the evaluation of the design, and if needed, a redesign or refinement of the system. Fig. 1 shows the top-level process of a PPS design method using the Business Process Modeling Notation (BPMN). The illustrated process can be applied to the case of a new PPS design, or an adaptation and enhancement of an existing PPS. The resulting PPS design should meet the earlier identified PPS objectives within the operational, safety, legal, and economic constraints of the facility.

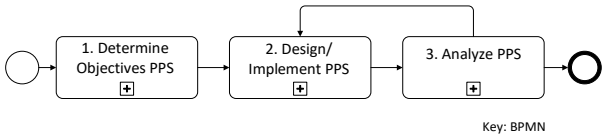

Fig. 1. Design and Evaluation Process for Physical Protection Systems

In this paper, we focus on the design process of the PPS, which focuses on its turn on the three key sub-activities, detection, delay, and response, as shown in Fig. 2.

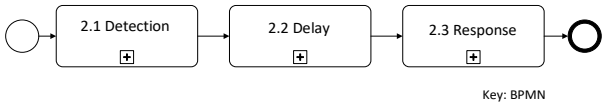

Fig. 2. PPS Design Process

Specific guidelines are included in the PPS design method. For example, PPS detection should be as far from the target as possible and delays near the target. Detection without assessment is not detection, and thus these two should be aligned appropriately. Another guideline considers the close association between response and response force communications. The PPS should be designed with effective communication calls for a response to provide the necessary response to neutralize the adversarial attacks. These and many other guidelines for designing effective PPSs are provided to ensure that the designer takes advantage of the strengths of each piece of equipment and uses equipment in combinations that complement each other and protect any weaknesses [8][6][26].

B. Product Line Engineering

In the previous sub-section, we have indicated that a systems engineering method is, in general, agnostic to the particular domain. In addition to this, the conventional systems engineering methods do not explicitly consider reuse, which is mostly an implicit concern. In particular, the notion of PLE is not well integrated, besides of recent studies that have addressed this issue [11]19][20].

In the traditional approach which does not adopt PLE, usually, a product portfolio can exist, but products are developed separately. This means that no PLE practices such as explicit commonality variability modeling, a product family architecture, and a shared asset base is adopted. The usually adopted process is shown in Fig. 3.

The primary activity in this process is to identify among all the already manufactured or delivered products, which is the closest one to the requirements and needs expressed formally (through a request for proposal or RFP) by a new potential customer [19]. Then the selected engineering artifacts of the previously existing product are reused and modified to completely fulfill the RFP requirements through multiple iterations.

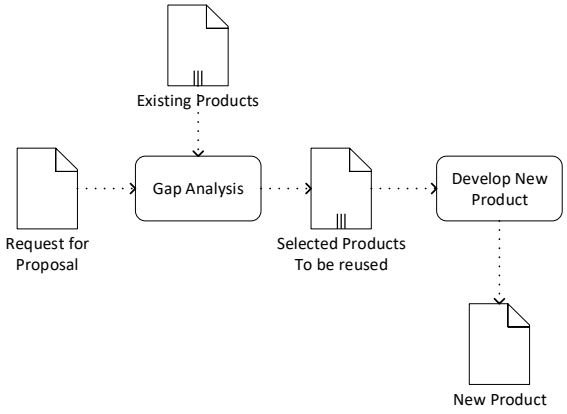

Fig. 3. Ad-hoc, non-PLE reuse strategy (adapted from: [19])

In case the products in the product portfolio share a substantial percentage of features, then the single system non-PLE based approach is considered less efficient. To exploit the potential for reuse, a systematic PLE method is then required. Compared to single system development, applying a product line engineering approach requires additional investments. The initial investment will result in a so-called *return on investment* (ROI). Altogether, PLE includes one or more of the benefits of large-scale productivity gains, decreased time to market, increased product quality, decreased product risk, increased market agility, increased customer satisfaction, more efficient use of human resources, ability to effect mass customization, ability to maintain a market presence, and ability to sustain unprecedented growth [1][13][15][16].

Different product line engineering processes have been proposed in the literature. As shown in Fig. 4., the common PLE process usually consists of two different activities [15][23][18]. In domain engineering, the focus is on developing reusable assets, while in the application engineering, these reusable assets are used to develop products. The terms domain engineering is also called

core/reusable asset development, while application engineering is sometimes termed *product development*.

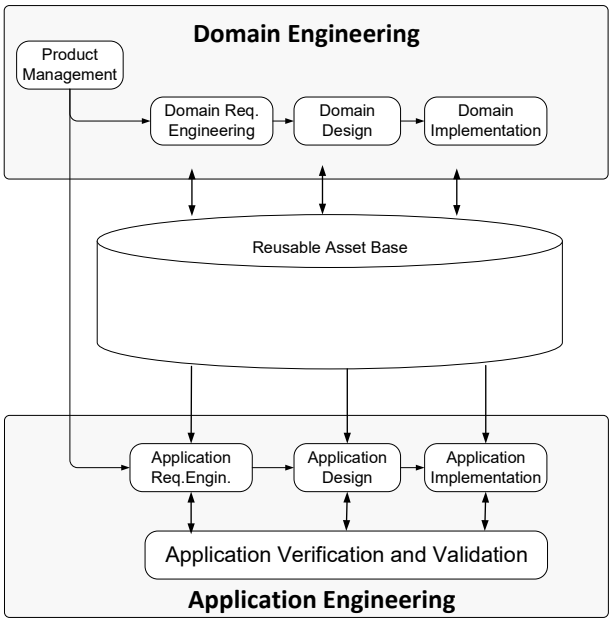

Fig. 4. SPLE Process

C. Architecture Design

Setting up a PLE for PPS would require realizing all the activities of the process, as shown in Fig. 4. In this paper, however, we do not elaborate on the entire product line engineering but instead focus on product line architecture. Since we are dealing with software-intensive PPS, we will focus on the design of software product line architecture.

It is generally accepted that software architecture design plays a fundamental role in coping with the inherent difficulties of the development of large-scale and complex software. Architectural drivers define the concerns of the stakeholders. A stakeholder is defined as an individual, team, or organization with interests in or concerns relative to a system. Each of the stakeholders' concerns impacts the early design decisions that the architect makes. A common practice is to model different "architectural views" for describing the design according to the stakeholders. An architecture view is a representation of a set of system elements and relations associated with them to support a particular concern. Having multiple views helps to separate the concerns and, as such, support the modeling, understanding, communication, and analysis of the software architecture for different stakeholders. Architectural views conform to viewpoints that represent the conventions for constructing and using such a representation. An *architecture framework* organizes and structures the proposed viewpoints.

A recent software architecture framework approach is the so-called Views and Beyond (V&B) approach [4][5]. The approach distinguishes three different categories of viewpoints or styles, including module, component-and-connector, and allocation styles. Module views deal with concerns related to implementation, such as decomposition and generalization. The C&C views deal with the interaction structure, such as data flow and message routing. The allocation views describe how software elements are allocated to the environment of the software system, such as hardware or development teams. A software architecture that addresses the concerns of specific stakeholders is here referred to as an *application architecture*. Application architectures can be viewed as specific implementations of product line architecture, which is the generic design for a family of systems.

III. Product Line Scoping

As stated before, the product line architecture defines the gross level structure for a family of products. For different product portfolios, we might end up with a different product line architecture. The products in the product line are not randomly selected. Typically, products are targeted that are likely to achieve the most economic benefit; and can be efficiently developed with the core assets either planned or in hand.

In this section, we focus on the product management activity of product line engineering, a sub-process of domain engineering for controlling the development, production, and marketing of the software product line and its applications. The input for product management consists of the company goals defined by top management. The purpose of product management is to make a significant contribution to entrepreneurial success by integrating the development, production, and marketing of products that meet customer needs. An essential task of product management is thus the management of a company's product portfolio, which is defined as the product types that are provided by the product line organization. Portfolio management is a dynamic decision process that continually checks and updates the portfolio according to the market and business requirements. The product management sub-process specifies a *product roadmap*, which outlines the estimated product line and defines the major common and variable features of all applications of the product line. The product roadmap is, on its turn, provided to the domain requirements engineering, which defines the product requirements based on which the product line architecture will be designed.

Product management employs *scoping* techniques to define what is within the scope of the product line and what is outside. The success of the product line architecture and herewith the overall product line process depends largely on the appropriate product line scope. If the scope is too large, the different product members will typically vary too much, and likewise, it will be more difficult to realize commonality and variability. The risk is then that the product line will collapse into the one-at-a-time product development effort. On the other hand, if the scope is too small, then the core assets might not be able to accommodate future growth, and the product line will stagnate. As a result, it will be difficult to realize economies of scope and achieve the expected return on investment. Scoping should be done carefully to mitigate these risks.

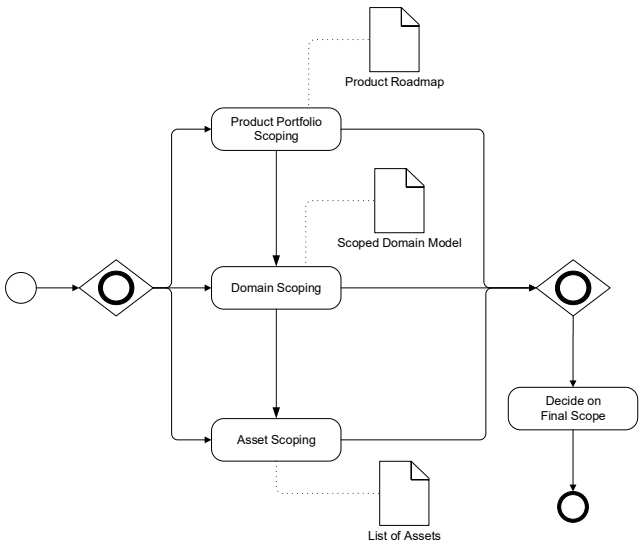

Fig. 5. Different Scoping Techniques and the output of each process

As shown in Fig. 5. three different forms of scoping can be distinguished [1]:

- *Product Portfolio Scoping*

The focus here is on identifying the products that need to be developed together with the features the products should provide. This process is usually driven from marketing aspects and likewise is a topic beyond system/software engineering aspects.

- *Domain Scoping*

Here, the boundaries for the domain under consideration that are supposed to be relevant to the product line are defined.

- *Asset Scoping*

This task aims at identifying the particular (implementation) components that should be developed in a reusable manner.

These three product line scoping processes build on each other in the sense that each of them refines the decisions made on the previous level. Correspondingly they can be connected with different stages in the product line development process: the product portfolio scoping relates to the overall set-up phase and is usually driven by a market study.

The overall product management process at the company is shown in Fig. 6. . Based on the company goals, the product roadmap is envisioned from which a product contract is derived. The product contract is then used to develop the system requirements and support the systems engineering process. Obviously, in this case, it is harder to identify the products beforehand. Thus the product line scoping will less rely on product portfolio scoping whereby an explicit product roadmap is prepared. Therefore, we have decided to aim to apply product line scoping using domain scoping and asset scoping. In essence, the product line scope includes the broad scope of the PPS domain with focusing on particular PPSs (such as railway PPS, highway PPS, etc.).

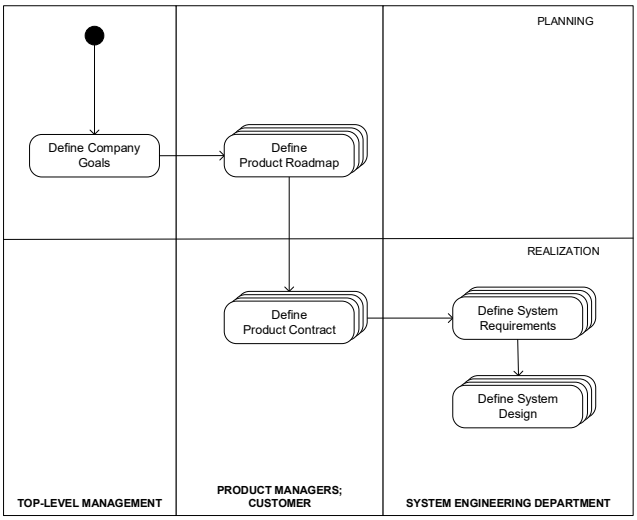

Fig. 6. Adopted Product Management Process

The term PPS by itself is also very broad and includes several sub-domains. Further, the relationship between domains and systems is often many-to-many. Systems do not necessarily cover a whole domain and may belong to several domains. For example, an application concerning distributed banking practices covers at least the following domains: banking practices, commercial bank information systems, workflow management, user interfaces, database management systems, and networking. Also, a domain can be used in several systems, and several domains may be scattered under one system.

Fig. 7. depicts the overall domain with the identified sub-domains that will be considered. In the figure, several PPS domains have been identified, but the adopted global scope is not only limited to these domains. This decision has, of course, also its implications for the product line architecture and the product portfolio. We will elaborate on this in the next paragraphs and subsections.

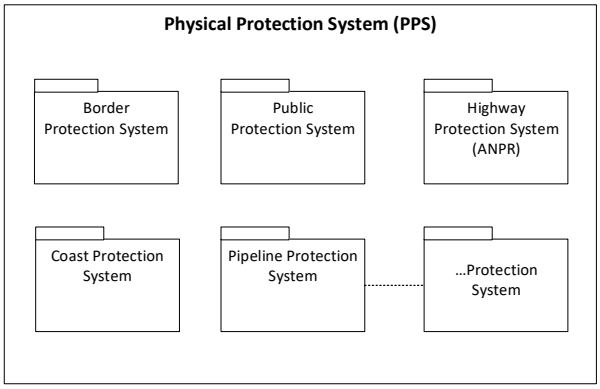

Fig. 7. Overall Scope for PPS product line

IV. PPS Product Line Architecture

Developing PPS product line architecture requires the selection of an architecture framework that is then used to design the PPS product line architecture, and based on this, the specific PPS application architectures. Fig. 8. shows the distinct architect roles in the overall product line engineering process.

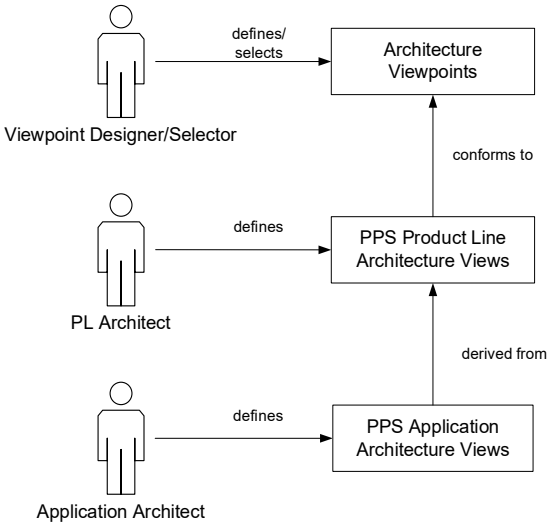

Fig. 8. Viewpoint definition and view development in the PLE process

The roles are the following:

- *PL Viewpoint Designer/Selector* – the person who designs or selects the architecture framework and viewpoints for the PL
- *PL Domain Architect* – the person who designs the product line architecture using the PL viewpoints that have been defined by PL Viewpoint Designer/Selector
- *PL Application Architect* - the person who develops the application architecture based on the PL Domain Architecture Views developed by PL Domain Architect.

Hence for designing the PL architecture, the first thing that is needed is the selection of PL Architecture Viewpoints. For designing the architecture, we have adopted the Views and Beyond (V&B) approach from which we have selected the decomposition view, layered view, aspects view, client-server view, publish-subscrivew view, and deployment views. In the following, we will discuss the selected views.

A. Decomposition View

The product line decomposition view for PPS is shown in Fig. 9. The decomposition view represents the overall decomposition of the system as a set of implementation units. The decomposition view includes all the modules that can be used in the design of a family of specific PPSs. It should be noted that the view has been shown for illustration purposes and (due to confidentiality reasons) should not be considered complete. Further, we have assumed that each module represents a software implementation unit, but on the other hand, it could also relate to hardware/system elements. We have not made a distinction between software and system elements. In our future work, we will explore this distinction in more detail.

Each of the modules of Fig. 9 can have their own sub-decomposition, which has also been documented. Due to space limitations, we can not show the complete decomposition but provide the decomposition view for PPS Detection (Fig. 10).

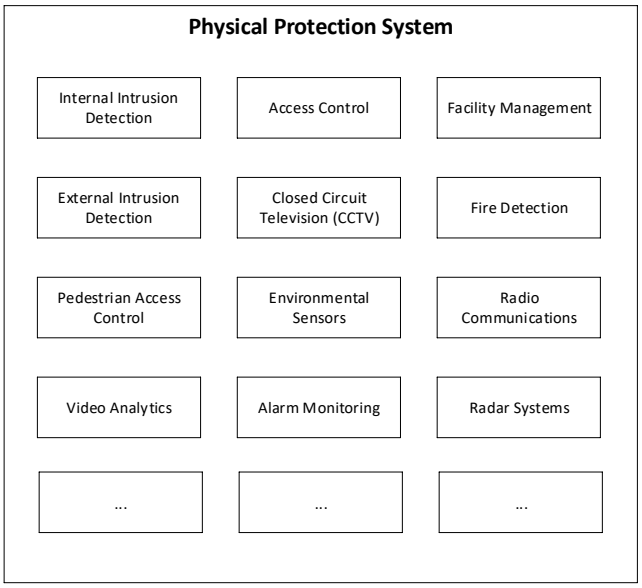

Fig. 9. Product Line Decomposition View for PPS

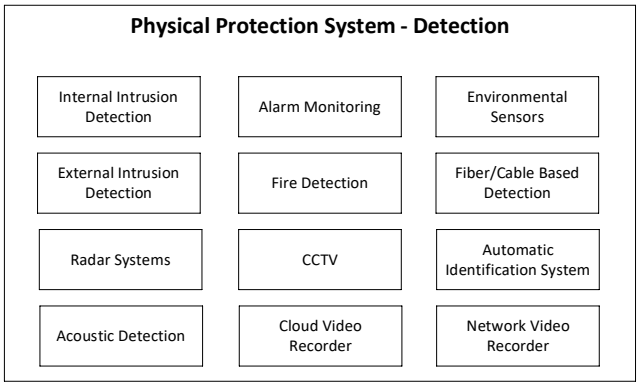

Fig. 10. Product Line Decomposition View of PPS Detection

B. Layered View

The layered view for the PPS is shown in Fig. 11. The lowest layer represents the *Facility Layer* that includes the physical elements. The *Protection Layer* accesses this facility layer by the use of sensors and actuators. The overall business logic with the key functionalities such as detection, alarm assessment, alarm classification, and response deployment and communication, is provided in the *Business Logic Layer.* This layer also includes the software modules for analytics and decision making for protecting the facility. Typically it includes the modules as defined in the decomposition view. The higher-level layer *Application UI Layer* defines the client applications that use the modules in the business logic layer. The architecture also includes a side layer *Configuration Layer,* which includes the modules for configuring the modules in the other layers for customizing the PPS for various domains.

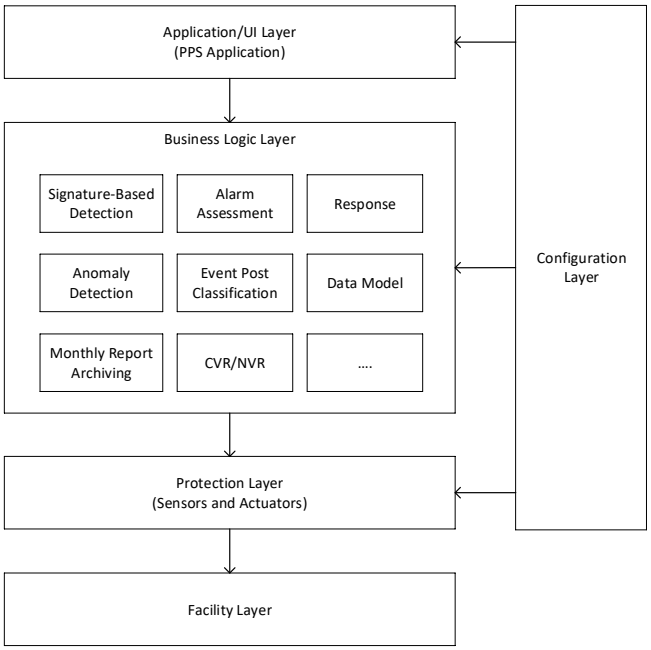

Fig. 11. Product Line Layered View for PPS

C. *Aspects View*

As stated before, we have chosen for a broad scope of the PPS product line. This required to design the architecture in a generic manner from which we can derive many concrete PPSs. However, to support the easy configuration and reuse, we have implemented an explicit configuration layer, as it is shown in the layered view in Fig. 11. Part of the configuration requires the implementation of aspects to cope with cross-cutting concerns such as logging and monitoring [2]. Fig. 12 shows the aspect views with an illustration of the key aspects. We have adopted the notations as proposed by the V&B approach [4].

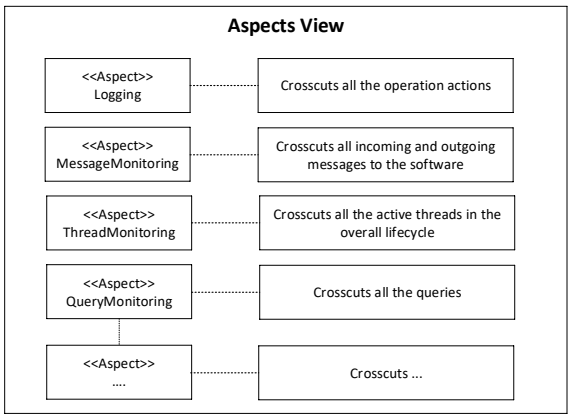

Fig. 12. Product Line Aspects View of PPS

D. *Client-Server View and Publish-Subscribe View*

A software-intensive PPS is digitally controlled and can consist of multiple components that are located on networked computers. Each PPS can be deployed on different platforms and thus a proper platform needs to be selected [21]. The computers can be connected by a local network and be physically close to each other, or they can be combined in a wide area network and geographically distant. The components in such a distributed system can communicate and coordinate their actions by passing messages to achieve a common goal. There are many alternatives for the message passing mechanism in distributed systems, including the request-reply pattern and publish-subscribe pattern [4]. The Client-Server pattern adopts a request-reply pattern in which we distinguish between server components that provide the services and client components that can access these services to synchronous, blocking service operations. Alternatively, in the publish-subscribe, so-called subscribers express their interest in an event, or a pattern of events, and are subsequently asynchronously notified of events generated by publishers. In our PPS product line architecture, both patterns are required and modeled using the so-called Client-Server View and Publish-Subscribe View. Fig. 13. shows the PPS Publish-Subscribe view [22].

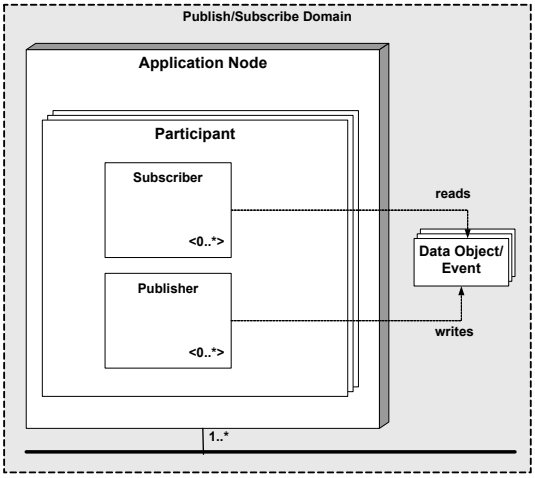

Fig. 13. Product Line Publish-Subscribe View for PPS

E. *Deployment View*

The deployment view considers the allocation of software modules to hardware nodes. Since we do not have a fixed configuration, we cannot directly provide a reference deployment architecture view. However, every PPS can be considered as a system-of-systems (SoS) configuration consisting of multiple systems, which, on their turn, can consist of sub-systems, which finally consists of components [25]. This generic SoS structure is shown in the left part of Fig. 14.

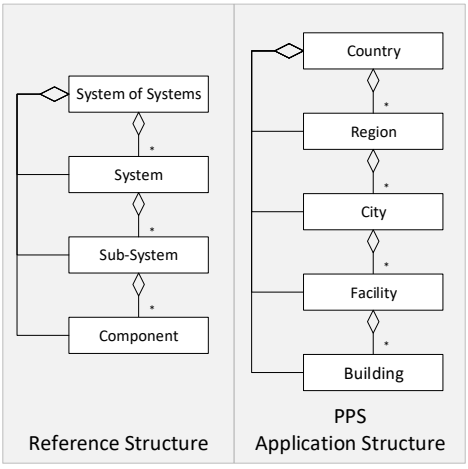

Fig. 14. Decomposition of PPS systems as System of Systems

An example configuration of PPS is shown in the right part of Fig. 14, which represents the configuration for the protection of buildings. A building is part of a facility, which is located in a city, which belongs to a region, which is part of a country. Even for this structure, we could derive multiple PPSs that focus on the protection of the mentioned system elements (building, facility, city, region, country).

Each element of an SoS forms a node to which the modules of the earlier views can be allocated to realize the necessary protection measures. For the example configuration of PPS, this implies that the layers and corresponding modules of Fig. 11 are allocated to the different SoS elements (e.g., Building, Facility, City, Region). This is typically done after the modules are configured for a particular context. In this way, we could have a different kind of system architectures ranging from local control whereby all the PPS business logic is deployed at the local entity (e.g., Building), to the centralized control whereby the PPS business logic is allocated at the central entity (e.g., Facility, City, Region, or even Country level). In the case of local control architecture, the sensors and actuators, as well as the controlled equipment, are within proximity, and the scope of each controller is limited to a specific system or subsystem. For the given example, this would mean that protection is localized for a building, and no further notification and protection measures are provided beyond the building boundary. Local PPS controllers can accept inputs from a supervisory controller to initiate, configure, or terminate locally-controlled automatic sequences, but the control action itself is determined in the local controller. The required operator interfaces and displays are also local.

In a centralized controlled PPS (e.g., building), the protection layer (sensors, actuators) within the facility are connected to a single controller or group of controllers located in a higher SoS element (e.g., City common control system). An example view of a centrally controlled PPS is shown in Fig. 15. Note that here we have just one level of control, that is, the City level. The control could be hierarchically allocated to the upper levels of the SOS. It can be observed that different PPS configurations can thus be defined with the architecture.

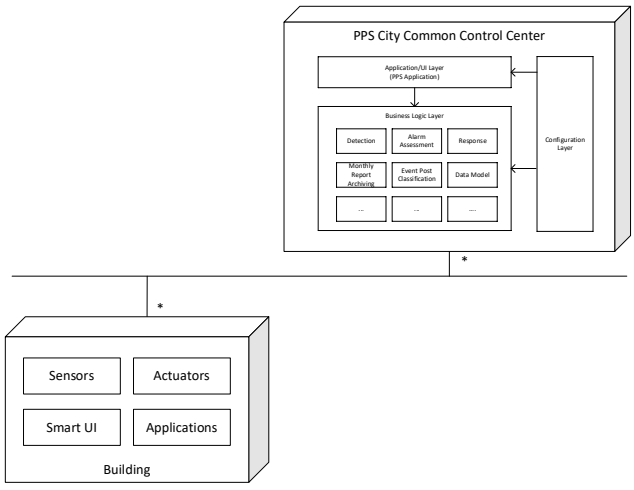

Fig. 15. Centrally Controlled PPS

V. Application Engineering

Once the domain assets have been developed, we can continue with the application engineering process in which a specific PPS is designed and implemented. For this, we will follow the conventional PLE process, as shown in Fig. 4. However, the traditional PPS process has been designed primarily from the perspective of a single system, and thus without reuse. Fig. **16.** shows the integration of PPS with the PLE process, which we will use to develop concrete PPSs. In essence, the output of the PPS domain engineering process are the family artifacts (such as family feature model and reference architecture), while the PPS application engineering focuses on reusing these artifacts to develop a particular PPS.

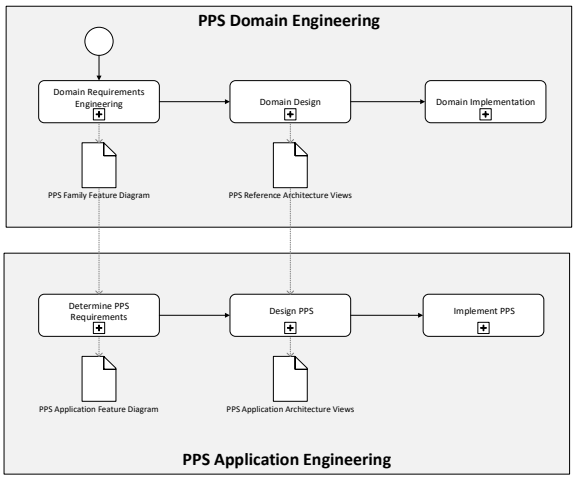

Fig. 16. PPS Design Process integrated with PLE Process

VI. Related Work

A comprehensive overview of PPS methods is provided in several textbooks and reports [6][8][10]. In our own study, we have provided an explicit process model that integrates the PLE process with the PPS process [24].

Several reference architectures are related to or quite close to PPS architectures. Software-Intensive PPSs adopt sensors and actuators to support physical protection. The detection is typically connected to a central server that assesses and classifies the alarm, and if needed, triggers the necessary response. In this sense, PPS uses the notion of Internet of Things (IoT) [12] that can be described as connecting devices to the internet. Various reference architectures have been provided for the IoT. In general, IoT architecture is integrated as a layered architecture, including device/datalink layer, network layer, session layer, and application layer [17]. The device layer includes the capabilities for the things in the network. The network layer provides functionality for networking connectivity and transport capabilities. The IoT layered architecture consists of functionality for generic support capabilities (such as data processing or data storage), and specific support capabilities for the particular applications. In addition to these, two side-car layers relating to the other four layers are provided. A security layer includes the functionality to ensure security over the layers. A management layer supports capabilities such as device management, local network topology management, and traffic and congestion management.

An intrusion detection system (IDS) monitors a network or systems for malicious activity or policy violations [1][14]. In case of an intrusion or violation activity, this is typically reported either to an administrator or collected centrally using a security information and event management system. The latter combines outputs from multiple sources and uses alarm filtering techniques to distinguish malicious activity from false alarms. In the PPS, the detection functionality largely uses the techniques as proposed by IDS and IDPS (intrusion detection and prevention system).

Supervisory control and data acquisition (SCADA) is a system of software and hardware elements to control industrial processes locally or at remote locations. The basic SCADA architecture begins with programmable logic controllers (PLCs) or remote terminal units (RTUs), which are microcomputers that communicate with an array of objects such as factory machines, sensors, HMIs, and end devices. The information from these objects is then routed to computers with SCADA software, which on its turn, processes and analyses the data to support the decision making of operators. The PPS is, in essence, a control system with various hierarchical levels of control.

VII. CONCLUSION

A physical protection system (PPS) is a systems engineering approach that integrates people, procedures, and equipment for the protection of assets or facilities against adversarial attacks. Dedicated PPS methods have been proposed that provide the steps for designing a PPS, including the critical measures of deterrence, detection, delay, and response. For developing multiple PPS, it pays off to adopt a product line engineering approach that supports the large scale systematic reuse and herewith includes several benefits such as reduced time-to-market, reduced cost of development, and increased quality. The products in the product line share a common product line architecture. The architecture, as such, is a key artifact that guides the development of systems, and supports the communication among stakeholders, guides the design decision, and supports the analysis of a system. Proper design and documentation of the architecture are crucial for the success of a product line.

In this paper, we have reported on the design of a product line architecture for PPS. Earlier PPS methods have focused on the overall process but did not consider the design and documentation of PPS architectures. In alignment with model-based systems engineering, we have adopted an architecture framework approach to model the PPS product line architecture using multiple views. Each PPS architecture view helps to focus on the system from a particular concern perspective.

To the best of our knowledge, this is the first time that a PPS architecture is documented using a model-based approach. In our future work, we will further develop the PPS product line using the presented product line architecture, thereby also addressing the other PLE lifecycle activities.

REFERENCES

[1] S. Axelsson. Intrusion Detection Systems: A Survey and Taxonomy. 2000.

[2] J. Bakker, B. Tekinerdogan, M. Aksit. Characterization of early aspects approaches. In Proc. of the Early Aspects Workshop at AOSD, The Netherlands, 2005.

[3] P. Clements, L. Northrop. Software Product Lines: Practices and Patterns. Boston, MA:Addison-Wesley, 2002.

[4] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, P. Merson, R. Nord, J. Stafford. Documenting Software Architectures: Views and Beyond. Second Edition. Addison-Wesley, 2010.

[5] E. Demirli, B. Tekinerdogan. Software language engineering of architectural viewpoints. in Proc. of European Conference on Software Architecture, p. 336-343, Springer, 2011.

[6] L. Fennelly. Effective Physical Security, Fifth Edition (5th. ed.). Butterworth-Heinemann, USA, 2016.

[7] ML. Garcia. Vulnerability assessment of physical protection systems. Amsterdam: Elsevier Butterworth-Heinemann; 2006.

[8] ML. Garcia. The design and evaluation of physical protection systems. 2nd ed. Amsterdam: Elsevier, 2008.

[9] Guide to the Systems Engineering Body of Knowledge (SEBoK), October 2016.

[10] IAEA, Handbook on the Physical Protection of Nuclear Material and Facilities, IAEA-TECDOC-127, March 2000.

[11] INCOSE Product Line Engineering International Working Group, http://www.incose.org/ChaptersGroups/WorkingGroups/analytic/product-lines, accessed October 2017.

[12] A. McEwen, H. Cassimally. Designing the Internet of Things. Wiley, 2014.

[13] J. D. McGregor, L. M. Northrop, S. Jarrad, and K. Pohl, Initiating software product lines, IEEE Software, 19(4), pp. 24–27, Jul. 2002.

[14] NIST (National Institute of Standards and Technology) – Guide to Intrusion Detection and Prevention Systems (IDPS), February 2007.

[15] K. Pohl, G. Böckle, F. van der Linden. Software Product Line Engineering – Foundations, Principles, and Techniques, Springer, 2005.

[16] K. Schmid, M. Verlage. The Economic Impact of Product Line Adoption and Evolution. IEEE Software, Vol. 19, No. 4, July/August 2002, 50-57.

[17] B. Tekinerdogan, Ö. Köksal. Pattern-Based Integration of Internet of Things Systems. In: Georgakopoulos D., Zhang LJ. (eds) Internet of Things – ICIOT 2018. Springer LNCS, vol 10972. 2018.

[18] B. Tekinerdogan, O. Özköse Erdogan, O. Aktug. Supporting Incremental Product Development using Multiple Product Line Architecture, International Journal of Knowledge and Systems Science (IJKSS) 5(4), 2014.

[19] B. Tekinerdogan, S. Duman, Ö. Gümüşay, and B. Durak. Devising Integrated Process Models for Systems Product Line Engineering. 2019 International Symposium on Systems Engineering (ISSE), Edinburgh, United Kingdom, 2019.

[20] B. Tekinerdogan, S. Duman, H. Caner, and B. Durak. Customizing a Feature Ontology for Product Line Engineering within a System-of-Systems Context. 2019 International Symposium on Systems Engineering (ISSE), Edinburgh, United Kingdom, 2019.

[21] B. Tekinerdogan., S. Bilir S., C. Abatlevi. Integrating Platform Selection Rules in the Model Driven Architecture Approach. In: Aßmann U., Aksit M., Rensink A. (eds) Model Driven Architecture.,Springer LNCS. vol 3599, Berlin, Heidelberg, 2005.

[22] B. Tekinerdogan, T. Celik. Architecting Feasible Deployment Alternatives for Publish-Subscribe Systems. International Journal of Computer & Software Engineering, Vol 2. No. 117, 2017.

[23] E. Tüzün, B. Tekinerdogan, M.E, Kalender, S. Bilgen. Empirical Evaluation of a Decision Support Model for Adopting Software Product Line Engineering, Information and Software Technology, Elsevier, Vol. 60, Pages 77–101, April 2015.

[24] B. Tekinerdogan, S. Yağız, K. Özcan, İskender Yakin. Integrated Process Model for Systems Product Line Engineering of Physical Protection Systems, in: Proc. Of 10th International Symposium on Business Modeling and Software Design, Springer, Berlin, 2020.

[25] B. Tekinerdogan. Multi-Dimensional Classification of System-of-Systems, in Proc. Of 14th Annual System of Systems Engineering Conference:"Internet of Things as System of Systems, p. 278-283, 2019.

[26] J.D. Williams, Physical Protection System Design and Evaluation, IAEA-CN-68/29, Vienna, 10–12 November 1997.