A Secure Development Decomposition Argument Pattern for Structured Assurance Case Models

JASON JASKOLKA, Carleton University
BRAHIM HAMID, IRIT, University of Toulouse
ALVI JAWAD, Carleton University
JOE SAMUEL, Carleton University

An important aspect of security assurance, certification, and evaluation is related to the secure development of systems and their components. Demonstrating compliance with secure development methodologies is often a cornerstone for an effective security assurance argument for complex and critical systems. In this paper, we propose a security assurance argument pattern called Secure Development Decomposition. This pattern is derived from the secure development processes and procedures required for developing a secure system. As a result, the Secure Development Decomposition pattern can be instantiated in the context of a domain-relevant secure development methodology or security standard to demonstrate compliance and adequate security considerations throughout system development as part of a structured security assurance case. To illustrate its applicability, we use an example from the automotive domain to show how to use the pattern to demonstrate compliance with a relevant security standard and its prescribed secure development methodology.

Categories and Subject Descriptors: D.3.3 [**Language Constructs and Features**] Patterns; K.6.5 [**Security and Protection**]

General Terms: Documentation; Management; Security

Additional Key Words and Phrases: argument pattern, assurance case, compliance, security, standards, secure development

ACM Reference Format:

Jaskolka, J., Hamid, B., Jawad, A. and Samuel, J. 2021. A Secure Development Decomposition Argument Pattern for Structured Assurance Case Models. HILLSIDE Proc. of Conf. on Pattern Lang. of Prog. 1 (August 2021), 11 pages.

1. INTRODUCTION

Security standards and guidelines related to secure system development and management play a critical role in security assurance, certification, and evaluation [Jaskolka 2020]. Effective security assurance often demands compliance with domain-relevant secure development methodologies and/or security standards. Documenting and demonstrating such compliance is challenging as we are often dealing with complex systems with many requirements and intertwined processes. This makes it difficult to gather sufficient evidence to support assurance claims and to support traceability for compliance checks.

Structured assurance case models are a popular tool for developing an evidence-based approach for arguing, assessing, and assuring that a specified set of critical claims regarding a system's properties are adequately justified for a given application in a given environment [GSN Working Group 2018]. Using patterns to construct

This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) grant RGPIN-2019-06306.
Author's address: J. Jaskolka, A. Jawad and J. Samuel, Department of Systems and Computer Engineering, 1125 Colonel By Drive, Ottawa ON K1S 5B6, Canada; email: {jason.jaskolka,alvi.jawad,joe.samuel}@carleton.ca; B. Hamid, IRIT, University of Toulouse, 118 Route de Narbonne, 31062 Toulouse Cedex 9, France; email: brahim.hamid@irit.fr

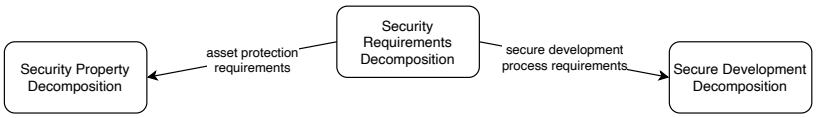

Fig. 1. Argument patterns for structured security assurance case models

assurance case models enables us to maintain the structure of the argument by generalizing specific details so that they can be instantiated in multiple different situations as appropriate [Alexander et al. 2011]. This helps to facilitate reuse and incremental refinement of the assurance case. In this paper, we present a security assurance argument pattern called *Secure Development Decomposition*. This pattern is one of several (see Fig. 1) that form the primary building blocks of a structured security assurance case model. The intents of these patterns are:

—*Security Requirements Decomposition*: Provides a claim decomposition to argue that if the system under consideration satisfies its asset protection requirements and secure development process requirements as prescribed by a relevant security standard, then it is adequately secure and compliant with the standard.

—*Security Property Decomposition*: Provides a claim decomposition to argue that if the system under consideration exhibits each of its security properties (e.g., confidentiality, integrity, availability, authenticity, accountability, etc.), then it satisfies all of its asset protection requirements [Jaskolka et al. 2021].

—*Secure Development Decomposition*: Provides a claim decomposition to argue that if the system under consideration is developed using an adequate set of secure development processes that are defined and implemented in accordance with a relevant secure development methodology and/or security standard, then it satisfies all of its secure development process requirements.

We use the pattern template provided by Kelly and McDermid [1997] for describing assurance case patterns to describe the Secure Development Decomposition pattern. The template is adapted from the Gang of Four template [Gamma et al. 1994]. It uses the Goal Structuring Notation (GSN) [GSN Working Group 2018] to graphically describe the structural details of the pattern, and rather than sample source code, it includes sample text for the eventual assurance case. The proposed pattern supports compliance for secure development processes by providing a claim decomposition derived from a variety of activities and tasks that must be defined and implemented as prescribed by existing secure development methodologies or security standards from different application domains. The pattern is intended for those certifying an already developed system at certification time and can enable system manufacturers, evaluators, regulators, and certifiers, as well as researchers and students, to formulate and evaluate well-reasoned and compelling arguments supporting the development of secure systems in compliance with a secure development methodology or standard. We show how to apply the proposed pattern for a system from the automotive domain to demonstrate compliance with ISO/SAE 21434 [2020].

2. BACKGROUND

Secure development methodologies refer to the set of processes and activities that are incorporated and undertaken as part of a typical system development lifecycle. Examples of such methodologies include UMLsec [Jürjens 2002], SecureUML [Lodderstedt et al. 2002], and ASE [Uzunov et al. 2015]. Secure development methodologies are process-centric, and aim to improve the security attributes of a given system [Uzunov et al. 2015]. They can be the products of research, or they can be included or prescribed in security standards. Compliance with secure development methodologies is an important aspect for an effective security assurance argument that can be given in the form of assurance case models.

Assurance case models provide a reasoned and compelling argument formed by combinations of structured claim decompositions, supported by a body of evidence, that a system, service, or organization will operate as intended for a defined application in a defined environment [GSN Working Group 2018]. Assurance case argument patterns have been proposed to support the modular and incremental development of large and complex

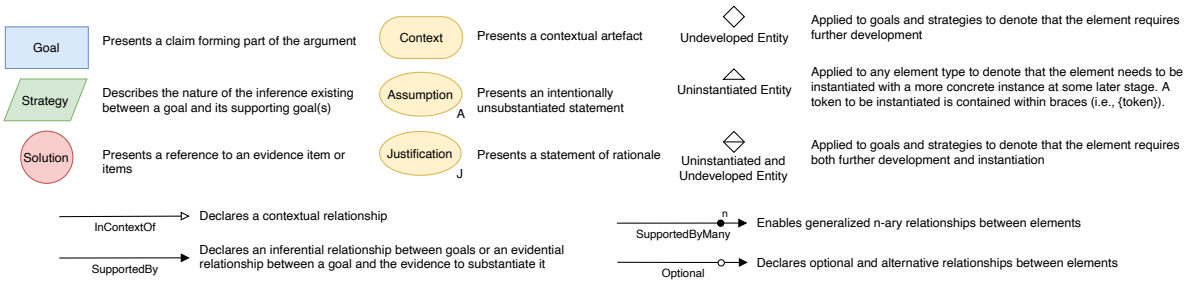

Fig. 2. Principal elements of the GSN Pattern Notation

assurance cases [Firestone and Cohen 2018; Fong and Wheeler 2018; Hawkins and Kelly 2013; Kelly 1998; Warg and Skoglund 2019; Wassyng et al. 2015]. The idea with argument patterns is that an assurance case (whether for safety, security, or another concern) can be developed by assembling and reusing a collection of successful argument structures. This idea has been compared to software design patterns [Gamma et al. 1994] and architecture patterns [Garlan and Shaw 1993]. The aim of using argument patterns is to address the issues of ad hoc reuse of components and structures from previously constructed assurance cases, such as inappropriate reuse of artefacts, lack of consistency, lack of traceability, and loss of knowledge [Kelly and McDermid 1997].

Assurance cases and argument patterns must be clearly documented to enable them to be developed, applied, discussed, challenged, presented, and reviewed among stakeholders, and maintained throughout the product lifecycle. There are a number of notations and existing tools for developing and documenting assurance cases, and the most popular of these is a graphical notation that can be used to visualize arguments called Goal Structuring Notation (GSN) [Kelly 1998]. Presenting and structuring arguments using GSN can help provide assurance of critical properties, such as those related to safety, security, and resilience of systems, services, or organizations. This paper adopts the GSN Pattern Notation to visualize and present the argument structure. The GSN Pattern Notation is an extension of core GSN and a summary of its graphical elements is provided in Fig. 2. The description of each of these elements is extracted from [GSN Working Group 2018].

3. SECURE DEVELOPMENT DECOMPOSITION PATTERN

We present a security assurance argument pattern called *Secure Development Decomposition*. The pattern provides a claim decomposition to demonstrate that the complex secure development processes and procedures required by many security standards and methodologies have been adequately performed and are supported by evidential artefacts.

3.1 Derivation of the Pattern

Evidence that secure development processes and all the associated activities and tasks have been adequately defined, executed, and managed in compliance with secure development methodologies or standards relevant to the specific domain context forms an important part of a security assurance argument. Developers need to provide sufficient evidence that they have defined and implemented adequate processes and procedures for developing a secure system and certifiers need to evaluate the provided evidence and accompanying argumentation to determine compliance to gain confidence that the system has been developed in a adequate manner with consideration for security concerns at all stages of the development lifecycle. NIST Special Publication 800-160 [Ross et al. 2016] is a well-known resource describing the wide-variety of activities and tasks that are performed as a part of a security engineering practice throughout the system development lifecycle. However, depending on the system context and the relevant security standard or development methodology for which the compliance is required, the specific forms of evidence to support security claims may vary. As a result, a claim

decomposition demonstrating that secure development process requirements that can be instantiated with respect to specific systems and standards and methodologies is a useful component for constructing security assurance cases and lends itself well to a pattern-based approach.

Deriving an argument pattern is not so much an exercise in development or invention, but rather one of discovery. As the primary goal of the proposed argument pattern is to support compliance, we studied existing secure development methodologies such as UMLsec [Jürjens 2002], SecureUML [Lodderstedt et al. 2002], and ASE [Uzunov et al. 2015], as well as security standards from several prominent domains for which high-levels of security assurance is required. This included ISO/SAE 21434 [2020] for automotive security, CNSS Policy 12 [2012] for space systems security, ISO 13485 [2016] and ISO 14971 [2019] for medical device security and risk management, and ISO/IEC 27019 [2017] and NISTIR 7628 Revision 1 [2014] for smart grid security, just to name a few. We also considered more general standards such as ISO 27001 [2018b], the Common Criteria [2017], the SafSec Methodology: Standard 3.1 [Dobbing and Lautieri 2006], and DoD Instruction 8510.01, Risk Management Framework [USA Department of Defense 2014]. By studying the requirements in these standards and methodologies, we identified and synthesized the commonalities and variabilities in the specific activities and tasks that ought to be part of the secure system development processes. This helped to inform and identify the candidates for the claim decomposition.

3.2 Pattern Description

3.2.1 *Pattern Name.* Secure Development Decomposition

3.2.2 *Intent.* The pattern provides a claim decomposition to argue that a system satisfies its secure development process requirements if an adequate set of secure development processes have been defined and implemented in accordance with a relevant secure development methodology or security standard throughout the development of the system.

3.2.3 *Motivation.* A rigorous, systematic, and well-documented process is critical to providing the required assurance for a developed system as it can help generate, gather, and manage the supporting evidence needed for security assurance and compliance [Jaskolka 2020]. Many security standards and methodologies require the satisfaction of a set of secure development *process requirements*. As a result, the pattern aims to provide support that the process requirements are satisfied. These process requirements are not necessarily tied to the specific functionality of the systems and components, but rather to the preparedness to deal with security and resilience challenges throughout the system's lifespan. These requirements are concerned with the resilient and robust design of critical core functionalities and infrastructures, conformance to implementation standards, adequate testing, verification and validation, and suitable specification and documentation for all system aspects, including the physical systems, information systems, and networks. They are also concerned with the development of personnel security, awareness and training, and the development of pre-defined responses based on detailed threat modelling and vulnerability analyses to ensure that operations and services continue to be provided in the case of a potential disruption or attack.

3.2.4 *Applicability.* The pattern should be applied when compliance is required in the secure development processes for the given system. The pattern assumes that the relevant standard or development methodology provides sufficient context for its instantiation. It is important to note that the decomposition and goals of the pattern refer to the processes by which the product is developed and not the product itself.

3.2.5 *Pattern Structure.* The decomposition strategy is captured by the argument pattern shown in Fig. 3.

3.2.6 *Participants.* The pattern participants are as follows:

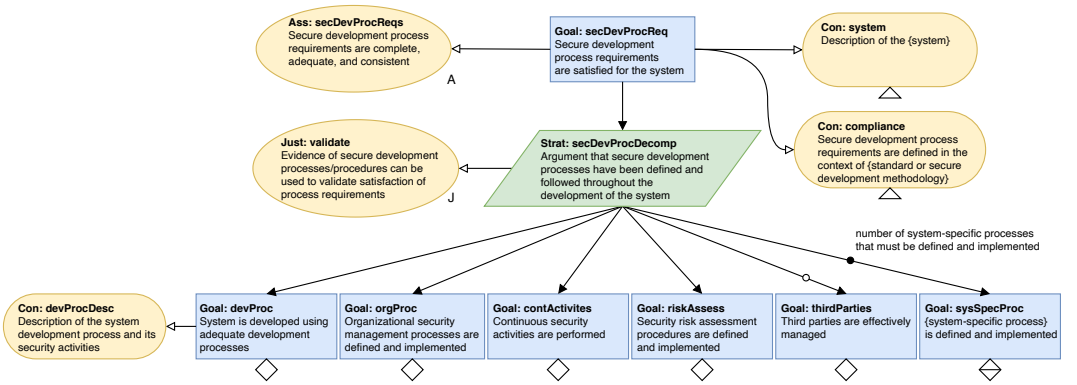

Fig. 3. Secure Development Decomposition Pattern

—Goal: secDevProcReq: The overall objective of the argument; to support the claim that the secure development process requirements are satisfied in compliance with a security standard or development methodology for a given system.

—Con: system: The description of the system for which the security assurance case is being developed.

—Con: compliance: The secure development process requirements are provided in the context of a {standard or secure development methodology} which defines the specific requirements that must be satisfied by the secure system development process. The {standard or secure development methodology} parameter needs to be instantiated with respect to the specific system under consideration and the regulatory and compliance requirements associated with the system's application domain.

—Ass: secDevProcReqs: The secure development process requirements can only be satisfied if they are complete, adequate, and consistent. This determination must be made as part of the system requirements engineering process, and therefore, this assumption is acceptable as long as the completeness, adequacy, and consistency of the security process requirements are demonstrated elsewhere at the system-level. If an argument to support this assumption exists, then that argument can be linked here in place of the assumption.

—Strat: secDevProcDecomp: The decomposition strategy arguing that the applicable secure development processes are developed and performed throughout the system development.

—Just: validate: The evidence that the applicable secure development processes are defined and performed throughout the system development can be traced to the secure development process requirements and can be used to validate the satisfaction of the requirements.

—Goal: devProc: This goal states that the system is developed using adequate development processes by demonstrating that security activities are performed throughout the system development lifecycle.

—Con: devProcDesc: The system development processes are provided in the context of the specific security activities that are tailored for the given system.

—Goal: orgProc: This goal states that organizational security management processes such as governance, tool management, information sharing, incident response, etc., are defined and implemented.

—Goal: contActivities: This goal states that continuous security activities such as monitoring, event assessment, and vulnerability analysis and management are performed regularly.

—Goal: riskAssess: This goal states that security risk management procedures including asset identification, threat scenario identification, attack analysis, and risk treatment decision-making processes and procedures are defined and implemented.

—Goal: thirdParties: This goal states that, if applicable, third parties involved in the system development process are appropriately managed from a security perspective.

—Goal: sysSpecProc: An instance of this goal is created for each {system-specific process} and states that such processes are defined and implemented. The required {system-specific process} is determined from the specific {standard} for which compliance is required.

3.2.7 *Collaborations.* The specific system and standard or secure development methodology are required to set the overall context for the argument. Therefore Con: compliance and Con: system are shared with all of the sub-goals in the decomposition. Strat: secDevProcDecomp is based on the argument that sufficient evidence demonstrating that the decomposed secure development processes and procedures were defined and performed throughout the system development will support the satisfaction of the secure development process requirements. As a result, each sub-goal (with the possible exception of Goal: thirdParties in the case that third parties are not involved in the development process) is required to support the top-level goal.

3.2.8 *Consequences.* The pattern enables a separation of concerns in the development of the assurance argument. Decomposition of the secure development requirements supports argumentation over the secure development activities and tasks prescribed by a relevant security standard, one at a time. This enables the explicit and more complete consideration of the security compliance goals and process requirements for the system under consideration. However, after instantiating the pattern, a number of undeveloped goals will remain. Specifically, Goal: orgProc, Goal: contActivities, Goal: riskAssess, Goal: devProc, Goal: thirdParties, and Goal: sysSpecProc will require further decomposition and support. In some cases, the decomposition of these goals can be accomplished by adopting other argument patterns recast in the context of security.

3.2.9 *Implementation.* First, Con: system is instantiated with a description of the system being developed. Second, Con: compliance is instantiated with the standard or secure development methodology for which compliance is required. Next, the context Con: devProcDesc is instantiated with the development process and security activities adopted for the development of the system. Then, with respect to Con: system and Con: compliance the required number of instances of Goal: sysSpecProc are created based on the system-specific processes required by the standard. Finally, the arguments for the remaining goals are elaborated with further goal decomposition and/or solutions to support these goals and demonstrate compliance with the defined standard.

Possible Pitfalls when implementing the pattern include:

—Failing to support Ass: secDevProcReqs. This can result in an incomplete or inadequate argument.

—Not having a clear description, or providing complete coverage, of system-specific processes (instances of Goal: sysSpecProc) as part of the decomposition. This can result in leaving out important processes that must be adequately undertaken and lead to an incomplete argument.

—Expressing Con: devProcDesc in a way that is difficult to apply to the information to further decompose Goal: devProc. This can result in difficulty in determining the granularity by which Goal: devProc is decomposed so that sufficient evidence can be generated to fully support the goal.

3.2.10 *Sample Text.* The text below may be used to describe the pattern in the final security assurance case:

—The Secure Development Decomposition pattern argues that the system has been developed using adequate secure development processes. This is equivalent to showing that the development of the system has considered security concerns at each state of its development lifecycle and that the processes have been applied correctly.

—The Secure Development Decomposition pattern enables explicit consideration of the assurance requirements for the system and its development processes and procedures.

3.2.11 *Known Uses.* A similar claim decomposition has been used in security assurance arguments for:

—General dependability concerns in automotive, railway, aerospace, and industrial automation [Vallée et al. 2018]: The AMASS project has contributed several assurance case arguments for multi-concern assurance cases covering the interplay of safety and security.

—Advanced metering infrastructure [Jaskolka 2018]: A security assurance case template is provided with a claim decomposition covering claims the about process requirements related to the adoption of adequate development processes, preparation of training and policy plans, preparation of incident response plans, and effective management of third-party suppliers.

—Connected Automated Driving Systems [Warg and Skoglund 2019]: An assurance argument showing that generic quality attributes (that can be instantiated) are adequate for a system is provided with considerations to process requirements such as risk mitigation, management, and maintenance.

—Best Practices Badge Application (BadgeApp) [Fong and Wheeler 2018]: An assurance argument is provided demonstrating that security is implemented by software life cycle processes including acquisition and supply processes, life cycle model management, infrastructure management, and technical management processes such as project planning, risk management, and configuration management.

3.2.12 *Related Patterns.* The pattern has several undeveloped goals which can be the overall objective of other argument patterns. For example, Goal: riskAssess may instantiate the ALARP (As Low As Reasonably Practical) pattern [Kelly 1999] considering security threats in place of safety hazards. This pattern is part of a collection of high-level security argument patterns which also includes the *Security Property Decomposition* pattern [Jaskolka et al. 2021] and the *Security Requirements Decomposition* pattern. Whereas the Security Property Decomposition provides a claim decomposition to argue the satisfaction of a system's asset protect requirements, this pattern is focused on the system's secure development process requirements.The two patterns are related by the Security Requirements Decomposition pattern (see Fig. 1).

4. APPLICATION OF THE SECURE DEVELOPMENT DECOMPOSITION PATTERN

In this section, we illustrate how to apply the Secure Development Decomposition pattern in the development of a security assurance case in the automotive domain. We focus on a headlamp system adopted from [SAE International 2020]. The headlamp system turns the headlamp on/off according to a switch by demand of the driver. If the headlamp is in high-beam mode, the headlamp system automatically switches the headlamp to the low-beam mode when an oncoming vehicle is detected, and it automatically returns the headlamp to the high-beam mode if the oncoming vehicle is no longer detected. In [Jaskolka et al. 2021], we showed how to apply the Security Property Decomposition argument pattern on this example to argue the satisfaction of the headlamp system's asset protection requirements with respect to ISO/SAE 21434. Now we turn our attention to arguing the satisfaction of the headlamp system's secure development requirements.

To apply the Secure Development Decomposition pattern, we instantiate Con: system with a description of the system being developed. The description of the headlamp system given above is used for this purpose. We also provide Con: compliance with the standard for which compliance is required. In the automotive domain, ISO/SAE 21434 [2020] provides high-level objectives and requirements in the development of road vehicle systems. Then, we instantiate Con: devProcDesc with the development process and security activities adopted for the development of the system. For the purpose of this example, we assume that the headlamp system is developed using the V-model for software development which is common in the automotive domain (e.g., [ISO 2018a]), supplemented with consideration to security-related activities as in [SAE International 2020], for example.

Next, we create the required number of instances of Goal: sysSpecProc based on the system-specific processes required by the standard. ISO/SAE 21434 describes several "Project Dependent Cybersecurity Management" requirements. These include the description of cybersecurity responsibilities and their assignment, the development of a cybersecurity plan, tailoring the cybersecurity activities, and developing plans for dealing with component reuse, out-of-context components, and off-the-shelf components that are captured in work products related to

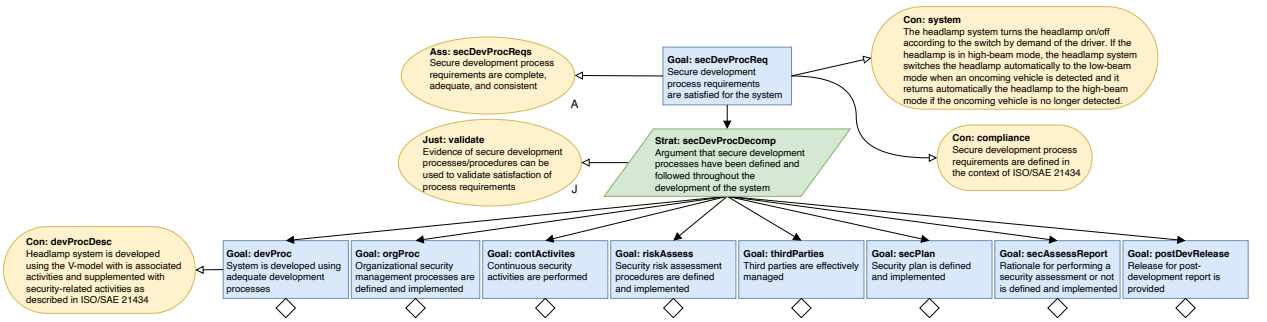

Fig. 4. Application of the Secure Development Decomposition pattern arguing ISO/SAE 21434 compliance of an automotive headlamp system

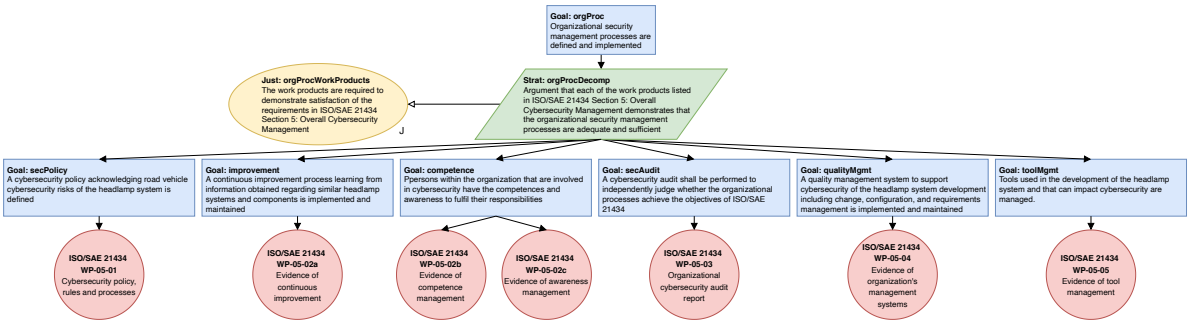

Fig. 5. Decomposition of Goal: orgProc supported by the work products associated with Overall Cybersecurity Management in ISO/SAE 21434

developing a cybersecurity plan, cybersecurity assessment report, and release for post-development report. We instantiate Goal: sysSpecProc by creating Goal: secPlan, Goal: secAssessReport, and Goal: postDevRelease to argue the satisfaction of these specific secure development process requirements for the headlamp system. The resulting application of the Secure Development Decomposition pattern in this context is shown in Fig. 4.

To complete the argument, we need to further decompose the lower-level goals and provide solutions to demonstrate compliance with ISO/SAE 21434. One way to achieve this is by using the specific work products defined in ISO/SAE 21434 corresponding to the secure development processes to support the remaining goals. These work products give the specific evidence (solutions) required to demonstrate that the security process requirements associated with these activities have been satisfied in compliance with ISO/SAE 21434. For example, Goal: orgProc is supported by the work products associated with Overall Cybersecurity Management as shown in Fig. 5. Similar arguments can be made using the work products identified in ISO/SAE 21434 for the other undeveloped goals left after applying the Secure Development Decomposition pattern.

5. RELATED WORK

In this paper, we presented a security assurance argument pattern which provides a claim decomposition supporting standards compliance. Several works have built on the above-mentioned foundations of assurance argument patterns to support standards compliance as we have. Some works in this area have again focused on safety arguments. For example, through pattern-based justification, Gallina et al. [2017] argued that Model-Based Assessment (MBA) is partly compliant with the CENELEC-EN 50128 railway safety standard [2020]. Other works have turned their attention to defining security argument patterns based on international security standards. Yamamoto et al. [2013] presented a security argument pattern to demonstrate that a system is secure based on the Common Criteria [2017]. Ankrum and Kromholz [2005] mapped three diverse standards, including the Common Criteria for security, to the Adelard Safety Case Editor (ASCE) tool to demonstrate how to structure

an assurance case for a practical security-critical system compliant with the standards. Delmas et al. [2020] described a general process for specifying assurance case patterns for a complex standard. Further, the Standards Conformity Framework (SCF) [Cyra and Gorski 2011] presented the concept of conformance templates. Our proposed argument pattern has the potential to support such a framework as it provides a structured argument decomposition to demonstrate standards compliance for a specified system and application domain.

6. DISCUSSION

The Secure Development Decomposition argument pattern supports compliance efforts by providing a claim decomposition of the complex activities and tasks that must be defined and implemented that are found in a variety of secure development methodologies or security standards from different application domains. Many existing assurance argument patterns supporting security compliance have been presented with respect to a single (fixed) standard. The Secure Development Decomposition argument pattern presented in this paper is a more generic pattern that can be instantiated with different standards from different domains. For example, in Section 4 we demonstrated the instantiation of the Secure Development Decomposition pattern for a system from the automotive domain with respect to ISO/SAE 21434. But we could also apply the pattern for a system from the energy domain with respect to NISTIR 7628 Revision 1 or for a system from the health domain with respect to ISO 13485. More generally, we could also apply the pattern to show compliance with a secure development methodology such as such as UMLsec [Jürjens 2002], SecureUML [Lodderstedt et al. 2002], and ASE [Uzunov et al. 2015]. This provides additional flexibility and adaptability of the proposed pattern for use in security assurance efforts for different systems in different application domains.

This idea draws inspiration from the area of situational method engineering (e.g., [Ralyté et al. 2003; Hug et al. 2009]) which aims to construct systems development methods to match specific organizational situations. The proposed pattern and our pattern-based approach for building structured security assurance case models for similarly aims to provide the flexibility to adapt to specific organizational situations when developing critical systems. The modular argumentation of the proposed pattern facilitates easy integration of changes, enabling the model to be constantly updated and stay relevant amid changes to requirements, standards, and/or the threat landscape. This gives the flexibility to describe precise forms of evidence later on and to adapt to changes in particular circumstances. For example, it may be the case at some point in the future that changes to the ISO/SAE 21434 standard require more support to Goal: thirdparties perhaps by requiring additional work-products. In this case, the proposed pattern provides a mechanism by which we can reason about these concerns by, for example, determining where this goal is supported and how such changes would be propagated through the argument.

7. CONCLUSIONS AND FUTURE WORK

Developing compelling, evidence-based arguments supporting compliance with secure development methodologies and/or security standards for many critical systems is at the heart of effective security assurance. In this paper, we proposed the Secure Development Decomposition assurance argument pattern. It can be adopted to better support the development of secure systems through the formulation of well-reasoned security assurance arguments. The pattern supports compliance for secure development processes by providing a claim decomposition derived from studying a variety of activities and tasks that must be defined and implemented in existing secure development methodologies and/or standards. We provided an illustrative application of the proposed argument pattern for an automotive system in support of compliance with ISO/SAE 21434 [2020]. We also discussed how the proposed pattern could be instantiated for other system domains and standards or development methodologies.

In future work, we seek to define more argument patterns to support a pattern-based approach for modular and incremental security assurance. Next, we plan to turn our attention to describing the *Security Requirements Decomposition* pattern (see Fig. 1). We envision a repository of security argument patterns that can be selected and assembled to help manage the complexity of large security assurance cases. We also aim to explore the

interplay between security and other dependability aspects such as safety, robustness, and reliability in support of expanding our efforts to consider multi-concern assurance and compliance to multiple standards.

ACKNOWLEDGEMENTS

The authors would like to thank Eduardo Fernandez for his insightful comments and feedback during the shepherding for PLoP 2021 that greatly helped to improve this paper.

REFERENCES

Rob Alexander, Richard Hawkins, and Tim Kelly. 2011. *Security Assurance Cases: Motivation and the State of the Art*. Technical Report CESG/TR/2011/1. University of York, York, UK.

T Scott Ankrum and Alfred H Kromholz. 2005. Structured assurance cases: Three common standards. In *Ninth IEEE International Symposium on High-Assurance Systems Engineering (HASE'05)*. IEEE, Heidelberg, Germany, 99–108.

Committee on National Security Systems. 2012. *National Information Assurance Policy for Space Systems Used to Support National Security Missions*. CNSS Policy 12. Committee on National Security Systems.

Common Criteria Recognition Arrangement. 2017. *Common Criteria for Information Technology Security Evaluation (CC), Version 3.1, Revision 5*. Number CCMB-2017-04-001.

Lukasz Cyra and Janusz Gorski. 2011. SCF—A framework supporting achieving and assessing conformity with standards. *Computer Standards & Interfaces* 33, 1 (2011), 80–95.

Kevin Delmas, Claire Pagetti, and Thomas Polacsek. 2020. Patterns for Certification Standards. In *International Conference on Advanced Information Systems Engineering*. Springer, Cham, 417–432.

Brian Dobbing and Samantha Lautieri. 2006. *SafSec Methodology: Standard 3.1*. SafSec: Integration of Safety & Security Certification S.P1199.50.2. Altran Praxis.

European Committee for Electrotechnical Standardization. 2020. CENELEC - EN 50128 Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems. (June 2020).

Justin Firestone and Myra B. Cohen. 2018. The Assurance Recipe: Facilitating Assurance Patterns. In *Computer Safety, Reliability, and Security*, Barbara Gallina, Amund Skavhaug, Erwin Schoitsch, and Friedemann Bitsch (Eds.). Springer, Cham, 22–30.

E. Kenneth Hong Fong and David A. Wheeler. 2018. *A Sample Security Assurance Case Pattern*. IDA Paper P-9278. Institute for Defense Analyses, Alexandria, VA, USA.

Barbara Gallina, Elena Gómez-Martínez, and Clara Benac-Earle. 2017. Promoting MBA in the rail sector by deriving process-related evidence via MDSafeCer. *Computer Standards & Interfaces* 54 (2017), 119–128.

Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. 1994. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.

David Garlan and Mary Shaw. 1993. An Introduction to Software Architecture. *Advances in Software Engineering and Knowledge Engineering* I (Dec. 1993), 1–39.

GSN Working Group. 2018. GSN Community Standard Version 2. Available: `https://scsc.uk/r141B:1?t=1`. (Jan. 2018).

Richard Hawkins and Tim Kelly. 2013. *A Software Safety Argument Pattern Catalogue*. Technical Report. The University of York.

Charlotte Hug, Agnès Front, Dominique Rieu, and Brian Henderson-Sellers. 2009. A Method to Build Information Systems Engineering Process Metamodels. *Journal of Systems and Software* 82, 10 (2009), 1730–1742.

ISO. 2016. ISO 13485 Medical devices–Quality management systems–Requirements for regulatory purposes. (Mar. 2016).

ISO. 2017. ISO/IEC 27019 Information technology–Security techniques–Information security controls for the energy utility industry. (Oct. 2017).

ISO. 2018a. ISO 26262: Road Vehicles–Functional Safety. (Dec. 2018).

ISO. 2018b. ISO/IEC 27000 Information technology–Security techniques–Information security management systems–Overview and vocabulary. (Feb. 2018).

ISO. 2019. ISO 14971 Medical devices–Application of risk management to medical devices. (Dec. 2019).

Jason Jaskolka. 2018. Challenges in Assuring Security and Resilience of Advanced Metering Infrastructure. In *18th Annual IEEE Canada Electrical Power and Energy Conference (EPEC 2018)*. IEEE, Toronto, ON, Canada, 1–6.

Jason Jaskolka. 2020. Recommendations for Effective Security Assurance of Software-Dependent Systems. In *Intelligent Computing, SAI 2020*, Kohei Arai, Supriya Kapoor, and Rahul Bhatia (Eds.). Advances in Intelligent Systems and Computing, Vol. 1230. Springer, 511–531.

Jason Jaskolka, Brahim Hamid, Alvi Jawad, and Joe Samuel. 2021. A Security Property Decomposition Argument Pattern for Structured Assurance Case Models. In *The 25th European Conference on Pattern Languages of Programs (EuroPLoP 2021)*. 10. (To Appear).

Jan Jürjens. 2002. UMLsec: Extending UML for Secure Systems Development. In *UML 2002 — The Unified Modeling Language*, Jean-Marc Jézéquel, Heinrich Hussmann, and Stephen Cook (Eds.). Springer, Berlin, Heidelberg, 412–425.

Tim Kelly. 1998. *Arguing Safety. A Systematic Approach to Safety Case Management*. Ph.D. Dissertation. University of York, York, UK.

Tim Kelly. 1999. ALARP (As Low As Reasonably Practicable) Pattern. Available: `https://scsc.uk/rgsn25:1`. (Feb. 1999).

Tim Kelly and John McDermid. 1997. Safety Case Construction and Reuse using Patterns. In *16th International Conference on Computer Safety, Reliability and Security (SafeComp '97)*, P. Daniel (Ed.). Springer, London, 55–69.

Torsten Lodderstedt, David Basin, and Jürgen Doser. 2002. SecureUML: A UML-Based Modeling Language for Model-Driven Security. In *UML 2002 — The Unified Modeling Language*, Jean-Marc Jézéquel, Heinrich Hussmann, and Stephen Cook (Eds.). Springer, 426–441.

Jolita Ralyté, Rébecca Deneckère, and Colette Rolland. 2003. Towards a Generic Model for Situational Method Engineering. In *Advanced Information Systems Engineering*, Johann Eder and Michele Missikoff (Eds.). Springer, Berlin, Heidelberg, 95–110.

Ronald S. Ross, Michael McEvilley, and Janet C. Oren. 2016. *Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems*. Special Publication (NIST SP) 800-160 Volume 1. NIST, Gaithersburg, MD, USA.

SAE International. 2020. ISO/SAE DIS 21434: Road Vehicles–Cybersecurity Engineering. (Feb. 2020).

The Smart Grid Interoperability Panel. 2014. *Guidelines for Smart Grid Cybersecurity: Volume 1–Smart Grid Cybersecurity Strategy, Architecture, and High-Level Requirements*. Interagency Report NISTIR 7628 Revision 1. NIST.

USA Department of Defense. 2014. DoD Instruction 8510.01, Risk Management Framework for DoD Information Technology. (Mar. 2014).

Anton V. Uzunov, Eduardo B. Fernandez, and Katrina Falkner. 2015. ASE: A Comprehensive Pattern-Driven Security Methodology for Distributed Systems. *Computer Standards & Interfaces* 41 (2015), 112–137.

Frédérique Vallée, Mohamed Bakkali, Marc Sango, Fredrik Warg, Irfan Sljivo, Barbara Gallina, Jose Luis de la Vara, Victor Sacristán, Christoph Schmittner, Thomas Gruber, Zhendong Ma, Petr Böhm, Morayo Adedjouma, Stefano Puri, Garazi Juez, Alejandra Ruiz, Huascar Espinoza, Helmut Martin, and Benito Caracuel. 2018. *Baseline and requirements for multi-concern assurance*. Technical Report AMASS_D4.1_WP4_A4T_V1.1. Architecture-driven, Multi-concern and Seamless Assurance and Certification of Cyber-Physical Systems.

Fredrik Warg and Martin Skoglund. 2019. Argument Patterns for Multi-Concern Assurance of Connected Automated Driving Systems. In *4th International Workshop on Security and Dependability of Critical Embedded Real-Time Systems (CERTS 2019)*, Mikael Asplund and Michael Paulitsch (Eds.), Vol. 73. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 3:1–3:13.

Alan Wassyng, Neeraj Kumar Singh, Mischa Geven, Nicholas Proscia, Hao Wang, Mark Lawford, and Tom Maibaum. 2015. Can Product-Specific Assurance Case Templates Be Used as Medical Device Standards? *IEEE Design & Test* 32, 5 (Oct. 2015), 45–55.

Shuichiro Yamamoto, Tomoko Kaneko, and Hidehiko Tanaka. 2013. A Proposal on Security Case Based on Common Criteria. In *Information and Communication Technology*, Khabib Mustofa, Erich J. Neuhold, A. Min Tjoa, Edgar Weippl, and Ilsun You (Eds.). Springer, 331–336.

Received June 2021; revised September 2021; accepted TODO