Dipl.-Ing. Andreas Daniel Sinnhofer, BSc.

Advances of the Pre-Personalization Process for Secure Embedded Systems

DOCTORAL THESIS

to achieve the university degree of

Doktor der technischen Wissenschaften

submitted to

Graz University of Technology

Supervisor

Prof. Dr. (ETH) Kay Uwe Römer

Institute of Technical Informatics

Advisor
Ass.Prof. Dipl.-Ing. Dr.techn. Christian Steger
Dipl.-Ing. Dr.techn. Christian Kreiner

Graz, September 2017

AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present doctoral thesis.

....
date (signature)

Acknowledgments

During my studies, and especially during my PhD, I had the chance to work with a lot of great people for which I am very thankful. I think it is not possible to adequately acknowledge everyone, but there are a few persons I would like to give an extra "thanks".

First of all, I would like to mention that this thesis was carried out at the Institute of Technical Informatics at Graz University of Technology, in cooperation with our industrial research partner NXP Semiconductors Austria GmbH. I would like to thank NXP for letting me participate in an interesting industrial project and for enabling my PhD study.

I would like to thank my supervisor Prof. Kay Uwe Römer for being open for discussions and for the excellent support during the official and organizational part of my PhD study. I would also like to thank Prof. Andreas Riel who kindly agreed to examine my thesis as a second assessor. Further, I would like to express my thanks to my advisor Christian Steger, who not only gave me the chance to work in this interesting research project, but also supported every decision I made and gave me the freedom regarding my research topics. Thanks a lot and I hope we stay in contact!

Big thanks go to my second advisor Christian Kreiner. I think it was back in December 2011 when we had our first discussion about my bachelor thesis and since then, I could rely on his continuous support. Without him, I wouldn't have met people like Peter Pühringer or Christian Steger, which both played key roles during my studies. Thanks a lot for your support over the last years, and I hope we stay in contact!

I also want to thank my colleagues who made my work maybe not always easier, but at least more enjoyable! I would like to especially thank my colleague Florian Oberhauser for his support during my studies. Thanks to Reinhard Berlach, Wolfgang Raschke, Massimiliano Zilli and Michael Lackner, who all supported me especially at the beginning of my PhD. Thanks also to Georg Macher, Johannes Iber, Tobias Rauter, Harald Sporer, Michael Kripser, Andrea Höller, Ralph Weissnegger, and Michael Spörk with which I had a lot of enjoyable coffee-meetings. Special thanks go to my colleague Felix Jonathan Oppermann who joined the research project for about a year. Without his help I wouldn't have had the time to work on my research. Further, I would like to deeply thank my colleague Klaus Potzmader who is probably one of the best software architects I have ever worked with. I learned a lot from him and his experience, and he always supported my (sometimes) crazy PhD ideas. I would also like to thank my team leader at NXP Clemens Orthacker. Without his support, I would not have the position I currently have.

Finally, I would like to thank my family for their lifelong support and for their encouragement during my whole live. Thanks to my brother Christian, who supported me during my whole life. A heartfelt gratitude goes to Inge Siegl for her support during the last seven years. Thanks for finding the right words during challenging periods and your love.

Graz, September, 2017

Abstract

We are at the dawn of the era of the Internet of Things (IoT). While such devices are increasingly and excessively used in our daily lives, their security vulnerabilities are often not recognized by the users: Each of these devices is usually equipped with a wide variety of different sensors and interfaces to communicate with its environment or via the Internet; as a consequence, hacking such devices is a valuable goal for adversaries to gain personal financial benefit. Since a lot of personal information can be collected from such devices, also national agencies may be interested in hacking such devices to monitor the daily lives and interactions of the users. Furthermore, also the number of cyberattacks, operated by IoT-based Bot-Nets, has increased over the last years. The most prominent example was the Mirai hack at the end of last year which took out a big Internet infrastructure company in the USA for several hours. One part of this bot-net was a massive number of hacked cameras and digital video recorders connected by the Internet. The devices could be captured by the Mirai malware since weak and partially static keys were used to protect the system against intrusion.

The lesson that should be learned is that the protection of these devices is vital to ensure the privacy of the users as well as of their data. One essential step towards this goal is to secure the pre-personalization of such critical infrastructure: During the pre-personalization process, the device's individual security-related setup – like keys – is loaded in order to guarantee a secure operation in the field. In combination with tamper-proof hardware protection mechanisms and crypto co-processors, a Root of Trust can be established during the production process. Since security usually means additional development costs, the pre-personalization process has to be established in such a way that the additional costs can be kept as low as possible to enable low-cost but secure devices.

In this thesis, new methods will be developed in order to establish product-line approaches for the pre-personalization process of secure embedded systems. Thus, the process will be designed to be compatible with a wide variety of different products which reduces the overall costs. This will by achieved by introducing a generic and configurable data generation system to generate the device-individual data. In order to ensure high security requirements, formal methods will be applied during the process to safeguard the security properties of the generated data as well as the provided customer data. Furthermore, business-process-driven approaches will be presented which can be used to automatically configure the pre-personalization process for the current product requirements. This also plays a vital role in security certifications since the product requirements can be directly traced to the pre-personalization steps. As such, a comprehensive overview will be generated of how the data was generated and loaded to the device.

Kurzfassung

Wir befinden uns im Anbruch des Zeitalters des Internet of Things (IoT). Während IoT-Geräte in unserem alltäglichen Leben Einzug halten, werden die damit verbundenen Sicherheitslücken oftmals von den Benutzern nicht erkannt: Jedes dieser Geräte besitzt eine Vielzahl von Sensoren und Kommunikationsschnittstellen, mit denen sie die Umgebung überwachen und über Netzwerke wie das Internet kommunizieren. Infolgedessen ist das Hacken solcher Geräte ein lohnendes Ziel, um persönlichen finanziellen Gewinn zu erzielen. Da persönliche Daten von solchen Geräten gesammelt und gespeichert werden, könnten auch nationale Sicherheitsbehörden daran interessiert sein, IoT-Geräte zu hacken, um Informationen über unser tägliches Leben zu erhalten. Aber auch die Anzahl an Cyber-Attacken, die durch Bot-Netze durchgeführt werden, nahm in den letzten Jahren stark zu. Eines der bedeutendsten Beispiele war der Denial of Service (DoS) - Angriff vergangenes Jahre auf einen großen amerikanischen Internet-Infrastrukturanbieter: Ein großer Teil des Bot-Net bestand aus Webcams und digitalen Videorekordern, die von der Schadsoftware Mirai befallen waren. Diese Geräte konnten über das Internet gehackt werden, da sie schwache bzw. statische Schlüssel verwendeten, um unbefugten Zugriff zu verhindern.

Die Lektion, die wir daraus lernen sollten, ist, dass der Schutz von IoT-Geräten essenziell ist, um die Privatsphäre der Benutzer sowie deren Daten zu schützen. Ein bedeutender Schritt ist dabei eine sichere Pre-Personalisierung dieser Geräte: Während der Pre-Personalisierung wird die initiale Gerätekonfiguration bestehend aus kryptografischen Schlüsseln und anderen Daten auf die Geräte aufgespielt. Wird dieser Prozess mit sicherem Speicher und kryptografischen Co-Prozessoren kombiniert, kann eine Root of Trust etabliert werden. Dies ermöglicht eine sichere Verwendung der Geräte im Feld. Da Sicherheitsanforderungen üblicherweise zu höheren Entwicklungskosten führen, muss der Prozess so gestaltet werden, dass die zusätzlichen Kosten möglichst gering gehalten werden.

Diese Arbeit beschäftigt sich mit neuen Methoden, um den Pre-Personalisierungsprozess von sicheren eingebetteten Systemen in einem Produktlinienansatz zu etablieren. Somit wird ein breites Spektrum an unterschiedlichen Produkten unterstützt, wodurch die Gesamtkosten für einzelne kompatible Geräte gesenkt werden können. Dies wird durch die Einführung eines generischen und flexiblen Systems erreicht, welches zur Datenerzeugung für die individuellen Geräte verwendet wird. Um die Sicherheitseigenschaften der Kundendaten und der erzeugten Daten während des gesamten Prozesses sicherzustellen, werden Methoden aus der formalen Verifikation angewendet. Darüber hinaus werden Methoden für das Management von Geschäftsprozessen vorgestellt, welche dazu verwendet werden, das Pre-Personalisierungssystem automatisiert zu konfigurieren. Dies spielt speziell in der Zertifizierung eine große Rolle, da mithilfe dieser Methoden eine direkte Verbindung zwischen den Anforderungen des Kunden und den erzeugten Daten hergestellt werden kann. Somit können Berichte generiert werden, die klar aufzeigen, wie Kundendaten während des gesamten Prozesses geschützt und generiert wurden.

Contents

List of Figures

List of Tables

List of Listings

List of Abbreviations

API Application Programming Interface.

BP Business Process.

BPM Business Process Management.

BPMN Business Process Model and Notation.

CAP Composed Assurance Packages.

CC Common Criteria.

CIA Confidentiality, Integrity and Availability.

CPS Cyber-Physical System.

CRC Cyclic Redundancy Check.

DoS Denial of Service.

EAL Evaluation Assurance Level.

EEPROM Electrically Erasable Programmable Read-Only Memory.

ERP Enterprise Resource Planning.

ETR Evaluation Technical Report.

FD Function Description.

FDL Function Description Language.

FODA Feature-Oriented Designed Modeling.

FTP File Transfer Protocol.

HMAC Keyed-Hash Message Authentication Code.

HSM Hardware Security Module.

IC Integrated Circuit.

InfoSec Information Security.

IoT Internet of Things.

MNO Mobile Network Operator.

OS Operating System.

PCI Payment Card Industry Security Council.

PP Protection Profile.

PT Production Template.

ROM Read-Only Memory.

SIM Subscriber Identity Module.

SPL Software Product Line.

SPLE Software Product Line Engineering.

ST Security Target.

ToE Target of Evaluation.

VP Variation Point.

XML Extensible Markup Language.

XSD XML Schema File.

1 Introduction

"Begin at the beginning, and go on till you come to an end; then stop."
Charles Lutwidge Dodgson (Lewis Carroll), Alice in Wonderland

Cyber-Physical System (CPS) and Internet of Things (IoT) devices are increasingly used in our daily lives. Generally speaking, IoT refers to the connection of our everyday objects with a network like the Internet [75]. Each of these devices is usually equipped with different kinds of sensors to observe its environment, making the device a smart object. In combination with embedded systems, IoT promises to increase the quality of our daily lives by taking over simple tasks like controlling the room temperature and cooking coffee [47]. As a consequence, smart objects like wearables are becoming more and more interesting for adversaries due to their increasing functionalities like Internet capabilities, cameras, microphones, GPS trackers and other sensor devices (Paper E). Furthermore, such smart objects are often deployed in unsupervised and untrusted environments turning the question about privacy and security into a crucial topic. As a consequence, a robust and secure software design is required for the implementation of cryptographic communication protocols and encryption algorithms. Moreover, tamper-proof solutions like secure elements and trusted-platform modules are necessary to securely calculate cryptographic functions and to store confidential data or cryptographic keys. While cryptographic protocols and secure hardware architectures are much discussed and subject to further research activities, the issue of provisioning the initial confidential device setup is still widely uncovered. However, the protection of this initial setup is as important as the protection of the confidential data during the time in use. Especially the protection of master keys is essential because otherwise, all security measures which are based on such keys are futile.

One prominent example of hacking such master keys was revealed by Edward Snowden in 2015: The secret master key of Subscriber Identity Module (SIM) cards, securing the 3G and 4G mobile communication channel, was victim to an attack perpetrated by the American "National Security Agency" (NSA) and the British agency "Government Communications Headquarters" (GCHQ) [54, 55, 66]. The company Gemalto, which produces – amongst other devices – smart cards in form of SIM cards and EMV chip cards, was victim to this attack. More precisely, Gemalto generates and inserts a chip-individual symmetric encryption key into each manufactured SIM card during the personalization process of the manufacturing process. The inserted key is used to encrypt the communication between the mobile phone and the cell tower of the mobile network operator. Mobile network operators purchase these SIM cards in batches from Gemalto. Additionally, Gemalto pro-

vides a file containing a list of all the master keys for the mobile network operator to allow the SIM cards to be recognized by the network. The primary goal of this hack was to steal the symmetric encryption keys to wiretap and decipher the encrypted mobile phone communication. By using the stolen symmetric encryption keys, the national agencies did not need any assistance from the mobile operators or permissions from the legal official court to spy on private conversations. To get inside Gemalto's network, social engineering attacks like phishing and faking Facebook posts were used to take over the employees' PCs [55]. Once inside the network, the agencies were able to retrieve the encryption keys since they have been sent via unencrypted or weakly protected File Transfer Protocol (FTP) to the smart card manufacturing factories. According to Scahill and Begley [55], millions of keys were stolen by GCHQ in a three-month period in 2010. This is a good example to show the impact of insecure data handling and how many users can be affected by hacking the personalization system of secure integrated circuits.

1.1 The challenge of reusing security artifacts

The development of a security-critical system or device is in general expensive and time-consuming. One of the most expensive aspects during the development of secure systems is the implementation and assessment of the security attributes confidentiality, integrity, and availability (referred to as the CIA triad, see section 2.1). The achievement of these attributes greatly relies on specialized technologies, processes and personnel. Higher security requirements are usually associated with the application of strict and formal methods which are in general expensive. Furthermore, security certification standards may especially require the application of such methods to ensure a mature development process. To reduce the engineering costs of security-critical systems, companies are increasingly starting to adopt a product-line approach. This is done by exploiting common and reusable features and artifacts within a specific domain [13, 46]. Security requirements are linked to such product-line approaches [40] to reduce the overall evaluation time for new product variants. With respect to the scope of this thesis, especially the Common Criteria (CC) certification is of importance. Traditionally, a product-line approach for CC-certified products foresees that the Security Target (ST) of the Target of Evaluation (ToE) are generated based on the application requirements. This means that for every individual product, a new product certification process is required which evaluates the generated ST with respect to the implemented security measures.

The systematic reuse of software components and routines is a frequently discussed topic since early software development [26]. But still, the design and implementation of applications remain expensive and error-prone. Most costs and failures arise from the continuous rediscovery and reinvention of core patterns, which is often done using a copy-and-clone process [57]. As a consequence, the development of high-quality, reusable software components is a great challenge [23]. Similar to the domain of functional safety, security is

a context-specific and complete-system attribute [67]. While the reuse of algorithms and protocols is highly recommended, the combination of different secure systems and their interactions is mostly context- and application-specific [3]. As identified by Anderson [3], a secure system can be vulnerable due to unidentified flaws of the Application Programming Interface (API). The reuse of security modules in different applications can even decrease the level of security since security risks of the new system have not been considered in the original analysis of the reusable component. To counteract these flaws, formal verification methods are widely used to analyze the interactions between all system components, including the interactions of external operators. Hence, every possible combination of interactions has to be evaluated to guarantee that the system security requirements are not violated. As such, formal analysis tools require a huge amount of time to model all the interactions and a huge amount of computational power to actually analyze the formal model of the system. The following challenges will specifically be addressed in this thesis:

- Integration of the security assessment into the product-line process by verifying product configurations at runtime against the system security requirements. This includes the design and implementation of reusable security-critical software components.
- Management of the impact of changes on the product line by explicitly tracing and analyzing the changes from the business level to the implemented domain artifacts.

1.2 The challenge of reuse in business-process modeling

Business Process (BP) - oriented organizations are known to perform better regarding highly flexible demands of the market and fast production cycles [30, 37, 71, 74]. These goals are achieved through the introduction of a management process in which business processes are modeled, analyzed and optimized in iterative ways. Nowadays, the Business Process Management (BPM) is also coupled with a workflow management, providing the ability to integrate the responsible participants into the process and to monitor the correct execution of the business process in each process step (Paper F). To administer the rising requirements, so-called BPM tools are used to cover the steps of process modeling, optimization and execution. In combination with an Enterprise Resource Planning (ERP) system, the data of the real process can be integrated into the management process. Additionally, having flexible BPs supports the use of agile development techniques with which incremental product improvements are continuously integrated into the development process [1, 2]. Combining all those aspect should lead to an organization which can be swift to react to the changes of the market.

However, in many cases, business processes do only vary in some points, which leads to the situation that new process variants are created through a copy-and-clone of old solutions. As a result, such process templates are instantiated in many process variants,

which makes the propagation of process improvements time- and cost-intensive for a bigger company (Paper E). In addition, the consistency of the documentation of this huge number of process variants is a challenging task. As a consequence, more maintenance effort is required to ensure that all processes are up to date and efficient with respect to the current market requirements. Thus, the benefits of having a BP-oriented organization are almost neglected due to this organizational overhead. The following challenges will specifically be addressed in this thesis:

- The management of business-process models in a software product-line approach to systematically generate process variants based on the current requirements. This includes the design of (sub-)processes and the transformation of these (sub-)processes into a feature-oriented model.
- Management of process improvement processes so that improvements of processes can be rolled out systematically to the whole product line.

1.3 Thesis hypothesis

> *The secure generation of pre-personalization data can be **feasible** and **traceable** through a process-driven product-line approach. Support of **multiple target platforms** can be assured through the use of platform configurations.*

The following key terms will be used as a foundation to evaluate this thesis (see chapter 6):

- **Feasible**: the application of the results in an industrial context
- **Traceable**: explicitly identifying and managing associations of product configurations and processes
- **Multi-platform support**: the developed system can be used for a wide variety of products as long as the platform supports basic (security) requirements

1.4 Contributions

The scientific contributions of this thesis are summarized in Table 1.1. Figure 1.1 illustrates the suggested approaches for improving the pre-personalization process of secure ICs grounded on a template-based process. First, we propose combining the variability management of business-process models and software designs to create a ***traceable*** link between customer requirements and software implementations. Second, we will examine methods to ***support multiple platforms*** to ensure a ***low-cost*** pre-personalization process for secure devices. To achieve these goals, we suggest the use of mature software-product-line

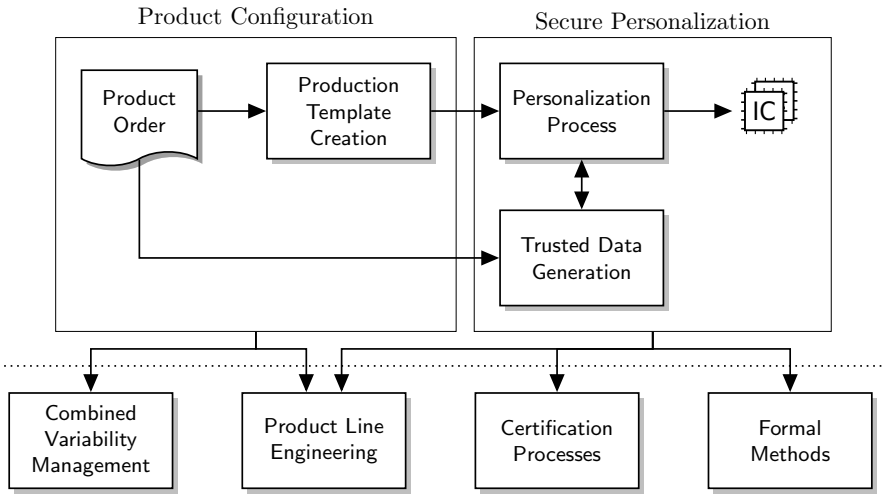


Figure 1.1: Overview of the suggested approaches to improve the pre-personalization process of secure ICs

development techniques and the application of formal verification methods to ensure security requirements during the pre-personalization process. Furthermore, we will investigate the use of incremental and modular certification processes to reduce the certification costs of different products.

1.4.1 Combined variability management

The reuse of software components is an important step for an industrial company to survive in a flexible and competitive market. But only with an integrated view of the according business processes, it is possible to raise the efficiency of the overall business. Thus, we propose a framework which allows to combine the variability modeling of software components and business-process models. This permits us to efficiently design software product line architectures that can directly be configured by the underlying business process of order processes. The developed framework is able to synchronize changes of the process model with software artifacts that have to be implemented by the developers. Methods for data binding are used to attach the specific software artifacts to a specific set of configurable settings that have to be provided by internal or external customers during the order process. Thus, software artifacts are automatically instantiated based on the provided customer data. This essentially reduces the development costs and the required time to react to the market's changes. Moreover, the overall robustness of the software toolchain is increased since the same code base is shared by a lot of different product families, leading to a higher customer satisfaction. Finally, the traceability of customer requirements to the actual product configurations is raised which is also a certification requirement.

1.4.2 A model-based formal verification method

Flexibility usually contradicts security requirements since an arbitrary combination of interactions, which are secure in isolation, may cause a system to leak confidential data in combination (see Section 3.2 for examples). Traditionally, such security vulnerabilities are circumvented by reducing the customization options and by analyzing the remaining options using formal verification methods prior to the development of the system. Since the field of application for secure ICs is growing in multiple directions, it is necessary to find new methods which enable the reuse of the pre-personalization system for a wide variety of different products in different domains. To overcome this limitation, we developed a model-based formal verification method which allows us to enforce security properties during the runtime of the pre-personalization system. Thus, more flexibility is guaranteed as long as specific interactions with the system do not violate the overall security model. Since traditional formal verification methods tend to have high resource requirements, we developed a new method based on concepts from the symbolic execution and abstract interpretation. Furthermore, we restricted the use of conditional branches and loops to guarantee a small memory consumption of the verification tool.

1.5 Thesis structure

The thesis is structured in the following way:

- Chapter 2: Defines the basic taxonomy and describes basic processes and techniques which lay the foundation of this thesis.
- Chapter 3: Summarizes state-of-the-art literature on (pre-)personalization processes and software-product-line engineering techniques for secure systems and business-process models. Moreover, related work will be presented which deals with formal verification tools and API attacks.
- Chapter 4: Introduces the design principels for an order-process-based software-configuration framework. This framework is based on a product-line approach for the creation of business-process models.
- Chapter 5: Elaborates the design principles of a flexible data generation system which is used to generate configuration scripts for individual products. Furthermore, a formal verification tool will be introduced which is used to guarantee security requirements during the customization process.
- Chapter 6: Presents the evaluation results of the developed tools and processes in industrial use cases.
- Chapter 7: Summarizes the thesis and presents future research directions.

Challenge	*Contribution*
Assessment of system security requirements throughout the pre-personalization product line	• A model-based, formal verification tool for analyzing Security Requirements during the Data-Generation-Process of the pre-personalization process (Section 5.1, Patent I) • A flexible runtime-configurable data generation system to generate chip individual pre-personalization data (currently in a company-internal patent process; Patent J)
Management of business-process models	• A method for the management of business-process models in a software-product-line approach, deriving process variants based on the current production requirements (Section 4.1, Paper B) • Support of process improvement processes through the systematic reuse of process models in various process variants (Paper B)
Management of the impact of changes on the security-critical product line	• A method for combined variability management to trace the impact of changes from the business level to the implementation level (Section 4.3, Paper D, F, G). • Methods to support agile and incremental product development techniques by systematically analyzing the impact of changes especially with respect to the security certification (section 6.3, Paper A, C).

Table 1.1: Overview of the contributions of this thesis

2 Background

An investment in knowledge pays the best interest.

Benjamin Franklin

This chapter provides a brief introduction to the theoretical background and key terms used in this thesis. Portions of this chapter were previously published as Paper H [66], Paper A [61], and Paper C [63], and have been reproduced with permission.

2.1 Information security

Information Security (InfoSec) is defined in various standards as the discipline of protecting information against unauthorized persons. One of the most common definition is the following:

> *The protection of information and information systems from unauthorized access, use, disclosure, disruption, modification, or destruction in order to provide confidentiality, integrity, and availability* [41].

The basic concepts that are of importance to this thesis will be described in the following.

2.1.1 Information security attributes

Saltzer and Schroeder [53] defined the information security attributes Confidentiality, Integrity and Availability (CIA) in 1975 as follows:

> **Unauthorized information release (Confidentiality)**: *an unauthorized person is able to read and take advantage of information stored in the computer. This category of concern sometimes extends to "traffic analysis," in which the intruder observes only the patterns of information use and from those patterns can infer some information content. It also includes unauthorized use of a proprietary program* [53].

> **Unauthorized information modification (Integrity)**: *an unauthorized person is able to make changes in stored information – a form of sabotage. Note that this kind of violation does not require that the intruder see the information he has changed* [53].

> **Unauthorized denial of use (Availability)**: *an intruder can prevent an authorized user from referring to or modifying information, even though the intruder may not be able to refer to or modify the information* [53].

Over the past years, extensions have been developed to integrate additional attributes like non-repudiation, authenticity, risk, control and practices [35, 44, 45], but the CIA form the core attributes of all models. These attributes are composed of the three main aspects Hardware, Software and Communication to help identify security vulnerabilities and to apply specific countermeasures.

2.1.2 Classification of attacks

Figure 2.1 shows a broad classification of attacks on embedded systems according to Ravi, Raghunathan, and Chakradhar [50] and Wasicek [72]. From a top-level perspective, attacks are grouped as functional and agent-based classes. Functional attacks are classified into three main categories, namely *Integrity Attacks*, *Privacy (Confidentiality) Attacks*, and *Availability Attacks*, which corresponds to the basic security attributes of the CIA triad. The agent(s) who initiate(s) an attack may either be passive or active. Passive attacks are attacks in which the adversary does not interfere with the system execution, but observes the system, all of its components, and the exchanged messages. Active attacks are attacks in which the adversary actively interferes the normal system execution by – for example – manipulating messages or specific hardware components of the system.

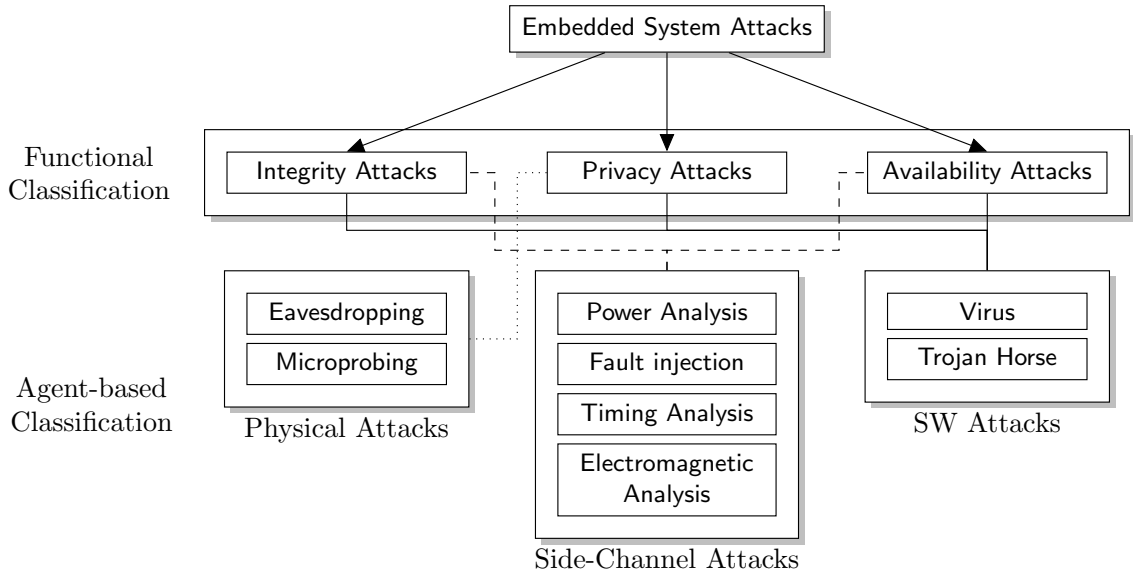


Figure 2.1: Classification of attacks on embedded systems, adopted from [50, 72]

Attacker	Motivation	Possible attack vectors
Competitors	Gaining knowledge of the system / pre-personalization process Damaging the reputation of the involved companies	Manipulation of the production equipment (HW and/or SW) Integration of manipulated devices into the pre-personalization process
National Agencies	Gaining knowledge of the users of the system Gaining access to specific functionality of the devices (e.g., accessing cameras or microphones)	Manipulation of the production equipment (HW and/or SW) Integration of manipulated devices into the pre-personalization process Attacks on social level against involved operators
Insider	Gaining personal financial profit	Manipulation of the production equipment (HW and/or SW) Deliberate misapplication of the production equipment

Table 2.1: Classification of attackers and attack vectors during the pre-personalization process, based on Rankl and Effing [49]

In practice, adversaries usually perform a combination of the described attacks to break a system. For example, physical attacks may be used to identify weak points in the system which can be utilized in side-channel attacks to break the system. In the context of this thesis, some of those attacks have already been countered or mitigated due to operational measures, which will be described in the next section.

2.1.3 Classification of attackers and attacks during the pre-personalization process

The attackers' basic motivation is usually greed or the gain of fame [49]. Greed is not only limited to personal financial profit, but also includes the gain of knowledge. National agencies, for instance, are not interested in gaining financial profit, but are interested in gaining knowledge of the users or the environment which the system is deployed in.

If details about hacks become known, the reputation of the companies involved may be seriously damaged. Thus, another potential attack vector includes competitors who either want to gain knowledge of the system or want to destroy another company's reputation. Similar damage to a company's reputation can be caused by scientific researchers who aim at breaking the system to gain reputation in their scientific field. As such, these attackers are only satisfied if their successful hacking attempt is published in scientific publications.

But also individual adversaries may only be motivated in gaining fame for being the first person breaking the system.

With respect to this thesis, operational countermeasures have already been implemented by the environment of the pre-personalization process. Since the pre-personalization process is executed in a secured production environment with restricted access, it is not possible for an arbitrary hacker to place specific hardware devices within the production environment. Such kinds of attacks may only be possible by competitors or organizations with a high amount of resources. Moreover, the control logic of the personalization equipment and the equipment itself are disconnected from any external network. Thus, malicious software cannot be loaded to such devices via public networks like the Internet. It is assumed that data between the pre-personalization company and any involved stakeholder is sufficiently protected while transferred between them. To summarize, Table 2.1 illustrates the attackers and attack vectors within the scope of this thesis. Although confidential pre-personalization data could be leaked through an attack during the in-field use, such kind of attacks are beyond the scope of this thesis.

2.2 Lifecycle of secure integrated circuits

Over the last decades, the development and production processes of secure smart cards have been continuously improved with respect to security and costs. Since the market of IoT devices is expected to grow similarly over the next years [36], the research findings of the smart card domain lay a good foundation to define a lifecycle for secure IoT devices [66]. The lifecycle can be separated into six phases which are illustrated in Figure 2.2 and will be briefly discussed in the following [49, 70] (Paper H):

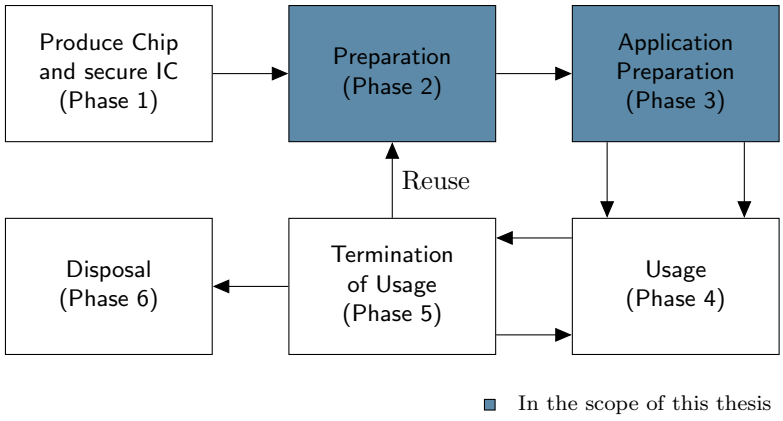


Figure 2.2: Lifecycle of secure CPS or IoT devices, adopted from the Smart Card standard ISO-10202-1 [68]

- **Production of Components (Phase 1)**: During this phase, all components of the secure CPS or IoT device are designed and manufactured. In many cases, the individual components will be developed and produced by different vendors. With respect to security, the production is an essential step since no matter how high the quality of the system and its cryptographic protection mechanisms are, they are of little help if all the confidential data is leaked during the production process. Besides security considerations, also functional testing is an important topic during this phase, especially since the production yields of chips may be rather low for new processes. Thus, excessive testing is used on chip level to ensure the proper functional operation of the IC. As such, wafer testing is used to ensure the correct electrical operation of the chip, while module testing is employed to test the according functionality.

- **Preparation (Phase 2)**: Here, all manufactured components are assembled to the final device. During this process, the common device data is loaded, including configuration settings, files, and secret keys which are shared by all devices in a production batch. In contrast to Phase 1, this phase is usually carried out by a single company [49, 70] called personalization company. For security reasons, the manufacturing of the hardware must be kept completely separated from loading the device data even if the same company is responsible for the manufacturing and the personalization. The loaded device data usually contains sensitive data which has to be protected properly.

- **Application Preparation (Phase 3)**: In contrast to Phase 2, this phase is dedicated to loading the device-individual data. This can also include the visual personalization of the device if – for example – unique identifiers are engraved on the body housing. During this process, device-individual secret keys are generated and loaded. As in Phase 2, this data has to be encrypted so that it is not exposed to any operator or third party during the production. In the case of symmetric key material, the personalization company is further responsible for a secure distribution of the generated data to the according stakeholder. In general, the process of loading additional application(s) requires authentication of the operator to the device in order to ensure that only authorized personnel can load data to the devices. This phase can be carried out by an external company or even by the customer in the field (see Phase 4).

- **Usage (Phase 4)**: During this phase, the secure CPS or IoT devices are used by the end customer. Depending on the use case, this phase also includes the loading of additional applications or the deactivation or deletion of applications. Thus, confidential data may be generated and loaded to the system by the customer. To ensure that this mechanism is protected, the customer needs to be aware of cryptographic keys which are used to securely communicate with the device. In the case of sym-

metric channel encryption, there are two possible ways to personalize the according key. The first way is that a key which is securely shared with the customer (e.g., via mail) is generated by the personalization company during Phase 3. The second way requires that the customer or vendor of the product shares the encryption key with the personalization company. Either way, it has to be ensured that no single operator or man-in-the-middle gains knowledge of this key during the personalization process.

- **Termination of Usage (Phase 5)**: Usually, devices are thrown away by the end customer if they are no longer needed. Nevertheless, a reuse of the device or components is principally possible if the device is returned to the vendor. To do so, the customer needs to deactivate all the applications and delete the sensitive, confidential data. The latter should always be carried out, even if the device is disposed of, to ensure that malicious third parties may not gain knowledge of any secret device data.
- **Disposal (Phase 6)**: As already summarized in Phase 5, the secure device should completely be erased before disposal to ensure that the sensitive, confidential data is not leaked. As stated by Rankl and Effing [49], the recycling of such devices is an interesting topic since rare components like gold and others are part of the devices.

2.3 Common Criteria

In the domain of information security, the Common Criteria (CC) [17] is widely used to evaluate the security measures. The CC defines a common set of requirements that need to be implemented by the security functionality of the Target of Evaluation (ToE) (Paper C). Thus, the evaluation process creates a level of confidence to the security functionality which is implemented in hardware, software, or both. For higher security levels, the CC evaluation also considers the maturity of the development processes, the production processes, as well as the used toolchains. The assurance level of the CC is rated on a scale from Evaluation Assurance Level (EAL) 1 to EAL 7, where EAL 1 is the lowest level and EAL 7 is the highest level. The key components and stakeholders of such an evaluation are listed in the following (Paper C):

- **Evaluation Facility**: Is responsible for testing and evaluating the implemented security functionalities of a ToE. Tests and evaluation methods are derived based on an ST. The results of the evaluation are collected in form of an Evaluation Technical Report (ETR). Based on this report, a certification body issues the CC certificate.
- **Target of Evaluation (ToE)**: Is defined *"as a set of software, firmware and/or hardware"* [17] that is the target of a CC certification. The ToE describes the whole system and the according configuration. The customer's configuration freedom of

the final product may lead to problems during the evaluation process, since all possible configurations of the ToE must meet the defined security requirements. As a consequence, it is often the case that the configuration options are limited to "meaningful" configurations. The limitations are documented within the ToE in form of guidance documents. Using the ToE beyond the defined guidance leads to a loss of the certificate. The ToE is compiled by the vendor(s) of the product.

- **Security Target (ST)**: The ST is a description of the *"implementation-specific statement of security needs for a specific identified Target of Evaluation"* [17]. It describes the assets, their threats and the implemented countermeasures. During the evaluation process, it is determined whether the stated countermeasures are sufficient to counter the threats. Countermeasures can be divided into two separate groups:
  - Security objectives for the ToE: This/these countermeasure(s) are directly implemented by the system. The correctness with respect to the threats and risks is determined during the evaluation process.
  - Security objectives for the operational environment: This/these countermeasure(s) is/are not implemented by the system but need to be provided by the operational environment. The correctness of this/these countermeasure(s) is/are not determined during the evaluation process.

  The ST is written by the vendor(s) of the system.

- **Evaluation Technical Report (ETR)**: Is a document which is assembled by the evaluation facility during the evaluation process. It documents the overall verdict of the evaluation facility and justifies this decision based on the collected evidence of the implemented security mechanisms. It is submitted to a certification body which issues the certificate in case of a positive attestation.

- **Evaluation Processes**: The CC standard and its supporting documents define formal and informal evaluation processes. Formal processes are explicitly defined in the standard, whilst informal processes are not explicitly defined. As a consequence, informal processes strongly depend on the instructed evaluation facility.

- **Protection Profile (PP)**: To allow groups and communities of interest to express their security requirements, the CC defines the concept of Protection Profiles (PPs) [17]. While the ST always describes a specific ToE, a PP is designed to describe a group of ToE so that it can be reused in different STs. A ToE is either fully compliant to a PP or non-compliant. Furthermore, being compliant to a PP does not necessarily mean that a specific EAL is reached since the PP describes the common requirements for a group of ToE and not a single specific one.

2.3.1 Evaluation processes

With respect to this thesis, modular and incremental evaluation processes are usually used for the certification. The following list gives an overview of the most common processes (Paper C):

- **Delta Evaluation** (informal process, defined in [18]): The delta evaluation is a CC certification process used to maximize the reuse of previously compiled evidences. To do so, the standard specifies that the following documents need to be shared between the evaluation facilities:
  - Product and supporting documentation
  - New security target(s)
  - Original evaluation technical report(s)
  - Original CC certificate
  - Original evaluation work packages (if available)

  Providing these data means that the current evaluation facility should not have to repeat the analysis of parts of the system where the requirements have not changed nor been impacted by any other change. Such changes are identified by performing a delta analysis between the old and new ST. A drawback of this approach is that the ETR contain information about the evaluation process and the applied measures to prove the security objectives of the ToE. As a consequence, ETR are usually considered as proprietary of the evaluation facility.

- **Composite Evaluation** (formal process, defined in [19]): Although the composite evaluation was designed for smart card products, it can be applied to a wide variety of products which fulfill the condition that *"an independently evaluated product is part of a final composite product to be evaluated"* [19]. The only restriction is that *"The composite product is a product consisting of at least two different parts, whereby one of them represents a single product having already been evaluated and certified ... The underlying platform is the part of the composite product having already been evaluated"* [19]. This means that a Layers pattern [10] is used for the overall system architecture. Compositional evaluations are technically similar to delta evaluations, but with the difference that additional conditions need to be fulfilled by the overall structure of the product. Due to these restrictions, it is not necessary to share the ETR. The lowest EAL of all components is the highest possible composite EAL.

- **Composed Evaluation** (formal process, defined in [17]): Is used to certify products which consist of independently certified (or going through an independent certification process) products/modules which are assembled to a new final product. It is similar to the composite evaluation but with the difference that the overall system

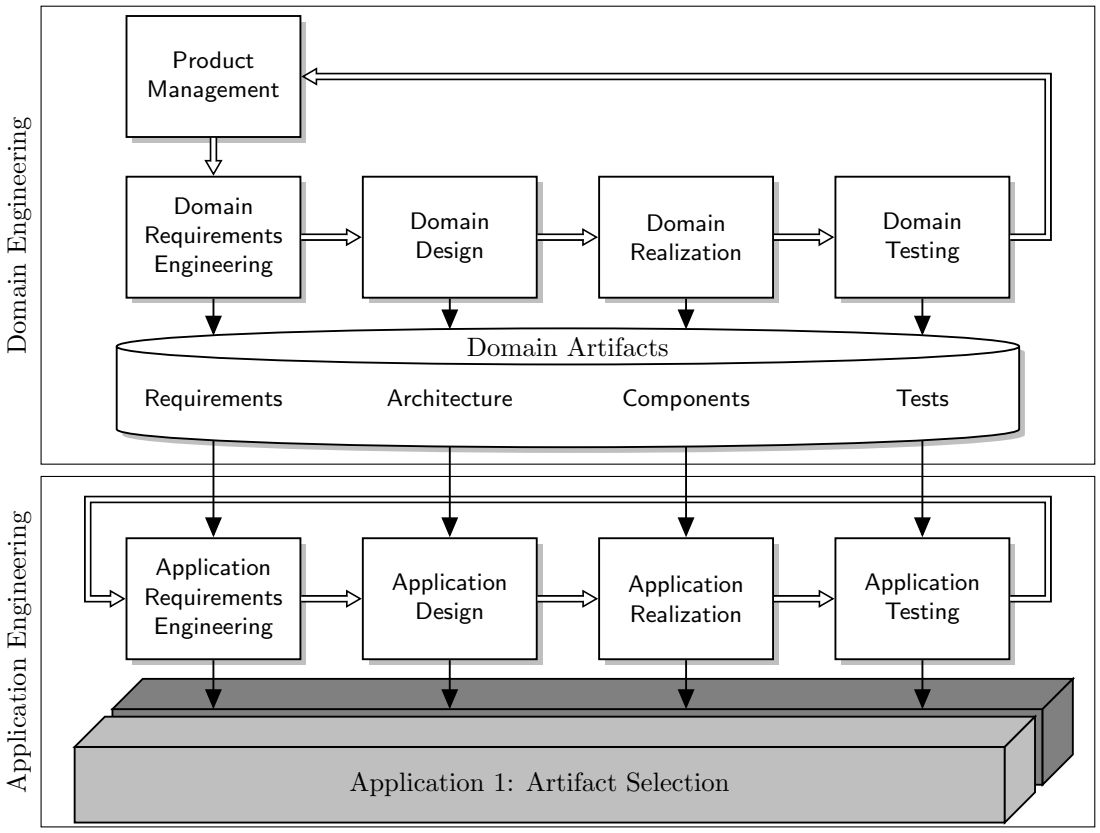


Figure 2.3: The software product-line engineering framework, adopted from [46]

architecture is not limited. Additionally, it is applicable for situations in which a delta evaluation is not possible since the ETR are not shared.

Since the individual components are already certified, the composed evaluation mainly focuses on the interface between the components and their according interaction(s). As a consequence, new evaluation assurance levels were introduced. These levels range from Composed Assurance Packages (CAP) A to CAP C, where A is the lowest level and C is the highest level. The assurance level CAP C stands for *Attack potential "Enhanced Basic"*, which is approximately comparable with EAL-4 (see [17], Part 3: pages 38 and 47). Due to this limitation, the composed evaluation has been performed much less successfully than other modular CC certification processes.

2.4 Software product-line engineering

Software Product Line Engineering (SPLE) applies the concept of product lines to software products. Thus, SPLE promises to create diverse, high-quality software products of a product family in a short time and at low price [46]. Instead of writing software for every individual system, a Software Product Line (SPL) is used to automatically generate software products by combining the required domain artifacts. The general framework is illustrated in Figure 2.3. As stated by Pohl, Böckle, and Linden [46] and Weiss and Lai [73], the SPLE can be split into two main phases, the *Domain Engineering* and the *Application Engineering*, which will be described in the following sections.

2.4.1 Domain engineering

During the domain engineering, the domain artifacts, the variabilities and the commonalities of the according domain are identified and implemented. The modeling of the domain is usually carried out using a Feature-Oriented Designed Modeling (FODA) [32] to explicitly state all dependencies. Domain artifacts are reusable development artifacts like the software architecture, or software components including their corresponding unit-tests [46]. As illustrated in Figure 2.3, the domain engineering is split into five sub-processes. The *Product Management* is the first process and deals with the economic aspects of the product line and defines the product roadmap. During the *Requirements Engineering* process, reusable and model-based requirements are defined. Hence, the requirements are not specified for a single particular application but contain the common and variable requirements for all products of the SPL. During the *Domain Design* processes, the variable software architecture is designed and the variability model that was created in the previous process is refined. The architectural design further defines common ways to deal with variability in the *Application Design* and *Application Realisation* of the application engineering. Finally, all software components are implemented during the the *Domain Realization* process and are tested during the last process on a component level.

2.4.2 Application engineering

In the application engineering phase, the final products are created by combining the domain artifacts which were implemented in the previous phase. Similar to the domain engineering, the application engineering is split into the four phases *Requirements Engineering*, *Application Design*, *Application Realization* and *Testing*. In contrast to the domain engineering, the application engineering mainly focuses on reusing domain artifacts. Based on the current requirements of the product, specific domain artifacts are chosen and assembled to the final product. Ideally, the application engineering makes use of software generators to automatically derive product variants without the need of implementing any new logic. This enables a rapid creation of high-quality products within

a defined product family. The amount of reused domain artifacts greatly depends on the application requirements. Hence, a major concern of the application engineering is the detection of deltas between the application requirements and the available capabilities of the SPL.

3 Related work

I don't believe you have to be better than everybody else. I believe you have to be better than you ever thought you could be.

Ken Venturi

First, this chapter presents the state-of-the-art pre-personalization process that is used in the domain of secure Smart Cards. After that, publications of API related attacks will be introduced which could be used to break the Data Generation System of the personalization process. Furthermore, formal methods will be described which could be used to minimize the risk of API-related attacks. Finally, related work dealing with the topic of Software Product Line (SPL) for secure systems and Business Process Management (BPM) will be outlined. Portions of this chapter were previously published as Paper H [66], and Paper B [58], and have been reproduced with permission.

3.1 State-of-the-art pre-personalization process

As stated by various standardization organizations like [6, 14], dedicated Hardware Security Modules (HSMs) are recommended for a secure generation and storage of cryptographic data. HSMs are tamper-resistant hardware modules, designed for generating random numbers and calculating cryptographic functions. Additionally, the hardware is optimized to accelerate the computation of cryptographic functions to ensure that a large amount of data can be processed securely and efficiently. Thus, HSMs are the best option for generating the required confidential personalization data in a secure and efficient way.

Based on the requirement to use HSMs, Chan and Ho [11] defined a system for a secure personalization of smart cards which is illustrated in Figure 3.1. As shown in the figure, HSMs are utilized to generate and to protect the data during the entire personalization process. This includes also data that is shared between the customer and the issuer of the product. The *Personalization Equipment* generally consists of special hardware which ensures that the data is sent to the individual chips cryptographically protected and authenticated [12, 31, 49]. The *Issuer System* is an IT infrastructure via which order and product-specific data and settings are shared with the personalization company. The shared data can also contain static (confidential) data that is provided by a stakeholder (e.g., a mobile network operator).

The data that is shared by any stakeholder with the *Issuer System* has to be encrypted so that the data is not leaked during the process. In the case of symmetric encryption,

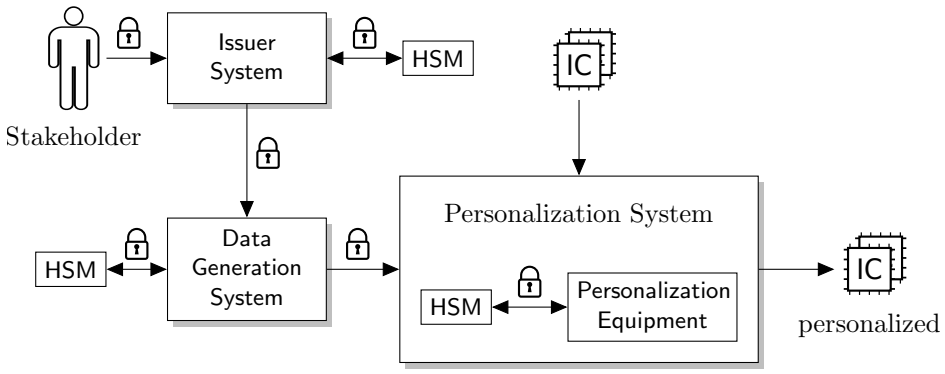


Figure 3.1: Principal structure of a system for an in-line personalization system, adopted from [11]

the corresponding protection key needs to be transported separately from the data as recommended in [14]. Moreover, the Payment Card Industry Security Council (PCI) [14] foresees that symmetric keys are split into multiple shares which are sent in multiple independent ways. Only after successful reception of one share, the next share is sent. Usually, such key exchange processes are performed several days or weeks prior to the production.

The *Personalization Equipment* usually executes scripts that are prepared prior to the production by the personalization company [25]. The script contains placeholders for the data that is either generated from the *Data Generation System* or directly provided by the *Issuer System*. In the case of directly provided data, the data is re-encrypted for the *Personalization System*. For security reasons, the API of the *Data Generation System* is tailored for single products in order to reduce the risk of API related attacks (see Section 3.2).

The investigated personalization process is based on the concepts described in this Section. In difference, a generic solution is introduced for the *Data Generation System* in order to support a wide variety of different products and product families. In order to allow flexibility while also ensuring high security requirements, a formal verification tool was developed and integrated into the process in order to verify security requirements during the process execution (see Section 5.1).

3.1.1 Establishing an authentication token

To start the personalization process of an individual chip, the personalization system has to use a secret authentication token so that the personalization of the chips can only be triggered by authorized personnel/equipment. The authentication token has to be inserted during an earlier and independent phase of the production. Rankl and Effing [49] describe a simple but common approach from the smart card domain:

During the production of the secure IC, the producer of the Operating System (OS) incorporates a secret value into the Read-Only Memory (ROM) code of the IC. The ROM contains the static functions and data which were already programmed during the lithography process of the production process. Thus, the code is hard-wired into the memory. To prevent attacks on the ROM code, the code is protected against optical attacks, including etching attacks. The secret ROM value is combined with a random value which was written into the non-volatile memory (Electrically Erasable Programmable Read-Only Memory (EEPROM) or Flash) of the chip by the semiconductor manufacturer to create an authentication token. The idea is that neither the OS producer nor the semiconductor manufacturer has enough knowledge to generate this authentication token on his/her own. The described method can be used to generate authentication tokens which are valid for one production batch. The method can be refined if a chip individual number is combined with the secret values. Thus, the authentication tokens are only valid for specific chips. This increases the level of security in case the chips have fallen into the wrong hands before having been completed [49].

In the scope of this thesis, an NXP proprietary approach was used in order to establish a first authentication token. Since the method itself is not relevant for this thesis – as long as the overall security requirements are fulfilled – it can be assumed that the approach from Rankl and Effing [49] was used.

3.1.2 Detection of manipulated devices

Another attack that has to be prevented in this scenario is the manipulation of the IC during the production process. An attacker with sufficient resources (e.g. a national agency or a competitor) may be able to smuggle manipulated ICs into the personalization process. These manipulated ICs function like the real product but with the difference that e.g. additional functionality is integrated into the OS, like dump routines, which can be used to retrieve the personalized confidential data in plain. Rankl and Effing [49] describes a simple, but common approach form the smart card domain in order to detect manipulated devices:

The semiconductor manufacturer stores a chip individual key in the non-volatile memory during the production process. The chip individual secret key and the unique chip identifier are securely shared with the personalization company. Before sending the personalization data to the card, the personalization system transmits a challenge consisting of a random number to the chip. The chip answers this request by computing a Keyed-Hash Message Authentication Code (HMAC) using the secret key. The message that is hashed consists of the ROM code and the received random number. The random number is necessary to prevent replay attacks. Moreover, the unique chip identifier is retrieved from the card. Consequently, the personalization system is able to verify the retrieved HMAC by comparing it with a recomputed value. Thus, manipulation of the ROM code can be

detected. The verification takes place in the HSM of the personalization system. Thus, it can be assured that the secret key is never exposed to any operator.

3.1.3 Loading of common and individual data

The procedure of loading data during the personalization process can be classified into two main groups: The first category requires the secure IC to perform basic file and data management commands such as *Create*, *Install* and *Update*; and the second category requires the secure IC to load the personalization data from a defined memory region after issuing a specific command [49, 66]. The following methods are currently state-of-the-art:

Loading personalization data using logical addresses [49]: This method is employed to avoid the specification of the real physical addresses by using symbolic names so that the secure IC is able to identify the real physical address. The benefit of this approach is that the data can be sent independently form the used micro-controller if the API is written in a platform-independent way (i.e., fixed endianness, etc.). The drawback is that the personalization process takes more time since the IC needs to resolve the symbolic identifier before the data is written. This overhead is small, but in high-volume production processes, it can lead to huge costs.

Loading personalization data using physical addresses [49]: Opposite to the first method, the real physical addresses of the individual datasets are used to write the data. In order to write this data, the personalization system has to consider how the data is expected (e.g. endianness) by the device. This is a huge drawback since it is an additional source of error which can render all produced chips unworkable. To be able to generate the data as expected, mostly, a sample device is used which contains a dump memory functionality. The sample device is configured as the final product, but with the difference that specific pattern values are used for the device-individual data. After the initialization, the memory content is dumped and the physical addresses are identified by searching the according pattern values. The memory dump is used as a template during the production of the secure ICs. Only the placeholder data is replaced during the personalization process with the real data. The critical aspect is the process which needs to assure that it is not possible to manufacture a production device having the dump memory functionality inside. Otherwise, an attacker is able to read confidential data from the dumped memory.

Loading personalization data using a dedicated personalization memory region (Paper H): The last common method for loading personalization data is using a dedicated memory region into which the personalization data is written as a whole block. The written data can even be loaded during the last stages of the production process. To actually load the data, the personalization system authenticates to the IC and sends the decryption key necessary to decrypt the loaded data. Additional meta-information is necessary so that the IC can determine the purpose of the data. Generally, the meta-information uses symbolic names which can be interpreted by the secure IC, but also real physical addresses could be used in this process. The personalization company together

with the OS producer needs to agree on the format of the data which is one of the drawbacks of this approach. The additional computational overhead that is required to process the data is compensated since the communication overhead between the personalization system and the IC can be kept as low as possible. The biggest drawback of this approach is that the memory region for the personalization data needs to be allocated by the secure IC which can be a problem for small, low-cost ICs. Of course, this additional memory can be used for customer data during the in-field use, but during the production process, this may be a limiting factor.

In practice, the first or third method is usually used for low-volume or prototype production processes, while the second approach is used for high-volume processes where the financial benefit is higher than the additional risk. Consequently, the second approach will be used in this thesis.

3.2 API attacks

A large and growing number of embedded systems make use of dedicated security processors to compute cryptographic operations and to store confidential data [9]. Thus, a less secure processing unit communicates with the security processors via a dedicated API. As identified by Anderson [3] and Bond [9], the most common API failure mode of secure systems is that functions/transactions that are secure in isolation become insecure in combination. Especially if a trusted system communicates with a less secure system, an adversary can use any unexpected combination of function calls in order to break the trusted system. One prominent example for such an API attack was the vulnerability of the HSMs which were provided by VISA [3, 4, 9]. In this scenario, it was possible to insert a known master key (having all zeros) into the HSMs by combining a specific sequence of API calls. After successfully loading the known key, they were able to export other confidential keys, like the PIN verification key, encrypted under the manipulated master key. By knowing the PIN verification key, a potential attacker is able to retrieve all PIN codes of any customer account of the affected bank.

In the context of the personalization process of secure ICs, API attacks can be used to break the *Data Generation System* (as illustrated in Figure 3.1) by maliciously executing a sequence of functions of the HSM API. Traditionally, such APIs are evaluated manually by experts during the development of the API. Due to the complexity of the systems and their interactions, failures which are hard to detect manually tend to arise. This also includes failures where information is leaked slowly during each interaction, leading to fatal failures when an adversary can inject many transactions per second. To circumvent the security flaws, the development of robustness principles is recommended. These principles are used to guide the designers of secure systems to reduce the complexity of the API to the essential transactions. Another alternative approach to evaluate the security of dedicated APIs is the use of formal verification tools, which will be discussed next.

3.2.1 Formal verification tools

Traditionally, formal verification was applied in the domain of hardware designs to ensure the correctness of the design with respect to its functionality [7]. Due to the increasing computational power of modern personal computers, the application in the domain of software development in large-scale industrial projects is now becoming more interesting [15].

One formal verification method which is of interest for this thesis, is the concept of abstract interpretation. The application of abstract interpretation is the static analysis of program code, that is the compile time determination of runtime properties of programs [15, 16]. Abstract interpretation uses a model of the actual program, using symbolic values for the input parameters which are mapped to specific properties. Thus, it is possible to show the correctness of a program with respect to its abstract model. The most important phase during this process is the generation of the abstract model: is the model incorrect or does not reflect all the program states, the assumptions about the bahaviour of the program are incorrect [16]. Thus, abstract interpretation always corresponds to a loss of precision, but with the benefit of getting a result in a feasible amount of time.

The second formal verification method that is of interest for this thesis is the concept of symbolic execution. Similar to abstract interpretation, the symbolic execution uses symbolic values as input parameter to analyze the program flow [34]. The execution proceeds as in the normal execution, but with the difference that symbolic formulas of the given input symbols are used as values. After execution, the program correctness can be ensured by applying the method presented by Floyd [22]: Under the assumption that the input parameter fulfill a defined predicate, it can be shown that the output value fulfills or violates a defined output predicate.

Since formal verification tools are consuming a lot of resources, a new approach was developed based on a method for symbolic execution to ensure that the Data Generation System of the personalization process does not leak any confidential data during the process (see Section 5.1).

3.3 Software-product-lines for business-process-management

As stated in a survey by Fantinato et al. [20], major challenges in the field of business process variability modeling are related to the reaction time of process changes and of the creation and selection of the best fitting business process variant. Starting from a business goal level, the framework by Valença et al. [71] presents a systematic way to reuse process goal models in variable process structures. He describes three methods of how to automatically track changes of the model by performing match detection [71]:

- **Perfect match**: The current variant is already part of the model. Consequently, no operation is performed during the model update.

- **No match**: The current variant is not part of the model. Thus, the variant is added as a new leaf to the according variation point.

- **Partial match**: The process variant – which is identified by the contained business goals – partially contains a sequence of business goals that are already part of the model. User interactions are often required to resolve such conflicts, since an automatic mapping is not always possible.

As a drawback of this approach, business-process models are considered as a linear sequence of business goals, which is practically not always the case. On the process model level, the PROVOP project (Hallerbach et al.[27, 28] and Reichert et al. [51]) define a framework to manage the variability of individual processes. This is achieved through well-defined change operations that are performed on a process model when deriving a process variant. Supported operations are the deletion, addition or moving of model elements or the adaptation of element attributes. This means that the basic process definition already expresses all possible variants at once, which can be hard to understand. Similar to the process level variability, a workflow model can be used to activate/deactivate specific actions within a process by using a dedicated modeling language [24]. The term workflow model is used for the specification of a business process which enables its execution in an enterprise and workflow management system. Extensions of the modeling language can be used to integrate roles and objects [52]. As a consequence, the variability of the process is not only addressed on the control flow level, but also on the ERP level.

In contrast to the related work, our approach reduces the overall process complexity by splitting up the process into layers with increasing details. Further, we use model based transformation techniques in order to combine the variability of business process models with the variability model of software-product-lines (see Section 4 and 5).

3.4 Software-product-lines for secure systems

Generally, managing requirements for SPL can be classified into two main categories: approaches based on specialized notations, and approaches that rely on extending existing notations [56].

The key concept of the first category is the FODA approach [21, 32, 33]. As such, FODA-based approaches are centered on the feature model which usually is of a tree-like form. This tree representation is used to explicitly state the dependencies of each feature. For example, two features are alternatives, one feature is optional or mandatory, and additional restrictions like one feature excludes or requires another feature.
The key concept of the second approach is an orthogonal variability management [5, 46], where the variability is modeled in a separate variability model instead of an integrated view.

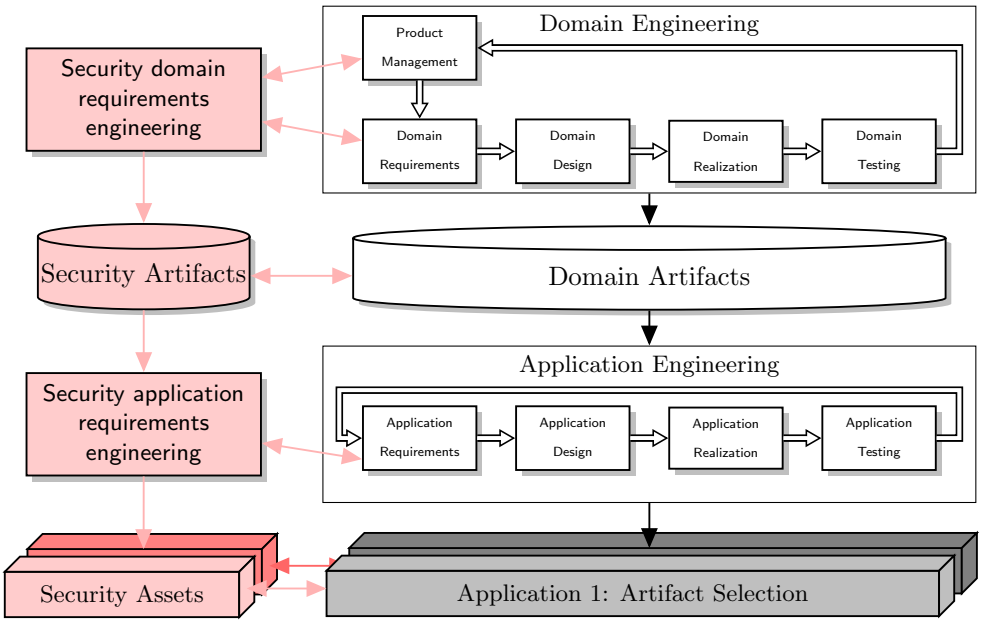


Figure 3.2: Security requirements extensions of the SPLE framework, adopted from [40]

A similar distinction can be made for security requirements management within the scope of an SPL. As stated by Mellado, Fernández-Medina, and Piattini [38, 39, 40], an orthogonal variability management process for security requirements enables a systematic and intuitive treatment of SPL security requirements. As such, the basic SPL framework – which was described in Section 2.4 – is extended with two additional subprocesses to link the security requirements to the product line. The additional parts are highlighted in red in Figure 3.2. The authors designed the subprocesses to be compliant with the Common Criteria standard.

The *Security Domain Requirements Engineering* process is used to develop common and variable security requirements and their precise documentation in the form of a CC PP. The PP is used as the generic security framework which every application specific Security Target (ST) has to be compliant with. Furthermore, security artifacts are derived from the requirements which are implemented during the domain realization of the product line.

The *Security Application Requirements Engineering* process is used to select the application-specific security requirements and their according security artifacts. This selection is documented in the form of the ST. To reduce the evaluation time for new applications, the differences between the security application artifacts and the security domain artifacts are identified and analyzed using a *Delta Analysis* (cf. Section 2.3.1).

Findings of this delta analysis are further used to increase the security requirements quality for subsequent projects [69].

In difference to the related work, our approach aims for a generic product line which uses formal verification tools in order to ensure security requirements during the product creation process. Consequently, the personalization system does not need to undergo a re-certification for new products / product families as long as the formal model is not changed. Since the scope of this thesis is a CC certification for Smart Cards or similar devices, the certification has to be renewed after it has expired [17, 19].

4 Process-driven software configuration

Understanding variation is the key to success in quality and business.
W. Edwards Deming

As described in Section 3.1.3, the pre-personalization process for high-volume production processes is driven by a template approach. The template – abbreviated as Production Template (PT) in the following – is a memory dump of a device with the same configuration as the production devices, but using specific pattern values as placeholder data. This PT functions as model for the production process. Only the pattern values are replaced during the pre-personalization process. Using a PT process has several benefits: the amount of data that is loaded during the production process can be kept as small as possible; and the product can be tested using a dummy configuration prior to the production. The verification of the product configuration can even be carried out by the customer to test the requested features. Only if the product adheres to the requirements, the green light is given for the production. Moreover, it is also possible that the Customer prepares such a PT on his own and provides it to the personalization company. A drawback of this approach is that the PT needs to be prepared manually by domain experts. Writing pre-personalization scripts is an error-prone task, especially for complex product configurations that may have a huge number of individual personalization steps. Furthermore, security certifications (CC certification) require that every product configuration has to be traced back to the customer requirements. This is necessary to create evidence that the provided customer data is handled in a protected way during the process, and in addition, is not confused with any other customer data. To solve the listed issues, we developed frameworks to cover the variability of the business processes and the software toolchain. In order to trace the product requirements from the process model to the actual software implementations, we linked the two variability models so that every single customer requirement can be traced back to PT configuration steps. This approach will be explained in more detail in the following Sections. Portions of this chapter were previously published as Paper B [58], Paper D [59], Paper G [60], and Paper F [65], and have been reproduced with permission.

4.1 A product line for business process models

A business process can be seen as a sequence of specific activities or subprocesses which need to be executed in a dedicated sequence to produce output with value to the customer

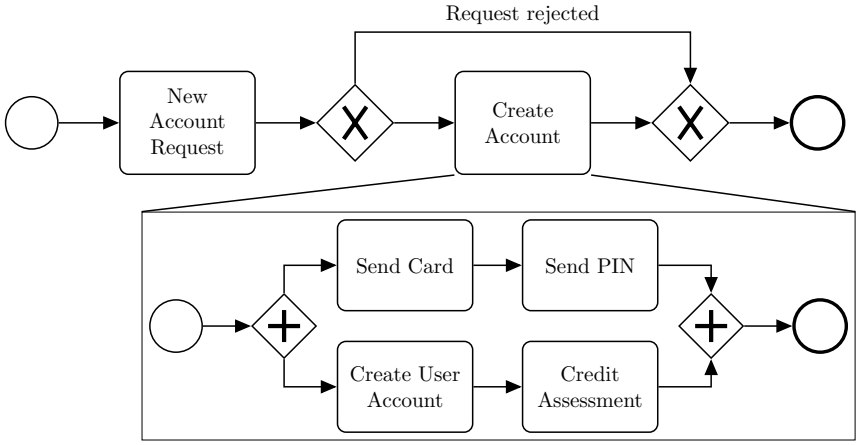


Figure 4.1: Exemplary layered business process showing a bank account creation request

[30, 65]. According to Österle [43], the process design is split up into different layers: The top layer (macroscopic level) is split into subprocesses, which are detailed out in the lower layers until the microscopic level is reached. This is achieved when all tasks are detailed enough so that they can be used as work instructions. An exemplary process showing a bank account creating request is illustrated in Figure 4.1. As shown in the Figure, the top level is a highly abstract description of the request process, while each subprocess is further described in the lower levels. For example, the Credit Assessment process can even be further refined with an additional process description. Variability of such a process structure can be expressed in two ways (Paper B): a variable structure of the process/subprocess by manipulating the sequence of tasks (adding/removing/shifting tasks); or by replacing the (sub)process refinements with different processes. With respect to the scope of this thesis, we used the following elements of the Business Process Model and Notation (BPMN) [42] to express business processes (Paper F):

- **Events:** Is something that occurs during the execution of a process or subprocess. Events affect the flow of the process and usually have a cause and an impact. For example, the start of an Activity and the completion of an Activity are typical Events. According to the modeling guidelines [42], Events are used only for types that affect the sequence or the timing of Activities of a process.

- **Activities:** Is an amount of work that a company or organization accomplishes during the execution of a process. Two different types of activities can be distinguished: An activity not broken down to a finer level of activities is called an atomic activity (i.e., a task). Sequences of tasks (e.g., a subprocess) belonging to the same activity are called non-atomic activities.

- **Gateways:** A Gateway controls how the process develops through different sequence flows. Each gateway can have multiple input and/or output paths. An example is a decision, where one of many possible alternative paths is selected based on a specific condition. Another example is a parallel Gateway which splits a single flow into multiple flows which are executed in parallel.
- **Data:** Data objects represent information flowing through the process such as documents or e-mails. A Data object can be required by a specific Activity as input parameter which has to be provided internally or by an external party involved in the process. Data objects which are generated by a specific Activity are called Output Data.
- **Pool** and **Lanes:** Are used to model responsibilities for specific Activities or sequences of Activities in a process. The responsibilities can be assigned to an organization, a specific role, a system or even a dedicated employee.

Our developed framework – to model the variability of process models – can be split into four different phases. These four phases are illustrated in Figure 4.2 and will be described in the following.

Process Modeling: In the first phase, process designers use a BPM tool of their choice and create process templates; that is, defining the sequence of steps of subprocesses using the BPMN notation. Furthermore, they add artifacts to the BPM Tool for required features like documentation artifacts, responsible workers or resources. As indicated in the Figure, the design of (sub)processes and the creation of the according domain model go hand in hand in an iterative manner.

Domain modeling: In the second phase, the created processes are imported into the SPLE tool and added to a feature model. During this process, the domain engineer (Process Designer) chooses the set of available (sub)processes and defines which parts of these processes are variable. Consider the following example, a company responsible for forming metal uses different production planning strategies for different customers. E.g., for customer X, the company employs event-driven Kanban, and for customer Y, the company uses Kanban with a quantity signal. As a consequence, the principal sequence of the production steps is the same, but the production planning is scheduled differently based on the used Kanban system. Thus, the domain engineer chooses the production planning strategy to be variable by defining a Variation Point (VP). He deposits the different Kanban implementations as possible variants for the VP so that two process models can be generated. To limit the possible configuration space, the domain engineer can define a comprehensive set of rules and restrictions so that only meaningful and valid process variants can be derived.

Feature Selection: In this phase, production experts use the defined set of process models (within the SPLE tool) to derive process variants for their current requirements. This is done by selecting the wished features from the feature model which was created in

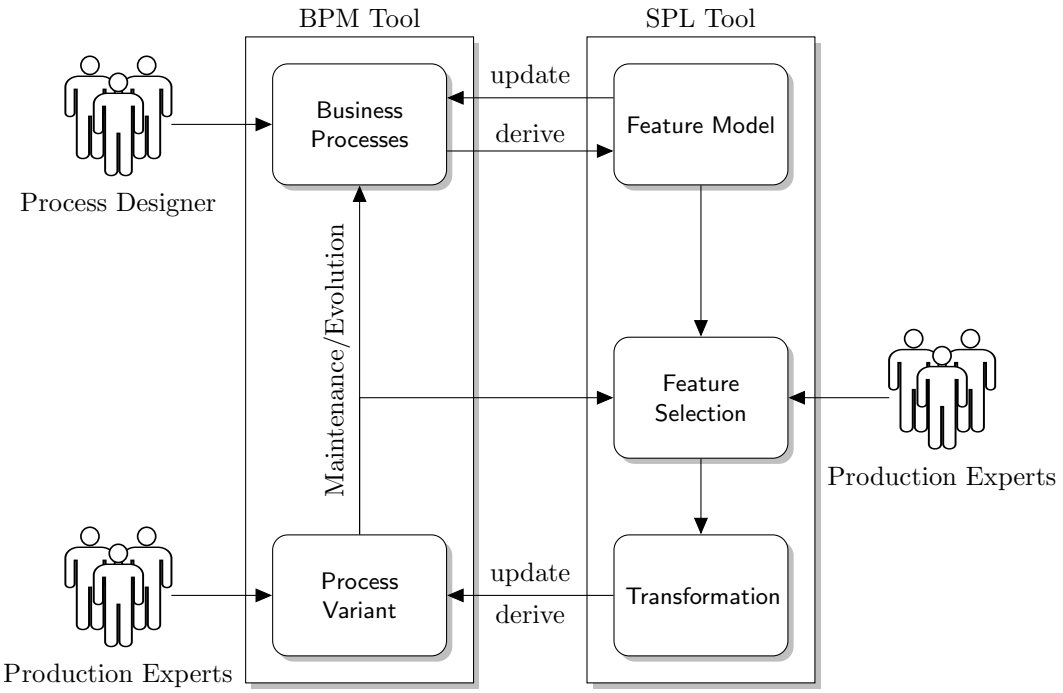


Figure 4.2: Product-line framework for the creation of business-process models (adopted from Paper B)

the second phase. For example, the production experts will choose event-driven Kanban if customer X has placed an order and will choose Kanban with a quantity signal if customer Y has placed an order (Paper F).

Maintenance and **Evolution:** In the last phase, the derived processes are used in the production and observed by production experts. Based on the collected data, the experts can either improve the feature selection of the used process (i.e., iteration back to phase 3), or issue a process improvement process (i.e., iteration back to step 1). For example, during the production for customer X, it was observed that event-driven Kanban was too slow to react to the customer needs. As a consequence, the production experts changed the production planning strategy to quantity-based Kanban to tackle these problems. Another possibility could be the observation that quantity-based Kanban was too general. For example, the production experts recognized that only one bin Kanban and three bin Kanban are valuable for the production processes. Consequently, new processes are designed and integrated into the existing feature model.

4.2 Software-product-line for product configurations

To overcome the issue of complex product configurations, we developed a framework for modular PT pre-personalization based on library scripts. These library scripts are actively configured via a software-product-line. The concept is illustrated in Figure 4.3. As shown in the Figure, a Domain Expert is responsible for selecting the product features which are mapped via specific generator rules to the modular personalization scripts. During the feature transformation, the modular script blocks are instantiated and configured with the specific product configuration which is provided by the Domain Expert. The resulting pre-personalization script contains dummy data with which the product configuration can be tested. The rules that are used to map the feature model to the PT personalization scripts are mainly defined by the used platform and the implemented interface standards like the GlobalPlatform[1]. Thus, Domain Experts are required who are responsible for defining/maintaining the PT scripts and for maintaining the transformation rules. For certification purposes, each product configuration (i.e., the feature selection) is logged together with the applied transformation rules. This log can be used to manually verify that the resulting pre-personalization script(s) adhere(s) to the product requirements.

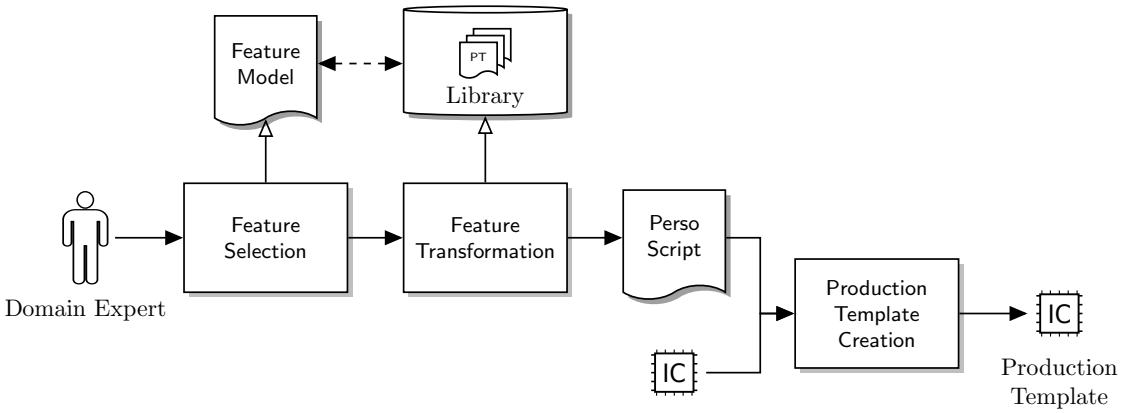


Figure 4.3: Framework for the generation of product configurations

4.3 Combining process variability and software variability

On the one hand, we have a system which is capable of reflecting the variability on business process level, and on the other hand, we have a system which is capable of reflecting the variability on product level. Both models basically represent the same variability on different levels designed for different stakeholders. As a consequence, we tried to combine

[1] GlobalPlatform is a non-profit organization that creates and publishes specifications for secure IC technology. https://www.globalplatform.org/

these two models using dedicated generation tools so that the configuration of the PT process can be directly mapped to the order process of the product. To achieve this binding, it is necessary that every configuration option is mapped to configuration settings within order entry forms (Paper F). These order entry forms need to be filled out by external customers (customer requirements), by internal stakeholders (system requirements) or a combination of both. To maintain an automatic link between the order process and the process model, the process model is tagged with additional information so that order entries can be generated automatically. The following type model was developed during this thesis (Paper F):

- **Inputs:** Are the abstract concepts of different input types which will be described below. Additionally, each Input is mapped to an input type defining the format of the input. For example, input data could be delivered as a file (structured or binary) or delivered in the form of configuration settings like strings, integers, etc. as well as cryptographically protected (encrypted and/or authenticated).
- **None:** No special data need to be submitted and, hence, a process node marked with none will not appear as a setting in the order entry form.
- **Customer Input:** Specific data need to be added by an external customer. A process node marked with this type generates an entry in the order entry form of a specific type. For example, a file upload button will appear if a customer needs to submit a specific file.
- **Internal Input:** Specific data needs to be added by an internal stakeholder. This information is directly forwarded to the internal stakeholder as a separate order.
- **Input Group:** A set of Inputs that are logically linked together. As a consequence, all of these inputs need to be provided for a single feature.

Moreover, the following default rules can be applied to support the domain experts in tagging process models which are based on the BPMN notation (Paper F):

- **Activities:** Non-atomic Activities can be used to group a specific set of input parameters as a single feature. For example, a process designed for configuring an application may require several input parameters. For the order entry form, all the input parameters should be displayed as one group of configuration settings. Any atomic Activity is automatically tagged as input type "None". This also applies if a non-atomic Activity does not contain any Data nodes.
- **Gateways:** Are used to define the structure of the generated order entry form. An example is a decision when the internal/external customer can choose between multiple customization options. Every branch appears as a separate group that can

be selected. If a branch does not contain any data, it will only appear as a checkbox that can be ticked. The system ensures that only one of the branches may be selected from the possible choices. In contrast, flows in a parallel flow are all mandatory.

- **Data:** Data to be provided by any entity involved in the process. The type of information needs to be added manually by the Domain Experts. With respect to the scope of this thesis, it was found that "string" is a meaningful default value.
- **Pools** and **Lanes:** Are used to define the responsibility of the input data. This means that data in a company-internal lane are automatically tagged with "Internal Input", while Data nodes in external lanes are tagged with "External Input". With respect to the scope of this thesis, it was found that "Internal Input" is a meaningful default value in case Pools and Lanes are not used.

All default rules can be manually overwritten by the Domain Experts. Furthermore, changes of the processes of the model will be traced and illustrated as diff-model so that changes can be reviewed by the Domain Experts before they are applied to the process feature model. If the process model is completely tagged (either manually or using the automatic rules), the according order entry forms can be generated. In general, web-based forms are commonly used to illustrate such configurable order forms, but every other format can be supported by implementing the required transformation operation. This systematic generation process can be used to derive the feature selection of the PT process

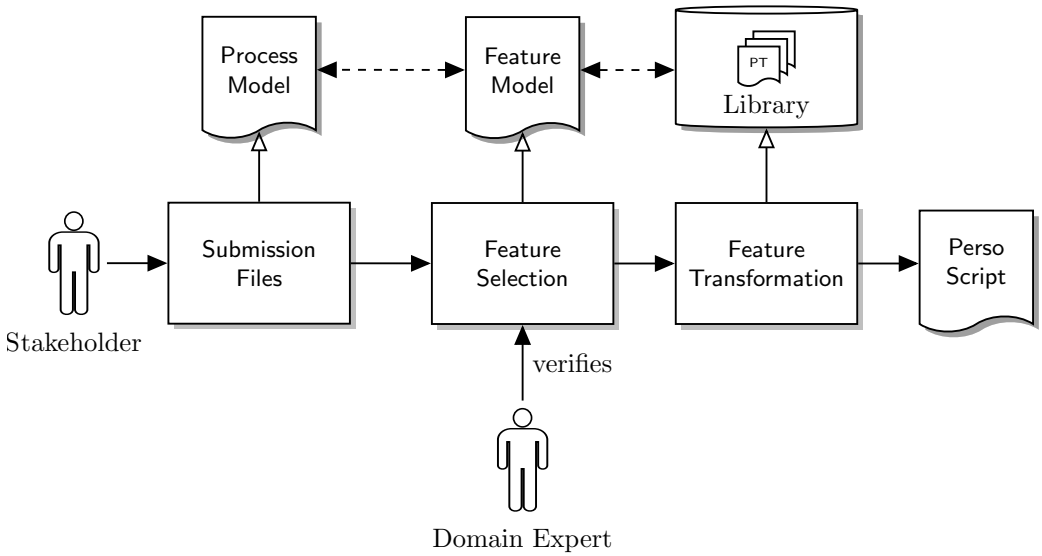


Figure 4.4: Process flow for the automatic generation of product configurations, based on order entries (adapted from Paper F)

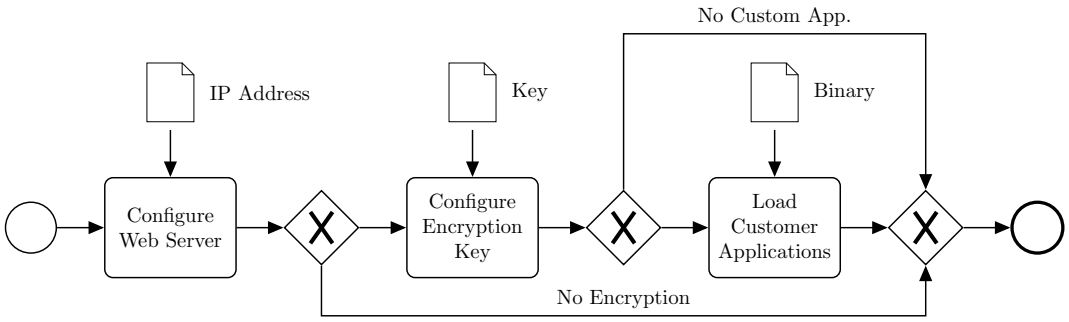


Figure 4.5: Exemplary sample process showing the basic configuration settings (adopted from Paper F)

in an automatic way (Paper F). For this purpose, the feature model of the PT process was mapped to the configuration items of the generated order entries. This process is a manual process, executed by a Domain Expert once for every supported product family. Thus, the defined mapping is valid for a wide variety of products. Especially for new products it is likely that knowledge on how to improve the efficiency of the process(es) is gained during the production process. This requires specific changes of the process model like the addition of new configuration settings, or a change of the sequence of process steps. In the case of changing configuration options (adapt, add, remove), the mapping between the process model and the feature model of the pre-personalization product line needs to be adapted as well. Since this feature model is valid for a wide variety of products, the maintenance costs of such models can be kept low. This is achieved since the improvements are rolled out for the entire product family.

The resulting overall process for creating pre-personalization configurations is illustrated in Figure 4.4. As illustrated, the feature selection is automatically derived from the submitted order entries of the internal/external customers. For certification purposes, it is necessary that the product configuration is manually verified even though automatic transformation steps are used. To overcome the limitation of the manual configuration verification, a tool evaluation could be performed. Since tool evaluations are cost-intensive processes with unknown outcome, performing a manual verification is less evil. After successfully verifying the product configuration, a digital signature is added to the configuration. As such, evidence can be generated to ensure that only properly signed – and, thus, verified – configurations were operated, meaning that the provided customer data is not confused with any other data or that incorrect features are instantiated.

4.3.1 Exemplary sample process

It is economically infeasible to pre-personalize each individual device manually if it is sold in high quantities. Furthermore, establishing personalization toolchains for dedicated

versions of the product may solve the issue of manually configuring products but leads to high maintenance costs with respect to the involved tools. Thus, a flexible system has to be implemented that ensures that variable parts of the process are reflected by variable parts of the software architecture of the pre-personalization toolchains. For illustration purposes, we will discuss a very basic pre-personalization process which is illustrated in Figure 4.5. The example shows the process of a company selling small embedded systems which are used as sensing devices for the IoT. The device is offered in three different variants with the following features (based on Paper F):

- **Version 1**: Senses the data in a given time interval and sends the recorded signal to a web server which is used for post-processing the data. In this first version, the communication channel between the web server and the device is unprotected.
- **Version 2**: Additionally to the basic features of the first version, this version allows the encryption of the communication channel between the server and the node using symmetric cryptography. As such, it should be circumvented that third parties are able to read the data in plain. For simplicity concerning this example, it is assumed that this key is provided by the customer and sent in plain to the pre-personalization company.
- **Version 3**: Additionally to the features of the previous versions, this device also allows customer applications to be run (e.g., data pre-processing routines) on the system. This requires that the customer submits a binary file which is loaded to every individual device to the pre-personalization company.

Establishing three different order processes using three different versions of customization toolchains will result in high maintenance efforts. To be able to define a stable software architecture that covers all three pre-personalization processes is the main goal of the Domain Engineering process which will be described in the following.

4.3.2 Domain engineering

As described in Section 2.4, the main goal of the Domain Engineering phase includes the identification and implementation of reusable Domain Artifacts. As such, we apply the transformation roles defined in Section 4.3 to identify the required artifacts. The default order entry (i.e., automatically generated by the process variability framework as described in Section 4.1) is generated by the following (Paper F): A string input box for defining the IP address of the web server and a decision whether to send the data encrypted or not. In case of encryption, a string input box can be used to specify the plain encryption key; if set, an option appears that allows a customer to load custom applications also as a string input box. The only problem of the default order entry process is the type of the Input "Binary". Thus, the Domain Expert adopts the Process Model by adding the hint that a

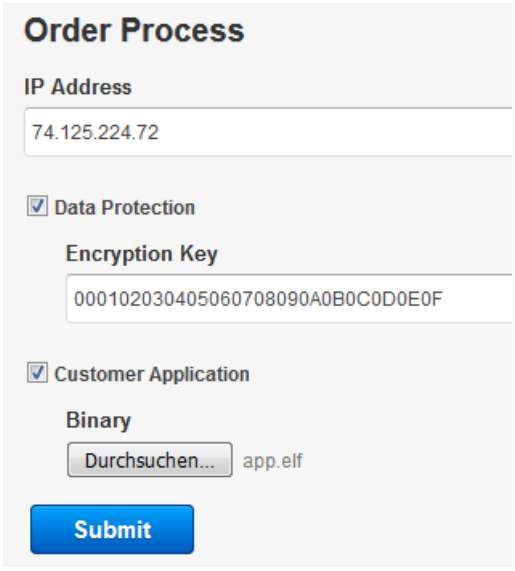


Figure 4.6: Order entry which is generated from the example process model (adopted from Paper G)

binary file input should be used. Consequently, a file upload button appears in the order entry with which binary files can be added. Additionally, the Domain Expert could refine the types of the other inputs so that runtime checks can be performed when the order entry is filled out by a customer. For example, rules can be applied so that the IP Address has to be valid, or that the key has the correct length and format (e.g., Base64-encoded). These additional hints are directly mapped to the processes using the framework defined in Section 4.1. Each node modeled in the BPM node has an unique identifier which is used for the mapping. Thus, reusing parts of processes in other processes will automatically propagate this additional information. The generated order entry is illustrated in Figure 4.6. The customization options for the Customer Application is only visible if the checkbox "Data Protection" is checked.

Based on the identified structure of the order entry, the features of the Feature Model can automatically be derived. The resulting Feature Model is depicted in Figure 4.7. As illustrated, the "Web Server" is modeled as a mandatory feature, while the "Data Protection" and "Customer Application" Feature is modeled as optional. Furthermore, the "Customer Application" requires that the "Data Protection" is selected. The "IP Address", "Encryption Key" and "Executable" nodes are used to symbolize the configuration dependency of the Feature which they are attached to. These configuration dependencies, however, are not real Features in the classical sense defined by Kang et al. [32]. Finally, the three identified main features have to be implemented in the form of PT scripts so that they can be instantiated during the Application Engineering. Principally, this would

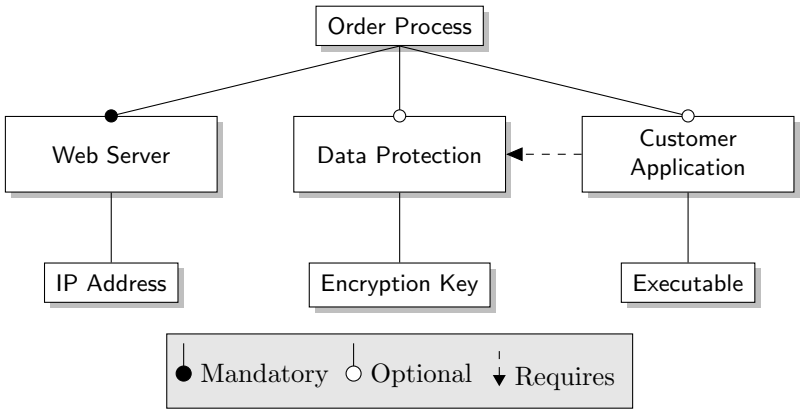


Figure 4.7: Feature Model that is derived from the exemplary process model (adopted from Paper G)

result in the desired output since the scripts for whole product configurations could be generated automatically. The problem with this approach is that every individual feature would result in a dedicated script designed for a specific hardware. Thus, a lot of different scripts would need to be developed although they share a lot of commonalities. As a consequence, an improvement was developed which enables a runtime-configurable script generation. This improvement will be discussed in Chapter 5.

4.3.3 Application engineering

As described in Section 2.4, the main goal of the Application Engineering is the combination of domain artifacts to whole products. As such, a software toolchain is required which is able to transform the order entries into feature selections. Based on these feature selections, code generators are used to generate the script for the final product. To be able to automatically derive the feature selection from the order entries, a file format has to be specified which can be interpreted by the following SPLE Tool. We have chosen zip archives as transport containers since they are easy to process and can help minimize the memory footprint. Each internal or external customer automatically generates such a submission file when the order entry form is filled out via the web-based front end. Additionally, a file is generated which contains the meta-information that was provided via the web form. We have chosen Extensible Markup Language (XML) as the file format. For example, the XML file illustrated in Listing 4.1 is generated for the order entry illustrated in Figure 4.6:

The platform attribute is a system attribute which cannot be set by the provider of the order entry. The attribute is used internally to identify how the data is processed during the script generation, which will be explained in more detail in Chapter 5. Data that is provided via the submission files is referenced by a file URL. Using the XML

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <configuration platform="PROD">
3    <WebServer>
4      <IPAddress>74.125.224.72</IPAddress>
5    </WebServer>
6
7    <DataProtection>
8      <EncryptionKey>0x000102030405060708090A0B0C0D0E0F</EncryptionKey>
9    </DataProtection>
10
11   <CustomerApplication>
12     <Binary>//file:app.elf</Binary>
13   </CustomerApplication>
14 </configuration>
```

Listing 4.1: Generated XML based on the exemplary Order Entry (adopted from Paper G)

format has two main benefits: First, it is human readable, which is a huge benefit if these configurations need to be verified by a human. And second, it is easy to process since many programming languages offer APIs for reading/writing XML files. Additionally, libraries can be found which are able to generate the class hierarchies based on the XML Schema File (XSD). As illustrated in the Listing 4.1, additional restrictions are not part of the XML file. They are only enforced during the creation of the submission files and during the script generation process.

In order to be able to generate the required PT script, a code generator has to be implemented which instantiates the developed domain artifacts (modular GS scripts) with the provided data. For each possible child element of the configuration XML (Figure 4.6), a mapping has to be defined for the modular PT scripts. In the case of the described example, three different scripts need to be present in the Domain Artifact repository: a loadData script, which is used to load the IP address of the web server to the chip; a putKey script, which is responsible for loading the encryption key to the chip; an upload and install script which loads the given customer application and installs it. The resulting product configuration script calls all the above mentioned scripts with the given parameter.

Reproducibility of results is a demanding topic for certification purposes. As a consequence, the generated script and the used modular GS scripts are archived in a database including the information about the executing operator and the submission files. Speaking of certification leads further to the topic of traceability: Every single customer requirement needs to be reflected in the product configuration. Traceability is achieved by the automatic mapping rules of the developed toolchain. Thus, every internal or external customer requirement is reflected in the according order entry and can be traced to the according scripts. To circumvent a time- and cost-intensive tool evaluation process, a Domain Expert is responsible to verify the final result (as illustrated in Figure 4.4). To guarantee that only valid configurations are further processed, digital signatures can be used.

5 A flexible runtime-configurable data-generation system

Stay committed to your decisions, but stay flexible in your approach.
Tom Robbins

Portions of this chapter were previously published as Paper G [60], and Paper F [65], and have been reproduced with permission.

Having a system which is able to generate product configurations based on a script library is a good starting point for product pre-personalization. Nevertheless, to support a wide variety of different platforms (i.e., Hardware and OS), a pure script-based approach would lead to a huge number of pre-personalization scripts which may share a lot of commonalities. Thus, the overall system would not be economically feasible since the maintenance costs of the script library would be too high.

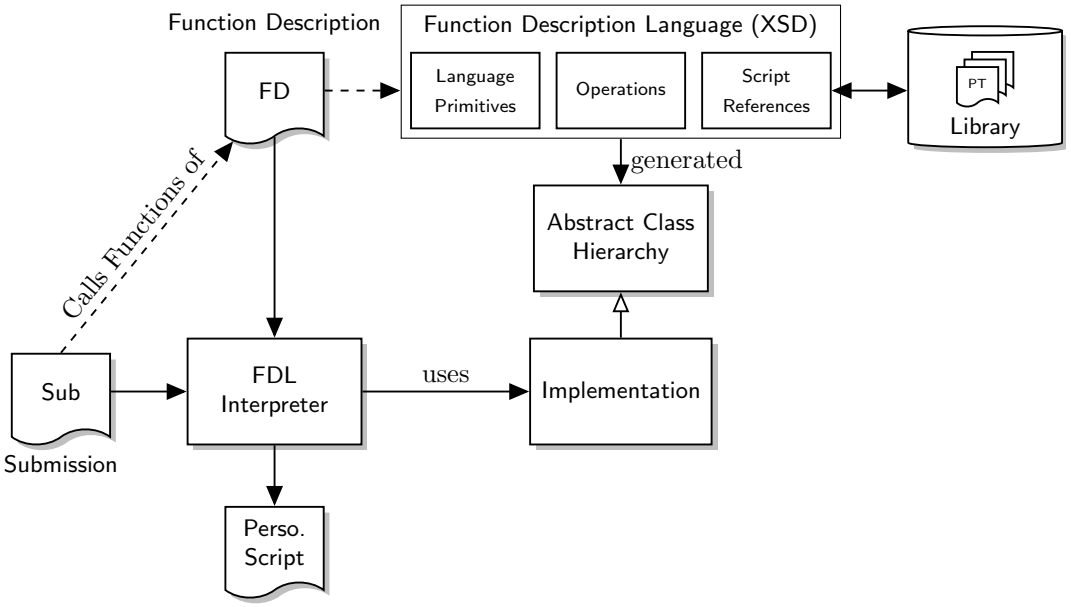


Figure 5.1: Framework for a flexible, runtime-configurable script-generation-system (based on Paper G)

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <FunctionDescription platform="...">
3    <Defines>
4      <!-- Global defines that can be used in this FD -->
5    </Defines>
6
7    <Macros>
8      <!-- Reusable Macros that can be called from Functions -->
9    </Macros>
10
11   <Functions>
12     <!-- List of Functions that can be called from Submission Files -->
13   </Functions>
14 </FunctionDescription>
```

Listing 5.1: Principal structure of a Function Description (FD) file

As a consequence, a flexible system is required which can be configured for the current requirements. Additionally, it has to be runtime-configurable to circumvent the implementation and deployment of new software releases, which may demand great effort. Especially the deployment of a new software release for protected production equipment may take a considerable amount of time. Usually, numerous independent persons are required to verify the release version and to load the software to the production equipment. Furthermore, a new software release may also require a re-certification process since the production equipment was changed.

To overcome these issues, the framework illustrated in Figure 5.1 was developed. Basically, it consists of three main components: The first component is the Function Description Language (FDL), which is a domain-specific language designed for the creation of pre-personalization scripts. The second component is the FDL Interpreter, which interprets a given submission file (Sub) and calls the referenced functions of a Function Description (FD) to generate the resulting pre-personalization script. The third component is a library consisting of small reusable PT scripts which are designed to work with a wide variety of different platforms. All three components will be described in more detail in the following.

Function Description Language (FDL): The FDL is a domain-specific language that is designed for the creation of pre-personalization scripts. The language is based on XML and uses a specific XSD to define the available language primitives and operations that can be performed on the input data. Each FD has to be compliant to the XSD. The overall structure of an FD is illustrated in Listing 5.1. Basically, the FD consists of three main blocks: Defines, Macros and Functions. Defines are read-only variables that can be used as global variables in every Macro or Function of this FD. Macros are reusable code segments that can be called in Functions. If, for example a specific post-processing action should be triggered after each function call, a Macro could be used to achieve this result. The third block is the most important block and contains the list of functions and

restrictions for each function. Specifically, the following restrictions are supported by the framework:

- **Function Choice**: A specific subset n-out-of-m possible functions has to be instantiated to form a correct product configuration. For example, if a function A and a function B are alternatives, either A or B has to be chosen (i.e., 1 out of 2), but not both.
- **Restricts**: A specific function may restrict the use of other functions. For example, if function A is selected, the functions B and C are not allowed to be present in the product configuration.
- **Requires**: A specific function requires that other functions are used as well. For example, in the illustrated example (see Figure 4.7), the function Customer Application requires that the Data Protection function is called as well.

The principal structure of Functions is illustrated in Listing 5.2. The configuration block basically needs to match the configuration of the generated order XML and can be verified using a generated XSD. The translate block is used to define the operations that the interpreter needs to perform in order to generate the according pre-personalization script. For example, the *WebServer* function – that is required by the described example – could instantiate a *loadData* script from the modular PT library which is responsible for writing the IP address to the chip. Available operations can be classified into two main groups: the first group defines operations which will directly result in instructions of the pre-personalization script (e.g., call of scripts, execution of commands, etc.); the second group defines operations that are only executed by the interpreter. For example, if the encryption key of the customer was provided using Base64 encoding, but the chip can only

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <FunctionDescription platform="...">
3    ...
4    <Function name="...">
5      <Configuration>
6        <!-- List of Parameter that need to be provided -->
7      </Configuration>
8      <Restrictions>
9        <!-- List of Restrictions -->
10     </Restrictions>
11     <Translate>
12       <!-- List of operations that are executed by the Interpreter -->
13     </Translate>
14   </Function>
15 </FunctionDescription>
```

Listing 5.2: Principal structure of a Function

interpret it as plain hex, the interpreter tool may convert the Base64 encoding into a plain hex encoding before instantiating a script with the key value.

5.1 A model-based formal verification tool

Generating pre-personalization scripts alone is not enough to cover all requirements of a modern, secure IoT device. As a consequence, a flexible system is required which is capable of generating the chip individual initial device setup. This device setup usually contains cryptographic keys in order to guarantee a secure and proper function of the device in the field. For example, on a SIM card for 3G communication, a symmetric encryption key has to be loaded to ensure that a mobile phone is able to communicate with the base stations of the Mobile Network Operator (MNO). The concept of such a data generation system is basically quite similar to the introduced method to generate pre-personalization scripts (see Section 5). HSMs are used to generate the required data by providing a dedicated API to the pre-personalization company. As mentioned in Section 3.2, introducing such an API can lead to security vulnerabilities which could be used to break the whole system. Thus, we developed a formal verification tool which aims to ensure security properties during the runtime of the system [8]. The basic concept is illustrated in Figure 5.2. As illustrated, the formal verification tool protects the input API of the HSM. The tool is based on a formal model and uses concepts of abstract interpretation and symbolic execution to derive this model. As a consequence, the tool is sound as long as the assumptions of the model are correct. Thus, we used a formal language to formalize the model such that it can be proofed using a standard theorem solver. Using this concept also helps in a security certification since evidences can be generated to show that the model is sound.

The execution of the formal verification tool can be split into three main phases:

- **Phase 1**: Classification of Input Data: During this phase, the input data is parsed and accordingly classified. Input Data can be classified in *constants*, *private keys* and *public keys*.
- **Phase 2**: Symbolic execution of Instructions: All instructions within one data generation script are executed. Each instruction defines specific rules that have to be met by the input or output data; for example, an encryption operation requires that a key object is used for the encryption operation and that the output buffer has a sufficient length. In case that the pre-conditions are not met, the execution of the script is rejected. If the pre-conditions are met, specific - instruction and input data related - properties are assigned to the output fields; for example, if a Cyclic Redundancy Check (CRC) is calculated over a key object, the output field will leak a specific amount of bits of the key.
- **Phase 3**: Investigation of Output Fields: After the symbolic execution of all instructions was done, all output fields are investigated to determine if confidential

data was leaked to one of the output fields. Each leak will be traced back to the origin and will lower the confidentiality of the origin. If the confidentiality is lower than a specific threshold, the execution of the script is rejected.

Additional to the classification of the input values, confidential data (like private and public keys) are further classified into several levels: *System Secret*, *Customer Secret* and *Generated Secret*. System Secrets are usually secrets that are not loaded to the device itself, but are required to protect the confidentiality of the device data during the pre-personalization process. For example, the transport protection key between the personalization equipment and the device is usually a System Secret. Customer Secrets are data that is provided by any stakeholder of the chip, including static keys and derivation master keys. Generated Secrets are random keys that are generated inside the HSM during the execution of the script. *System Secrets* and *Customer Secrets* are secrets that could be reused in several scripts. As a consequence, it has to be prohibited that even a single bit is leaked of such keys during the script's execution. Otherwise, these keys could be completely leaked through multiple runs. Quite the opposite, randomly generated secrets cannot be reused in several runs and, thus, a strict restriction is not necessary. Consequently, a random key can be considered as secure as long as the confidentiality of the key does not drop below a certain threshold.

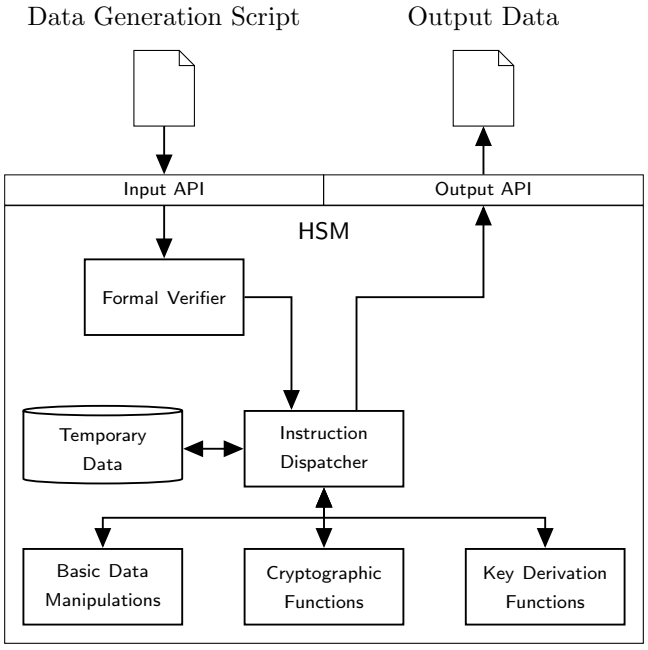


Figure 5.2: Concept of the formal verification tool to ensure system security requirements (based on Patent I)

6 Evaluation

Before software can be reusable, it has first to be usable.

Ralph Johnson

The results of the studied use case will be presented in this chapter. The implemented methodologies will be compared with the hypotheses of this thesis at the end of this chapter. Portions of this chapter were previously published as Paper B [58], and Paper D [59], and have been reproduced with permission.

6.1 Managing process variability

During this thesis, several case studies in different domains were adduced to investigate the performance and the overhead of our developed business-process product-line. The investigated domains are the manufacturing of car parts and the domain of customizing software for IoT or more generally embedded systems. To measure the performance of the framework, a sample process structure was developed and used by the respective domain experts. The following actions where investigated (Paper B):

- **Use Case 1:** The time was measured that a domain expert requires to create a new process variant by manually creating the process in the BPMN tool and by using our developed business-process feature-model. The manual process consisted of the tasks to assemble existing process templates to the whole business process. To get a better estimate, the experiment was repeated with several domain experts and different process setups which followed a common structure (illustrated in Figure 6.1). In total, the process structure had four Variation Points (VPs): two on the top level and two on lower levels with several variants. The results were divided by the number of VPs to get a rough estimate for the time saving per VP.
- **Use Case 2:** To get an estimate of the scalability of the developed framework, the experiment of Use Case 1 was repeated with a process database containing 200 additional unrelated processes.
- **Use Case 3:** To measure the performance regarding maintaining processes, a process template was changed and all created process variants had to be updated. For the manual process, the domain expert only received the name of the changed process template. As a consequence, this Use Case covers the case in which a process

designer is responsible for maintaining processes he has not created by himself. The changes applied to the process template included adding/deleting a task in a process template and altering the task sequence in a process. The resulting time for the manual maintenance was divided by the number of processes to receive an estimate of the maintenance effort for a single process.

- **Use Case 4:** An extension of Use Case 3: In this case, the domain expert received a list of process names that are affected by the change. Thus, the Use Case gives an estimate of the maintenance effort if the processes are maintained by the expert who was responsible for developing them.
- **Use Case 5:** A new Process Variant was created and had to be included in the existing process feature model. Additionally, a new process had to be derived based on this additional process variant. The new variant was an addition to "VP 3" as illustrated in Figure 6.1.
- **Use Case 6:** In this case, the initial Feature Model was created (see Figure 6.1) to measure the initial overhead of creating a process model.

The above use-cases were done with a focus group of fifteen people and averaged in order to get the final results. All participants were trained in order to be able to use our

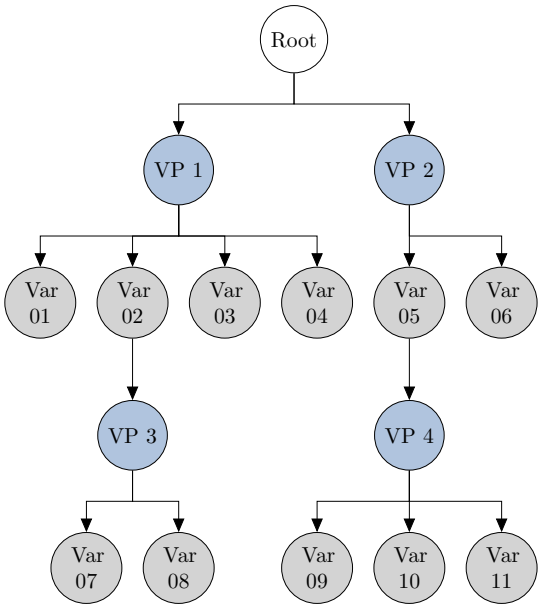


Figure 6.1: The variability structure of the process feature-model for the investigated use cases (Paper B)

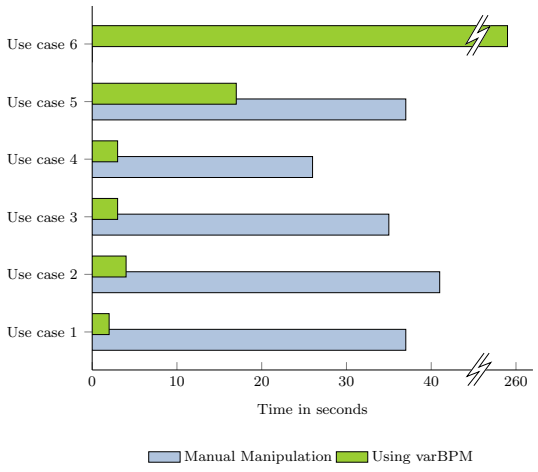


Figure 6.2: Overview of the results of the studied process management use cases (Paper B)

developed framework / the SPLE tool. All of them are familiar with the used BPM tool which is used by them on a regular basis.

The results are illustrated in Figure 6.2. The results of Use Case 2 were somewhat surprising since the framework performed worse compared to Use Case 1. This was due to the fact that the code had not been optimized for efficient queries in the process database. Thus, a human was able to find the according processes more efficiently because a folder structure understandable by humans was used in the database. The implementation was improved in a later version of the framework when efficient identifier-based queries were used to find the according processes. Furthermore, we identified another bottleneck which is related to the communication overhead: The SPLE tool was communicating via a web-service interface with the BPM tool which queried the process database from a local network.

The break even point strongly depends on the use cases: For example, if maintenance activities of process templates – which will be reused in several processes – is common, the initial overhead pays off more quickly. Through our experiments, we identified that the initial overhead amortizes if around three processes are derived from a single model (Paper B).

6.2 Software-product-line for product configurations

We published first results of the combined Software Variability Toolchain in Paper D, and F in which three different product families were supported. The third supported product family was a revision of the second one which supported new configuration settings, and the underlying hardware platform was changed: respectively, the endianness changed and new sensors were added. To evaluate the developed system, we compiled an effort estimation in which the implementation effort for the initial system was compared to the development effort for a traditional software development. The term "traditional software development" is used for a software development with ad-hoc (or nearly ad-hoc) software architecture, which means that different systems are designed almost independently from each other. This leads to the situation that only a little code base is shared between each software project since almost the entire code is optimized for single purposes. However, more code could have been reusable if adaptations of interfaces/implementations would have been considered.

The effort for the traditional software development was based on the real implementation effort for the first system and an effort estimation for the second and third system. These estimates where given by the responsible software architects. The results are presented in Table 6.1. As illustrated, the effort for supporting the third Product Family is 20 to 30 times higher in the case of a traditional development. This number seems a bit too high at first sight but can be explained due to the change of the endianness: In the traditional development, all functions which produce integer numbers have to be

	Effort in man-month	
	New Framework	Traditional System
Base System	12	-
Product Family 1	1	6
Product Family 2	0.5	Estimate: 4–5
Product Family 3	0.05	Estimate: 1–1.5
Overall	14.55	11–12.5

Table 6.1: Comparison of the implementation effort for the product configuration toolchain (Paper F)

identified and adapted; in case of our developed toolchain, a post-processing operation was introduced to the "translate" blocks. The development effort for the third supported family also gives a good estimate for maintenance activities: A modular and reusable code base is more flexible with respect to changes.

Considering that the third product was just a revision of the previous product family, the break even point for the product line will be around three to four major product families. This number also correlates to the typical pay-off estimate of other software product lines [46].

6.3 Process-driven software configuration

To measure the effectiveness of the combined variability framework, the number of lines of codes and the time to create a new PT were measured [29]. The lines of codes are the lines of code which are generated from the data generation system as well as the lines from the modular script library. These lines of code need to be verified by the Domain Expert during the PT creation with respect to the configuration and test coverage. For the evaluation, two products were investigated which were created manually as well as using our newly developed framework. Every product was created by a different developer, in both variants from scratch. Both developers were experts in creating product configurations. The results are illustrated in Table 6.2. The time measurement contains the time for creating/generating the scripts and running a verification of the configuration on a sample device or an FPGA, using a test configuration (i.e., no production keys). As illustrated in the table, the lines of code are reduced by around 30 percent and the expenditure of time by 50 percent. Another advantage of having a process-driven approach is that the tools may be operated by non-experts. This means that also product managers are able

	Product A		Product B	
	manual	framework	manual	framework
Lines of Code	4838	3363	4507	3137
Time effort in hours	16	8	14	7

Table 6.2: Comparison of a manual product creation with a process-driven product creation, done by domain experts [29]

to generate the configuration scripts. Only the final configuration verification has to be done by a developer.

Another advantage of having a process-driven product configuration is the fact that every single product configuration can be traced to the according scripts. Furthermore, the configuration settings can be traced to the semiconductor factory and can be verified with the log of the production process. This means that evidences can be generated which link the product configuration to the final production report. This is an important point for the security certification in order to be able to ensure that customer data is not confused with other data or leaked during the process.

6.3.1 Security considerations

During the production process, confidential data has to be generated and inserted into the final product. To ensure the proper confidentiality of this data, the formal verification tool is used as described in Section 5.1. Furthermore, operational measures have to be applied to ensure that the production equipment cannot be manipulated by single individuals. With respect to the attacker model (see Section 2.1.2), the formal verification tool prevents the leakage of confidential data through the data-generation-system. This leakage could be done intentionally (e.g., by an attacker), or accidentally if the security architecture of the product has undiscovered weaknesses. In any case, the formal verification tool is used to identify and prevent these security leaks.

The tool implementation is based on a formal model which describes how specific operations propagate information from the input parameter to the output parameter. Further, formal conditions are defined in order to ensure that operations are executed only on valid data. This means for example, that an encryption operation only accepts valid key objects with respect to the employed algorithm. The correctness of the model can be proven using an automated theorem prover, based on state-of-the-art assumptions regarding the strength of cryptographic operations. The assumptions of the current model are based on the recommendations published by the NIST [6].

Traditionally, the software which is used to generate the chip individual data is designed for dedicated products. The formal verification tool helps to develop a generic software that can be configured for the actual product without costly redevelopment of the software. Through the formal model, it is assured that no combination of API calls leaks the confidential device data. This means that security certification can be reused for different products and product families without costly security evaluations of pre-personalization equipment.

Table 6.3 presents typical attacks during the pre-personalization process as well as their according countermeasures.

Attack	**Attacker**	**Description / Countermeasure**
Tapping data communication	Insider, Organizations, Hackers	**Description**: During the whole process, confidential data is transferred between multiple stakeholders. This includes also the communication channel between the personalization equipment and the actual device during the production. **Countermeasure**: Secure messaging has to be used throughout the whole personalization process in order to secure the confidentiality of the data. Only dedicated HSMs are able to decrypt the received data in order to ensure that no operator has access to the confidential customer data. The formal verification tool is used to ensure that the data which leaves the HSM is properly encrypted or does not contain any confidential data.
Manipulation of data transfers	Insider, Organizations, Hackers	**Description**: Related to the previous attack. Proper origin of data streams has to be ensured throughout the whole process in order to properly identify it's origin. **Countermeasure**: Authentication methods have to be used throughout the whole process to ensure the proper origin of the data, as well as message authentication methods to ensure that the data was not manipulated during the transfer. The formal verification tool is used to enforce the use of authentication mechanisms as well as integrity protection mechanisms.

Attack	Attacker	Description / Countermeasure
"Fake" devices	Insider, Organizations	**Description**: Fake devices could be used to leak data that is loaded during the personalization process. In case that static keys are used, all keys can be leaked if a fake device was smuggled into the pre-personalization process. Even worse, if faked HSMs are used to generate the data, all confidential customer and system data could be leaked. **Countermeasure**: Bi-Directional authentication of the personalization equipment and the actual devices has to be ensured. Additionally, it has to be ensured that only authorized personal is able to load new software to the personalization equipment. Multiple eye principal is required for the development and the maintenance of the personalization equipment as well as restricted access to the equipment. Additionally, the integrity of the ROM code has to be ensured in order to identify manipulated IC functionality (See Section 3.1.2).
Social Engineering	Insider, Organizations, Hackers	**Description**: Attacks on the social level can be carried out to gain access to (restricted) company networks, or to establish backdoors for other attacks. **Countermeasure**: Split knowledge such that no single person is able to break the whole system. Confidential data has to be stored either encrypted (using reasonable encryption algorithms) or inside of dedicated HSMs. Additionally, the security awareness of the involved personal shall be increased through trainings.

Table 6.3: List of typical attacks during the pre-personalization process (based on Paper H)

By using the formal verification tool, security certification of the data generation process does not need to be redone in order to support new products: Reasonable encryption of the

confidential data is enforced during the process. This requires that the product in general is capable of loading confidential data in a secure way (i.e. encrypted). If a product does not fulfill this requirement, it is not supported by the process and consequently, will not be certified. The CC security certification for smart cards and similar devices expires after two years maximum. Which means that the data-generation system needs to be re-evaluated at least every two years. This means that the formal model of the verification tool is re-evaluated in order to ensure that all security assumptions are still valid. The re-evaluation costs can be reduced significantly, since the formal model can be verified using state-of-the-art automated theorem prover.

6.4 Limitations

After having discussed the benefits of using the developed tools and processes, the drawbacks and limitations have to be debated as well. Having a flexible system which is configurable for a lot of different products is not always a benefit since the performance (i.e., the throughput time) is lower compared to using specialized and optimized software implementations. During the assessment of this thesis, we evaluated the computational overhead during the generation of the confidential chip individual data since this is the most time-critical part: the longer the generation of the data takes, the more time in the factory is required, which again leads to higher costs. To evaluate the introduced overhead, we measured the time to generate the device-individual data for a typical product configuration using the old software toolchain and the newly developed toolchain. It was investigated that the computational overhead is around 6 percent (160 ms vs. 151 ms per chip) for complex product configurations. In many cases, this overhead is still feasible, but in some cases, it may be too much. For example, if ten million chips are produced, the computational overhead will lead to a delay of around 17 hours. To circumvent this overhead, a system could be developed in which the common data is loaded in advance to the HSM. This would mean that for the actual data generation process, less data would need to be sent to the HSM, leading to a shorter period of time. According to our measurements, most of the time is spent to generate RSA key pairs. As a consequence, lower numbers would only be possible with HSMs that are able to calculate RSA keys more quickly.

While we were able to fully generate the required scripts for simple product configurations, we were able to only partially generate the scripts for complex product configurations due to the high number of inter-feature constraints of the different product features. This is not a technical problem of the approach, but having a complete coverage of all inter-feature constraints is a time-consuming and iterative activity. Furthermore, modeling all the constraints in advance is usually not possible for complex systems. Consequently, the initial constraint model contained only those constraints, which were immediately visibly by the developers. The created constraint model is updated with additional constraints

with every new product (if required).

With respect to security certification, it is usually difficult to reuse the developed system for different products. While the developed formal verification tool helps for reusing evaluation evidences of the data-generation system, evaluation facilities usually still investigate the (supportive) processes and software which is used during the whole flow. As a consequence, certification needs to be done for different product families, but the effort can be reduced significantly since the certification of the data-generation system itself does not need to be redone.

6.5 Overall evaluation

In this Section, the evaluation results with respect to the three major aspects of the thesis hypotheses will be summarized: **Feasibility**, **Traceability**, and **Multi-platform Support**.

- **Feasible**: We successfully applied the developed concepts in an industrial context. While the initial development effort is higher for the developed concepts, the maintenance costs and efforts can be reduced significantly. This means that the developed system is able to react more quickly to changes while delivering a high level of quality since the same system is used for many different products.
- **Traceable**: Through the use of a process-driven process, it is possible to link the settings of the product order to the actual configuration scripts. Moreover, it is possible to verify that the delivered data was used during the production process by comparing the order log with the production log. Consequently, it can be guaranteed that data is not confused with other products.
- **Multi-platform support**: We developed a flexible and generic system which is able to generate the required device-individual data. Through the use of a formal verification tool, it is possible to guarantee high security requirements while still being able to support a wide variety of products. Currently, the developed system is used for three different product families and will be extended in the future for other products. The only restriction is that confidential data has to be protected (i.e., encrypted) when transferred to the chip.

7 Conclusion

"Always in Motion the Future is"

Master Yoda, Star Wars

We are living in an ever-changing and interconnected world. The dawn of IoT and CPS devices further increases the number of connected devices which could be manipulated to collect sensitive and confidential data about their users. To ensure that such devices will not be misused requires a Root of Trust to be part of every device. This Root of Trust is basically used to store confidential data and to calculate cryptographic operations to ensure secure communication channels. As such a secure pre-personalization process is required during which the initial confidential device data is loaded to the chips. To guarantee that these devices are still low-cost requires that the pre-personalization process is cheap and flexible, such that it can be used in many different products. This thesis showed that it is possible to develop generic concepts for a secure pre-personalization for a wide variety of security critical devices. Through the use of software-product-line engineering techniques, it is possible to reduce the costs of the system while still guaranteeing high quality. Via the usage of formal verification tools it is possible to analyze security properties during the execution of the process, while maintaining a high amount of flexibility. Furthermore, the effort for security certifications can be reduced by using formal methods and automatic methods for logging and impact of change detection. Thus, strong arguments can be formulated why only specific parts need to be re-evaluated in case of changes.

7.1 Future work

Since security is an ever-changing domain, possible future work could address the formal verification method and the used security model. A method could be developed which is used to derive the implementation code from the security model to ensure that changes of the security model can be easily implemented and evaluated during a security audit.

Besides security considerations, also the creation of inter-feature constraints is an interesting topic for future research: In the ideal case, a customization tool is developed which can be operated by non-experts (i.e., product managers, operations, etc.) and which creates product configurations based on user-friendly graphical user interface. To be able to solve this problem, new methods have to be investigated on how to model complex inter-feature constraints so that only valid configurations can be created. Since different

products may have different configuration possibilities, it needs to be analyzed on how to efficiently model commonalities and differences in such a customization tool as well.

While this work combines the variability modeling of order processes with the variability modeling of product configurations, it would be an interesting topic on how to combine process variability and software variability in general. Combining these two worlds promises to increase the awareness of developers as well as product managers. This could have the benefit that the amount of unused features can be reduced and that the IT landscape will be aligned with the corresponding business processes, leading to reduced costs and increased business efficiency.

8 Publications

This thesis is based on the following peer-reviewed workshop or conference papers as well as one book chapter publication. The publications are ordered according to their publication date:

A Andreas Daniel Sinnhofer, Wolfgang Raschke, Christian Steger, and Christian Kreiner. "Evaluation paradigm selection according to Common Criteria for an incremental product development." In: *International Workshop on MILS: Architecture and Assurance for Secure Systems* 1 (2015). Available at http://mils-workshop-2015.euromils.eu/, pp. 1–5

B Andreas Daniel Sinnhofer, Peter Pühringer, and Christian Kreiner. "varBPM — A Product Line for Creating Business Process Model Variants." In: *Proceedings of the Fifth International Symposium on Business Modeling and Software Design - Volume 1: BMSD 2015*. 2015, pp. 184–191. ISBN: 978-989-758-111-3. DOI: 10.5220/0005886901840191

C Andreas Daniel Sinnhofer, Wolfgang Raschke, Christian Steger, and Christian Kreiner. "Patterns for Common Criteria Certification." In: *Proceedings of the 20th European Conference on Pattern Languages of Programs*. EuroPLoP '15. Kaufbeuren, Germany: ACM, 2015, 33:1–33:15. ISBN: 978-1-4503-3847-9. DOI: 10.1145/2855321.2855355. URL: http://doi.acm.org/10.1145/2855321.2855355

D Andreas Daniel Sinnhofer, Peter Pühringer, Klaus Potzmader, Clemens Orthacker, Christian Steger, and Christian Kreiner. "A Framework for Process Driven Software Configuration." In: *Proceedings of the Sixth International Symposium on Business Modeling and Software Design - Volume 1: BMSD 2016*. 2016, pp. 196–203. ISBN: 978-989-758-190-8. DOI: 10.5220/0006223701960203

E Andreas Daniel Sinnhofer, Felix Jonathan Oppermann, Klaus Potzmader, Clemens Orthacker, Christian Steger, and Christian Kreiner. "Patterns to establish a secure communication channel." In: *Proceedings of the 21st European Conference on Pattern Languages of Programs*. EuroPLoP '16. Kaufbeuren, Germany: ACM, 2016, 33:1–33:20. ISBN: 978-1-4503-4074-8. DOI: 10.1145/3011784.3011797. URL: http://doi.acm.org/10.1145/2855321.2855355

F Andreas Daniel Sinnhofer, Peter Pühringer, Klaus Potzmader, Clemens Orthacker, Christian Steger, and Christian Kreiner. "Software Configuration Based on Order Processes." In: *Business Modeling and Software Design: 6th International Symposium, BMSD 2016, Rhodes, Greece, June 20–22, 2016, Revised Selected Papers*. Ed. by Boris Shishkov. Cham: Springer International Publishing, 2017, pp. 200–220. ISBN: 978-3-319-57222-2. DOI: 10.1007/978-3-319-57222-2_10. URL: http://dx.doi.org/10.1007/978-3-319-57222-2_10

G Andreas Daniel Sinnhofer, Andrea Höller, Peter Pühringer, Klaus Potzmader, Clemens Orthacker, Christian Steger, and Christian Kreiner. "Combined Variability Management of Business Processes and Software Architectures." In: *Proceedings of the Seventh International Symposium on Business Modeling and Software Design - Volume 1: BMSD,*. INSTICC. SciTePress, 2017, pp. 36–45. ISBN: 978-989-758-238-7. DOI: 10.5220/0006527300000000

H Andreas Daniel Sinnhofer, Felix Jonathan Oppermann, Klaus Potzmader, Clemens Orthacker, Christian Steger, and Christian Kreiner. "Where do all my keys come from." In: *Handbook of Research on Solutions for Cyber-Physical Systems Ubiquity*. Ed. by Norber Druml, Andreas Genser, Armin Krieg, Manuel Menghin, and Andrea Hoeller. 701 E. Chocolate Ave. Hershey, PA 17033, USA: IGI Global, 2017, pp. 278–300. DOI: 10.4018/978-1-5225-2845-6.ch011

Moreover, the thesis is based on a patent which is currently patent pending:

I Florian Böhl, Klaus Potzmader, Clemens Orthacker, Andreas Daniel Sinnhofer, and Christian Steger. "Method for Symbolic Execution on Constrained Devices." Patent 82086638 (82019770US01) (US). Status: Application; App. No. 15/482462. Apr. 2017

Furthermore, we are currently preparing a second patent for a product line for the generation of chip individual data. Additionally, an extension of Paper G was accepted for publication in a Springer LNBIP series. The paper will elaborate on how the identification of business drivers can be improved when a combined variability management of business processes and software architectures is applied.

J Klaus Potzmader, Clemens Orthacker, Andreas Daniel Sinnhofer, Christian Steger, and Christian Kreiner. "A runtime-configurable in-HSM secure IC intialization process." Patent – (EU). Currently in a company internal review process. 2017

K Andreas Daniel Sinnhofer, Peter Pühringer, Klaus Potzmader, Clemens Orthacker, Christian Steger, and Christian Kreiner. *Identification of business drivers using traceable links between process models and software architectures*. Working title; Accepted for publication in a LNBIP series in 2018

Additionally, we are currently preparing a paper for a shepherding process describing secure IoT devices in a pattern form which rely on a secure pre-personalization process to establish a Root of Trust.

The context of the publications listed above are highlighted in Figure 8.1.

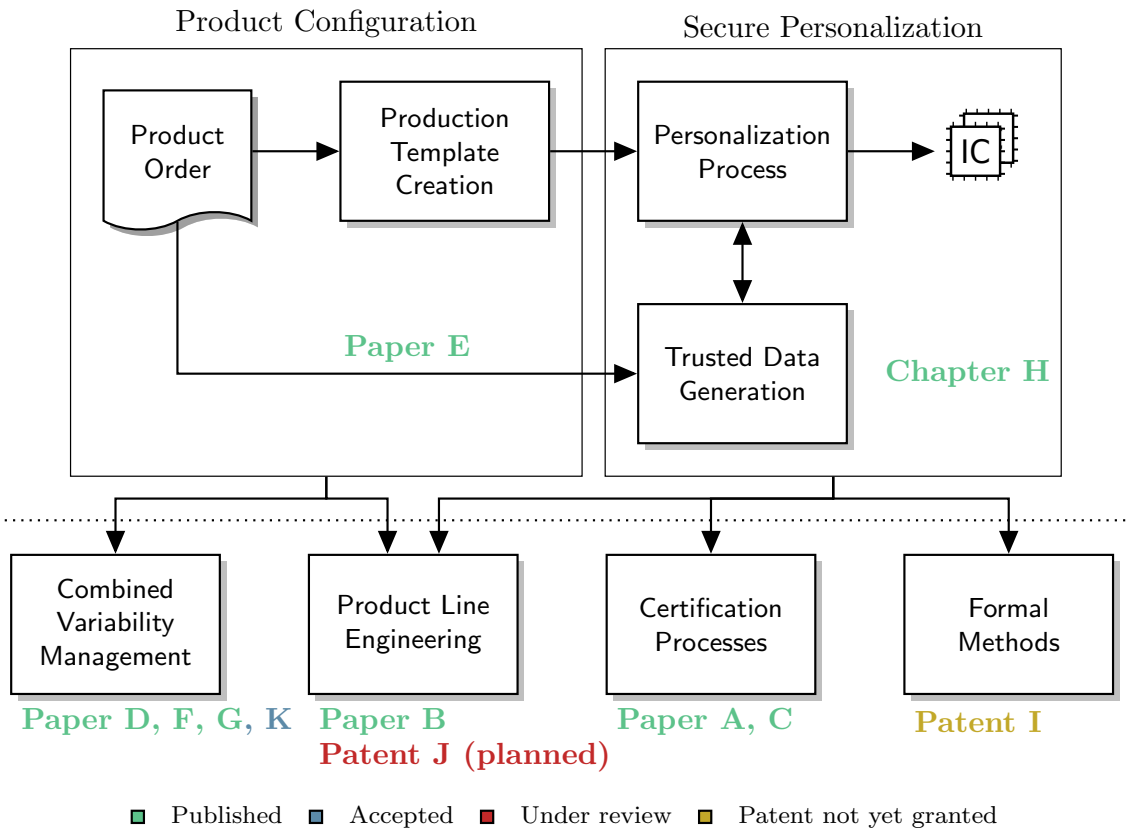


Figure 8.1: Overview of the suggested approaches to improve the pre-personalization process of secure ICs, highlighting the position of the publications

Evaluation paradigm selection according to Common Criteria for an incremental product development

Andreas Daniel Sinnhofer
Institute for Technical Informatics
Graz University of Technology, Austria
a.sinnhofer@tugraz.at

Wolfgang Raschke
Institute for Technical Informatics
Graz University of Technology, Austria
wolfgang.raschke@tugraz.at

Christian Steger
Institute for Technical Informatics
Graz University of Technology, Austria
steger@tugraz.at

Christian Kreiner
Institute for Technical Informatics
Graz University of Technology, Austria
christian.kreiner@tugraz.at

ABSTRACT

Today, agile product development techniques are widely used providing a rapidly and steadily progression of incremental product improvements. Traditionally, a product certification is issued in a late stage of the development process, although some Common Criteria evaluation paradigm would exists to support an agile or modular development process. The usage of such a paradigm would result in a beneficial certification process, since the evaluator gains experience through the maturing product. To provide a systematic way to integrate the evaluation process into the development process — and thus saving money and time — we have identified use case scenarios with the according evaluation paradigm, providing a selection scheme for the right paradigm.

Categories and Subject Descriptors

D.2.13 [**Software Engineering**]: Reusable Software—*Reuse models*

General Terms

Design, Security

Keywords

Common Criteria, Security Evaluation

1. INTRODUCTION

Today, agile product development techniques are widely used providing a rapidly and steadily progression of incremental product improvements, based on common parts and a modular product architecture [7]. This leads principally to a faster time to market and enables the ability to survive and compete in a competitive market. A problem with this flexible and adaptive development paradigm comes up when a certification of the product should be issued, since — traditionally — agile methods are already not used for the development and evaluation process of secure products.

At present, a common approach is to start the certification process of a product in a very late phase of the development, which can result in huge costs when the evaluation facility gives a negative attestation, because a redesigned must be issued. As identified by Boehm [9], the later changes are introduced in the development process, the higher the costs are.

Another problem with such an approach is the long period of time an evaluation process can take, even when the certification of the product is positive. E.g. the certification process of Microsoft Windows 7 took one year and eight months[1]. This can lead to a delayed release if a certificate is a condition for the disposal of a product (e.g. the CE certificate for resale within the EU) or a big gap between the date of release and the date a certificate is issued. Either way, both situations can potentially result in a loss of customers when a competitor is already selling a certified product.

To overcome these drawbacks, Raschke et al. [14] introduced two processes capable for a modular or agile product development, where the certification process is started in parallel. Furthermore he provides a method to automatically detect the actual impact set, so that only those modules are reevaluated which have an effect to the security assurance of the product. This approach has the key benefit that the evaluator is integrated since the early stages of the process. In fact, the evaluator is gaining experience with the maturing system. Moreover, the feedback of the evaluator can be directly integrated in the next iteration step leading to lower redesign costs [8] [6]. The Common Criteria certification process itself is not further specified, which means that any possible paradigm can be chosen, such as the assurance

[1]see the 14th International Common Criteria Conference (ICCC) `https://www.commoncriteriaportal.org/iccc/ICCC_arc/presentations/T2_D2_2_30pm_Grimm_Evaluating_Windows.pdf`

continuity, a compositional evaluation or a delta evaluation, depending on the current development environment regarding the number of involved developing companies and the number of involved certification facilities.
The contribution of our paper is the identification of the appropriate evaluation scheme for a Common Criteria certification for an agile or modular product development which is applicable in combination with the processes from Raschke et al. [14]. The proposed selection scheme is also applicable for products which are based on previously certified products or modules (e.g. for bug-fix releases).
Section 2 gives a short introduction into the evaluation paradigms according to Common Criteria and the processes identified by Raschke et al. [14]. Section 3 gives an overview over the use case scenarios, providing further information on the according evaluation paradigm and Section 4 summarizes the findings from the use case scenarios in the proposed selection scheme. Finally the results of this paper are summarized and related work is presented.

2. BACKGROUND

2.1 Assurance Continuity

As proposed in Common Criteria Assurance Continuity [1], an evaluation paradigm for the maintenance and re - evaluation of already Common Criteria certified products exists. The flow chart of this approach is illustrated in Fig. 1. It can be seen that based on an impact analysis report (IAR) a decision is made whether the changes to the target of evaluation (TOE) is minor (does not affect the assurance baseline) or major. In the case of minor changes, the previously issued certificate is updated with a maintenance addendum and a maintenance report. The Common Criteria Assurance Continuity [1] states, that *"Maintenance may, in general, continue for up to two years beyond the certification date"*. Due to the fact, that we only consider major changes, the maintenance process is not further contemplated.

In case of major changes, a re-evaluation needs to be performed regarding all affected parts and a new certificate is issued. This can be achieved using an informal modular evaluation scheme (i.e. Delta-Evaluation) through re-evaluation of only the changed and affected modules as stated in the Common Criteria Information Statement on the reuse of evaluation results (see Section 2.2).
The drawback of the assurance continuity approach is, that it is only applicable in those situations, where the evaluation facility is not changed and where a certificate was already issued. Therefore, this approach is intended to be used for bug-fix releases/revisions of old products.

2.2 Delta Evaluation

As stated in the "Common Criteria Information Statement on the reuse of evaluation results" [2] the following evidences must be shared to reuse previously created evidences:

- Product and supporting documentation
- New security target(s)
- Original security target(s)
- Original evaluation technical report(s)

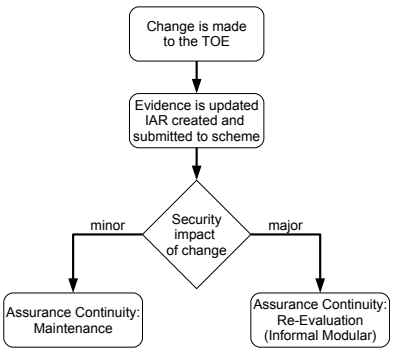


Figure 1: Common Criteria Assurance Continuity flow chart

- Original certification/validation report(s)
- Original Common Criteria certificate(s)
- Original evaluation work packages (if available)

It is specified that

> "*... the evaluation facility conducting the current evaluation should not have to repeat analysis previously conducted where requirements have not changed nor been impacted by changes in other requirements ...*"

where such changes are identified through a so called delta analysis:

> "*... The evaluation facility would be required to perform a delta analysis between the new security target and the original security target(s) to determine the impact of changes on the analysis and evidence from the original evaluation(s) ...*"

which is similar to an impact analysis.
As a result, a product re-evaluation can be performed by an analysis of the impacts of changes and through evaluation of only the changed and affected modules. Unaffected modules need not be reconsidered for the overall evaluation process. Drawback of this approach is that the evaluation technical report is typically generated by the evaluation facility and thus, in some cases, is considered as proprietary to that facility, which makes the interchange of evidences between different certification facilities difficult.

2.3 Composite evaluation

As stated in the Common Criteria Mandatory Technical Document on the composite product evaluation for smart cards and similar devices [3] a composite evaluation can be performed for all kind of products where

> "*... an independently evaluated product is part of a final composite product to be evaluated ...*"

and hence is not limited to smart cards only, but with the limitation that

> "*... The composite product is a product consisting of at least two different parts, whereby one of them represents a single product having already been evaluated and certified ... The underlying platform is the part of the composite product having already been evaluated ...*"

Thus it is applicable for example for an embedded system whereas an application runs on a certified OS, respectively the OS is running on a certified hardware. I. e. a layers pattern is used for the product, whereby trust is established through each layer. The lowest EAL of all components is the limiting factor of the composite product.

2.4 Composed evaluation

As stated in the Common Criteria part 3 (see [4]), the composed evaluation is intended for situations, where independently certified (or going through an independent certification process) products/modules are assembled to a new product which should be certified. It is applicable, where a composite evaluation is not suitable and a delta evaluation cannot be performed due to missing evidences (proprietary documents are not shared). At present, a composed evaluation for higher assurance levels (higher than CAP-C[2] is not supported through the composed scheme and hence a re-evaluation of the whole product is necessary. Due to this, composed evaluations have been performed much less successful than composite evaluations.

2.5 Informal: Identification of the impact set

Due to the fact that it is not necessary to perform unaffected evidences twice, it is meaningful to use change detection analysis to determine the actual affected modules so that only these modules need to be reconsidered in the evaluation. It is important to understand, that modules can interact with each other and hence not only the directly changed module but all other interacting modules need to be reconsidered. This can be achieved through the use of the change impact analysis process proposed by Bohner [10] or the refined processes by Raschke et al. [14]. Our work only mentions the processes proposed by Raschke et al. since he also describes a tool for an automatic change detection analysis, which is well-suited for an partially automatic generation of the Impact Analysis Report (respectively delta analysis), but every other approach is also applicable.
The change detection analysis is based on the so-called *Security Model*, which describes the properties and relationships of the developer evidences, based on the security target, the design documentation, the implementation and the tests (see Figure 2 explanatory graphical representation). Therefore it is applicable to trace and detect all dependencies between each module.

[2]Attack potential "Enhanced Basic"; approximately comparable with EAL-4 (see[4] pages 38 and 47)

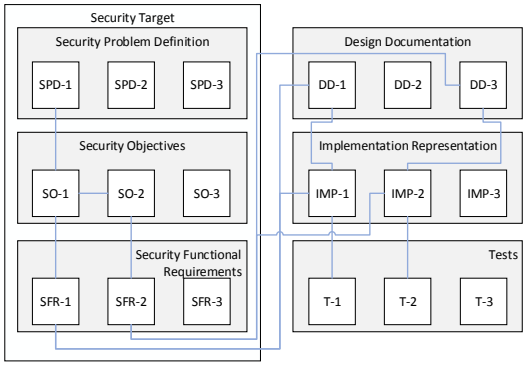


Figure 2: Explanatory Security Model, showing some exemplary artefacts and traces

3. PROPOSED USE CASES AND ACCORDING EVALUATION PARADIGM

Overall situation: Aforementioned, we consider an agile or modular product development process, where the (final) certified product is assembled using a number of modules. In each development iteration new modules can be added or old modules can be changed or removed. Various companies can be involved in the development process of the product and any number of evaluation facilities can be integrated in the certification process.
The selection scheme is applicable for the following scenarios:

- *Use case 1:* One company develops a number of modules which are all evaluated at the same evaluation facility. Since the evaluation facility has full access to all modules and all related evidences, an evaluation can be achieved by a simple informal modular evaluation. If during the development process the evaluation facility is changed, a formal modular paradigm would need to be chosen.

- *Use case 2:* One company develops a number of modules, whereas a number of evaluation facilities ($n > 1$) are involved in the certification process, interchanging all kind of evidences. Therefore, a delta evaluation can be issued.

- *Use case 3:* One company develops a number of modules, whereas a number of evaluation facilities ($n > 1$) are involved in the certification process, but unfortunately they do not interchange evidences. Depending on the architecture of the developed product a composite (Use case 3.a) evaluation or an composed (Use case 3.b) evaluation can be issued.

- *Use case 4:* Several companies are involved in the development process of the product, but one central evaluation facility is used. In this scenario an informal modular evaluation can be used since the certification facility has direct access to every contribution of every

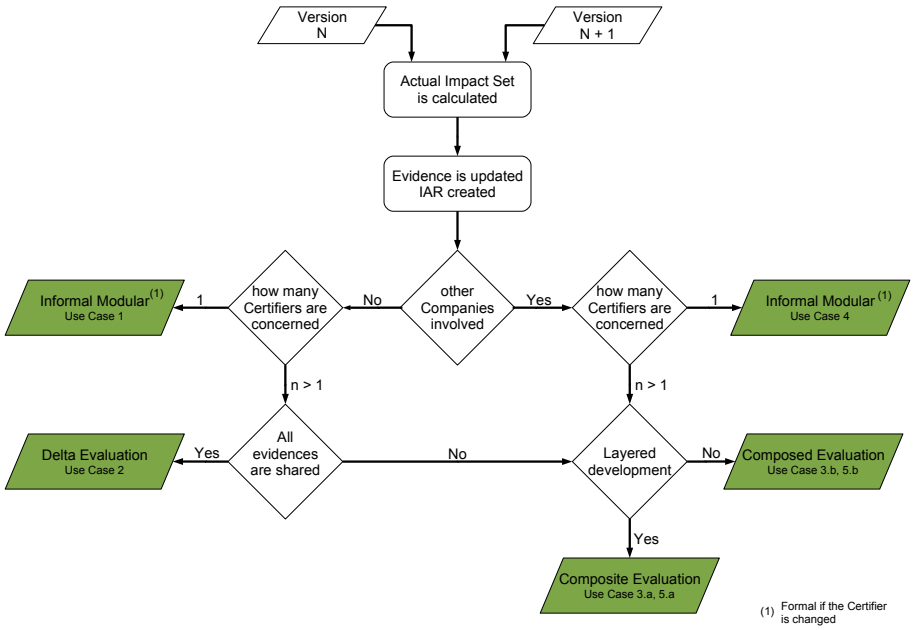


Figure 3: Proposed paradigm selection scheme

company. If during the development process the certification facility is changed, a formal modular scheme would need to be chosen.

- *Use case 5:* Several companies are involved in the development process of the product and any number of evaluation facilities (n > 1) are included into the certification process (e.g. each company consults a different evaluation facility). A delta evaluation is possible if the different evaluation facilities interchange all kind of evidences, which can be a problem since the evaluation facilities would need to provide information on their evaluation process and their used methods. In practice, a composite (Use case 5.a) or composed (Use case 5.b) evaluation scheme is used, depending on the used architecture.

4. PARADIGM SELECTION SCHEME

Based on the activities during the assurance continuity process [1], a selection scheme for the presented use cases was created. The first steps towards the reuse of any evidence is the analysis of the impacts on the assurance of the current Target of Evaluation which is intended to be done by one of the processes proposed by Raschke et al. [14]. The selection scheme is split-up into two main leafs, where one is applicable if a product is developed from a single company and the other one for a product which is developed from many companies. As identified in the use cases, another factor which must be considered is the number of certification facilities and the fact if these certification facilities do interchange all needed evidences so that the evaluation results can be reused efficiently. Another criterion which needs to be reconsidered is derived from the composed evaluation scheme, whereby the developed product is structured in a layered approach. The lowest layer must be already certified.
Generally spoken an informal approach can be used if certification facilities do interchange evidences or a single certification facility is issuing the product evaluation and formal approaches must be chosen in all other cases which are usually more time and money intensive.
The next enumeration provides a short description of the according paradigms:

- *Informal Modular:* The certification facility has full access to all modules and evidences, therefore only the affected modules are re-evaluated (Delta Evaluation).
- *Formal Modular:* The certification facility was changed and hence, all modules need to be reconsidered in the evaluation process. Previously created evidences (e.g. certified modules) can be reused, if all needed information is available.
- *Composed Evaluation:* This evaluation is based on the Composed Assurance Package (CAP) of the Common Criteria part 3 (see Section 2.4). Drawback is that the highest achievable CAP level is CAP-C, which is comparable to EAL-4. Higher levels of assurance are only possible through a complete re-evaluation of the assembled product.
- *Composite Evaluation:* This evaluation paradigm is based on a layered product development, where trust is gained through the combination of all layers. In difference to the composed evaluation, the composite product is the final product for which an EAL level

certification is issued. This allows a direct comparison with similar products certified after a single evaluation. [3]

- *Delta Evaluation:* This is the delta evaluation as described in Section 2.2. A concrete process for the certification is not provided through the Common Criteria standard and thus the according certification facility needs to be consulted.

5. RELATED WORK

Klohs [12] provides observations and thoughts on the modularisation concepts for the development of a smart card operating system according to Common Criteria. He points out that the JIL document [5] on the security architecture requirements for smart cards and similar devices, establishes a first starting point for the reuse of software components, based on a description of the security interface and the implemented security mechanism which is implemented from the component independent of a concrete security target.
The Assert4SOA[3] project focuses on the development of methods for the certification of service oriented architectures (SOAs), reusing existing certification processes to overcome the challenging tasks for an evolving software ecosystem. The project itself does not focus on the Common Criteria scheme, but provides a guidance to integrate the Common Criteria certification scheme into a service oriented architecture in [13].
The Euro-MILS[4] project focuses on providing a framework for trustworthiness by design and high assurance based on *Multiple Independent Levels of Security (MILS)* [11]. In fact, assurance of the whole product is gained through the composition of assurance arguments of its components and the system's security architecture. The developed framework is based on the Common Criteria evaluation schemes.

6. CONCLUSION

Today's industry is embossed through fast changing requirements regarding functional and security needs. These circumstances are tried to be solved through the usage of agile or incremental manufacturing techniques. We have identified a scheme for the selection of the appropriate evaluation paradigm to support an agile or modular development processes regarding the security certification to reduce the time shift between the successful certification and the time the product development finished. Furthermore the costs for re-evaluating the developed product/modules can be kept as low as possible since the most suitable paradigm is chosen, maximizing the reuse of already evaluated modules and providing a direct integration of the evaluation facility in the process so that the feedback is directly integrated into the next development iteration.

7. ACKNOWLEDGEMENT

Project partners are NXP Semiconductor Austria GmbH and the Technical University of Graz. The project is funded by the Austrian Research Promotion Agency (FFG).

[3] www.assert4soa.eu
[4] http://www.euromils.eu

8. REFERENCES

[1] Common Criteria. Assurance Continuity CCRA Requirements. Version 2.1 (June 2012).
[2] Common Criteria Information Statement. Reuse of Evaluation Results and Evidence. (October 2002).
[3] Common Criteria Supporting Document Mandatory Technical Document - Composite product evaluation for Smart Cards and similar devices. Version 1.2 (April 2012).
[4] Common Criteria for Information Technology Security Evaluation. Part 3 Security assurance compnents. Version 3.1 Revision 4 (September 2012).
[5] Common Criteria Supporting Document Guidance - Security Architecture requirements (ADV_ARC) for smart cards and similar devices. Version 2.0 (April 2012).
[6] S. Ambler. *The Object Primer: Agile Model-Driven Development with UML 2.0 - Third Edition.* Cambridge University Press, 2004.
[7] D. Anderson. *Agile Product Development for Mass Customization: How to Develop and Deliver Products for Mass Customization, Niche Markets, Jit, Build-To-Order and Flexible Manufacturing.* Irwin Professional Pub., 1997.
[8] K. Beck and C. Andres. *Extreme Programming Explained: Embrace Change (2Nd Edition).* Addison-Wesley Professional, 2004.
[9] B. W. Boehm. *Software Engineering Economics.* Prentice Hall, Englewood Cliffs, NJ, 1981.
[10] S. A. Bohner. Extending software change impact analysis into cots components. In *Proceedings of the 27th Annual NASA Goddard Software Engineering Workshop (SEW-27'02)*, SEW '02, pages 175–, Washington, DC, USA, 2002. IEEE Computer Society.
[11] H. Blasum, S. Tverdyshev, B. Langenstein, J. Maebe, B. De Sutter, B. Leconte, B. Triquet, K. Müller, M. Paulitsch, A. Söding- Freiherr von Blomberg, A. Tillequin. Secure European Virtualisation for Trustworthy Applications in Critical Domains - MILS Architecture, 2014.
[12] D. K. Klohs. Software modularisation and the common criteria - a smartcard developer's perspective.
[13] M. B. Samuel Paul Kaluvuri and Y. Roudier. Bringing common criteria certification to web services.
[14] W. Raschke, M. Zilli, P. Baumgartner, J. Loinig, C. Steger and C. Kreiner. Supporting evolving security models for an agile security evaluation, 2014.

varBPM
A Product Line for Creating Business Process Model Variants

Andreas Daniel Sinnhofer[1], Peter Pühringer and Christian Kreiner[1]
[1]*Institute for Technical Informatics, Graz University of Technology, Austria*
{*a.sinnhofer, christian.kreiner*}*@tugraz.at, p.puehringer@inode.at*

Keywords: Software Product Lines, Feature Oriented Modelling, Business Processes, Tool Integration.

Abstract: Business processes have proven to be essential for organisations to be highly flexible and competitive in today's market. To manage the life-cycle from modelling such business processes over the execution and the maintenance, Business Process Management Tools are used in the industry. In many cases, different business processes do only vary in few points. This leads to the situation that new business process variants are formed through copy or clone of previous solutions leading to a high number of instantiated process templates. However, this means that changes to a template affects many processes, where all of them need to be manually updated, which can lead to a considerable amount of work and money for a bigger company. In this paper, we will present a framework for the integration of business process modelling tools and software product line engineering tools to provide a systematic way to reuse and trace process variations of whole process families.

1 INTRODUCTION

Business Process (BP) oriented organisations are known to perform better regarding highly flexible demands of the market and fast production cycles (McCormack and Johnson (2000); Hammer and Champy (1993); Valena et al. (2013); Willaert et al. (2007)). These goals are achieved through the introduction of a management process, where business processes are modelled, analysed and optimised in iterative ways. Nowadays, the business process management is also coupled with a workflow management, providing the ability to integrate the responsible participants into the process and to monitor the correct execution of the business process in each process step. To administer the rising requirements, so called business process management tools are used (BPM-Tools) which cover process modelling, optimization and execution. In combination with an Enterprise-Resource-Planning (ERP) system, the data of the real process can be integrated into the management process.
In many cases, business processes do only vary in some points, which leads to the situation, that new process variants are created through a copy and clone of old solutions (often called as templates). As a result, such templates are instantiated in many various processes which makes the propagation of process improvements time and cost intensive for a bigger company. Also the consistency of the documentation of this huge number of process variants is a challenging task.
Software Product Line Engineering (SPLE) techniques have been successfully applied for almost any domain, providing a technique for the systematic reuse of domain artefacts. Although the topic of product line techniques in the domain of business process modelling is not new (e.g. Gimenes et al. (2008); Rosa et al. (2008); Fantinato et al. (2012); Derguech (2010)) only little work is found for the issues related to the correct configuration of whole process families (e.g. Hallerbach et al. (2009a,b)), the integration into existing toolchains and the reuse throughout various production plants. Thus, our approach is focused on developing a framework for the integration of a SPLE Tool and a BPM Tool, to provide a generic way to generate process variants of whole process families. In particular, we use the SPLE Tool for a systematically reuse of expert knowledge in form of valid process variations, designed in an appropriated BPM Tool. The integrity of the process variations is secured by the capabilities of the BPM Tool and a rich constraint checking in the SPLE Tool. Furthermore, our proposed approach enables the abilities to automatically trace all process variants for an automatic propagation of changes and process improvements and the systematic integration into the capabilities of the BPM Tools such as documentation generation, workflow engines, process optimisation tools, etc.

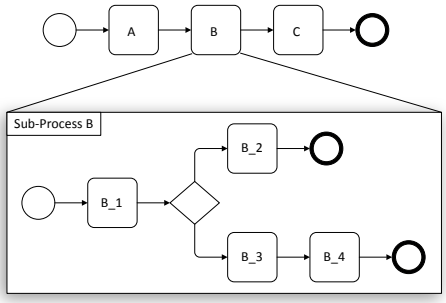


Figure 1: Principal structure of a business process according to Österle (1995) which is used for our approach. Starting with an abstract description of the process, the tasks are further described in sub-processes until a complete work description is reached (microsopic level).

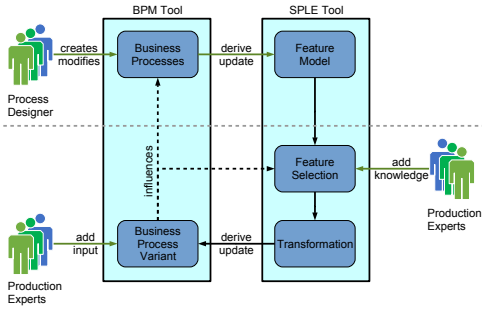


Figure 2: Overall conceptual design. The upper side of the Figure describes the domain engineering part and the lower side of the Figure the application engineering part.

This work is structured in the following way: Section 2 gives an overview over the concepts of tool integration and the design paradigm for business processes which is needed for our framework. Section 3 presents the conceptual design of the framework and states construction rules of the according feature models and some design rules for the BPM Tool. In section 4 we will introduce our case study regarding some metrics and implementation details. Section 5 summarizes the related work and finally section 6 concludes this work and gives an overview of open issues.

2 BACKGROUND

2.1 Tool Integration

According to the work of Karsai et al. (2005), two possible patterns exists for tool integration. The first approach is named ”Integration based on integrated models” and is based on the idea of a common data model which is shared between each participating tool. This means that each tool needs two model converters, one for the conversion of the native data model into the common data model and one for the opposite direction. The data is shared over a so called integrated model server where each tool can publish or consume data. For obvious reasons this approach is meaningful applicable if each participating tool has a similar data model. A drawback of this approach is that it does not scale very good with the number of connected tools.
The second pattern is named ”Integration based on process flows” and is based on the idea of a point to point message based communication. Each participating tool registers itself at a backplane providing information about what data is shared and what data is intended to be consumed. As a result, this approach scales better with the number of participants since potentially fewer model transformations are needed at which each model can be better optimized regarding the communicating tools. This approach is used in those situations, where the data is processed in a specific sequence.
For our framework both patterns would be possible. Due to the fact, that the number of participants is small (in most circumstances there is only one SPLE Tool and one BPM Tool) and since the representation of business processes is very similar throughout various tools, the first approach is more applicable.

2.2 Business Processes

A business process can be seen as a sequence of tasks/sub-processes which needs to be executed in a specific way to produce a specific output which is of value to the costumer (Hammer and Champy (1993)). According to Österle (1995) the process design on the macroscopic level (high degree of abstraction) is split up into sub-processes until the microscopic level is reached. This level is reached, when all tasks are detailed enough, so that the process employees can use it as work instructions.
In other words, a complete business process is designed in layers, where the top layer is a highly abstracted description of the overall process, while the production steps are further refined on the lower levels. As a result, the lowest level is highly dependable on the concrete product and production environment, providing many details for the employees. In fact the top layers are – mostly – indepen-

dent from the concrete plant and the supply chain and could be interchanged throughout the production plants, whereas the lower levels (the refinements) of the processes would need to be reconsidered. Figure 1 gives an overview of such a structure. Variability of such a process structure can either be expressed through a variable structure of a process/sub-process (e.g. adding/removing nodes in a sequence) or by replacing the process refinement with different processes. The current version of our developed prototype focuses on the second method but the framework is not limited to it.

2.3 Informal: Feature Model

A feature is defined by Kang et al. (1990) as a *"prominent or distinctive user-visible aspect, quality, or characteristic of a software system or system"*. In context of a Software Product Line, a feature model is a model which defines all these features and explicitly states their relationships, dependencies and additional restrictions between each other. It enables the ability to visually represent the variable parts of a system and the options available for all products of a product line.

Table 1: Needed process information within the SPLE Tool

property name	description
id	A unique id to identify the process/task
category id	A unique id of the category of the process
display name	A human readable and understandable name of the process/task
children	A list of ids which references the processes/tasks of the process itself (empty if the microscopic level is reached)
additional data	A list of additional data which is needed for the concrete instantiation of the process. E.g. data for a workflow engine, the responsible workers, etc. This data can be provided through the BPM Tool (almost static) or can be design variable in the SPLE Tool.

3 VARIABILITY FRAMEWORK

3.1 Conceptual design

The overall conceptual design is based on a feature oriented domain modelling approach and is displayed in Figure 2. It is intended, that the domain experts (process designer) design process templates in the according BPM Tool, providing also all needed information for e.g. a workflow engine. Based on these templates and the abstract process model (the top level description of the process) a feature model is partially automatically created/updated with the guidance of the domain experts. This is done by identification of the variation points and the linkage of the according variations on every process level. Each variation can contain additional variability by either defining new variation points where further refinement can be linked to or by a variable process structure. Additional constraints regarding the possible combination of features are intended to be modelled in the SPLE Tool, but are not limited to it. In application engineering the domain experts (not necessarily a process designer, but someone who knows the current needs of the production) adds his knowledge and selects the needed features. The found description is then automatically transformed in a real business process which can be executed by the workers. During the process execution, loads of data is generated regarding the performance and efficiency of the process. Thus, it is possible that some additional information is added to the derived processes which leads to a possible influence of the according business process templates or a possible influence of the feature selection. This flowback mechanism is an important task and needs to be considered for the maintenance and the evolution throughout the lifecycle of a process.
For illustration a short example for the flowback mechanism is given: Task B of the process displayed in Figure 1 could be dependent on the logistic chain of a supplier of a specific part or material. If during the execution of the process the supplier is changed, it is also likely that the overall control of the logistic chain is changed due to the fact that the newly integrated supplier can only deliver goods in a specific way. If the process of the logistic chain was not already modelled, then the process designer would need to create a new process variant and would need to update the existing feature models first. After this is done, our proposed toolchain needs to update the feature selection of the respective process in the SPLE Tool. This automatically updates the structure of the overall process variant, leading to an almost on the fly update of the complete process workflow especially when the process variation was already modelled.
Summarizing this concept means that the SPLE Tool is responsible for the following points: It needs to assist the domain experts (process designer) during the

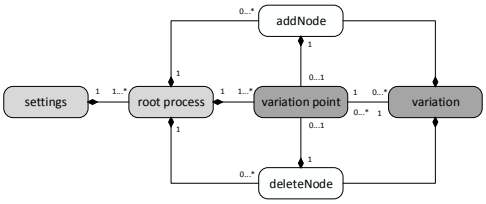


Figure 3: Feature model construction rules. The white parts cover the variability on the process level itself and the dark grey parts cover the variability between each layer of the overall process structure.

(partially) generation of the feature models and during the selection and creation of the concrete product variant. Furthermore, it has to keep track of all generated process variants to automatically apply process template improvements or changes to the overall process structure and to flow back information added during the execution in the BPM Tool. To do so, the tool needs information about the properties of the process displayed in Table 1. The BPM Tool is responsible for the creation of semantically correct process variants and to provide capabilities which are of value for the developing company e.g. automatic documentation generation, workflow engine, etc. For obvious reasons, each Tool must have rich import/export capabilities or the ability to integrate user defined plug-ins to extend the functionality.

3.2 Type model

To model the variability within the SPLE Tool in a structured way, the feature model should support the following type models:

- **settings**: Is a data model, containing information for the tool adaptors to identify the right datasets (e.g. identifier of the database of the BPM Tool from which the process structure is imported; log-in settings, etc.)
- **root process**: A process model of the top level process and therefore an abstract description of the overall process sequence. It consists of various nodes where some of them deal as variation points.
- **variation point**: A node in a process where at least one variation can be linked to.
- **variation**: A process model for a task or a subprocess which can be linked to a number of variation points. If it is a process, it can also contain variation points or a variable process structure (addition/deletion of nodes).
- **addNode**: Adds a node (task or process model) at a specific location of the process structure. This added node can also be a variation point for further refinements. The addition of the node can be dependent on the feature selection.
- **deleteNode**: Deletes a node (task or process model) at a specific location of the process structure. The deletion of the node can be dependent on the feature selection and hence it is also possible to link further refinements to this node.

The construction rules for this type model can be seen in Figure 3. The settings node is only instantiated once in such a model. The root process is somehow very similar to a variation, but with the difference that it must contain variation points (to prohibit a ”Blob” anti pattern [Brown et al. (1998)]).

3.3 Design rules for business processes

Aforementioned, the processes should be designed as stated by Österle (1995). Secondly we have noticed, that almost every bigger BPM Tool supports the assignment of specific group identifiers to groups of processes, providing a more structural design of the processes. Thereby it is possible to automatically map specific groups of processes to a specific variation point. This leads to the situation that new variations can be automatically detected and can be advocated for an integration into existing feature models. Furthermore, the structuring in groups of processes enables the ability to introduce a constraint check so that variation points are limited to specific groups of processes. This increases the assurance in the creation of semantically valid processes.

4 INDUSTRIAL CASE STUDY: VARBPM

In this section an overview over our industrial case study is given, which describes the domain of our industry partner and the developed toolchain.

4.1 Industrial project partner

Our project partner Magna Cosma[1] is an international company in the metal stamping and assembly industry – specialised on class-A car body panels and closure parts (e.g. doors) – with several plants all over the globe. The implemented business processes

[1] http://www.magna.com/de/kompetenzen/karosserie-fahrwerksysteme

are mostly controlled by an SAP infrastructure and are designed with the BPM-Tool Aeneis[2]. Although some plants are specialised on the same production parts, almost every plant develops and maintains their own business processes, which makes it difficult to compare processes, mark bottlenecks, optimise the processes and publish the changes to other plants.

4.2 Tool integration

As mentioned before, the tool integration of the SPLE Tool (pure::variants[3]) and the BPM Tool (Aeneis) is based on the Pattern "Integration based on Integrated Models". The integrated data model contains the relevant data enumerated in Table 1 where all fields are of type String respectively an array of Strings for the children and data field. This means that the native data model of the SPLE Tool is the integrated data model and hence only the BPM tool adaptors need to implement a data conversion. To support an update mechanism without sending the complete process, each published dataset can be assigned to a specific type indicating what should happen with this dataset. Possible types are:

- **New**: Indicating that this dataset was not published before and hence it should be integrated directly just as is.
- **Update**: Contains the id of the according dataset and the data which shall be updated. Non specified attributes are not affected.
- **Remove**: Contains the id of the according dataset which should be deleted out of the system. Linked variations of such a process are not affected.

The developed tool adapters are also applicable to get notified when data is added/updated so that such changes are processed almost immediately. If this mechanism is not supported by the participating tool connector, an operator needs to trigger this update mechanism manually. In the current development, the SPLE Tool needs to check manually for updated data since this task is intended to be supported by an domain expert, whereas the BPM Tool uses the benefits of the immediate notification system. The communication of the tools is done by an XML based file exchange and due to some consistency issues the communication is only possible if both tools are running. I.e. deriving process variants is only possible if the BPM Tool is running too.

[2]http://www.intellior.ag

[3]http://www.pure-systems.com

4.3 pure::variants

pure::variants is a feature oriented domain modelling tool and is based on Eclipse. As such, it can easily be extended based on java plug-in development. During the implementation of this project, five different plug-ins where developed:

- An import plug-in, which is capable of importing the process structure - including the definition of variation points and the according variations - and converting it into a feature model compliant to the construction rules displayed in Figure 3 without the blue parts
- An extension to the internal model compare engine so that different versions of created feature models can be compared with each other
- An update mechanism to automatically search for deleted / added variations or updated process structures, providing graphical assistance for the domain expert.
- An extension to the internal model transformation engine so that the feature selection is automatically converted into a business process in the common data model; This process is then delivered to the attached BPM Tools, so that a native version of the process can be created/updated and executed
- Additions to the internal model check engine to model and create only valid processes (e.g. checks related to the feature selection, the consistency of the feature model, etc.)

To keep track of all generated business process variants, a list including all ids of the processes is stored and maintained in a file located in the same directory as the variant description model (feature selection), but hidden from the user perspective. This list is automatically updated when a process variant is deleted in the BPM Tool or created with the SPLE Tool. pure::variants also provides a framework for the comparison of different models, which enables the ability to compare different process variants in an efficient way.

4.4 Studied use cases and results

In the list below, use cases can be found which we investigated during the development of our approach regarding the performance of our toolchain (time saving). The according results are displayed in Figure 5, where the white bars are related to the manual case and the grey bars to our approach. Although the varBPM approach automatically integrated each derived process into the capabilities of the BPM Tool

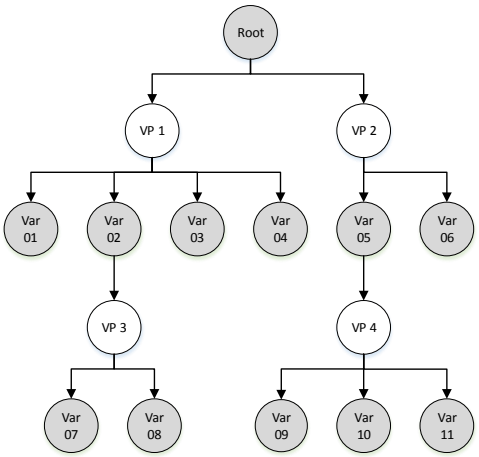


Figure 4: The variability structure of the process for the evaluation examples

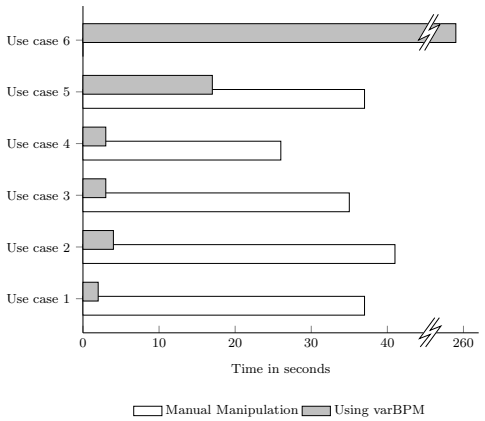


Figure 5: The results of the evaluation use cases. The grey parts are the time spent using the varBPM approach and the white parts states the time taken if a manual manipulation is used.

(documentation generation, workflow engine), the manual approach only reflects the time used for the creation of the bare process structure. The used process structure is part of a bigger process used for an on-demand manufacturing of spare parts for different car manufacturer.

Use case 1: For this use case, the time was measured that a domain expert needs to create a new process variant manually or using an existing feature model. The process setup consisted of four variation points (two top-level variation points and two variation points on lower levels) where a total number of twenty-seven different process variants were possible (for illustration, the variability structure can be seen in Figure 4). The number of other processes in the database of the BPM tool was considered to be low (20 other processes). The experiment was repeated with different experts and different process setups (the overall variability structure was kept the same but the process structures changed). The results of this use case are divided with the number of variation points to get a rough estimate for the time saving per variation point.

Use case 2: This use case is an addition to the first one, with the difference of a high number of other processes in the database (200 processes).

Use case 3: Is related to the topic of maintenance. In this scenario a process template was changed and all process variants should be updated. The domain expert was told that there is a number of six variants he needs to update providing only the name and the id of the changed process template. The size of the database was limited to 50 processes. The change to the template was a deletion/addition of a node. The results displayed in Figure 5 are normed to the update of one process.

Use case 4: This use case is an addition to the previous case, but with the difference that the domain expert was now told how the process variants where called (assuming that the domain expert was responsible for the creation of the process variants and exactly knows the variants).

Use case 5: In this situation, a new process variation was created by a process designer and the domain expert should now derive a new product variant out of it (in fact, a new variation for the Variation Point ”VP 3” was developed). It is assumed that the processes are designed according to our proposed design rules. The number of other processes in the database was considered as low (20 processes). This use case also gives a good estimate on how much overhead is produced, to update an existing feature model.

Use case 6: In this case, a new feature model was developed according to the variability structure displayed in Figure 4 to generate a metric on how much overhead is produces for the initial creation of the feature model. To get a rough estimate on the overhead per variation point, this number is divided by four.

The results of ”Use case 2” where surprising, since our developed approach performs worse in relation to the previous scenario. The reason for this is that the project now consists of ten times more processes, leading to a more time demanding search for the right processes. For humans, it was still quite easy to find the right processes since they were organised in a clear ”human understandable” manner. As a result the

increasing number of processes do not have a high influence to a manual manipulation if the processes are structured in a clear and meaningful way. Otherwise the time would increase more significantly.
To get a rough estimate of the break-even point, the following equation is used:

$$p \approx \frac{\text{Overall Overhead}}{\text{Average time saving}} = \frac{259}{4 \cdot 29} \approx 2.2 \quad (1)$$

The overall overhead is the time spent to create the feature model ("Use case 6") and the average time saving is calculated using the average time saving per variation point multiplied by the number of variation points. Interestingly, when software product line techniques are applied to pure software systems, the break-even point is also located at around three systems (according to Pohl et al. (2005)).

4.5 Restrictions

Depending on the API of the used BPM Tool, your approach can be limited in terms of the available features. This means that if the BPM Tool only provides access to the basic process structure, our framework is limited to the creation of derived processes without the ability to automatically integrate the models into the capabilities of the BPM Tool (e.g. workflow engine).

5 RELATED WORK

As stated in the survey of Fantinato et al. (2012), major challenges in the field of business process variability modelling are related to the reaction time of process changes and of the creation and selection of the right business process variants, which are also main topics in our approach.
Derguech (2010) presents a framework for the systematic reuse of process models. In contrast to our approach, it captures the variability of the process model at the business goal level and describes how to integrate new goals/sub-goals into the existing data structure. The variability of the process is not addressed in his work.
Gimenes et al. (2008) presents a feature based approach to support e-contract negotiation based on web-services (WS). A meta-model for WS-contract representation is given and a way is shown how to integrate the variability of these contracts into the business processes to enable a process automation. It does not address the variability of the process itself but enables the ability to reuse business processes for different e-contract negotiations.
While our approach reduces the overall process complexity by splitting up the process into layers with increasing details, the PROVOP project (Hallerbach et al. (2009a,b) and Reichert et al. (2014)) focuses on the concept, that variants are derived from a basic process definition through well-defined change operations (ranging from the deletion, addition, moving of model elements or the adaptation of an element attribute). In fact, the basic process expresses all possible variants at once, leading to a big process model.
The work Gottschalk et al. (2007) presents an approach for the automated configuration of workflow models within a workflow modelling language. The term workflow model is used for the specification of a business process which enables the execution of it in an enterprise and workflow management system. The approach focuses on the activation or deactivation of actions and thus is comparable to the PROVOP project for the workflow model domain.
Rosa et al. (2008) extends the configurable process modelling notation developed from Gottschalk et al. (2007) with notions of roles and objects providing a way to address not only the variability of the control-flow of a workflow model but also of the related resources and responsibilities.
The work of Leitner and Kreiner (2010) addresses the process variability through a bottom up approach by examining the possible configurations through the scan of the according ERP System (SAP). In contrast to this approach, we focus on an top down method to abstract the complexity of the underlying ERP System.
The Common Variability Language (CVL Haugen et al. (2013)) is a language for specifiying and resolving variability independent from the domain of the application. It facilitates the specification and resolution of variability over any instance of any language defined using a MOF-based meta-model. A CVL based variability modelling and a BPM model with an appropriate model transformation could lead to similar results as presented in our work.

6 CONCLUSION AND OUTLOOK

The reuse of business process models is an important step for an industrial company to survive in a competitive market. With our work we have proposed a way to combine the benefits of software product line engineering techniques with the capabilities of a business process modelling tool to provide a framework for the systematic reuse of business processes. With the proposed design rules, our approach results in an automatic detection and propagation of new and/or

changed business process variations. On the other hand it leads to an automatic integration of new assembled process variants into the BPM capabilities such as an automatic integration into a workflow engine, integration of responsibilities and resources, etc. Our developed framework covers the variability of the process in two different ways: Through the linkage of different process variations to variation points and through a variable process structure (deletion / addition of nodes) in each layer. Due to the fact that our developed framework is in an early stage of usage, further research efforts would address the collection and evaluation of data regarding the evolution and maintenance of the process models. In this context, an integration of Six Sigma[4] into our framework is aimed to provide a complete framework from modelling and improving process models. Additionally the customization of the ERP system of the underlying system (in this case SAP) is an interesting topic, providing a complete framework for the topics of process modelling, execution and maintenance including the planing of the resources of the concrete production facility.

ACKNOWLEDGEMENT

We want to gratefully thank Magna Cosma for sponsoring this project, Danilo Beuche from pure::systems and Intellior AG for their support.

REFERENCES

Brown, W. J., Malveau, R. C., McCormick, H. W. S., and Mowbray, T. J. (1998). *AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis: Refactoring Software, Architecture and Projects in Crisis*. John Wiley & Sons.

Derguech, W. (2010). Towards a Framework for Business Process Models Reuse. *In The CAiSE Doctoral Consortium*.

Fantinato, M., Toledo, M. B. F. d., Thom, L. H., Gimenes, I. M. d. S., Rocha, R. d. S., and Garcia, D. Z. G. (2012). A survey on reuse in the business process management domain. *International Journal of Business Process Integration and Management*.

Gimenes, I., Fantinato, M., and Toledo, M. (2008). A Product Line for Business Process Management. *Software Product Line Conference, International*, pages 265–274.

[4]Six Sigma is a set of techniques and tools for a systematic management of process improvements based on quality management methods, including some statistical methods.

Gottschalk, F., van der Aalst, W. M. P., Jansen-Vullers, M. H., and Rosa, M. L. (2007). Configurable Workflow Models. *International Journal of Cooperative Information Systems*.

Hallerbach, A., Bauer, T., and Reichert, M. (2009a). Guaranteeing Soundness of Configurable Process Variants in Provop. In *Commerce and Enterprise Computing, 2009. CEC '09. IEEE Conference on*, pages 98–105. IEEE.

Hallerbach, A., Bauer, T., and Reichert, M. (2009b). Issues in modeling process variants with Provop. In Ardagna, D., Mecella, M., and Yang, J., editors, *Business Process Management Workshops*, volume 17 of *Lecture Notes in Business Information Processing*, pages 56–67. Springer Berlin Heidelberg.

Hammer, M. and Champy, J. (1993). *Reengineering the Corporation - A Manifesto For Business Revolution*. Harper Business.

Haugen, O., Wasowski, A., and Czarnecki, K. (2013). Cvl: Common variability language. In *Proceedings of the 17th International Software Product Line Conference*, SPLC '13.

Kang, K., Cohen, S., Hess, J., Novak, W., and Peterson, A. (1990). Feature-oriented domain analysis (foda) feasibility study.

Karsai, G., Lang, A., and Neema, S. (2005). Design patterns for open tool integration. *Software & Systems Modeling*, pages 157–170.

Leitner, A. and Kreiner, C. (2010). Managing erp configuration variants: An experience report. In *Proceedings of the 2010 Workshop on Knowledge-Oriented Product Line Engineering*, KOPLE '10, pages 2:1–2:6.

McCormack, K. P. and Johnson, W. C. (2000). *Business Process Orientation: Gaining the E-Business Competitive Advantage*. Saint Lucie Press.

Österle, H. (1995). *Business Engineering - Prozess- und Systementwicklung*. Springer-Verlag.

Pohl, K., Böckle, G., and Linden, F. J. v. d. (2005). *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer.

Reichert, M., Hallerbach, A., and Bauer, T. (2014). Lifecycle Support for Business Process Variants. In Jan vom Brocke and Michael Rosemann, editor, *Handbook on Business Process Management 1*. Springer.

Rosa, M. L., Dumas, M., ter Hofstede, A. H. M., Mendling, J., and Gottschalk, F. (2008). Beyond control-flow: Extending business process configuration to roles and objects. In Li, Q., Spaccapietra, S., and Yu, E., editors, *27th International Conference on Conceptual Modeling (ER 2008)*, pages 199–215, Barcelona, Spain. Springer.

Valena, G., Alves, C., Alves, V., and Niu, N. (2013). A Systematic Mapping Study on Business Process Variability. *International Journal of Computer Science & Information Technology (IJCSIT)*.

Willaert, P., Van Den Bergh, J., Willems, J., and Deschoolmeester, D. (2007). *The Process-Oriented Organisation: A Holistic View - Developing a Framework for Business Process Orientation Maturity*. Springer.

Patterns for Common Criteria Certification

ANDREAS DANIEL SINNHOFER, Graz University of Technology
WOLFGANG RASCHKE, Graz University of Technology
CHRISTIAN STEGER, Graz University of Technology
CHRISTIAN KREINER, Graz University of Technology

One step in the development of certifiable secure systems is to provide trust in the development process and in the implemented security mechanisms of the product. In the domain of information technology, the Common Criteria schemes are used to evaluate the implemented security mechanisms of a product. Traditionally, a product certification is issued at a late stage of the development process, even though some Common Criteria evaluation paradigm exists to support the development process. The usage of such a paradigm would result in a beneficial certification process, since the evaluator gains experience through the maturing product. We have identified patterns which are designed to support the development process of secure applications. Based on these patterns, a systematic approach to integrate the evaluation process into the development process can be defined.

Categories and Subject Descriptors: I.5.0 [**Pattern Recognition**] General; D.2.0 [**Software Engineering**]: General—*Standards*

Additional Key Words and Phrases: Common Criteria, Process Pattern, Security Certification

1. INTRODUCTION

One step in the development of certifiable secure systems (i.e. a banking cards) is to provide trust in the development process and in the implemented security mechanisms of the product so that customers do not have any worries using the product. In the smart card domain, this is usually done by a Common Criteria certification which states a degree of security on so called Evaluation Assurance Levels (EAL) [Mayes and Markantonakis 2009]. The levels range from EAL-1 to EAL-7, where 1 is the lowest level and 7 the highest. For each level more or fewer evidences are needed (e.g. verified design, tests, etc.) which are performed and collected from one or more evaluation facilities. Based on these evidences and the verdict of the evaluation facilities, a certification body issues the certificate.

In an abstracted point of view, this is very similar to gaining trust into money. Traditionally, we have trust in money because there is somewhere a big safe which holds a lot of gold or other things of high value. The "value" of the money is low if the gold in the safe is low and the "value" is high if more gold is in this safe. From a certification point of view the evaluation facilities are holding the evidence (gold) and the more positive evidences they have, the higher is the trust in the product.

This work is supported by the Austrian Research Promotion Agency (FFG).
Author's address: Andreas Sinnhofer, Institut fuer Technische Informatik, Inffeldgasse 16, 8010 Graz; email: a.sinnhofer@tugraz.at; Wolfgang Raschke, Institut fuer Technische Informatik, Inffeldgasse 16, 8010 Graz; email: wolfgang.raschke@tugraz.at; Christian Steger, Institut fuer Technische Informatik, Inffeldgasse 16, 8010 Graz; email: steger@tugraz.at Christian Kreiner, Institut fuer Technische Informatik, Inffeldgasse 16, 8010 Graz; email: christian.kreiner@tugraz.at

At present, a common approach is to start the certification process of a product in a very late phase of the development, which can result in huge costs when the evaluation facility gives a negative attestation, because a redesign must be issued. Furthermore, an evaluation process typically takes a long period even if the verdict of the evaluation facility is positive (e.g. The certification process of Microsoft Windows 7 took one year and eight months[1]). Either way, both situations can potentially result in a loss of customers when a competitor is already selling a certified product. Furthermore, the Common Criteria scheme allows a various number of different evaluation paradigms to evaluate a product. Each of these paradigms is most suitable for specific circumstances which leads to the situation that it is often difficult to choose the right paradigm.
In this paper we will investigate process patterns for a common criteria evaluation, stating the applicable evaluation paradigm to reduce the needed time and money for the certification. Using these patterns, it is possible to integrate the responsible evaluation facilities from the beginning of the product development, resulting in a beneficial process since the feedback of the evaluation facility can be directly integrated into the process. The patterns were extracted by investigating the evaluation reports of various different product developments which can be found on the Common Criteria website ([Common Criteria 2015]). Our proposes patterns are designed for all kind of development scenarios were one or many companies can be involved in the development of a product and one or many evaluation facilities can be used to certify the resulting product.
The document is structured in the following way: Section 2 shortly summarizes the Common Criteria concepts and evaluation paradigms which are referenced in this paper. Section 2.4 summarizes related work on the field of modular and compositional Common Criteria evaluation schemes. In Section 3 we will describe our found patterns and Section 4 summarizes this work.

2. BACKGROUND

2.1 Common Criteria

Common Criteria (CC) (see [Common Criteria 2012a], [Common Criteria 2012b], [Common Criteria 2012c] and [Common Criteria 2012d]) is a standard for providing a common set of requirements for the implemented security functionality of Information Technology (IT) products during a security evaluation. These IT systems may be implemented in Software, in Hardware or as a composition of both. The evaluation process establishes a level of confidence into the security functionality of such systems and also investigates the development processes and used toolchains in case of high security critical products.
A so called evaluation facility is responsible for testing and evaluating the implemented security functionality. It collects all results in form of an Evaluation Technical Report (ETR). During this process, the evaluation facility is in close touch with the developing companies.

2.1.1 *Security Target (ST).*

The Security Target is a document which contains the implementation dependent statement of security needs for a specific identified target of evaluation. It describes the assets, their threats and the implemented countermeasures and is compiled by the involved vendors of the system. During the evaluation, it is determined if the stated countermeasures are sufficient enough to counter the threats. There exists two groups of countermeasures. The first group specifies the security objectives for the target of evaluation which are directly implemented by the system and for which correctness will be determined during the evaluation process. The second group describes the security objectives of the operational environment for non IT related countermeasures (such as physical countermeasures like security guards, etc.). This means that the second group is not directly implemented by the system itself but by the operational environment in which the system lives. This can also include guidelines for the

[1] 14th International Common Criteria Conference (ICCC) *https://www.commoncriteriaportal.org/iccc/ICCC_arc/presentations/T2_D2_2_30pm_Grimm_Evaluating_Windows.pdf*

customers of the system (e.g. not writing a private PIN-Code onto the banking card).
The Security Target is a document which is assembled by the developing companies and provided to the evaluation facility and will be made public available after the successful evaluation on the Common Criteria website [2].

2.1.2 *Evaluation Technical Report (ETR).*

The Evaluation Technical Report is a document which is assembled by the evaluation facility. It reports and documents the overall verdict and its justification and is submitted to a certification body which is issuing the certificate in case of a positive attestation.

2.1.3 *Formal and Informal Evaluations Paradigms.*

The Common Criteria Standard and its supporting documents defines formal and informal evaluation paradigms. The term "formal evaluation paradigm" is used, if the process of the paradigm is specified in the Common Criteria Standard or one of its supporting documents. The term "informal evaluation paradigm" is used if the evaluation process is not specified and thus must be defined with the according evaluation facility.

2.2 Common Criteria Evaluation Paradigms

This section gives a short description of the common criteria evaluation paradigms, which are referenced later in the patterns.

2.2.1 *Informal: Delta Evaluation.*

The delta evaluation is a certification paradigm which aims for a maximum reuse of previously compiled evidences. As stated in the "Common Criteria Information Statement on the reuse of evaluation results" (see [Common Criteria 2002]) the following evidences must be shared to reuse previously created evidences:

—Product and supporting documentation

—New security target(s)

—Original security target(s)

—Original evaluation technical report(s)

—Original certification/validation report(s)

—Original Common Criteria certificate(s)

—Original evaluation work packages (if available)

It is specified that

> "*... the evaluation facility conducting the current evaluation should not have to repeat analysis previously conducted where requirements have not changed nor been impacted by changes in other requirements ...*"

where such changes are identified through a so called delta analysis:

> "*... The evaluation facility would be required to perform a delta analysis between the new security target and the original security target(s) to determine the impact of changes on the analysis and evidence from the original evaluation(s) ...*"

[2] *https://www.commoncriteriaportal.org/products/*

which is similar to an impact analysis.
As a result, a product re-evaluation can be performed by an analysis of the impacts of changes and through evaluating only the changed and affected modules. Unaffected modules are not required to be re-certified.
A drawback of this approach is that the evaluation technical report is generated by the evaluation facility and thus, in some cases, is considered as proprietary to that facility, which makes the exchange of evidences between different certification facilities difficult.

Example: A company has developed a product which was certified some time ago (the black parts in Figure 1). Now they are starting to develop a new product based on the previously certified product but with some additional functionality (the grey parts in Figure 1). Adding the new module M 6 only influences Module M 4 and hence, only M 4 would need to be re-evaluated. For module M 5, the modules M 2, M 3, M 4 would need to be reconsidered.

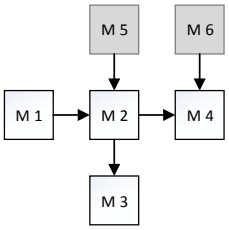


Fig. 1. Explanatory dependency graph for a delta evaluation

The dependencies of such a product can vary over time due to changing requirements and hence, tools for Change Impact Analysis (CIA) and Tools for a Traceability Impact Analysis (TIA) should be used to support the process of evolving security requirements (see Section 2.3).

2.2.2 *Formal: Composite evaluation.*

As stated in the "Common Criteria Mandatory Technical Document on the composite product evaluation for smart cards and similar devices" (see [Common Criteria 2012e]) a composite evaluation can be performed for all kind of products where

> "*... an independently evaluated product is part of a final composite product to be evaluated ...*"

and hence is not limited to smart cards only, but with the limitation that

> "*... The composite product is a product consisting of at least two different parts, whereby one of them represents a single product having already been evaluated and certified ... The underlying platform is the part of the composite product having already been evaluated ...*"

Example: The composite evaluation is applicable for an embedded system where an application runs on a certified OS (see Figure 2); respectively the OS is running on a certified hardware. I.e. a layers pattern is used for the product, whereby trust is established through each layer. The lowest EAL of all components is the limiting factor of the composite product. In this case maximum reuse is achieved due to the fact that the applications only influences the interface of the underlying OS. As one may notice, this is very similar to a delta evaluation with additional restrictions to the overall structure of the product. Due to these additional restrictions, it is not necessary to share the evaluation technical reports.

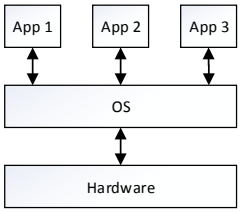


Fig. 2. Explanatory structure of a composite evaluation

2.2.3 *Formal: Composed evaluation.*

As stated in the Common Criteria part 3 (see [Common Criteria 2012c]), the composed evaluation is intended for situations, where independently certified (or going through an independent certification process) products/modules are assembled to a new product which should be certified. It is applicable, where a composite evaluation is not suitable and a delta evaluation cannot be performed due to missing evidences (proprietary documents are not shared). The composed evaluation mainly focuses on the interfaces between components and hence new evaluation levels were introduced. The levels are called Composed Assurance Packages (CAP) and ranges from CAP-A to CAP-C, where A is the lowest level and C the highest. Due to the fact, that composed evaluations are focused on the interfaces between components, a composed evaluation for higher assurance levels (higher than CAP-C[3]) is not supported through the composed scheme and hence a re-evaluation of the whole product is necessary. Due to this, composed evaluations have been performed much less successfully than composite evaluations.

Example: The hardware and software development of a product is done by two different companies. One of this company was responsible for the software and the other one for the hardware and each part was certified separately. For a low level product (e.g. an access control and time recording card for non security critical areas) only the interfaces between those two parts are evaluated.

2.3 Informal: Identification of the impact set

As mentioned in the above described paradigms, it is not necessary to perform unaffected evidence twice. Therefore, it is meaningful to use change detection analysis to determine the actual affected modules so that only these modules need to be reconsidered in the evaluation process. In the context of security, many modules can influence each other due to overlapping security requirements. To trace all these dependencies and to automatically detect the impact of changes, Raschke et al. [Raschke et al. 2014] introduced a change detection analysis based on the so-called *Security Model* using Change Detection Analysis (CDA) and Traceability Impact Analysis (TIA) techniques. The Security Model describes the properties and dependencies based on the security target, the design documentation, the implementation artefacts (i.e. the modules) and the tests. As such, it is possible to trace all impacts if modules or overall security requirements are changed.

[3]Attack potential "Enhanced Basic"; approximately comparable with EAL-4 (see [Common Criteria 2012c] pages 38 and 47)

2.4 Reuse and separation concepts for Common Criteria certification

The Euro-MILS[4] project focuses on providing a framework for trustworthiness by design and high assurance based on *Multiple Independent Levels of Security (MILS)*. In fact, assurance of the whole product is gained through the composition of assurance arguments of its components and the system's security architecture. It can be seen as the application of a layer pattern in combination with a strategy pattern applied to security artefacts. The developed framework is based on the Common Criteria evaluation schemes. They propose a layered and separated product development technique which can be seen in Figure 3. As it can be seen, it aims to be used in a composite product evaluation, where the hardware and the communication between applications is protected by a so called *Separation Kernel*. The presented separation kernel is a generic formal specification (see [Verbeek et al. 2014]) on how to achieve this kind of separation. This is essential for composite product evaluations with different levels of security since it needs to be proofed, that different components are not influencing each other. Further, this increases the re-usability of evidences since only new components need to be evaluated even if multiple evaluation facilities are involved into the certification process. Thus, it represents an implementation strategy for some of our presented patterns.

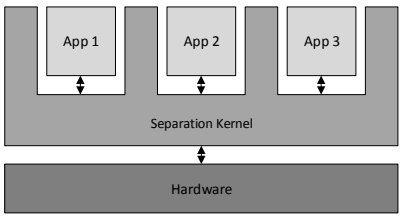


Fig. 3. MILS: Concept of the separation kernel

The work of Klohs [Klohs 2012] provides insights on the modularisation concepts for the development of an operating system for secure chip cards based on Common Criteria schemes. He strongly refers to the JIL document [Common Criteria 2012f] on the security architecture requirements for smart cards and similar devices and points out, that it is a first starting point to reuse software components. The reuse is based on a description of the security interface and the implemented security mechanisms which are independently developed from the concrete security target. This can be seen as an application of an abstraction pattern for a separation of concerns and thus we will use it in some of our patterns so that less information needs to be shared between different vendors or evaluation facilities.

The work of Mellado et al. is focused on the development of a generic and reusable framework for security requirements engineering for Common Criteria evaluated information systems [Mellado et al. 2007]. Using their framework it is possible to systematically reuse security requirements and the related countermeasures for different products and hence their framework helps developing reusable security architectures. In the context of our work the framework can be used together with processes for a change detection analysis to systematically trace all requirements to their according implementation. This increases the traceability of all requirements and illustrates the influence of modules to the system security in a more intuitive way.

[4] http://www.euromils.eu; [Blasum et al. 2014]

3. PATTERNS

3.1 **Pattern Name:** S-Ven: Single Vendor Evaluation Pattern

Context:

One company is developing a secure product which is evaluated at a central evaluation facility. The pattern is applicable for all situations where this condition is met, including for example the domain of operating systems (such as Microsoft Windows), secure hardware modules (random number generators, cryptographic co-processors, etc.) or access control devices. Products developed using this pattern are often components which are later integrated into other systems (e.g. a security critical computer consists of the certified OS and the certified hardware components).

Problem:

I want to develop a product which shall be evaluated at one central evaluation facility. Further I want to establish a good starting point for future products based on this certified product. I want to reduce the risk of a negative evaluation result to save time and money.

Forces:

—It is often difficult to argue why specific modules are not influencing others, due to complex dependencies among them.

—The management wants a "fast" and "cheap" process.

Solution:

Use a modular product development technique so that an informal[5] modular product certification process (i.e. Delta-Evaluation, as described in Section 2.2.1) can be used. Using techniques for an automatic detection of impacts (i.e. using the change impact analysis described in Section 2.3) produces strong arguments why specific modules of a product are unaffected during a development iteration and hence do not need to be reconsidered. Furthermore, these modules can be adopted from previously certified products if the evaluation facility is not changed. The basic concept of the certification process is illustrated in Fig. 4. In every new development iteration the updated and new modules are sent to the Evaluation Facility for a preliminary security evaluation. The resulting feedback of this evaluation should be directly integrated into the next iteration so that the maturity of the product is increased steadily during the development process.

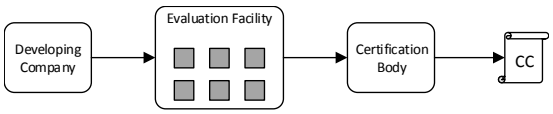


Fig. 4. Graphical representation of the S-Ven: Single Vendor Evaluation Pattern

Consequences:

—Pros:

—The evaluation facility is integrated from the early stages of the product development so that trust is gained through the whole development process and feedback can be directly integrated.

—The certificate can be issued shortly after the product development has finished.

[5] An informal certification process is a process which is not defined by the Common Criteria standard and needs to be arranged with the according evaluation facility

—Cons:
  —Since the evaluation process is informal, it can strongly vary according to the consulted evaluation facility.
  —Higher costs for the evaluation process itself since the evaluation facility is consulted multiple times compared to one single monolithic evaluation.

Known Uses:

An uncategorised list of all certified products can be found at [Common Criteria 2015] including all Security Targets and Evaluation reports. The following are some examples of products which were certified using this pattern:

—Secure Smart Card Controller, like the NXP P40, Infineon IC M7791 B12, etc.

—Hardware encryption modules and data encryption solutions, like SafeGuard Enterprise - Device Encryption

3.2 **Pattern Name:** S-Ven-M: Single Vendor multiple Evaluation Facilities Pattern

Context:

One company is developing a secure product, where a number of evaluation facilities (n $>$ 1) are involved in the certification process. This is for example the case if the developed product is a cross-domain product where specific parts of it are developed from different departments of a company (e.g. a split hardware and software development).

Problem:

I am developing a product which is partially evaluated at different evaluation facilities. I want to reduce the risk of a negative evaluation result to save time and money. Another common use case of this pattern is porting a certified operating system to a new certified hardware platform.

Forces:

—In many cases, the Evaluation Technical Reports (ETR) are considered to be proprietary to the evaluation facility and thus it is difficult to argue the security of these modules to the other facilities.

—Different Evaluation Facilities are performing different tests and hence it is difficult to establish trust between the involved evaluation facilities.

Solution:

First, one central evaluation facility must be chosen which coordinates the communication between each facility and which composes the final verdict for the certification body. Further the development techniques should be based on layered and separated product development methods as proposed by the Euro-Mils project (see Section 2.4) using the methods proposed by Klohs (see Section 2.4) based on a composite (see Section 2.2.2) or composed (see Section 2.2.2) evaluation scheme. Using this approach and integrating the evaluation facilities into the early product development process enables that the evaluation facilities are gaining trust through the architectural design of the product. This generates strong arguments for the security of the modules tested and evaluated by other evaluation facilities so that only less additional evidences are necessary to evaluate the whole product. Theoretically, it is also possible to use an informal approach (i.e. Delta Evaluation as stated in [Common Criteria 2002]) if the involved evaluation facilities are exchanging the Evaluation Technical Reports. The concept of this process is displayed in Fig. 5.

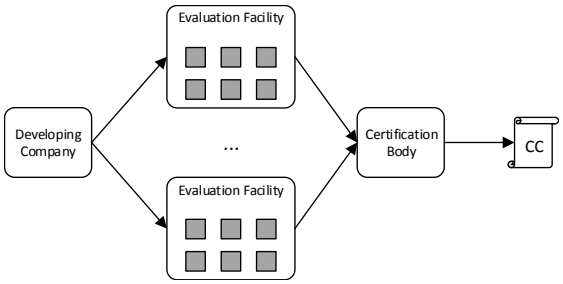


Fig. 5. Graphical representation of the S-Ven-M: Single Vendor multiple Evaluation Facilities Pattern

Consequences:

—Pros:
- —Using the Euro-Mils approach and using the proposed methods by Klohs results in a reusable architecture of the product so that supporting different modules/platforms and security targets requires less effort.
- —Early integration of the evaluation facilities into the development process so that their feedback can directly be considered in the next iteration.
- —The certificate can be issued shortly after the product development has finished.
- —Due to the architecture, trust can be established without exchanging ETRs.

—Cons:
- —Formal evaluation paradigm and thus tending to be more cost intensive. Further the costs for the evaluation process are higher since the evaluation facility is consulted multiple times compared to a single monolithic evaluation.
- —Arranging the communication and the exchange of evidences between the evaluation facilities is often time intensive.

Known Uses:

An uncategorised list of all certified products can be found at [Common Criteria 2015] including all Security Targets and Evaluation reports. The following are some examples of products which were certified using this pattern:

—Secure Smart Cards, like the PayPass or eTravel developed by Gemalto.

3.3 **Pattern Name:** Mu-Ven: Multiple Vendors Evaluation Pattern

Context:

Several companies (n $>$ 1) are involved in the development process of the product, but one central evaluation facility is performing the evaluation process. Each company is participating some modules which are finally assembled to a whole product. This is a very common situation in the industry since secure products are often developed in cooperation with many other companies. A secure computer would be one practical example, where a number of companies can be involved into the development process of the hardware, the software and the applications running on this computer (usually one company is responsible for the OS, but many can be involved in the development of the hardware).

Problem:

A product is developed in cooperation with different companies. I want to minimize the evaluation costs and do not want to expose too much information about my own development techniques to the other companies.

Forces:

—Vendors do not want to provide internal design documents and development process evidences to other vendors, even though confidentiality agreements are signed.

—Coordinating the development efforts of multiple companies is often difficult.

Solution:

Only little information needs to be shared between the companies since the central evaluation facility has full access to every contribution. The evaluation facility needs to be integrated from the early product development so that each vendor gains trust through the statements of the evaluation facility. Thus, it is possible to use an informal modular product certification process (i.e. Delta-Evaluation as described in Section 2.2.1). Using techniques for an automatic detection of impacts (i.e. using the change impact analysis described in Section 2.3) produces strong arguments why specific parts (modules) of a product are not affected during a development iteration and hence do not need to be reconsidered. Further, it enables a transparent tracing of the modules of each company on the overall system security. The usage of a modular development process enables each vendor to concentrate on their own parts with clear and defined interfaces to parts of other vendors. The concept of this process can be seen in Fig. 6.

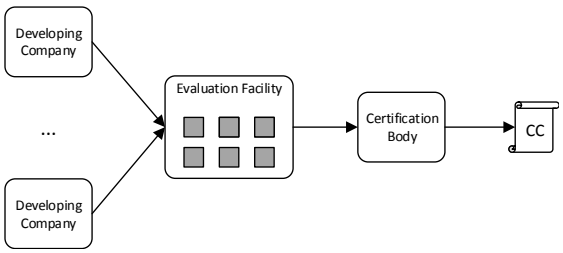


Fig. 6. Graphical representation of the Mu-Ven: Multiple Vendors Evaluation Pattern

Consequences:

—Pros:
- —The evaluation facility is integrated from the early stages of the product development so that trust is gained through the whole development process and feedback can be directly integrated.
- —Informal evaluation paradigm and thus, tending to be less time and money intensive compared to formal schemes.
- —Vendors do not need to expose all internal design documents and development processes to other vendors.

—Cons:
- —Since the evaluation process is informal, it can strongly vary according to the consulted evaluation facility.
- —Higher costs for the evaluation process itself since the evaluation facility is consulted multiple times compared to one single monolithic evaluation.

Known Uses:

An uncategorised list of all certified products can be found at [Common Criteria 2015] including all Security Targets and Evaluation reports. The following are some examples of products which were certified using this pattern:

—SmartCards which were developed from multiple companies, like the dragonFly (Oberthur Technologies and STMicroelectronics), Athena IDProtect (Athena Smartcard Solutions Inc. and Inside Secure S.A.), etc.

Related Patterns:

The S-Ven pattern presented in this work can be seen as a foundation of this pattern since every vendor can use the S-Ven pattern for all parts of the product which are separated from the parts of other companies.

3.4 **Pattern Name:** Mu-Ven-M: Multiple Vendors and Evaluation Facilities Pattern

Context:

Several companies are involved in the development process of the product and any number of evaluation facilities (n > 1) are included into the certification process (e.g. each company consults a different evaluation facility). This is a very common situation in the industry since secure products are often developed in cooperation with many other companies. A secure computer would be one practical example, where a number of companies can be involved into the development process of the hardware, the software and the applications running on this computer (usually one company is responsible for the OS, but many can be involved in the development of the hardware).

Problem:

A product is developed in cooperation with different companies and different evaluation facilities leading to a number of different requirements. I want to minimize the evaluation costs and do not want to expose too much information about my own development techniques to the other companies.

Forces:

—In many cases, the Evaluation Technical Reports (ETR) are considered to be proprietary to the evaluation facility and thus it is difficult to argue the security of these modules to the other facilities.

—Vendors do not want to provide internal design documents and development process evidences to other vendors.

—It is often difficult to argue why specific modules are not influencing others, due to complex dependencies among them.

Solution:

First, one central evaluation facility must be chosen which coordinates the communication between each facility and which composes the final verdict for the certification body. Further the developing techniques should be based on layered and separated product developing methods as proposed by the Euro-Mils project (see Section 2.4) using the methods proposed by Klohs (see Section 2.4) based on a composite (see Section 2.2.2) or composed (see Section 2.2.3) evaluation scheme. Thus, a modular and incremental product development process should be used. Using this approach and integrating the evaluation facilities into the early product development process enables that the evaluation facilities are gaining trust through the architectural design of the product. This generates strong arguments for the security of the modules tested and evaluated by other evaluation facilities so that only less additional evidences are necessary to evaluate the whole product. Vendors mainly need to give insight in the interfaces between each component / layer and hence do not expose too many business secrets to other vendors.

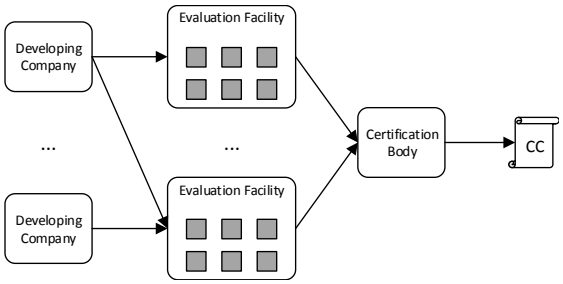


Fig. 7. Graphical representation of the Mu-Ven-M: Multiple Vendors and Evaluation Facilities Pattern

Theoretically, it is also possible to use an informal approach if the involved evaluation facilities also exchanges the Evaluation Technical Reports (ETR) and each company gives insights in their development techniques and methods (see [Common Criteria 2002]), but this is practically never happening. The concept of this process is illustrated in Fig. 7.

Consequences:

—Pros:
- —Potentially a reusable architecture of the product so that supporting different modules/platforms and security targets requires less effort.
- —Early integration of the evaluation facilities into the development process so that their feedback can directly be considered in the next iteration
- —Vendors do not need to expose all internal design documents and development processes to other vendors.
- —The certificate can be issued shortly after the product development has finished.

—Cons:
- —Formal evaluation paradigm and thus tending to be more cost intensive. Further the costs for the evaluation process are higher since the evaluation facility is consulted multiple times compared to a single monolithic evaluation.
- —Arranging the communication and the exchange of evidences between the evaluation facilities and the developing companies is often time intensive

Known Uses:

An uncategorised list of all certified products can be found at [Common Criteria 2015] including all Security Targets and Evaluation reports. The following are some examples of products which were certified using this pattern.

Multi-Application Smart Cards like the eTravel MultiApp on P60 (Gemalto and NXP) or on the M7820 A11 (Gemalto and Infineon)

Related Patterns:

The S-Ven-M and the Mu-Ven patterns presented in this work can be seen as a foundation of this pattern since every vendor can use these patterns for all parts of the product which are separated from the parts of other companies.

4. CONCLUSION AND FUTURE WORK

Today's industry is defined by fast changing requirements regarding functional and security needs. Patterns have been proven to increase the quality of the product by systematically applying strategies to common problems. We have identified patterns for security certifications based on the Common Criteria scheme. Selecting the appropriate pattern in the according situation reduce the time shift between the successful certification and the time the product development is finished. Furthermore, the costs for re-evaluating the developed product/modules can be kept as low as possible since the most suitable paradigm is chosen, maximizing the reuse of already evaluated modules and providing a direct integration of the evaluation facility in the process so that the feedback is directly integrated into the next development iteration.
However, reusing previously certified products is a difficult task and hence future research should focus on composite product evaluations so that the exchange of evidences is supported and documented in a standardized document. An objective should be an approach which is similar to the state-of-the-art process in the domain of functional safety (Safety element our of context [Schneider et al. 2012]). There, different evaluated products of different suppliers can be combined and the overall system safety can be determined using methods for composite products which are defined in the standard.

Acknowledgement

Project partners are NXP Semiconductor Austria GmbH and the Technical University of Graz. The project is funded by the Austrian Research Promotion Agency (FFG). Further, we want to gratefully thank our Shepard Azadeh Alebrahim for her continuous and constructive feedback. And finally we want to thank all participants of the Security Pattern Workshop at EuroPLoP'15 for their feedback.

REFERENCES

BLASUM, H., TVERDYSHEV, S., LANGENSTEIN, B., MAEBE, J., SUTTER, B. D., LECONTE, B., TRIQUET, B., MÜLLER, K., PAULITSCH, M., VON BLOMBERG, A. S.-F., AND TILLEQUIN, A. 2014. Secure European Virtualisation for Trustworthy Applications in Critical Domains - MILS Architecture.

COMMON CRITERIA. 2002. Common Criteria Information Statement. Reuse of Evaluation Results and Evidence. (October).

COMMON CRITERIA. 2012a. Common Criteria for Information Technology Security Evaluation. Part 1: Introduction and general model. Version 3.1 Revision 4 (September).

COMMON CRITERIA. 2012b. Common Criteria for Information Technology Security Evaluation. Part 2: Security functional components. Version 3.1 Revision 4 (September).

COMMON CRITERIA. 2012c. Common Criteria for Information Technology Security Evaluation. Part 3: Security assurance components. Version 3.1 Revision 4 (September).

COMMON CRITERIA. 2012d. Common Methodology for Information Technology Security Evaluation. Evaluation methodology. Version 3.1 Revision 4 (September).

COMMON CRITERIA. 2012e. Common Criteria Supporting Document Mandatory Technical Document - Composite product evaluation for Smart Cards and similar devices. Version 1.2 (April).

COMMON CRITERIA. 2012f. Common Criteria Supporting Document Guidance - Security Architecture requirements (ADV_ARC) for smart cards and similar devices. Version 2.0 (April).

COMMON CRITERIA. 2015. Common criteria certified products. `https://www.commoncriteriaportal.org/products/`. Accessed: 2015-03-30.

KLOHS, D. K. 2012. Software modularisation and the common criteria - a smartcard developer's perspective.

MAYES, K. AND MARKANTONAKIS, K. 2009. *Smart Cards, Tokens, Security and Applications* 1 Ed. Springer.

MELLADO, D., FERNÁNDEZ-MEDINA, E., AND PIATTINI, M. 2007. A common criteria based security requirements engineering process for the development of secure information systems. *Comput. Stand. Interfaces 29*, 2, 244–253.

RASCHKE, W., ZILLI, M., BAUMGARTNER, P., LOINIG, J., STEGER, C., AND KREINER, C. 2014. Supporting evolving security models for an agile security evaluation.

SCHNEIDER, R., BRANDSTAETTER, W., BORN, M., KATH, O., WENZEL, T., ZALMAN, R., AND MAYER, J. 2012. Safety element out of context - a practical approach. *SAE Technical Paper*.

VERBEEK, F., TVERDYSHEV, S., HAVLE, O., BLASUM, H., LANGENSTEIN, B., STEPHAN, W., NEMOUCHI, Y., FELIACHI, A., WOLFF, B., AND SCHMALTZ, J. 2014. Formal specification of a generic separation kernel. `http://afp.sf.net/entries/CISC-Kernel.shtml`, Formal proof development.

EuroPLoP '15, July 08 - 12, 2015, Kaufbeuren, Germany

ACM 978-1-4503-3847-9/15/07...$15.00
DOI: http://dx.doi.org/10.1145/2855321.2855355

A Framework for Process driven Software Configuration

Andreas Daniel Sinnhofer[1], Peter Pühringer, Klaus Potzmader[2], Clemens Orthacker[2], Christian Steger[1] and Christian Kreiner[1]
[1]*Institute of Technical Informatics, Graz University of Technology, Austria*
[2]*NXP Semiconductors, Gratkorn, Austria*
{*a.sinnhofer, christian.kreiner, steger*}*@tugraz.at, p.puehringer@inode.at,* {*klaus.potzmader, clemens.orthacker*}*@nxp.com*

Keywords: Software Product Lines, Feature Oriented Modelling, Business Processes, Tool Configuration

Abstract: Business processes have proven to be essential for organisations to be highly flexible and competitive in today's markets. However, good process management is not enough to survive in a market if the according IT landscape is not aligned to the business processes. Especially industries focused on software products are facing big problems if the according processes are not aligned to the overall software system architecture. Often, a lot of development resources are spent for features which are never addressed by any business goals, leading to unnecessary development costs. In this paper, a framework for a business process driven software product line configuration will be presented, to provide a systematic way to configure software toolchains.

1 INTRODUCTION

Business Process (BP) oriented organisations are known to perform better regarding highly flexible demands of the market and fast production cycles (e.g. McCormack and Johnson (2000); Valena et al. (2013); Willaert et al. (2007)). This is achieved through the introduction of a management process, where business processes are modelled, analysed and optimised in iterative ways. Nowadays, business process management is also coupled with a workflow management, providing the ability to integrate the responsible participants into the process and to monitor the correct execution of it in each process step. To administer the rising requirements, so called business process management tools are used (BPM-Tools) which cover process modelling, optimization and execution. In combination with an Enterprise-Resource-Planning (ERP) system, the data of the real process can be integrated into the management process.

In the domain of software products, different choices in business processes lead to different software configurations. To handle variability automatically is a challenging task because the variability of the process model needs to be reflected in the software architecture. Further, the actual customer choice during the ordering process needs to be mapped to the according software features. Due to this, software configuration is often done manually which takes a considerable amount of time during production. Particularly for resource constraint devices like embedded systems, it is vital to have a working software configuration process since unnecessary features may occupy a lot of memory. Further, it is important to have a software architecture which is synchronised with the business goals. Otherwise, a lot of resources are spent for developing and maintaining software components which are never used anyway. Thus, process awareness is crucial for an efficient development.

Context Aware Business Process modelling is a technique for businesses living in a complex and dynamic environment (Saidani and Nurcan (2007)). In such an environment a company needs to tackle changing requirements which are dependent on the context of the system. Such context sensitive business process models are able to adapt the execution of their process instances according to the needs, such that the company can react faster and more flexible. This is achieved by analysing the context states of the environment and mapping these states to the according business processes and their related software system. The problem with such approaches is, that the used software systems are often developed independently from each other, although they share a similar software architecture. Therefore, this work focuses on the development of a framework which covers the variability of process models and mapping such variable process structures to software configuration artefacts such that the software system can be adapted automatically with respect to its context. This

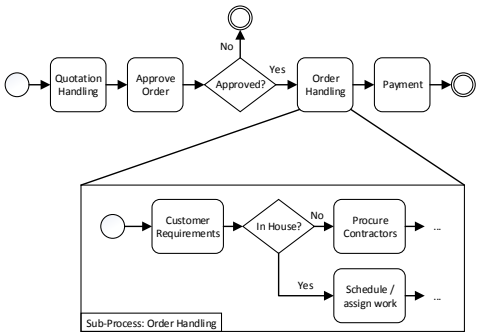


Figure 1: Exemplary order process to illustrate the basic concepts defined by Österle (1995): A high level description of the process is split into its sub-processes until a complete work description is reached.

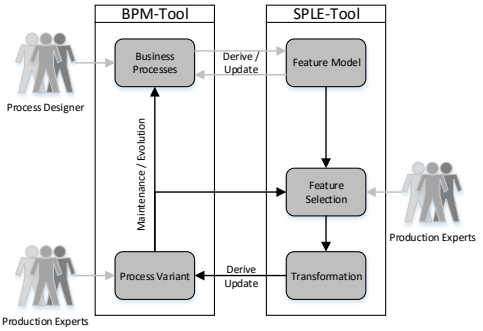


Figure 2: Used framework for an automatic business process variant generation (adapted from Sinnhofer et al. (2015)). The grey lines indicate process steps which need to be done manually.

is achieved through software product line engineering techniques. Thus, only one system needs to be developed and maintained for whole product families. The modelling of business process variability is based on our previous work, which can be found in Sinnhofer et al. (2015). In particular, a SPLE Tool was used to systematically reuse expert knowledge in form of valid process variations, designed in an appropriated BPM Tool. The integrity of the process variations is secured by the capabilities of the BPM Tool and a rich cross functional constraint checking in the SPLE Tool. This work will extend the framework in order to be able to map process artefacts to software configurations. Hence, software toolchains can be configured in an automatic way and the architecture can be kept aligned with the business goals.

This work is structured in the following way: Section 2 gives an overview over the used design paradigm for business processes modelling and Software Product Line Engineering techniques which were needed for the framework. Section 3 summarizes the concept of our work and Section 4 describes our implementation in an industrial use case. Finally, Section 5 summarizes the related work and Section 6 concludes this work and gives an overview over future work.

2 BACKGROUND

2.1 Business Processes

A business process can be seen as a sequence of tasks/sub-processes which need to be executed in a specific way to produce a specific output with value to the costumer (Hammer and Champy (1993)). According to Österle (1995) the process design on a macroscopic level (high degree of abstraction) is split up into sub-processes until the microscopic level is reached. This is achieved, when all tasks are detailed enough, so that they can be used as work instructions. An exemplary order process is illustrated in Figure 1. As illustrated, the top layer is a highly abstracted description, while the production steps are further refined on the lower levels. As a result, the lowest level is highly dependable on the concrete product and production environment, providing many details for the employees. Usually, the top layers are independent from the concrete plant and the supply chain and could be interchanged throughout production plants. Only the lower levels (the refinements) would need to be reconsidered. Variability of such a process structure can either be expressed through a variable structure of a process/sub-process (e.g. adding/removing nodes in a sequence) or by replacing the process refinement with different processes.

Traditionally, processes for similar products are created using a copy and clone strategy. As a result, maintaining such similar processes is a time consuming task, since every improvement needs to be propagated manually to the respective processes. To solve this issue, we proposed a framework to automatically derive process variants from business process models by modelling the variable parts of a process using Software Product Line Engineering techniques in a previous work (see Sinnhofer et al. (2015)). The presented framework can be split into four different phases which are illustrated in Figure 2. In the first phase, process designers create process templates in a BPM tool, adding all wished features like documentation artefacts, responsible workers or resources. In the second phase, the created processes are imported

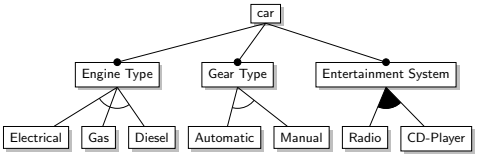


Figure 3: An exemplary feature model of a car.

into the SPLE tool and added to a feature model. Process experts define a comprehensive set of rules and restrictions so that only valid process variants can be derived from the model. The third phase is called the feature selection phase in which production experts will automatically derive processes for their needs based on a selection of features. The fourth phase consists of maintenance and evolution. There, data is collected and used to improve process designs or feature selections.

2.2 Software Product Line Engineering

SPLE applies the concepts of product lines to software products (Kang et al. (1990)). A Software Product Line can be seen as a set of domain features, which are automatically assembled and configured to a whole software project just by choosing the wanted features. Instead of writing code for a whole system, the developer divides the project into small lightweight features which are implemented in so called domain artefacts. For this, a software architecture is needed in which the variation points and the respective variants (features) are explicitly modelled. Further, a source code generator is needed which is able to generate the according software products, based on the according feature selection.

Features are usually modelled in so called 'Feature Models' which describe all features of a product and explicitly states their relationships, dependencies and additional restrictions between each other. Figure 3 illustrates an explanatory feature model for a car. A car consists of three mandatory variation points (Engine Type, Gear Type, Entertainment System) and their respective variants. For example, the Engine Type of the car could be Electrical, Gas or Diesel powered. The variants of the 'Engine Type' and 'Gear Type' variation point are modelled as alternative features which means that exactly one variant needs to be chosen. In contrast, the 'Entertainment System' is modelled in such a way, that either one or both options can be chosen.

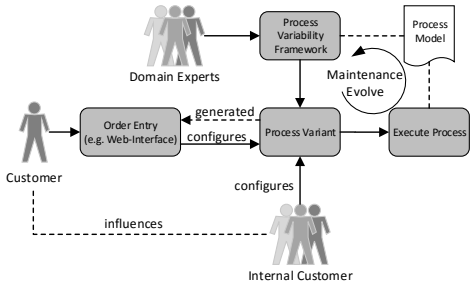


Figure 4: Overall conceptual design of the framework. The "Process Variability Framework" block is described in Figure 2.

3 VARIABILITY FRAMEWORK

The goal of the developed framework is to implement a systematic way to keep the business processes aligned with the IT infrastructure so that development costs can be reduced and the company is more flexible to changes of the market. The following Sections summarizes our developed framework.

3.1 Conceptual Design

The overall conceptual design is based on a feature oriented domain modelling framework and is displayed in Figure 4. As illustrated in the Figure, Domain Experts are responsible for operating the "Process Variability Framework" as already described in Section 2.1. They design process models based on their domain knowledge and generate process variants for various types of product platforms. Based on this variants, the used SPLE tool also generates a order entry form, stating explicitly which kind of information a customer needs to submit, to be able to order the product. For example, if the customer can decide which applications should run on his device or if the device can be personalized by adding signatures of the customer. Complex products usually tend to have a lot of internal stackholders which can be seen as internal customers. This means that based on the customer needs, specific stackholders may be addressed to further submit needed information or even parts of the product. For instance, if a product can run on multiple hardware platforms, each of these platforms may be developed by different departments or even different companies which need to be ordered and shipped accordingly. To be able to automatically generate the order entry forms, additional information needs to be added to the process models. This can be done by either adding this information into the process model itself (i.e. using the BPM tool) or by us-

ing the capabilities of the SPLE tool and mapping this information to the according process models. Option two is the more generic approach which also has the positive side-effect, that the processes itself are not ”polluted” with information that may change in different circumstances. On the other hand, it rises high requirements to the SPLE tool which needs to support product family models so that the process model and the additional information used for the order entry can be kept aligned, but separated which increases the reusability factor.

After all needed data is collected, the process can finally be executed and the ordered products are manufactured. Especially for new products, it is likely that during this manufacturing process knowledge is gained on how to increase the efficiency of the whole process(es) by introducing specific changes to the process model. Further, changes to the generated order entry may be identified, which means that specific parts of the product need to be made selectable. The advantage of using one core of process models for a specific family of products is that the gained knowledge can be rolled out in an automatic way for the whole product family. This means that the required changes only need to be implemented once.

3.2 Type model

To automatically generate order entry forms from a feature selection, the used model needs to support the following types:

- **Inputs**: Is the abstract concept of different Input types which are described below.
- **None**: No special data needs to be submitted and hence a node (i.e. task in a process) marked with none will not appear as a setting in the order entry form.
- **Customer Input**: Specific data need to be added from a customer. A node marked with this will generate an entry in the order entry form of a specific type. For example a file upload button will appear if a customer needs to submit specific files.
- **Internal Input**: Specific data or parts of the product needs to be delivered from an internal stackholder. This information is directly submitted to the internal stackholder as a separate order.

Furthermore, the family model should support the concept of choices (i.e. a customer needs to submit one of possible n options) and multiple inputs if multiple submissions are needed for a single node. Also multiple inputs of multiple different stackholder need to be supported.

3.3 Process driven Software Toolchain configuration

An established process management, which is able to generate order entry forms and trigger internal processes, is a big step towards good business management. However, to be successful on the market it is not enough to just focus on well managed processes, but also on an aligned IT infrastructure. Hence, the big remaining challenge is having an IT infrastructure which is able to be configured directly from the according business processes.

For illustration purposes let’s consider the following example: A company is developing small embedded systems which are used as sensing devices for the internet of things. The device is offered in three different variants with the following features:

- Version 1: Senses the data in a given time interval and sends the recorder signal to a web-server which is used for post-processing.
- Version 2: Additionally to the features of Version 1, this version allows encryption of the sensed data using symmetric cryptography before it is sent to the web-server. This prevents that third parties are able to read the data. For simplicity, we assume that this key is provided in plain from the customer.
- Version 3: Additionally to the features of Version 2, this version also allows customer applications to be run (e.g. data pre-processing routines) on the system.

It is not economic feasible to personalize each device manually if it is sold in high quantities. Further, establishing three different order processes using three different versions of customization toolchains will result in higher maintenance efforts. To summarize the findings of this short example, it is fundamental to have a software architecture which is synchronized with the according business process(es). This means that variable parts of the process model need to be reflected by a variable software architecture. Further, minor changes to the process model (e.g. addition of new configuration settings) should not lead to huge development efforts since – ideally – the software architecture does not need to be changed. These requirements lead to the architecture displayed in Figure 5. As illustrated, the tool is basically an interpreter which can be ”dynamically programmed” for the actual order. This means that variability of the architecture is gained by shifting features from the implementation phase to the configuration phase. To ensure that such freedom is not misused, it is necessary to enforce specific rules in the Interpreter Tool (e.g.

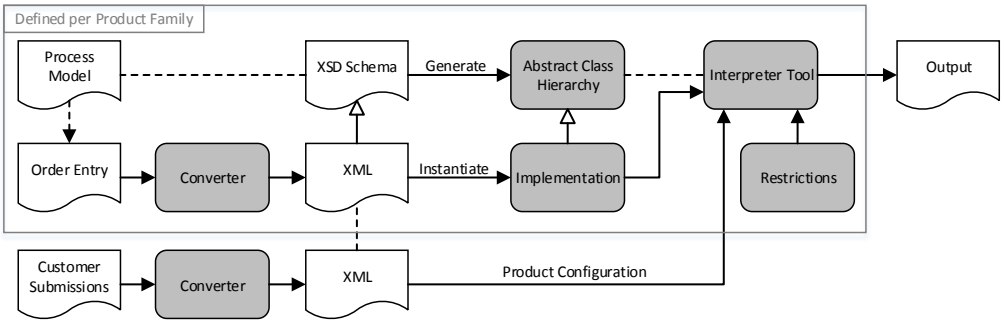


Figure 5: The architecture of the software tool responsible for generating the wished product outcome.

security requirements). Based on the Process Model of the Process Variability Framework, a schema file is created which states all possible operations and all additional language primitives (like conditional statements, etc.) the Interpreter Tool can perform. This step is semi-automatic which means that only a skeleton of the needed functionality can be generated automatically.

For illustration purposes we will reconsider the previous example: Basically, there are three different order processes, where in the first case a customer can customize a connection string for his web-server. In the second case he can further submit a key which is stored onto the nodes and in the third case executables can be submitted to be loaded to the chip. Taking this into account, the XML illustrated in Listing 1 can be generated. Each function consists of a *Configuration* block and a *Translate* block. The Configuration block is used to indicate which data needs to be provided from the customer submissions (i.e. from the "real" product configuration) and how often they can occur (configuration safety). This Configuration blocks are further used to generate a schema file which is used by the converter tool to convert the Customer Submissions into the needed XML structure. The Translate block defines how the submitted data is processed. This cannot be generated and hence a developer is needed who needs to define this transformation based on the language primitives of the Interpreter Tool. This needs to be done only once for a whole product family. For this particular example, only a Store-Data and an Install-Application routine would need to be offered by the interpreter. Additional restrictions are domain depended and could contain in that example the following checks: Verification that the submitted key is of reasonable strength (e.g. AES key with a

Listing 1: Generated XML based on the Order Entry. The Translate blocks need to be edited manually by a developer.

```
<?xml version="1.0" encoding="UTF-8"?>
<Functions>
  <Function id="WebServer"
            minOccurs="1"
            maxOccurs="1">
    <Configuration>
      <Parameter name="Connection" type="ipAddress" />
    </Configuration>
    <Translate> ... </Translate>
  </Function>
  <Function id="EncryptionKey"
            minOccurs="0"
            maxOccurs="1">
    <Configuration>
      <Parameter name="Key" type="hexstring" />
    </Configuration>
    <Translate> ... </Translate>
  </Function>
  <Function id="InstallApplication"
            minOccurs="0"
            maxOccurs="unbounded">
    <Configuration>
      <Parameter name="Binary" type="fileUri" />
    <Configuration>
    <Translate> ... </Translate>
  </Function>
</Functions>
```

Listing 2: Exemplary generated Configuration file based on customer submissions. Two different versions are shown. The first example illustrates a "Version 3" product and the second one a "Version 1" product.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomerOrder>
  <WebServer>
    <Connection>X.X.X.X</Connection>
  </WebServer>
  <EncryptionKey>
    <Key>0x01020304...<Key>
  </EncryptionKey>
  <InstallApplication>
    <Binary>file://orderXYZ/app1.elf</Binary>
  </InstallApplication>
  <InstallApplication>
    <Binary>file://orderXYZ/app2.elf</Binary>
  </InstallApplication>
</CustomerOrder>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomerOrder>
  <WebServer>
    <Connection>X.X.X.X</Connection>
  </WebServer>
</CustomerOrder>
```

minimum length of 16 bit) and that the submitted applications are protected by a signature of the customer to ensure that they are not replaced by a malicious third party. If a product is ordered, the filled order entry (i.e. customer submissions) is converted into a configuration file which instantiates the specific features of the product. For example Listing 2 shows the generated Configuration file for a "Version 3" product (top one) and a "Version 1" product (bottom one).

4 INDUSTRIAL CASE STUDY

In this section an overview over our industrial case study is given. The implemented business processes of our industrial partner are controlled by an SAP infrastructure and are designed with the BPM-Tool Aeneis. Further, pure::variants is used as SPLE tool to manage the variability of the business processes. Thus, our implemented prototype is also based on pure::variants and Java.

4.1 SPLE-Tool: pure::variants

pure::variants is a feature oriented domain modelling tool which is based on Eclipse. As such, it can easily be extended based on Java plug-in development. During the implementation of this project, five different plug-ins where developed:

- An extension to the import plug-in which was developed in our previous work. It assists the Process Designers in modelling cross functional requirements and providing the needed information for the code generators.
- An extension to the internal model compare engine for comparing different versions of created feature models with each other.
- An extension to the internal model transformation engine to convert the feature selection of the process model into the according order entry form. This also generates the back-end to trigger processes for internal stackholders.
- Additions to the internal model check engine to model and create only valid processes (e.g. checks related to the feature selection, the consistency of the feature model, etc.)
- Generator Tools which are able to generate the skeleton of the schema file (as described in Section 3.3) and the order entry form (a generated Web-Interface). Additionally, converter tools were written which are converting the generated forms and received submissions into the related XML files.

4.2 Implementation of the interpreter tool

As mentioned in Section 3.3, the class hierarchy should be generated from a schema file, thus we used the tool Jaxb (a Java architecture for XML binding) to generate the bare class hierarchy which needs to be implemented by the software developers. Since the creation of the schema file is semi-automatic, our developed framework (implemented in pure::variants) opens a dialogue which hints the domain expert to check the validity of the schema file to ensure that the changes to the processes are always propagated to the schema file. Since our industry partner is working in a safety and security critical domain, additional restrictions are implemented. Formal verification rules are implemented to check that confidential data is not leaked and rules are defined to check the configuration safety such that no invalid configuration can be submitted and executed. These restrictions will be part of a future publication.

4.3 Evaluation

The framework was successfully deployed for two different product families which are based on the same Process Model. The time was measured to implement the initial system and the overhead to support the two systems to get an effort estimation which can be compared with a traditional software development. We use the term "traditional software development" for a software development with ad-hoc (or near to ad-hoc) software architecture which means that multiple different systems are designed almost independently. This leads to the situation that only a little code base is shared between each software project since most of the code is optimized for single purposes. However, this code would be reusable if adaptations of the interfaces / implementations would have been considered. The effort for the traditional software development was based on the real implementation time for the first system and an effort estimation to port the existing family to the new one. These numbers were given by the responsible developers. As illustrated in Table 1, the break-even point will be between 3 to 4

Table 1: Effort measurements and estimations in man-month to develop the systems.

	Framework	Traditional
Base System	12	-
Product Fam. 1	1	6
Product Fam. 2	0.5	4 - 5
Overall	14,5	10 - 11

systems using a curve fitting interpolation. This number also correlates to the typical number presented in relevant software product line publications (e.g. Pohl et al. (2005)). Additionally, the maintenance cost can be reduced since fixing problems in one product family will fix this issue in all others as well.

5 RELATED WORK

As stated in the survey of Fantinato et al. (2012), major challenges in the field of business process variability modelling are related to the reaction time of process changes and of the creation and selection of the right business process variants, which are also main topics in our framework since the time to adopt the IT infrastructure to the changed business processes can be reduced with the new framework.

Derguech (2010) presents a framework for the systematic reuse of process models. In contrast to this work, it captures the variability of the process model at the business goal level and describes how to integrate new goals/sub-goals into the existing data structure. The variability of the process is not addressed in his work.

Gimenes et al. (2008) presents a feature based approach to support e-contract negotiation based on web-services (WS). A meta-model for WS-contract representation is given and a way is shown how to integrate the variability of these contracts into the business processes to enable a process automation. It does not address the variability of the process itself but enables the ability to reuse business processes for different e-contract negotiations.

While our used framework to model process variability reduces the overall process complexity by splitting up the process into layers with increasing details, the PROVOP project (Hallerbach et al. (2009a,b) and Reichert et al. (2014)) focuses on the concept, that variants are derived from a basic process definition through well-defined change operations (ranging from the deletion, addition, moving of model elements or the adaptation of an element attribute). In fact, the basic process expresses all possible variants at once, leading to a big process model. Their approach could be beneficial considering that cross functional requirements can be located in a single process description, but having one huge process is also contra productive (e.g. the exchange of parts of the process is difficult).

The work Gottschalk et al. (2007) presents an approach for the automated configuration of workflow models within a workflow modelling language. The term workflow model is used for the specification of a business process which enables the execution in an enterprise and workflow management system. The approach focuses on the activation or deactivation of actions and thus is comparable to the PROVOP project for the workflow model domain.

Rosa et al. (2008) extends the configurable process modelling notation developed from Gottschalk et al. (2007) with notions of roles and objects providing a way to address not only the variability of the control-flow of a workflow model but also of the related resources and responsibilities.

The Common Variability Language (CVL Haugen et al. (2013)) is a language for specifying and resolving variability independent from the domain of the application. It facilitates the specification and resolution of variability over any instance of any language defined using a MOF-based meta-model. A CVL based variability modelling and a BPM model with an appropriate model transformation could lead to similar results as presented in this paper.

The work of Zhao and Zou (2011) shows a framework for the generation of software modules based on business processes. They use clustering algorithms to analyse dependencies among data and tasks, captured in business processes. Further, they group the strongly dependent tasks and data into a software component.

6 CONCLUSION AND OUTLOOK

The reuse of business process models is an important step for an industrial company to survive in a competitive market. But only with an integrated view of the according IT landscape it is possible to raise the efficiency of the overall business. With this work we proposed a way to combine the benefits of software product line engineering techniques with the capabilities of a business process modelling tool. This work provides a framework for the systematic reuse of business processes and the configuration of software toolchains used during the actual production of the product. The new introduced framework is able to synchronize variable process structures with a variable software architecture. This means that changes to the processes will automatically generate a skeleton of the software artefacts which need to be implemented by the developers. For that, the framework uses XML data binding to bind specific software features to a specific set of configurable artefacts which need to be submitted by customers (internal and external) during the order process. This is done in an automatic and managed way so that the order interface is always aligned to the software toolchains. More-

over, the overall robustness of the software toolchains is increased since the same code base is shared for a lot of different product families leading to a higher customer satisfaction.

Future work will address the semi-automatic creation of the schema file which is used to keep the software architecture aligned to the process models. Another point for improvement is the fact that additional security requirements are implemented and mapped manually to the according product configurations. In a future work, we will investigate a way to map these security requirements to the according process model which enables an automatic way to bind these requirements to the product families and enforce them in the process. This is important especially if a certification of the products is intended.

ACKNOWLEDGEMENT

The project is funded by the Austrian Research Promotion Agency (FFG). Project Partners are NXP Semiconductor Austria GmbH and the Technical University of Graz. We want to gratefully thank Danilo Beuche from pure::systems for his support.

REFERENCES

Derguech, W. (2010). Towards a Framework for Business Process Models Reuse. *In The CAiSE Doctoral Consortium*.

Fantinato, M., Toledo, M. B. F. d., Thom, L. H., Gimenes, I. M. d. S., Rocha, R. d. S., and Garcia, D. Z. G. (2012). A survey on reuse in the business process management domain. *International Journal of Business Process Integration and Management*.

Gimenes, I., Fantinato, M., and Toledo, M. (2008). A Product Line for Business Process Management. *Software Product Line Conference, International*, pages 265–274.

Gottschalk, F., van der Aalst, W. M. P., Jansen-Vullers, M. H., and Rosa, M. L. (2007). Configurable Workflow Models. *International Journal of Cooperative Information Systems*.

Hallerbach, A., Bauer, T., and Reichert, M. (2009a). Guaranteeing Soundness of Configurable Process Variants in Provop. In *Commerce and Enterprise Computing, 2009. CEC '09. IEEE Conference on*, pages 98–105. IEEE.

Hallerbach, A., Bauer, T., and Reichert, M. (2009b). Issues in modeling process variants with Provop. In Ardagna, D., Mecella, M., and Yang, J., editors, *Business Process Management Workshops*, volume 17 of *Lecture Notes in Business Information Processing*, pages 56–67. Springer Berlin Heidelberg.

Hammer, M. and Champy, J. (1993). *Reengineering the Corporation - A Manifesto For Business Revolution*. Harper Business.

Haugen, O., Wasowski, A., and Czarnecki, K. (2013). Cvl: Common variability language. In *Proceedings of the 17th International Software Product Line Conference*, SPLC '13.

Kang, K., Cohen, S., Hess, J., Novak, W., and Peterson, A. (1990). Feature-oriented domain analysis (foda) feasibility study.

McCormack, K. P. and Johnson, W. C. (2000). *Business Process Orientation: Gaining the E-Business Competitive Advantage*. Saint Lucie Press.

Österle, H. (1995). *Business Engineering - Prozess- und Systementwicklung*. Springer-Verlag.

Pohl, K., Böckle, G., and Linden, F. J. v. d. (2005). *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer.

Reichert, M., Hallerbach, A., and Bauer, T. (2014). Lifecycle Support for Business Process Variants. In Jan vom Brocke and Michael Rosemann, editor, *Handbook on Business Process Management 1*. Springer.

Rosa, M. L., Dumas, M., ter Hofstede, A. H. M., Mendling, J., and Gottschalk, F. (2008). Beyond control-flow: Extending business process configuration to roles and objects. In Li, Q., Spaccapietra, S., and Yu, E., editors, *27th International Conference on Conceptual Modeling (ER 2008)*, pages 199–215, Barcelona, Spain. Springer.

Saidani, O. and Nurcan, S. (2007). Towards context aware business process modelling. In *8th Workshop on Business Process Modeling, Development, and Support (BPMDS07), CAiSE*, volume 7, page 1.

Sinnhofer, A. D., Pühringer, P., and Kreiner, C. (2015). varbpm - a product line for creating business process model variants. In *Proceedings of the Fifth International Symposium on Business Modeling and Software Design*, pages 184–191.

Valena, G., Alves, C., Alves, V., and Niu, N. (2013). A Systematic Mapping Study on Business Process Variability. *International Journal of Computer Science & Information Technology (IJCSIT)*.

Willaert, P., Van Den Bergh, J., Willems, J., and Deschoolmeester, D. (2007). *The Process-Oriented Organisation: A Holistic View - Developing a Framework for Business Process Orientation Maturity*. Springer.

Zhao, X. and Zou, Y. (2011). A business process-driven approach for generating software modules. *Software: Practice and Experience*, 41(10):1049–1071.

Patterns to establish a secure communication channel

ANDREAS DANIEL SINNHOFER, Graz University of Technology
FELIX JONATHAN OPPERMANN, Graz University of Technology
KLAUS POTZMADER, NXP Semiconductors
CLEMENS ORTHACKER, NXP Semiconductors
CHRISTIAN STEGER, Graz University of Technology
CHRISTIAN KREINER, Graz University of Technology

Nowadays, cyber-physical systems (CPS) are omnipresent in our daily lives and are increasingly used to process confidential data. While the variety of portable devices we use excessively at home and at work is steadily increasing, their security vulnerabilities are often not noticed by the user. Therefore, portable devices such as wearables are becoming more and more interesting for adversaries. Additionally, the increasing functionalities like internet capabilities, cameras, microphones, GPS trackers and other senor devices make them an interesting target for hacking. Furthermore, such CPS devices are often deployed in unsupervised and untrusted environments raising the question about privacy and security to a crucial topic. Thus, a robust and secure software design is required for the implementation of cryptographic communication protocols and encryption algorithms. In our opinion, Software-Patterns have proven to be an efficient way to support the development of such systems. Therefore, we will present patterns for solving the issue of Man-in-the-middle attacks. The presented patterns provide generic guidance on how to establish secure communication channels based on symmetric and / or asymmetric cryptography. Further, a selection graph is presented which helps to find the appropriate pattern in a specific context.

CCS Concepts: •**Security and privacy** → **Cryptography;** •**Software and its engineering** → **Patterns;**

Additional Key Words and Phrases: Man-In-The-Middle, Secure Channel, Secure Communication

ACM Reference Format:

Andreas Daniel Sinnhofer, Felix Jonathan Oppermann, Klaus Potzmader, Clemens Orthacker, Christian Steger, and Christian Kreiner. 2016. Patterns to establish a secure communication channel. *jn* , , Article (July 16), 21 pages.
DOI: http://dx.doi.org/10.1145/3011784.3011797

1. INTRODUCTION

Security was long treated as an orphan in our today's society since many customers have not cared much about privacy. Therefore, companies weren't interested in investing time and money into "features" a customer does not want to have and is not willing to pay. This traditional thinking of security changed when hacking systems was getting more frequent and the economic loss for companies started to get bigger [Bianca Stanescu 2012; Kevin M. McGinty 2015; Information is Beautiful 2016]. Further, the interest of the society in secure systems increased especially after Edward Snowden has published

This work is supported by the Austrian Research Promotion Agency (FFG).
Author's address: Andreas Daniel Sinnhofer, Felix Jonathan Oppermann, Christian Steger, Christian Kreiner, Institut fuer Technische Informatik, Inffeldgasse 16, 8010 Graz; Klaus Potzmader, Clemens Orthacker, NXP Semiconductors, Mikronweg 1, 8101 Gratkorn;

documents proving that national agencies hacked public systems and spies the daily life (Gemalto - Hack [The Intercept 2014]). As a consequence of hacks, private data like credit card numbers, pin codes and other user credentials are exposed to the hacker. Thus, more and more effort is invested into developing secure systems to countermeasure these trends. Especially communication systems like messaging services are noticing a steady growth in user numbers after the publication of the Gemalto Hack [Rob Price 2015].
In this work, we will present a catalogue of patterns which is designed to support students in choosing a pattern for a secure communication between parties to omit the Man-In-The-Middle (MITM) attack (see Section 2.7). Thus, basic cryptographic principles are briefly summarized in Section 2, such that the reader is aware about the used concepts. Since the variety of attacks strongly depends on the use-case, we concentrate our solutions to tackle the classic MITM attack and replay attacks an MITM could perform (see Section 2.7 and 2.8).
The paper is structured in the following way: Section 2 gives a short introduction to cryptographic principles and introduces the motivating Man-In-The-Middle problem. Section 3 presents the investigated patterns, Section 4 gives guidance on how to choose one of the presented patterns and Section 5 summarizes this work.

2. BACKGROUND

This Section summarizes the basic cryptographic concepts which are used in the presented patterns and illustrates the motivating MITM attack and replay attack.

2.1 Random number generators

Random number generators are used to generate random numbers which are required for almost every cryptographic system. For example, such generators can be used to generate new random keys to protect a specific message. The strength of such random numbers is called entropy which is a degree for *"the uncertainty of an outcome"*[1]. Such number generators can be split into the following two groups:

—True Random Number Generators (TRNG)
—Pseudo-Random Number Generators (PRNG)

PRNG are based on algorithms that use mathematical equations or a list of pre-calculated values to generate a sequence of numbers which appear to be random to the user. As a consequence, such systems can be broken by observing the output of the random number generator. Thus, the entropy of such generators are low compared to TRNG. In difference, TRNG are usually based on a specific physical phenomenon to produce a random sequence of numbers. A commonly used example is the thermal noise of a resistor or the radioactive decay of a material. As a result, the sequence of values is not guessable by observing the previous values. As such, most cryptographic system require TRNG to hold the security statements.

2.2 Symmetric cryptography

Symmetric cryptography is the term used for encryption algorithms which are using the same key for the encryption and the decryption operation. Thus, the receiver and the sender of a message need to know the secret key to be able to establish a secure communication. If the conditions for a one-time pad are fulfilled (see enumeration below), a symmetric key offers a security strength which is proportional to the key length. In contrast, public key cryptography requires longer keys to achieve the same bit security strength (cf. Section 2.3 and Table III). One of the most popular symmetric encryption

[1] https://www.vocabulary.com/dictionary/entropy

algorithms is the Advanced Encryption Standard (AES) [Contel Bradford 2014]. Today's CPUs usually have hardware acceleration units for AES encryption / decryption operations, so that such operations can be calculated fast and power efficient even on small embedded devices. Further, symmetric One-Time pads are mathematically unbreakable if the following conditions are met [Menezes et al. 1996]:

—The length of the key is as long as the message to encrypt.

—The key must be truly random (i.e. using a True Random Number Generator (TRNG)).

—The key must only be used once.

Solving the issue of finding the key is in this case as difficult as directly guessing the message which was encrypted. Consider the following example:
Alice wants to send the message "HELLO" to Bob encrypted with the one-time pad "IXELQ". The encryption operation is done by calculating the modular addition of the message and the Key (see Table I).

'H' (7)	'E' (4)	'L' (11)	'L' (11)	'O' (14)	Message
'I' (8)	'X' (23)	'E' (4)	'L' (11)	'Q' (16)	Key
(7 + 8) mod 26	(4 + 23) mod 26	(11 + 4) mod 26	(11 + 11) mod 26	(14 + 16) mod 26	Modular Addition
'P' (15)	'B' (1)	'P' (15)	'W' (22)	'E' (4)	Cryptogram

Table I. : Exemplary encryption using a one-time pad; Alphabet 'A' - 'Z' (26 letters, A = 0, B = 1, ...)

When an attacker tries to recover the message, he can choose keys with which every 5 letter long message can be decrypted. As a result, he is not able to reliably choose the right message out of all possible candidates. For example, if the attacker picks the key "TNYLA" he decrypts the cryptogram to the following message candidate:

'P' (15)	'B' (1)	'P' (15)	'W' (22)	'E' (4)	Cryptogram
'T' (19)	'N' (13)	'Y' (24)	'L' (11)	'A' (1)	Key Candidate
(15 - 19) mod 26	(1 - 13) mod 26	(15 - 24) mod 26	(22 - 11) mod 26	(4 - 1) mod 26	Modular Subtraction
'W' (22)	'O' (14)	'R' (17)	'L' (11)	'D' (3)	Message Candidate

Table II. : Exemplary decryption using a wrong key; Alphabet 'A' - 'Z' (26 letters, A = 0, B = 1, ...)

Instead of reconstructing the original message "HELLO", the attacker decodes the word "WORLD" which seems to be a valid message to the attacker if he does not have any further information about the message itself.

2.3 Public key / Asymmetric cryptography

In difference to symmetric cryptography, public key cryptography is based on two keys, where one key is used to encrypt messages (public) and another key is used to decrypt the cryptograms (private). One exception to this rule are digital signatures (see Section 2.5) where the private key is used for the encryption operation and the public key for the decryption operation. To ensure such a property, both keys are related to a specific mathematical function. For example the RSA algorithm is based on the factorization problem of big prime numbers [Jonsson and Kaliski 2003] and the ECC algorithm is based on the elliptic curve discrete logarithm problem [Mcgrew et al. 2011]. From a computational

point of view, the elliptic curve discrete logarithm problem is harder to break than the integer factorization problem. As a result, ECC keys can be shorter compared to RSA keys to achieve the same bit-security strength. Table III illustrates the key bit length recommendations published from the National Institute of Standards and Technology (NIST) (see [Elaine Barker 2016]).

Security Strength	Symmetric Encryption Algorithm	Asymmetric: RSA	Asymmetric: ECC
$\leq$ 80 Bits	2 TDEA (128 Bits)	1024 Bits	160 - 233 Bits
112 Bits	3 TDEA (192 Bits)	2048 Bits	224 - 255 Bits
128 Bits	AES-128 (128 Bits)	3072 Bits	256 - 383 Bits
192 Bits	AES-192 (192 Bits)	7680 Bits	384 - 511 Bits
256 Bits	AES-256 (256 Bits)	15360 Bits	512+ Bits

Table III. : Bit Security Strength and the according key bit length recommendations for different algorithms [Elaine Barker 2016]

2.4 Key derivation functions (KDF)

Key derivation functions (KDF) are functions which are used to systematically derive variants of a specific key using pseudo-random functions. These KDFs are cryptographic one way functions, which means that an attacker does not gain knowledge about the input of the KDF by observing the output value. One common use case are so called *Key hierarchies* where one secret master key is used to derive individual keys. An example would be a server which holds the secret master key. Each client will receive an individual derived key which is used to protect the communication between the client and the server. As a result, the Server is able to communicate with every client, but no client is able to eavesdrop the communication between the server and another client. Another common use case for KDFs is *Key Stretching* [Kelsey et al. 1998], where a given short or weak key (e.g. a user defined password) is stretched to a longer key. Further, KDFs are used in key-agreement protocols to derive session keys (e.g. Diffie-Hellman key agreement [Rescorla 1999], password based key agreements [Bellovin and Merritt 1993b], etc.).

2.5 Digital Signatures

Digital signatures are used to prove that a specific message was sent by a specific person. This is done by encrypting the hash of the message using the private key of an asymmetric key pair and appending this signature to the message itself. If the receiver of the message has the public key of the sender, he can check if the decrypted signature matches the hash of the message. If not, either the message was altered or the signature was created with a different key. This technique is for example used for signed email messages to prevent the problem of email spoofing [P Ramesh Babu D.Lalitha Bhaskari 2010].

2.6 Certificates

Certificates are used to prove that a specific public key belongs to a specific person. Usually, certificates are protected with a signature created from a certificate authority (CA) which is responsible to check the identity of the party requesting the certificate. As a result, if the receiver trusts the CA, he can be sure that messages encrypted with the public key can only be read by the specific person. Another option is the use of self-signed certificates, where the certificate is signed with the corresponding private key of the certificates public key. Thus, the issuer of the certificate acts as a CA for his own certificate. A Certificate is valid until the expiration date is reached or it was added to a revocation list. Revocation lists can be published by the according CA or can be created manually.

2.7 Man-In-The-Middle Attack

When two parties are communicating via an insecure channel (like the internet) every party which is involved in routing, can principally eavesdrop the communication of those two. Encrypting the communication channel can help to circumvent a Man-In-The-Middle attack, but encrypting alone does not guarantee a secure communication. For example, Figure 1 illustrates the Man-In-The-Middle attack. In this case one party referred to as Alice wants to talk to another party called Bob, while a third party (Eve) is trying to eavesdrop the communication. For instance Alice wants to receive a secret message from Bob and thus she sends Bob a public key so that Bob can encrypt his secret using that public key. Consequently only Alice should be able to decrypt the message. The problem is that neither Alice nor Bob can assure that they are directly talking to each other without having a Man-In-The-Middle. In this case Eve can act as such by interrupting the communication between Alice and Bob and replacing the public key of Alice with her own key. In that way she can always decrypt the received messages of Bob and send the re-encrypted message to Alice so that none of these two will ever notice that the communication was eavesdropped.

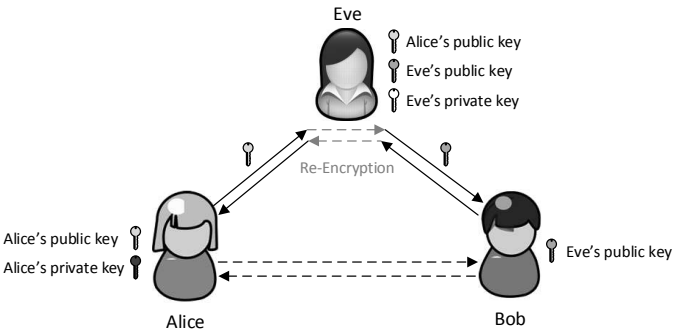


Fig. 1: The Man-In-The-Middle attack

One solution of this problem could be the use of certificates (cf. Section 2.6). In that way, the presented Man-In-The-Middle attack is no longer possible since Eve cannot replace the certificate of Alice without Bob noticing that the received certificate is either corrupt (signature is invalid) or belongs to someone else. Only if the CA is working together with Eve (the CA issues Eve a certificate with Alice's identity), or if Eve was able to break the signature creation key of the CA, eavesdropping is possible. Self-signed certificates can be considered as a solution to the problem of a compromised CA, but only if the self-signed certificate is known to Bob and is not replaced with any other certificate. This implies that the certificate was delivered to Bob in such a way, that he can be sure that the certificate was self-signed by Alice. Otherwise, Eve could have created a self-signed certificate, claiming that Eve is Alice.

2.8 Replay Attack

Another related attack is the replay-attack. In this scenario, the attacker was not able to eavesdrop the key agreement between Alice and Bob and hence was not able to read the communication in plain. At first sight, this does not seem to be a problem, but an attacker may be able to initiate specific actions by

resending previously eavesdropped encrypted messages. For example, opening a car door by sending a recorded signal of a wireless car door opener can be achieved[2]. In this example, an attacker placed a small device onto the car which was able to record signals from a wireless car door opener. Further this device was able to jam the communication channel so that the car is not able to receive commands properly. The jamming is necessary because the car remembers which codes were received and will not open the door if an already used code was received a second time. When the car owner pressed the button to open the door, the device recorded the command and jammed the channel so that the car was not able to receive the message. When the car owner pressed the opening button a second time, the device repeated the procedure, but this time it sent the old recorded command so that the car opened the door.

[2]http://www.techinsider.io/samy-kamkar-keyless-entry-car-hack-2015-8

3. PATTERNS

In this Section, we will present three patterns to protect the communication between multiple parties against Man-In-The-Middle and replay attacks. All presented patterns are extensions of the "*Secure Channels*" pattern [Schumacher et al. 2005] giving a more detailed solution in specific situations.
The first pattern uses the concept of symmetric cryptography (see Section 3.1), the second pattern uses third party based authentication (see Section 3.2) and the third pattern combines public key and symmetric key cryptography (see Section 3.3).

3.1 Pattern: Symmetric key cryptography

Context:

A device with limited computational power wants to establish a secure channel with another party via an insecure channel. A MITM can eavesdrop or change every sent message package. Further the MITM is able to resent previously eavesdropped packages (replay attack).

Problem:

No secret data shall be exposed to a third party listening or intercepting the communication. No previously recorded message package should trigger an unwanted action (e.g. a payment transaction).

Example:

A small embedded system – such as a payment card – is communicating with a host to perform a payment transaction. The device needs to answer requests from the host within a specified time interval.

Forces:

—**Computational Complexity**: While the communication shall be secure, the computational overhead of the applied protocol shall be as small as possible.

—**Resource constraints**: Storing long asymmetric keys may be a problem on devices with limited amount of memory (e.g. small Internet of Things (IoT) devices). According to Table III an AES-128 key is up to 48 times smaller than an RSA-3072 private key (stored as modulus and private exponent)

—**Eavesdropping**: Each sent communication package may be eavesdropped or replaced by a malicious third party.

Solution:

Core-Solution:
Encryption of the communication based on a shared symmetric secret between the two communication parties (abbreviated as Alice and Bob in the following). To establish the first communication, a handshake is required for authentication.

Detailed solution:
The solution to this problem is using a shared secret key. In the simple case, this shared secret is used to encrypt all messages sent between Alice and Bob. Thus, a potential eavesdropper (Eve) can only eavesdrop the communication by brute-forcing the shared secret key. However, symmetric encryption without further techniques does not solve the problem of replay attacks (as described in Section 2.8). To protect the system against such attacks, newly generated session keys should be used such that Eve cannot profit from recorded messages. Moreover, challenges sent for authentication should use random numbers and time-stamps. The principle sequence of operations is illustrated in Figure 2 – assuming that Bob wants to start a communication – and is described in the following:

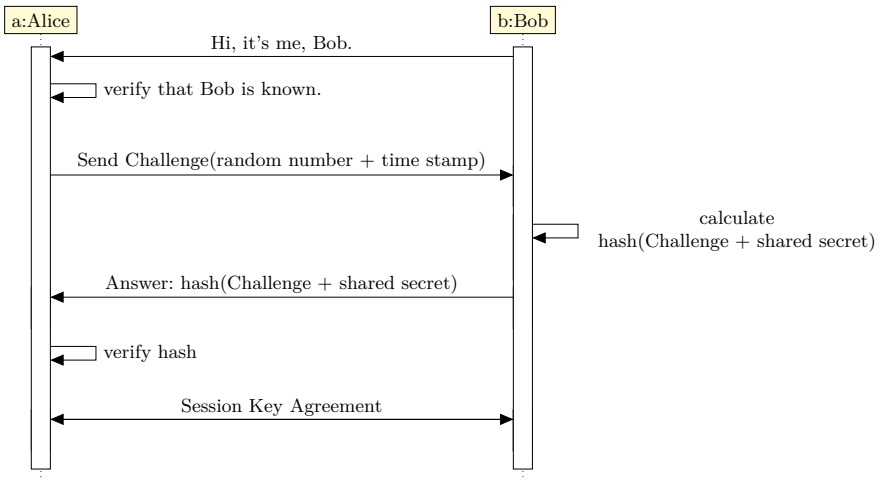


Fig. 2: Authentication based on a shared secret key. Bob is authenticated to Alice

(1) To start a communication, Bob sends an unencrypted message to Alice. This message contains information about Bob such that Alice knowns which shared secret she shall use for the further communication. If Bob is unknown to Alice, she will immediately terminate the communication.
(2) Alice answers the request with a challenge for authentication purposes. The sent challenge incorporates a random number and a time-stamp such that replay attacks can be detected.
(3) Bob answers this challenge with a calculated hash value (e.g. using an HMAC implementation). Input of the hash operation is the challenge received from Alice and the shared secret key.
(4) Alice calculates the same hash as Bob and verifies if both are matching. If not she will terminate the communication.

The presented solution covers the case that Bob is authenticated to Alice. For some applications this may not be sufficient. To ensure a mutual authentication, Bob also needs to send a challenge to Alice. In the simple case, this can be achieved by repeating the steps 2 to 4 with Bob as the sender of the challenge.

Further considerations:

The problem which remains is how to choose or how to establish the initial shared secret. This needs to be done using a different channel which can either be considered as secure, or the way the data is sent via this channel can be considered as being secure. For example, a password based secret can be used which is sent to Bob in a secure way. This could be achieved via a key shipment split on multiple key shares using trusted couriers (see [Payment Card Industry Security Standards Council 2016; National Institute of Standards and Technology (NIST) 2016; International Organization for Standardization (ISO) 2006] for recommendations). Depending on the use-case a ”face-to-face” meeting may also be an option, where the shared secret is exchanged via for example an NFC pairing of the devices.

Consequences:

+ There are a lot of open source libraries implementing symmetric cryptography algorithms (e.g. low-level C/C++ libraries such as cryptlib[3] or Crypto++[4]).
+ Computational overhead and resource consumption can be kept low.
– Establishing a shared secret without using certificate based public key cryptography is hard.
– Both parties need to agree on another channel via which the shared secret is established which reduces the overall usability of this pattern.

Known Uses:

—The Authentication and Key Agreement Protocol (AKA) for 3G systems [Choudhary and Bhandari 2015]. In that system, a shared secret key is used to mutually authenticate the Mobile Station (MS) and the Home Environment (HE; e.g. a mobile phone). Further, a key agreement is started to generate a Session Key to protect the further communication which is based on the shared secret.

—The Generic Bootstrapping Architecture (GBA) [Olkkonen 2006]. The GBA defines two ways to establish a secure communication. Only the first approach is based on symmetric cryptography which uses the same concept as the AKA for 3G system.

Related Patterns:

The presented pattern makes use of the "*Symmetric Encryption*" Pattern [Hashizume and Fernandez 2010] for encrypting all sensitive data after the session key agreement was done.

[3] http://www.cryptlib.com/
[4] https://www.cryptopp.com/

3.2 Pattern: Third party based authentication

Context:

A system wants to establish a secure channel with another party via an insecure channel. Both communication parties are not limited in terms of computational power or resources (e.g. a personal computer / smart phone communicates with a server), but may not know each other in advance. A MITM can eavesdrop or change every sent message package. Further the MITM is able to resent previously eavesdropped packages (replay attack).

Problem:

Since both communication parties are not known to each other (i.e. no shared secret knowledge, nor a known public key) a mechanism is required such that trust is gained. No secret data shall be exposed to a third party listening or intercepting the communication. No previously recorded message package should trigger an unwanted action (e.g. a payment transaction).

Example:

One example could be that a company X is developing a server hosting a web-shop. If a customer wants to buy something from the shop, he needs to send his credit card information to the server. The customer is not interested in sending his credit card information in plain and hence a secure channel has to be established. Further, the customer is not interested in starting a symmetric key exchange process via a second secure channel (i.e. a key shipment using trusted couriers), because he wants to perform the operation now and not after some days when he finally received the symmetric key.

Forces:

The following forces need to be considered:

—**User experience**: The security controls should not negatively influence the functional behaviour (e.g. real-time capabilities) of the system.

—**Computational complexity**: Public key cryptography is more time consuming compared with symmetric cryptography.

—**Eavesdropping**: Each sent communication package may be eavesdropped or replaced by a malicious third party.

Solution:

Core-Solution:
Use public key cryptography based on certificates for authenticating each communication participant (abbreviated as Alice and Bob in the following). After successful authentication, a session key is generated to secure the further communication.

Detailed solution:
The solution of this problem is to have a trusted third party who proves the authenticity of each participant. This is usually called a Certificate Authority which proves the identity of the communication parties and signs their public keys (see Section 2.6). The principle sequence of steps for certificate based secure channel establishment is illustrated in Figure 3 and described in the following:

(1) Before a communication is possible, Alice and Bob need to request a certificate issued by a trusted third party (CA). The CA proves the identity of Alice and Bob (e.g. by verifying the passport) and issues the certificate (i.e. signs the public key of Alice and Bob).

(2) To start the communication, Bob sends a message to Alice including a random number for the authentication process. A time stamp should be incorporated to harden the system against replay attacks.

(3) Alice answers the request with her certificate. Bob extracts the public key of Alice and verifies the certificate by using the public key of the CA.

(4) If the verification was successful, Bob sends his certificate to Alice such that she can verify the properness of Bob's certificate and to retrieve his public key.

(5) Until that point, no encryption is needed at all since the certificates are public anyway. In principle, these steps could have been performed by a malicious third party which has Bob's certificate. Thus, Bob sends the hash of all previous messages which is signed with his private key so that Alice can prove that Bob has the corresponding private key of the sent certificate. This is important since a malicious third party – which only owns the certificate – cannot compute the signature of the hash. As a result, Alice will terminate the communication if she cannot verify the signature.

(6) Both parties need to agree on a session key to encrypt all future messages. This can be achieved by using dedicated key derivation functions (e.g. by using a Diffie-Hellman key agreement where a symmetric session key is generated based on the used key-pairs [Rescorla 1999]) or by randomly generating a new key. This generated key has to be sent encrypted to Alice / Bob using their according public key. Usually symmetric session keys are chosen due to performance reasons.

The presented solution covers the case that Bob is authenticated to Alice. For some applications this may not be sufficient. To ensure a mutual authentication, Bob has to send a challenge to Alice. In the simple case, this can be achieved by repeating step 5 where Alice is calculating the hash and Bob is verifying the signature of the message.

Further considerations:

Each certificate has a specific date at which it gets automatically invalid. In some cases, it may happen that the owner of the certificate or the CA want to revoke the certificate earlier (i.e. the private key was broken / leaked, it was improperly issued, etc.). Therefore, so called revocation lists were introduced containing a list of invalid certificates. To ensure that such certificates are not used, each communication participant needs to consult the CA to retrieve this list. Thus, an online connection is necessary to regularly update these lists.

Consequences:

+ Through the use of certificates, Alice and Bob can assure that they are talking to each other and not to a Man-In-The-Middle.
+ There are open source libraries supporting such authentication processes (e.g. OpenSSL[5])
– The Certificates are based on public key cryptography which can be time-consuming when operated on small embedded systems.
– If the Man-In-The-Middle is cooperating with the Certificate issuer, a Man-In-The-Middle attack cannot be detected / prevented, since he is able to manipulate the certificates of Alice and Bob (as discussed in Section 2.6). Such "hacks" cannot be prevented since the CA deals as a root of trust.
– Requires an online connection or a frequent update of revocation lists to ensure that no blacklisted certificate is accepted as valid.

[5]https://www.openssl.org/

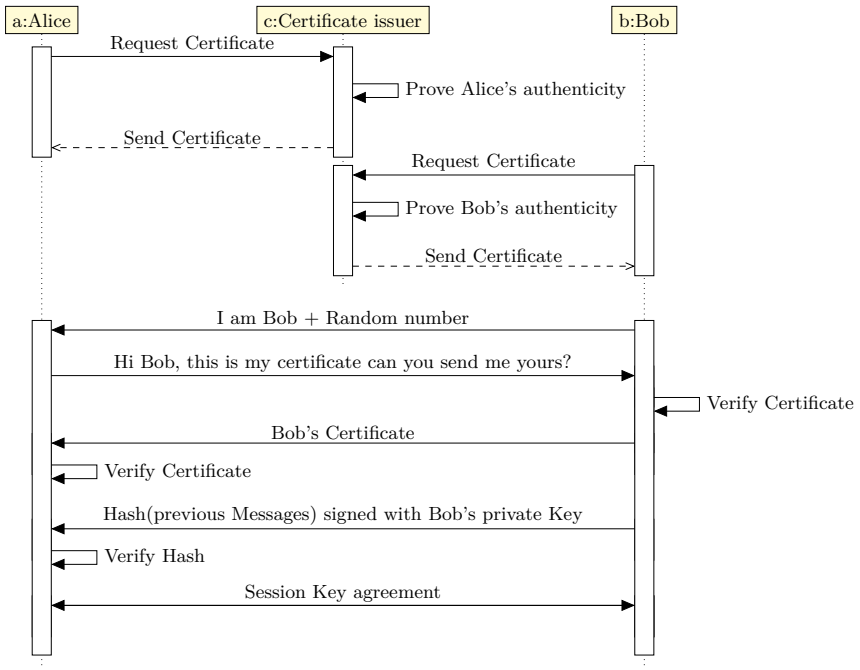


Fig. 3: Certificate based authentication

Known Uses:

—Secure Sockets Layer (SSL), Transport Layer Security (TLS) [Das and Samdaria 2014]. This system uses the process illustrated in Figure 3 using symmetric session keys.

—Secure key agreement algorithms like the extended Diffie-Hellman key exchange [Yooni and Yoo 2010]. This process implements the presented pattern with a bi-directional challenge response protocol to ensure that Alice and Bob are the owner of the corresponding private keys.

Related Patterns:

The "*Session Key Exchange with Server-side Certificate*" or the "*Session Key Exchange With Certificates*" pattern (both can be found in [Schumacher et al. 2005]) are similar implementations of this pattern, where the trusted third party is not a CA but any other common third party which knows Alice and Bob in advance.
The "*Session Key Exchange with Public Keys*" pattern (see [Schumacher et al. 2005]) presents a similar solution, but does not address the issue when Alice and Bob are both unknown to each other. Thus, an eavesdropper could create a self-signed certificate to claim the identity of Alice or Bob.
The "*Asymmetric Encryption*" [Fernandez-Buglioni 2013] pattern is used multiple times in the pre-

sented pattern: The signature operation performed by Bob and / or Alice is an asymmetric encryption operation using the private key and a specific message. During the session key agreement asymmetric encryption can be used to send the session key.
The *"Symmetric Encryption"* [Hashizume and Fernandez 2010] pattern can be used in case of symmetric session keys.

Extended Version: Web of Trust

Problem:

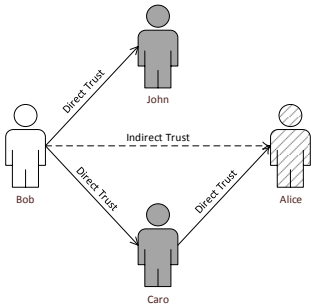
In opposite to the *Third party based authentication* pattern, Alice and or Bob do not want to trust a single third party.

Solution:

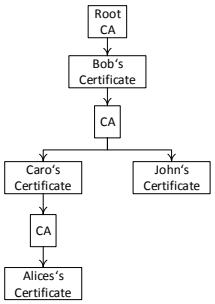
Core-Solution: Trust in a key is gained by manually verifying the fingerprint. Thus, no central CA is necessary to prove the individuals identity.

Detailed solution:

"Web of Trust" is a term which is used to describe a certificate based system in which not a single authority (CA) is used to prove that a specific public key belongs to a specific person, but the community. Each signature can be marked with a confidence level which indicates how the authenticity of the key owner was proved. The basic concept is illustrated in Figure 4a. As illustrated, Bob trusts John and Caro directly since he personally checked the identity of them (e.g. via a face to face meeting). Further, Bob indirectly trusts Alice since she is trusted by Caro. In difference to the CA structure John does not trust Alice because there is no direct path of trust leading to her. This is different in the CA case illustrated in Figure 4b. John will always trust Alice because they have the same root CA.



(a) The Web of trust structure



(b) The hierarchical CA structure

Fig. 4: Differences of the CA structure compared to the web of trust structure

Consequences compared to the CA case:

+ No single entity is used to prove the identity of a specific person / system.
– The usability is reduced since every communication party needs to prove the identity of the other parties. This is usually done during a face-to-face meeting.

Known Uses:

—Pretty Good Privacy (PGP)[6]. Trust is managed by the individual communication participants by manually verifying the fingerprints of received public keys / certificates. The user can decide if a key is fully trusted, partially trusted or untrusted. These levels can be used to define the behaviour of ”indirect trust”. This means, that a key which is signed by ”n” fully trusted parties or ”m” partially trusted parties may be accepted as trusted automatically.
—GNU Privacy Guard (GnuPG)[7]. A free and open source implementation of PGP.

Related Patterns:

The presented pattern makes use of the ”*Session Key Exchange with Public Keys*” pattern (see [Schumacher et al. 2005]) in combination with a face-to-face meeting to verify the fingerprint of the public keys. Thus, a MITM attack can be prevented.
The ”*Symmetric Encryption*” [Hashizume and Fernandez 2010] pattern can be used in case of symmetric session keys.

[6] https://www.symantec.com/products/information-protection/encryption
[7] https://www.gnupg.org/index.de.html

3.3 Pattern: Password based key exchange

Context:

A system wants to establish a secure channel with another party via an insecure channel. The connectivity of the device is limited such that it is not possible to securely load or store long private keys on the device (i.e. a user needs to manually enter it).

Problem:

A user needs to manually enter a password to start the communication. In a realistic scenario, such passwords are ”short” and designed to be easy to remember. Using only these passwords would result in an insecure system since dictionary attacks / brute force attacks could be performed by a MITM to break the password. Further the MITM is able to resent previously eavesdropped packages (replay attack).

Example:

One example could be a server which is operated by multiple users and need to be accessible from anywhere at any time. This requires that the operators may access the server from their private devices (like computers or smart phones) or via shared public available infrastructure (like public access points or internet coffee shops).

Forces:

The following forces need to be considered:

—**Limited access**: Symmetric key files, certificates or ”Web of Trust” are not suitable since it is not possible to securely load / store a private key file on the device.

—**User experience**: The security controls should not negatively influence the functional behaviour of the system.

—**Eavesdropping**: Each sent communication package may be eavesdropped or replaced by a malicious third party.

Solution:

Core-Solution:
Both communication parties (abbreviated as Alice and Bob) need to share a secret password. This password is used to protect randomly generated session key pairs which are used to further secure the communication.

Detailed solution:

Since Alice or Bob wants to transmit high confidential data using a weak (easy to remember) symmetric key they need to issue a password based key agreement process. Asymmetric cryptography has been proven in that context [Bellovin and Merritt 1993a; Bellovin and Merritt 1993b; Bellovin and Merritt 1995]. Thus, Alice and Bob need to agree on a shared secret which is incorporated into the key agreement protocol such that the used public key scheme is authenticated using this shared secret. The principle sequence of steps is displayed in Figure 5.

(1) Before starting a secure communication, Bob is generating a new random key pair (Public Key: P_B; Private Key: P_{priv_B}).

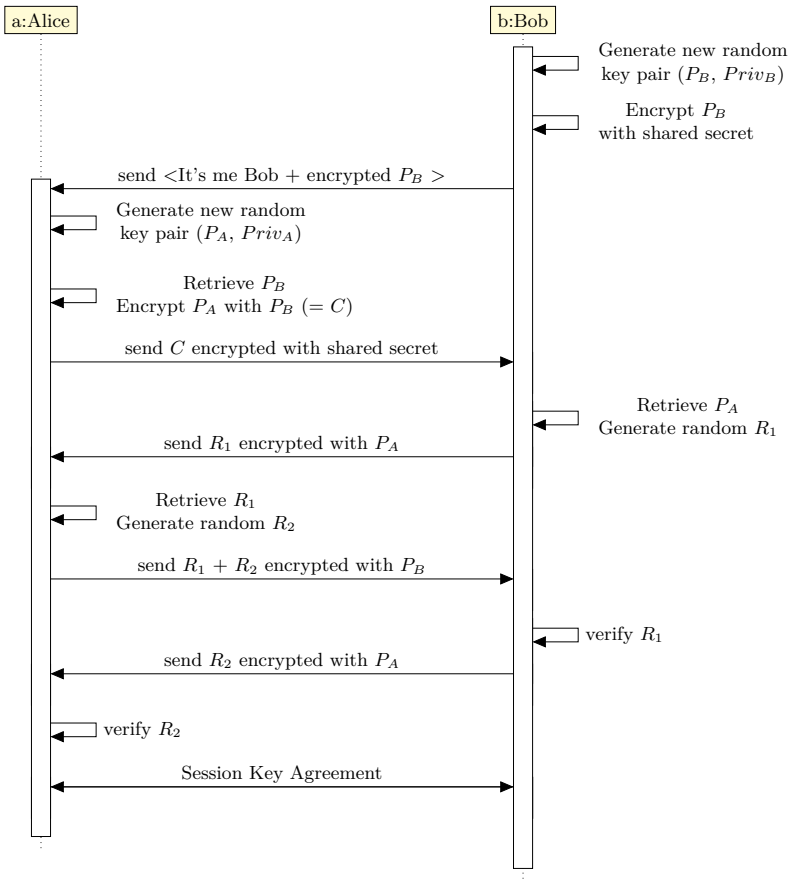


Fig. 5: Shared secret authenticated public key establishment.

(2) To start a communication, Bob sends a message to Alice containing the newly generated public key which is encrypted using the shared secret password.

(3) Alice receives the message and generates a new random key pair (Public Key: P_A; Private Key: P_{priv_A}). The fact that Alice and Bob are both generating new key pairs is essential to increase the overall strength of the protocol. By doing so, it is getting more difficult to achieve offline brute-force attacks (see [Bellovin and Merritt 1993b]).

(4) Alice retrieves Bob's public key P_B by decrypting the received message with the shared secret. Alice encrypts her generated public key P_A with the retrieved public key P_B (abbreviated as C). Further, the resulting cryptogram is encrypted with the shared secret password before sent to Bob.
(5) Bob can retrieve P_A by first decrypting the received message with the shared secret and by decrypting the result with his private key P_{priv_B}. After that, Alice and Bob are able to start a secure communication based on the generated key pairs. For large amount of data it is advisable to start a session key agreement to establish a strong symmetric key (i.e. AES-256).
(6) The next steps illustrated in Figure 5 describes a basic challenge-response protocol to verify the validity of the generated cryptographic keys. This is done in most cryptographic protocols to ensure that Alice and Bob are able to decrypt messages which were encrypted with the according public key P_A/P_B.
    (a) Bob generates a random number R_1 which he sends encrypted with the retrieved public key P_A to Alice.
    (b) Alice retrieves the generated random number R_1 by decrypting the received message. She generates a new random number R_2 which she concatenates with R_1. Alice uses the public key P_B to encrypt the concatenated random numbers and sends the cryptogram to Bob.
    (c) Bob decrypts the received message and verifies that R_1 is part of the received message. Further he retrieves the random number R_2 and sends it encrypted with P_A back to Alice.
    (d) Alice verifies the correctness of R_2
(7) Both parties need to agree on a session key to encrypt all future messages. This can be achieved by using dedicated key derivation functions (e.g. by using a Diffie-Hellman key agreement where a symmetric session key is generated based on the used key-pairs [Rescorla 1999]) or by randomly generating a new key. This generated key has to be sent encrypted to Alice / Bob using their according public key. Usually symmetric session keys are chosen due to performance reasons.

When to encrypt with the shared secret password:
The illustrated sequence used two times the shared secret password to encrypt a message (encryption of P_B and C). In most cases, one of these encryption operations can or should be omitted. As stated in [Bellovin and Merritt 1993b], the most obvious reason to skip one of the encryption operation is, that the message which shall be encrypted has to be indistinguishable from a random number. Otherwise an attacker has knowledge about the principle structure of the plain text. Thus, an attacker could perform more sophisticated attacks against the shared secret password to break it.

Further considerations:

The problem which remains is how to choose or how to establish the initial shared secret. This needs to be done using a different channel which can either be considered as secure, or the way the data is sent via this channel can be considered as being secure. For example, a password based secret can be used which is sent to Bob in a secure way. This could be achieved via a key shipment split on multiple key shares using trusted couriers (see [Payment Card Industry Security Standards Council 2016; National Institute of Standards and Technology (NIST) 2016; International Organization for Standardization (ISO) 2006] for recommendations).

Consequences:

+ Using a "weak" shared secret produces a good channel protection against a Man-In-The-Middle attack since an attacker can only hack the system by brute-forcing the randomly generated public key (which should be of reasonable length).
+ I do not need a third party claiming the authenticity of a communication partner.

– Establishing a shared secret without using certificate based public key cryptography is hard.
– Both parties need to agree on another channel via which the shared secret is established which reduces the overall usability of this pattern.

Known Uses:

Use-Cases are mainly found in the domain of password-authenticated key agreements like:

—Encrypted Key Exchange protocols (e.g. [Bellovin and Merritt 1993a; Bellovin and Merritt 1993b; Bellovin and Merritt 1995]). These are based on the process illustrated in Figure 5 but provides slightly adaptations for different public key schemes (like RSA, ECC). The reason for this is to reduce the complexity of the overall process due to security properties of the different schemes.
—Dragonfly (see [Zorn and Harkins 2015]). This protocol uses EC based cryptography in combination with a pseudo-random key derivation function to diversify the shared secret at the beginning of each session.

Related Patterns

The presented pattern makes use of the "*Session Key Exchange with Public Keys*" pattern (see [Schumacher et al. 2005]) in combination with shared secret knowledge in form of a password. The shared knowledge is required to ensure that Alice and Bob are who they claim to be by proofing their knowledge about the shared secret.
The "*Asymmetric Encryption*" [Fernandez-Buglioni 2013] pattern is used to protect the messages sent between Alice and Bob to establish a secure channel. Further, it can be used for the session key agreement.
The "*Symmetric Encryption*" [Hashizume and Fernandez 2010] pattern can be used in case of symmetric session keys.

4. PATTERN SELECTION

The presented patterns are applied in similar contexts. Thus, Figure 6 gives guidance on how to choose an appropriated pattern in the given scenario. The presented selection flow is not exhaustive in any means, but shall give a first starting point for students / developers which are new to the topic of security and secure communication.

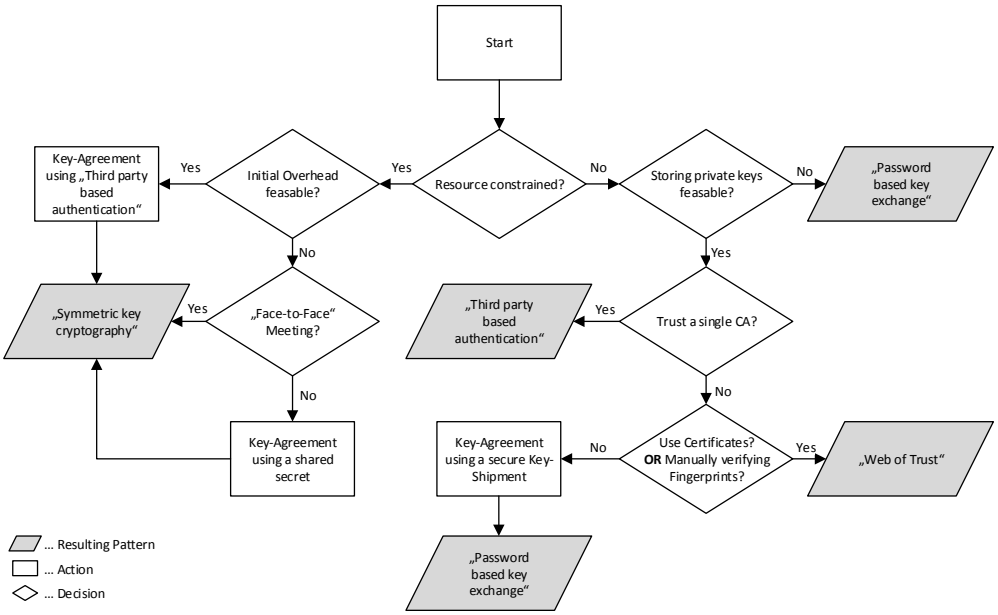


Fig. 6: Guidance flow chart to determine the pattern which can be implemented according to the context.

4.1 Example

Context

In the context of Internet of Things (IoT) devices, many small sensor nodes are collecting data from their human operators (e.g. heart rate sensor, gps sensor, etc.) and are communicating with a base station (e.g. the smart phone). To protect this communication against eavesdropping, the communication of the devices with the base station should be encrypted. Since IoT applications are evolving over time including new features and sensors, the system should be easy to configure without the need for timely setup procedures.

Applying the pattern selection:

(1) **Resource Constraint?** Yes. Although the base station may not be resource constraint (i.e. smart phone, home server, etc.), the sensor nodes are highly resource constraint (limited memory and computational power).

(2) **Initial Overhead feasible?** No. The system should be as simple as possible such that new sensor nodes can be deployed without time consuming setup procedures.

(3) **Face-to-Face Meeting?** Yes. In case of a smart phone as base station, it would be a good option to use NFC pairing to establish the initial cryptographic keys.

As a result, the pattern "*Symmetric Key Cryptography*" shall be used. If a face-to-face meeting is not possible (e.g. the base station is a web-server without direct access), a secure key shipment can be used to establish a first secure key (e.g. the vendor of the sensor nodes generates a random key which is shared with the customer via mail, etc.).

5. CONCLUSION

Having secure communication mechanisms is vital in our today's society since more and more confidential information is sent via insecure channels like the internet. Developing such systems is always a trade-off between security, usability and performance and hence making a decision is not always easy especially for developers which are not familiar with cryptographic concepts. Our work summarizes the most common patterns on how to establish a secure communication via an insecure channel to circumvent eavesdropping by third parties. They are designed for students to learn basic cryptographic principles and to increase the security awareness for such systems. The presented patterns are generic descriptions of the principle communication protocol which is necessary to establish secure channels and thus, they provide a good starting point to define what pattern should be used when designing a secure communication system and what standard protocol can be implemented to achieve secure messaging.

Acknowledgement

Project partners are NXP Semiconductor Austria GmbH and the Technical University of Graz. The project is funded by the Austrian Research Promotion Agency (FFG). We want to gratefully thank our Shepard Uwe van Heesch for his constructive feedback. Finally we want to thank all participants of the Security Pattern Workshop at EuroPLoP'16 for their feedback.

REFERENCES

S.M. Bellovin and M. Merritt. 1993a. Cryptographic protocol for secure communications. (08 1993). https://www.google.com/patents/US5241599 US Patent 5,241,599.

S.M. Bellovin and M.J. Merritt. 1995. Cryptographic protocol for remote authentication. (08 1995). https://www.google.com/patents/US5440635 US Patent 5,440,635.

Steven M. Bellovin and Michael Merritt. 1993b. Augmented Encrypted Key Exchange: A Password-based Protocol Secure Against Dictionary Attacks and Password File Compromise. In *Proceedings of the 1st ACM Conference on Computer and Communications Security (CCS '93)*. ACM, New York, NY, USA, 244–250. DOI:http://dx.doi.org/10.1145/168588.168618

Bianca Stanescu. 2012. Top 5: Corporate Losses Due to Hacking. (2012). http://www.hotforsecurity.com/blog/top-5-corporate-losses-due-to-hacking-1820.html.

Anilmit Choudhary and Randhir Bhandari. 2015. Analysis of UMTS 3G Authentication and Key Agreement Protocol AKA for LTE 4G Network. *International Journal on Recent and Innovation Trends in Computing and Communication* 3, 4 (april 2015), 2146–2149. DOI:http://dx.doi.org/10.17762/ijritcc2321-8169.150482

Contel Bradford. 2014. 5 Common Encryption Algorithms and the Unbreakables of the Future. (2014). http://www.storagecraft.com/blog/5-common-encryption-algorithms/.

Manik Lal Das and Navkar Samdaria. 2014. On the security of SSL/TLS-enabled applications. *Applied Computing and Informatics* 10, 12 (2014), 68 – 81. DOI:http://dx.doi.org/10.1016/j.aci.2014.02.001

Elaine Barker. 2016. NIST Special Publication 800-57 Part 1 Revision 4 - Recommendation for Key Management Part 1: General. (2016).

Eduardo Fernandez-Buglioni. 2013. *Security Patterns in Practice: Designing Secure Architectures Using Software Patterns* (1st ed.). Wiley Publishing, West Sussex PO19 8SQ, England.

Keiko Hashizume and Eduardo B. Fernandez. 2010. Symmetric Encryption and XML Encryption Patterns. In *Proceedings of the 16th Conference on Pattern Languages of Programs (PLoP '09)*. ACM, New York, NY, USA, Article 13, 8 pages. DOI:http://dx.doi.org/10.1145/1943226.1943243

Information is Beautiful. 2016. World's Biggest Data Breaches. (2016). http://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/.

International Organization for Standardization (ISO). 2006. ISO/IEC 11770-4: Information Technology - Security Techniques - Key Management - Part 4: Mechanisms Based On Weak Secrets. (2006).

J. Jonsson and B. Kaliski. 2003. RFC 3447 - Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1. (2003).

John Kelsey, Bruce Schneier, Chris Hall, and David Wagner. 1998. *Secure applications of low-entropy keys*. Springer Berlin Heidelberg, Berlin, Heidelberg, 121–134. DOI:http://dx.doi.org/10.1007/BFb0030415

Kevin M. McGinty. 2015. Target Data Breach Price Tag: $252 Million and Counting. (2015). https://www.privacyandsecuritymatters.com/2015/02/target-data-breach-price-tag-252-million-and-counting/.

D. Mcgrew, K. Igoe, and M. Salter. 2011. RFC 6090 - Fundamental Elliptic Curve Cryptography Algorithms. (2011).

Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. 1996. *Handbook of Applied Cryptography* (1st ed.). CRC Press, Inc., Boca Raton, FL, USA.

National Institute of Standards and Technology (NIST). 2016. Special Publication 800-57-1: Recommendation for Key Management - Part 1: General. (2016). Revision 4.

Timo Olkkonen. 2006. Generic Authentication Architecture. (2006).

CH.Satyanarayana P Ramesh Babu D.Lalitha Bhaskari. 2010. A Comprehensive Analysis of Spoofing. *International Journal of Advanced Computer Science and Applications(IJACSA)* 1, 6 (2010), 1–6. http://ijacsa.thesai.org/

Payment Card Industry Security Standards Council. 2016. Data Security Standard - Requirements and Security Assessment Procedures. (2016). Version 3.2.

E. Rescorla. 1999. Diffie-Hellman Key Agreement Method. RFC 2631 (Proposed Standard). (06 1999). http://www.ietf.org/rfc/rfc2631.txt

Rob Price. 2015. Germany's most popular paid app is a secure messenger. (2015). http://uk.businessinsider.com/threema-encryption-messaging-app-america-launch-isis-2015-6?IR=T.

Markus Schumacher, Eduardo Fernandez, Duane Hybertson, and Frank Buschmann. 2005. *Security Patterns: Integrating Security and Systems Engineering*. John Wiley & Sons, West Sussex PO19 8SQ, England.

The Intercept. 2014. The Great SIM Heist - How Spies Stole the Keys to the Encryption Castle. (2014). https://theintercept.com/2015/02/19/great-sim-heist/.

Eun-Jun Yooni and Kee-Young Yoo. 2010. A new elliptic curve diffie-hellman two-party key agreement protocol. In *Service Systems and Service Management (ICSSSM), 2010 7th International Conference on*. IEEE, Tokyo, Japan, 1–4. DOI:http://dx.doi.org/10.1109/ICSSSM.2010.5530179

Glen W. Zorn and Dan Harkins. 2015. Extensible Authentication Protocol (EAP) Authentication Using Only a Password. IETF RFC 5931. (14 Oct. 2015). DOI:http://dx.doi.org/10.17487/rfc5931

Software Configuration based on Order Processes

Andreas Daniel Sinnhofer[1], Peter Pühringer[3], Klaus Potzmader[2], Clemens Orthacker[2], Christian Steger[1], and Christian Kreiner[1]

[1] Institute of Technical Informatics, Graz University of Technology, Austria,
{a.sinnhofer,christian.kreiner,steger}@tugraz.at,
[2] NXP Semiconductors, Gratkorn, Austria,
{klaus.potzmader,clemens.orthacker}@nxp.com
[3] p.puehringer@inode.at

Abstract. Business processes have proven to be essential for organisations to be highly flexible and competitive in today's markets. However, good process management is not enough to survive in a market if the according IT landscape is not aligned to the business processes. Especially industries focused on software products are facing big problems if the according processes are not aligned to the overall software system architecture. Often, a lot of development resources are spent for features which are never addressed by any business goals, leading to unnecessary development costs. In this paper, we will present a framework for an automatic, order process driven, software configuration. For this, modern software product line engineering techniques are used to provide a systematic way to align the variability of the order processes with the software architecture.

Key words: Software Product Lines, Feature Oriented Modelling, Business Processes, Tool Configuration

1 Introduction

Business Process (BP) oriented organisations are known to perform better regarding highly flexible demands of the market and fast production cycles (e.g. [1, 2, 3]). This is achieved through the introduction of a management process, where business processes are modelled, analysed and optimised in iterative ways. Nowadays, business process management is also coupled with a workflow management, providing the ability to integrate the responsible participants into the process and to monitor the correct execution of it in each process step. To administer the rising requirements, so called business process management tools are used (BPM-Tools) which cover process modelling, optimization and execution. In combination with an Enterprise-Resource-Planning (ERP) system, the data of the real process can be integrated into the management process.

In the domain of software products, different choices in business processes lead to different software configurations. To handle variability automatically is

a challenging task because the variability of the process model needs to be reflected in the software architecture. Further, the actual customer choice during the ordering process needs to be mapped to the according software features. Due to this, software configuration is often done manually which takes a considerable amount of time during production. Particularly for resource constraint devices like embedded systems, it is vital to have a working software configuration process since unnecessary features may occupy a lot of resources. Further, it is important to have a software architecture which is synchronised with the business goals. Otherwise, a lot of resources are spent for developing and maintaining software components which are never used anyway. Especially big companies are facing problems to align the whole development team to the business goals. Thus, process awareness is crucial for an efficient development and production.

Context Aware Business Process modelling is a technique for businesses living in a complex and dynamic environment (Saidani and Nurcan [4]). In such an environment a company needs to tackle changing requirements which are dependent on the context of the system. Such context sensitive business process models are able to adapt the execution of their process instances according to the needs, such that the company can react faster and more flexible. This is achieved by analysing the context states of the environment and mapping these states to the according business processes and their related software system. The problem with such approaches is, that the used software systems are often developed independently from each other, although they share a similar software architecture. Therefore, this work focuses on the development of a framework which covers the variability of process models, and mapping such variable process structures to software configuration artefacts. This allows, that the software system can be adapted automatically with respect to its context. Software product lines have proven to be capable of achieving such flexible architectures. Additionally, the robustness of such systems can be increased, since only one system needs to be developed and maintained for whole product families. In this work, we limited the scope to reflect the variability of the order processes for software products. This includes the automatic detection of variable personalization options of different product configurations, and the automated configuration of the software toolchain responsible for generating the resulting software product. The modelling of business process variability is based on our previous work, which can be found in [5]. In particular, a SPLE Tool was used to systematically reuse expert knowledge in form of valid process variations, designed in an appropriated BPM Tool. The integrity of the process variations is secured by the capabilities of the BPM Tool and a rich cross functional constraint checking in the SPLE Tool. A more detailed description is given in Section 3.1. This work will extend the framework in order to be able to map process artefacts to software configurations. Hence, software toolchains can be configured in an automatic way and the architecture can be kept aligned with the business goals.

This work is structured in the following way: Section 2 summarizes the related work and Section 3 gives an overview over the used design paradigm for business processes modelling and Software Product Line Engineering techniques which

were needed for the framework. Section 4 summarizes the concept of our work, giving details about the conceptual design, the used type model and rules which the system applies. Section 5 describes our implementation in an industrial use case and Section 6 concludes this work and gives an overview over future work.

2 Related work

As stated in the survey of Fantinato et al. [6], major challenges in the field of business process variability modelling are related to the reaction time of process changes and of the creation and selection of the right business process variants, which are also main topics in our framework since the time to adopt the IT infrastructure to the changed business processes can be reduced with the new framework.
Derguech [7] presents a framework for the systematic reuse of process models. In contrast to this work, it captures the variability of the process model at the business goal level and describes how to integrate new goals/sub-goals into the existing data structure. The variability of the process is not addressed in his work.
Gimenes et al. [8] presents a feature based approach to support e-contract negotiation based on web-services (WS). A meta-model for WS-contract representation is given and a way is shown how to integrate the variability of these contracts into the business processes to enable a process automation. It does not address the variability of the process itself but enables the ability to reuse business processes for different e-contract negotiations.
While our used framework to model process variability reduces the overall process complexity by splitting up the process into layers with increasing details, the PROVOP project ([9, 10] and [11]) focuses on the concept, that variants are derived from a basic process definition through well-defined change operations (ranging from the deletion, addition, moving of model elements or the adaptation of an element attribute). In fact, the basic process expresses all possible variants at once, leading to a big process model. Their approach could be beneficial considering that cross functional requirements can be located in a single process description, but having one huge process is also contra productive. The exchange of parts during a process improvement process can be difficult.
The work of Gottschalk et al. [12] presents an approach for the automated configuration of workflow models within a workflow modelling language. The term workflow model is used for the specification of a business process which enables the execution in an enterprise and workflow management system. The approach focuses on the activation or deactivation of actions and thus, is comparable to the PROVOP project for the workflow model domain.
La Rosa et al. [13] extends the configurable process modelling notation developed from Gottschalk et al. [12] with notions of roles and objects providing a way to address not only the variability of the control-flow of a workflow model but also of the related resources and responsibilities.
The Common Variability Language (CVL [14]) is a language for specifying and

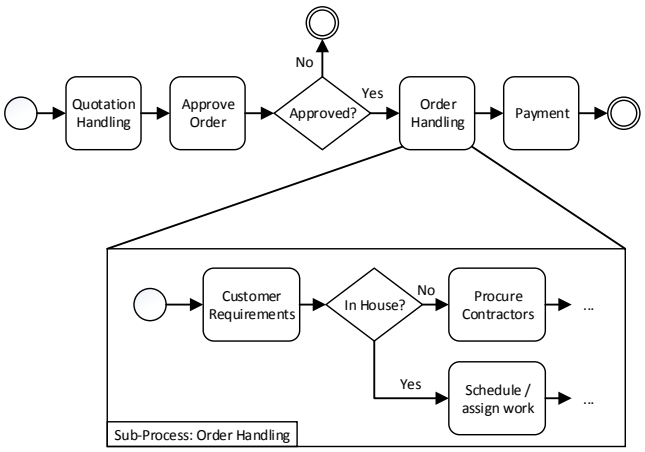


Fig. 1. Exemplary order process to illustrate the basic concepts defined by Österle [16]: A high level description of the process is split into its sub-processes until a complete work description is reached (adapted from [17]).

resolving variability independent from the domain of the application. It facilitates the specification and resolution of variability over any instance of any language defined using a Meta Object Facility (MOF) -based meta-model. A CVL based variability modelling and a BPM model with an appropriate model transformation could lead to similar results as presented in this paper.
The work of Zhao and Zou [15] shows a framework for the generation of software modules based on business processes. They use clustering algorithms to analyse dependencies among data and tasks, captured in business processes. Further, they group the strongly dependent tasks and data into a software component.

3 Background

3.1 Business Processes

A business process can be seen as a sequence of specific activities or sub-processes which need to be executed in a specific way to produce a specific output with value to the costumer [18]. According to Österle [16] the process design on a macroscopic level (high degree of abstraction) is split up into sub-processes until the microscopic level is reached. This is achieved, when all tasks are detailed enough, so that they can be used as work instructions. An exemplary order process is illustrated in Figure 1. As illustrated, the top layer is a highly abstracted description, while the production steps are further refined on the lower levels. As

a result, the lowest level is highly dependable on the concrete product and production environment, providing many details for the employees. Usually, the top layers are independent from the concrete plant and the supply chain and could be interchanged throughout production plants. Only the lower levels (the refinements) would need to be reconsidered. Variability of such a process structure can either be expressed through a variable structure of a process/sub-process (e.g. adding/removing nodes in a sequence) or by replacing the process refinement with different processes. One prominent way to design such processes is the use of the *"Business Process Model and Notation"* notation (BPMN; see [19]). The key components of such processes can be classified in the following way:

- **Events**: An Event is something that occurs during the execution of a Process. Such events affect the flow of the Process and usually have a cause and an impact. Examples are the start of an Activity, the completion of an Activity, the start of a Process, etc. According to BPMN [19], events are used only for those types, that affects the sequence or timing of Activities of a process.
- **Activities**: An Activity is work that a company or organization performs during the execution of a process. Two different types of activities can be distinguished: An activity not broken down to a finer level of Process Model is called atomic activity (i.e. a task). Sequences of tasks (e.g. a sub-process) belonging to the same activity are called non-atomic activities.
- **Gateways**: A gateway within a process controls how the process flows through different sequence flows. Each gateway can have multiple input and / or output paths. An example is a decision, where one of many possible alternative paths is selected based on a specific condition. Another prominent example is a parallel gateway which splits a single flow into multiple flows which are executed in parallel.
- **Data**: Data objects represents information flowing through the process such as documents or e-mails. A Data object can be required from a specific activity as input parameter which can be provided internally or by an external party involved in the process. Data objects which are generated by a specific activity are called Output Data.
- **Pool** and **Lanes**: Represent responsibilities for specific activities or sequences of activities in a process. The responsibilities can be assigned to an organization, a specific role, a system or even an dedicated employee.

Traditionally, processes for similar products are created using a copy and clone strategy. As a result, maintaining such similar processes is a time consuming task, since every improvement needs to be propagated manually to the respective processes. To solve this issue, we proposed a framework to automatically derive process variants from business process models (see [5]). Software Product Line Engineering techniques are used to model the variable parts of a process. The presented framework can be split into four different phases which are illustrated in Figure 2 and described below:

- **Process modelling**: In the first phase, process designers use a BPM tool of their choice and create process templates. That is, defining the sequence of

steps of sub-processes using the BPMN notation. Further, they add artefacts to the BPM Tool for required features like documentation artefacts, responsible workers or resources. As indicated in the figure, the design of (sub-) processes and the creation of the according domain model is done hand in hand.

– **Domain modelling**: In the second phase, the created processes are imported into the SPLE tool and added to a feature model (see Section 3.2 for a description of feature models). During this process, the domain engineers (Process Designer) chooses the set of available (sub-) processes and defines which parts of these (sub-) processes are variable. Consider the following example, a company responsible for forming metal uses different production planning strategies for different customers. E.g. for customer X the company is using event driven Kanban and for customer Y the company is using Kanban with a quantity signal. As a consequence, the principal sequence of production steps is the same, but the production planning is scheduled differently based on the used Kanban system. Thus, the domain engineer chooses the production plaining strategy to be variable by defining a Variation Point (VP). He deposits the different Kanban implementations as possible variants for this VP such that two process models can be generated. To limit the possible configuration space, the domain engineer can define a comprehensive set of rules and restrictions such that only meaningful and valid process variants can be derived.
– **Feature selection**: In this phase, production experts are using the defined set of process models (within the SPLE tool) to derive process variants for their current needs. This is done by selecting the wished features from the feature model, which was created in the second phase. For example, the production experts could choose event driven Kanban, since they know that customer X has placed an order.
– **Maintenance and Evolution**: In this phase, derived processes are used in the production and observed by production experts. Based on the collected data, the Production Experts can either improve the feature selection of the used process (iteration back to step 3), or issue a process improvement process (iteration back to step 4). For example, during the production for customer X, it was observed that event driven Kanban was to slow to react to the customer needs. As a consequence, the production experts changed the production planning strategy to quantity based Kanban to tackle these problems. Another example could be that it was observed that quantity based Kanban was to general. E.g. the production experts recognized that only one bin Kanban and three bin Kanban are valuable for the production processes. As a consequence, new processes need to be designed and integrated into the existing feature model.

Our today's business environment is focused on creating sustainable value by increasing the revenue of business drivers. The identification of these drivers or, equally important, the identification of drivers capable of destroying value is a crucial step for a process driven organization. Such drivers have a high impact on the organization in order to stay competitive or even survive on the market

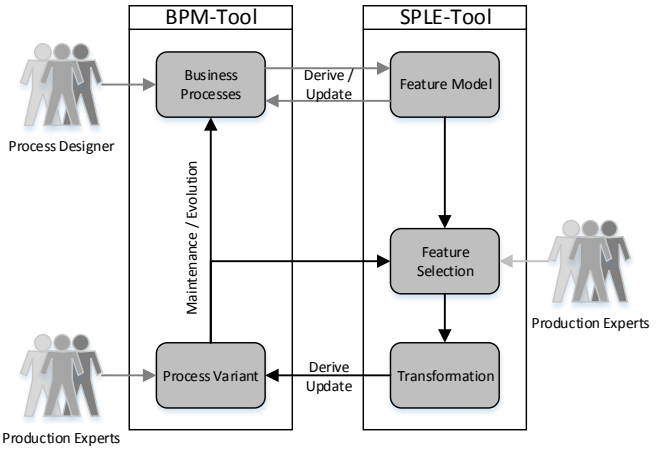


Fig. 2. Used framework for an automatic business process variant generation (adapted from [5]). The grey lines indicate process steps which need to be done manually.

[20]. The combination of SPLE engineering and business modelling is promising a potential way to improve the identification of the drivers and to react fast to changes of the market.

3.2 Software Product Line Engineering

SPLE applies the concepts of product lines to software products [21]. A Software Product Line can be seen as a set of domain features, which are automatically assembled and configured to a whole software project just by choosing the wanted features. Instead of writing code for every individual system, the developer divides the project into small lightweight features which are implemented in so called domain artefacts. For this, a software architecture is required in which the variation points and the respective variants (features) are explicitly modelled. Further, a source code generator is needed which is able to generate the according software products, based on the according feature selection. As identified by Pohl et al. [22] and Weiss et al. [23], the software product line engineering can be split up into two main parts, the *Domain Engineering* and the *Application Engineering*. The domain engineering is the procedure for defining the components, the variabilities and the commonalities of the product line. The application engineering is the procedure where the application itself is built, using some domain artefacts which were created in the domain engineering. The application engineering can be done manually, or automatically by using dedicated generators. This enables the rapid creation of similar software products of a product family.

During domain modelling, *Feature Models* are used to describe all features of a product and to explicitly state their relationships, dependencies and additional restrictions between them. Figure 3 illustrates an explanatory feature model for a car. A car consists of three mandatory variation points ('Engine Type', 'Gear Type', 'Entertainment System'), their respective variants, and an optional variation point ('Navigation System'). For example, the 'Engine Type' of the car could be Electrical, Gas or Diesel powered. The variants of the 'Engine Type' and 'Gear Type' variation point are modelled as alternative features which means that exactly one variant needs to be chosen. In contrast, the 'Entertainment System' is modelled in such a way, that either one or both options can be chosen. Further restrictions can be defined to ensure that only valid products can be generated. One example is illustrated in Figure 3; an electrical engine requires that an automatic gear type is selected. Another restriction, which is not illustrated in the figure, could be, that a specific feature cannot be selected if another feature was selected. This allows to implement complex rules to ensure that the generated systems are working as expected.
After defining the set of features, a design is created and the reusable artefacts are implemented. In case of our used process modelling approach [8], the design of the features and the implementation of reusable artefacts is done iteratively. The reason for that is, that the definition of the variable parts and their according restrictions, is the most critical aspect of designing an SPL. This ensures, that the reusable parts can be used in many different contexts.

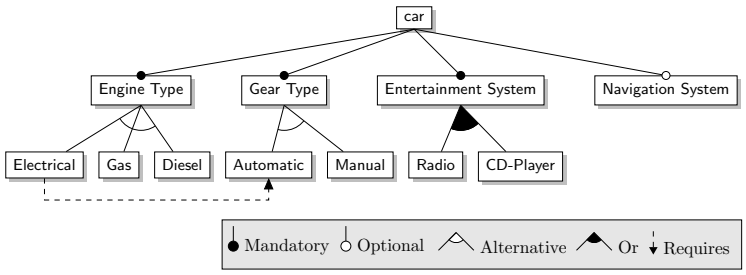


Fig. 3. An exemplary feature model of a car (adapted from [17]).

4 Variability Framework

The goal of the developed framework is to implement a systematic way to keep the customization options of the order processes aligned with the IT infrastructure such that development costs can be reduced and the company is more flexible to changes of the market. For this purpose, we are combining our previously developed product line for business process models (see [5]) with a product line

for creating software products. The following Sections summarizes the developed concepts.

4.1 Conceptual Design

The overall conceptual design is based on a feature oriented domain modelling framework and is illustrated in Figure 4. As shown in the Figure, Domain Experts are responsible for operating the "Process Variability Framework" as already described in Section 3.1. They design process models based on their domain knowledge and generate process variants for various types of product platforms. Based on this variants, the used SPLE tool also generates an order entry form, stating explicitly which kind of information a customer needs to submit, to be able to order the product. For example, if the customer can decide which applications should run on his device, or if the device can be personalized by adding signatures of the customer.

Complex products usually tend to have a lot of internal stakeholders which can be seen as internal customers. This means that based on the customer needs, specific stakeholders may be addressed to further submit needed information or even parts of the product. For instance, if a product can run on multiple hardware platforms, each of these platforms may be developed by different departments or even different companies which need to be ordered and shipped accordingly.

To be able to automatically generate the order entry forms, additional information needs to be added to the process models. This can be done by either adding this information into the process model itself (i.e. using the BPM tool) or by tagging the process model inside the SPLE tool. The additional type information designed in the SPLE tool is mapped to the according process models using unique identifiers. Using the SPLE tool to add the additional information

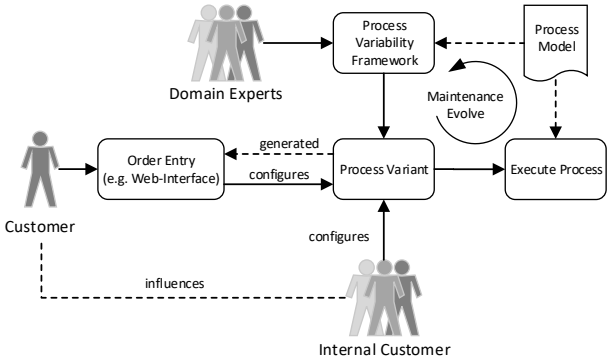


Fig. 4. Overall conceptual design of the framework (adapted from [17]). The "Process Variability Framework" block is described in Figure 2.

is the more generic approach. This also has the positive side-effect, that the processes itself are not "polluted" with information that may change in different circumstances. On the other hand, it rises high requirements to the SPLE tool which needs to support product family models such that the process model and the additional information used for the order entry can be kept aligned, but separated.

After all needed data is collected, the process can finally be executed and the ordered products are manufactured. Figure 5 shows a more detailed process of our framework to configure or generate software products. The filled order entry forms of the internal and external customer(s) are collected and converted into a format which can be automatically processed during the following Application Engineering. Here, the given submission files are imported and a feature selection is generated based on the provided input data. The feature model is directly linked to the process model and is maintained by Domain Experts operating the system. The maintenance can be done semi-automatic through the use of mapping rules which are described in Section 4.2.

The generated feature selection is verified manually by a Domain Expert against the customer requirements. The verification step is necessary to ensure the configuration safety of the final product. This is also an important topic for process certification, where evidence need to be provided to prove, that the ordered configuration is equivalent to the final product and not confused with any other customer configuration. After the verification, the Domain Experts triggers the "Feature Transformation" process. During this process, the submitted customer data is translated using product specific code generators. The result of this process strongly depends on the intended use case: It could be a binary file which is directly loaded during the production of the IC; another possibility is a set of configuration scripts which are executed to configure the product. In Section 5 we will present an industrial case study which generates a set of scripts which are executed to configure the ordered products.

Especially for new products, it is likely that during the manufacturing process knowledge is gained on how to increase the efficiency of the whole process(es) by introducing specific changes to the process model. Further, changes to the generated order entry forms may be identified, leading to more configuration possibilities. This also effects the Feature Model and the code generators used during the application engineering. The advantage of using one core of process models for a specific family of products is, that the gained knowledge can be rolled out in an automatic way for the whole product family. This means that the required changes only need to be implemented once for a whole product family of a product line.

4.2 Type model

To summarize the findings of the last Section, a type model is required which maps from BPMN nodes to some kind of input data that need to be provided from internal or external customers. Based on this information, order entry forms can be generated for the different stakeholders. As a result, the process model

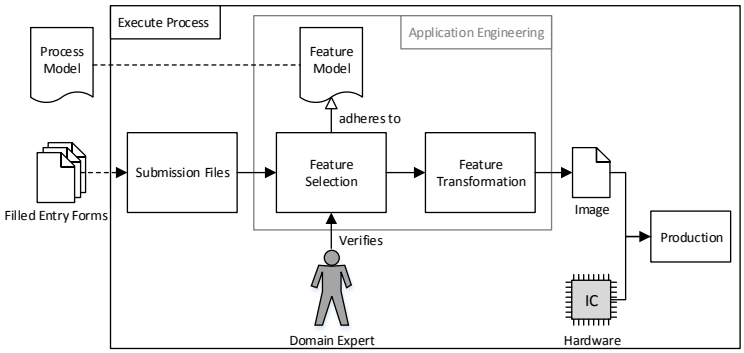


Fig. 5. The zoom-in version of the "Execute Process" task illustrated in Figure 4. The parts which are executed during the application engineering of the according SPLE workflow are framed grey.

needs to be tagged with additional information such that these order entry forms can be generated automatically for a process variant.

- **Inputs**: Is the abstract concept of different input types which are described below. Additionally, each Input is mapped to an input type defining the format of the input. For example, input data could be delivered as a file (structured or binary) or delivered in form of configuration settings like strings, integers, etc.
- **None**: No special data needs to be submitted and hence, a node marked with none will not appear as a setting in the order entry form.
- **Customer Input**: Specific data need to be added from a customer. A node marked with this type will generate an entry in the order entry form of a specific type. For example a file upload button will appear if a customer needs to submit a specific file.
- **Internal Input**: Specific data or parts of the product need to be delivered from an internal stakeholder. This information is directly propagated to the internal stakeholder as a separate order.
- **Input Group**: A set of Inputs that are logically linked together. As a consequence, all of these inputs need to be provided for a single feature.

To support the Domain Experts during the tagging of the process model, the following information can be examined automatically from the given process model:

- **Activities**: Non-atomic Activities (i.e. (sub-) processes) can be used to group a specific set of input parameter to a feature. For example a process designed for configuring an application may require several input parameter. For the order entry form, this should be displayed as one group of configuration set-

tings. Any atomic Activity will be automatically tagged as input type "None". This also applies if a non-atomic Activity does not contain any Data nodes.
- **Gateways**: Gateways are used to define the structure of the generated order entry form. An example is a decision where the customer can choose between multiple customization options. Every branch will appear as a separate group that can be selected. If a branch does not contain any data, it will only appear as a checkbox that can be ticked. The system ensures, that only one of the branches is selected from the possible choices. In contrast, flows in a parallel flow are all mandatory.
- **Data**: Data to be provided by an entity involved in the process. These nodes are tagged as Customer / Internal Input. The type information needs to be added manually by the Domain Experts. Per default, our implementations chooses "string" as a default type since – in our case – "string" is the most frequent type.
- **Pool** and **Lanes**: Gives information about the source of the input file. This means that a file in a company internal lane will be automatically flagged as an "Internal Input". Per default, our implementation chooses "Internal Input" as a default.

4.3 Domain Engineering

An established process management, which is able to generate order entry forms and trigger internal processes, is a big step towards good business management. However, to be successful on the market it is not enough to just focus on well managed processes, but also on an aligned IT infrastructure. Hence, the big remaining challenge is having an IT infrastructure which is able to be configured directly from the according business processes.

For illustration purposes let's consider the following example: A company is developing small embedded systems which are used as sensing devices for the internet of things. The device is offered in three different variants with the following features:

- **Version 1**: Senses the data in a given time interval and sends the recorder signal to a web-server which is used for post-processing.
- **Version 2**: Additionally to the features of **Version 1**, this version allows encryption of the sensed data using symmetric cryptography before it is sent to the web-server. This prevents that third parties are able to read the data. For simplicity, we assume that this key is provided in plain from the customer.
- **Version 3**: Additionally to the features of **Version 2**, this version also allows customer applications to be run (e.g. data pre-processing routines) on the system. This requires that the customer submits a binary file that shall be loaded to the system during the production.

It is economic infeasible to personalize each device manually if it is sold in high quantities. Further, establishing three different order processes using three different versions of customization toolchains will result in higher maintenance efforts. To summarize the findings of this short example, it is fundamental to

have a software architecture which is synchronized with the according order process(es). This means that variable parts of the process model need to be reflected by a variable software architecture. Further, minor changes to the process model (e.g. addition of new configuration settings) should not lead to huge development efforts since – ideally – the software architecture does not need to be changed. To be able to define such a stable software architecture it is necessary to understand and identify the variable parts of the processes to be able to define the basic set of features that need to be implemented during the Domain Engineering.

To identify the required features, we will take a second look at the previous example and identify how the order entry is generated: Basically, there are three different customization options, where in the first case a customer can customize a connection string for his web-server. In the second case, he can further submit a key which is stored onto the nodes, and in the third case, executables can be submitted to be loaded to the chip. A very basic process showing these customization options is illustrated in Figure 6. Applying the rules defined in Section 4.2, the default order entry is generated with the following properties: A string input box for defining the IP address of the web server and a decision whether to send the data encrypted or not. In case of encryption, a string input box can be used to specify the plain encryption key and an option will appear that allows a customer to load custom application also as a string input box. In this case, the only problem of the default behaviour is the default string type of the Input "Binary". As a consequence, the Domain Experts need to adapt the default process model by defining the correct input type for the "Binary" Input to get the final process model. If doing so, the system will automatically be able to generate a web form having a file upload button for the custom application. Additionally, the Domain Experts could refine the type of the other inputs to be able to perform automated constraint checks (i.e. verifying that the specified IP-Address is valid, that the given key is valid with respect to its length, etc.). The additional information is directly linked to the processes via the "Process Variability Framework" [5]. This mapping is based on unique identifiers which are unique for each node in every process designed in the BPM tool. Thus, re-factoring of the process sequence does not cause the loss of additional data. Only if the according nodes are deleted, the information is lost. Having this structured view shows that the features can be directly identified

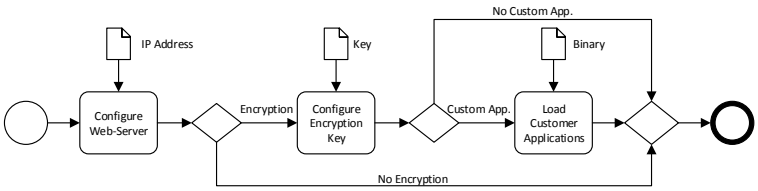


Fig. 6. Simple example process showing the customization process of the described example (Version 1, 2 and 3).

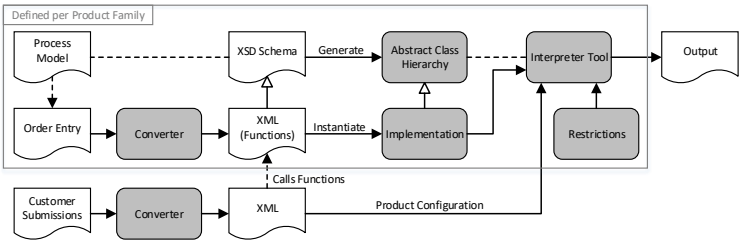


Fig. 7. The architecture of the software tool responsible for generating the wished product outcome (adapted from [17]).

that need to be supported by the following software toolchain to generate the different product variants. Thus, the Domain Experts are able to create / derive the according Feature Model consisting of the three features 'Web-Server Configuration' (mandatory), 'Encryption' (optional) and 'Custom Application' (optional; requires 'Encryption').

4.4 Application Engineering

Based on the knowledge gained from the Domain Engineering, the goal of the Application Engineering is to automate the process of creating product variants based on the submitted order entry forms. This means, that individual features of the defined Feature Model are automatically selected and assembled to the final product, based on the submitted order entry forms. As a consequence, the submitted order entry forms need to be in a format which is on the one hand easy to read for humans and on the other hand easy to process for the software toolchain. The requirement for a human readable configuration is directly linked to the verification step shown in Figure 5. If the configuration is not readable, a human won't be able to verify it against the customer requirements. Summing these requirements up, the architecture illustrated in Figure 7 can be derived. As illustrated, the Extensible Markup Language (XML) is used as the human readable and machine readable file format. But of course, any other markup language can be used as well.

The tool is basically an interpreter which can be "dynamically programmed" for the actual order. This means that variability of the architecture is gained by shifting features from the implementation phase to the configuration phase. To ensure that such freedom is not misused, it is necessary to enforce specific rules in the Interpreter Tool (e.g. security requirements). Based on the Process Model of the Process Variability Framework, a schema file is created which states all possible operations and all additional language primitives (like conditional statements, etc.) the Interpreter Tool can perform. This can be seen as a Domain Specific Language (DSL) which can be used to program the Interpreter Tool. Based on this schema, a XML file is created that reflects the Feature Model

which was created from the Domain Experts during the Domain Engineering. Each feature corresponds to a dedicated function which can be called from the according Product Configuration. Each function explicitly states which input parameter need to be provided such that the Interpreter Tool is able to generate the required output. Further, each function contains the sequence of operations that need to be executed by the Interpreter Tool.

For demonstration purposes, we will revisit the example from Section 4.3. There, three different Features where identified namely the 'Web-Server Configuration' (mandatory), the 'Encryption' (optional), and the 'Custom Application' (optional; may occur multiple times) feature. Taking this into account and the concept mentioned above, the XML illustrated in Listing 1 can be generated. For each feature, one dedicated functions is created, consisting of a *Configuration* block and a *Translate* block. The Configuration block is used to indicate which data needs to be provided from the order entries (i.e. from the "real" product configuration) and how often they can occur (configuration safety). These Configuration blocks are further used to generate a schema file which is used by the converter tool to convert the Customer Submissions into the needed XML structure. The Translate block defines how the submitted data is processed by combining the operations defined in the DSL of the Interpreter Tool. The definition of the translate blocks need to be done only once for a whole product family and only need to be adapted if the corresponding Process Model was changed.

Listing 1. Generated XML based on the Order Entry. The Translate blocks need to be edited manually by a developer calling operations defined in the XSD schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<Functions>
  <Function id="Web-Server Configuration"
            minOccurs="1"
            maxOccurs="1">
    <Configuration>
      <Parameter name="IP Address" type="ipAddress" />
    </Configuration>
    <Translate> ... </Translate>
  </Function>

  <Function id="Encryption"
            minOccurs="0"
            maxOccurs="1">
    <Configuration>
      <Parameter name="Key" type="hexstring" />
    </Configuration>
    <Translate> ... </Translate>
  </Function>

  <Function id="Custom Application"
            minOccurs="0"
            maxOccurs="unbounded">
    <Configuration>
      <Parameter name="Binary" type="fileUri" />
    <Configuration>
    <Translate> ... </Translate>
  </Function>
</Functions>
```

Listing 2. Exemplary generated Configuration file based on customer submissions. Two different versions are shown. The first example illustrates a "Version 1" product and the second one a "Version 3" product.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomerOrder>
 <WebServer>
   <Connection>X.X.X.X</Connection>
 </WebServer>
</CustomerOrder>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomerOrder>
 <WebServer>
   <Connection>X.X.X.X</Connection>
 </WebServer>

 <EncryptionKey>
   <Key>0x01020304...<Key>
 </EncryptionKey>

 <Custom Application>
   <Binary>file://orderXYZ/app1.elf</Binary>
 </Custom Application>

 <Custom Application>
   <Binary>file://orderXYZ/app2.elf</Binary>
 </Custom Application>
</CustomerOrder>
```

For this particular example, one possible solution could be that the interpreter tool offers a store-data and an install-application operation. The store-data operation could be used to write the IP address of the web-server to the correct memory location, as well as the encryption key and the according setting to enable the encryption.

Additional restrictions are domain depended and could contain in that example the following checks: Verification that the submitted key is of reasonable strength (e.g. AES key with a minimum length of 16 Byte); that the submitted applications are protected by a signature of the customer, to ensure that they are not replaced by a malicious third party; Verification that the 'Custom Application' feature is only used if encryption is activated.

To trigger the generation process, the filled order entry forms (i.e. customer submissions) are converted into a configuration file which calls the specific functions of the Functions XML (see Figure 7). For example Listing 2 shows the generated Configuration file for a "Version 1" product (the configuration on the top) and a "Version 3" product (the configuration on the bottom). Based on this configuration, the Interpreter Tool is able to generate the ordered product in an automatic way. The result of this process strongly depends on the intended use case; examples could be a binary file which can be directly flashed to the sensor nodes during productions, or script files which are executed for every individual node to configure the nodes.

5 Industrial Case Study

In this section an overview over our industrial case study is given. The implemented business processes of our industrial partner are controlled by an SAP infrastructure and are designed with the BPM-Tool Aeneis. Further, pure::variants is used as SPLE tool to manage the variability of the business processes. Thus, our implemented prototype is based on pure::variants and Java. The investigated products are smart objects designed for the Internet of Things (IoT). One possible application is the use in smart home environments to automatically control the temperature of the rooms.

5.1 SPLE-Tool: pure::variants

pure::variants is a feature oriented domain modelling tool which is based on Eclipse. As such, it can easily be extended based on Java plug-in development. It supports family models which was one of the fundamental tool requirements identified in Section 4.1. During the implementation of this project, five different plug-ins where developed:

- An extension to the import plug-in which was developed in our previous work. It assists the Process Designers in modelling cross functional requirements and providing the needed information for the code generators.
- An extension to the internal model compare engine for comparing different versions of created feature models with each other.
- An extension to the internal model transformation engine to convert the feature selection of the process model into the according order entry form. This also generates the back-end to trigger processes for internal stakeholders.
- Additions to the internal model check engine to model and create only valid processes (e.g. checks related to the feature selection, the consistency of the feature model, etc.)
- Generator Tools which are able to generate the skeleton of the schema file (as described in Section 4.3) and the order entry form (a generated Web-Interface). Additionally, converter tools were written which are converting the generated forms and received submissions into the related XML configuration files.

Additionally, the interpreter tool was written within the same development environment to ensure that the feature models are directly reflected in the implementation of the interpreter.

5.2 Implementation of the interpreter tool

As mentioned in Section 4.4, XML / XSD are used to model the operations of the interpreter tool and to automatically instantiate the requested features. To ease the implementation, the according class hierarchy was generated from

Table 1. Effort measurements and estimations in man-month to develop the system using traditional software development and our presented framework.

	Framework	Traditional
Base System	12	-
Product Fam. 1	1	6
Product Fam. 2	0.5	Estimate: 4 - 5
Product Fam. 3	0.05	Estimate: 1 - 1.5
Overall	14,55	11 - 12.5

the schema file using the tool Jaxb (a Java architecture for XML binding[1]). The generated bare class hierarchy was then implemented by the software developers. Since the creation of the schema file is semi-automatic, our developed framework (implemented in pure::variants) opens a dialogue which hints the domain expert to check the validity of the schema file to ensure that the changes to the processes are always propagated to the schema file. The domain of our industrial case study was the configuration of small embedded systems. The production was done in two major steps. During the first step, the initial device was produced having the default software flashed to the devices. During the second step, the order individual configuration was loaded to the nodes by applying scripts. This required that the nodes where mounted to a programming device which communicated with the nodes using their standard API.

5.3 Evaluation

The framework was successfully deployed for three different product families which are based on the same Process Model. The third supported product was a revision of the second, where new configuration settings where supported and the underlying hardware platform was changed (endianness changed and additional sensors where added). The time was measured to implement the initial system and the overhead to support the three systems to get an effort estimation which can be compared with a traditional software development. We use the term "traditional software development" for a software development with ad-hoc (or near to ad-hoc) software architecture which means that multiple different systems are designed almost independently. This leads to the situation that only a little code base is shared between each software project since most of the code is optimized for single purposes. However, this code would be reusable if adaptations of the interfaces / implementations would have been considered.

The effort for the traditional software development was based on the real implementation time for the first system and an effort estimation to port the existing family to the new products. These numbers were given by the responsible developers. As illustrated in Table 1, the "traditional" development effort for the third system was about 20 - 30 times higher than the implementation effort of the new developed framework. This number may seem too high at first sight,

[1] http://www.oracle.com/technetwork/articles/javase/index-140168.html

but can be directly mapped to the change of the endianness of the underlying hardware platform. With our proposed framework, it was a minor change which required a byte reversal of integer values of the generated outcome. This was done by introducing a post-processing step in every "Translate" block which resulted in integer numbers using already existing functionality. For the traditional development, the responsible developers argued that they would need to search the whole source base to identify the impacts of the endianness change and to fix the according implementations.

As illustrated in Table 1, the break-even point will be between 3 to 4 systems using a curve fitting interpolation. This number also correlates to the typical number presented in relevant software product line publications (e.g. [22]). The break-even point may also be reduced since these numbers do not consider the maintenance of the systems. The maintenance cost can be reduced since fixing problems in one product family will fix this issue in all others as well.

6 Conclusion and Outlook

The reuse of business process models is an important step for an industrial company to survive in a competitive market. But only with an integrated view of the according IT landscape, it is possible to raise the efficiency of the overall business. With this work we proposed a way to combine the benefits of software product line engineering techniques with the capabilities of a business process modelling tool. This work provides a framework for the systematic reuse of business processes and the configuration of software toolchains used during the actual production of the product. The new introduced framework is able to synchronize variable order process structures with a variable software architecture. This means that changes to the processes will automatically lead to software artefacts which need to be implemented by the developers. For that, the framework uses XML data binding to bind specific software features to a specific set of configurable artefacts which need to be submitted by customers (internal and external) during the order process. This is done in an automatic and managed way such that the order interface is always aligned to the software toolchains. This essentially reduces the development costs and time required to react to changes of the market. Moreover, the overall robustness of the software toolchain is increased since the same code base is shared for a lot of different product families leading to a higher customer satisfaction.

In the current state, the presented framework is focused on covering the variability of the order processes. As a consequence, the developed framework is only applicable in similar contexts, where the variability of the process models is mainly driven by the order processes. In a future work, we will investigate the consequences of different contexts and how these changes influences the process models and the according software toolchains.

Further, Future work will address the semi-automatic creation of the schema file which is used to keep the software architecture aligned to the process models. Another point for improvement is the fact that additional restrictions like

security requirements are implemented and mapped manually to the according product configurations. In a future work, we will investigate a way to map these security requirements to the according process model which enables an automatic way to bind these requirements to the product families. Thus, additional non-functional requirements can be automatically enforced during the process. This is important especially if a certification of the generated products is intended. Certification requires, that evidence is provided which ensures that the functional and non-functional requirements are met no matter which valid configuration is used. This requires mature development processes and verification gates to allow an automatic detection of violations.

Acknowledgements. The project is funded by the Austrian Research Promotion Agency (FFG). Project Partners are NXP Semiconductor Austria GmbH and the Technical University of Graz. We want to gratefully thank Danilo Beuche from pure::systems for his support. Further, the authors want to gratefully thank Felix Jonathan Oppermann for his support during the design and the implementation of the industrial prototype.

References

1. McCormack, K.P., Johnson, W.C.: Business Process Orientation: Gaining the E-Business Competitive Advantage. Saint Lucie Press (2000)
2. Valena, G., Alves, C., Alves, V., Niu, N.: A Systematic Mapping Study on Business Process Variability. International Journal of Computer Science & Information Technology (IJCSIT) (2013)
3. Willaert, P., Van Den Bergh, J., Willems, J., Deschoolmeester, D.: The Process-Oriented Organisation: A Holistic View - Developing a Framework for Business Process Orientation Maturity. Springer (2007)
4. Saidani, O., Nurcan, S.: Towards context aware business process modelling. In: 8th Workshop on Business Process Modeling, Development, and Support (BPMDS07), CAiSE. Volume 7. (2007) 1
5. Sinnhofer, A.D., Pühringer, P., Kreiner, C.: varbpm - a product line for creating business process model variants. In: Proceedings of the Fifth International Symposium on Business Modeling and Software Design. (2015) 184–191
6. Fantinato, M., Toledo, M.B.F.d., Thom, L.H., Gimenes, I.M.d.S., Rocha, R.d.S., Garcia, D.Z.G.: A survey on reuse in the business process management domain. International Journal of Business Process Integration and Management (2012)
7. Derguech, W.: Towards a Framework for Business Process Models Reuse. In The CAiSE Doctoral Consortium (2010)
8. Gimenes, I., Fantinato, M., Toledo, M.: A Product Line for Business Process Management. Software Product Line Conference, International (2008) 265–274
9. Hallerbach, A., Bauer, T., Reichert, M.: Guaranteeing Soundness of Configurable Process Variants in Provop. In: Commerce and Enterprise Computing, 2009. CEC '09. IEEE Conference on, IEEE (2009) 98–105
10. Hallerbach, A., Bauer, T., Reichert, M.: Issues in modeling process variants with Provop. In Ardagna, D., Mecella, M., Yang, J., eds.: Business Process Management Workshops. Volume 17 of Lecture Notes in Business Information Processing. Springer Berlin Heidelberg (2009) 56–67

11. Reichert, M., Hallerbach, A., Bauer, T.: Lifecycle Support for Business Process Variants. In Jan vom Brocke and Michael Rosemann, ed.: Handbook on Business Process Management 1. Springer (2014)
12. Gottschalk, F., van der Aalst, W.M.P., Jansen-Vullers, M.H., Rosa, M.L.: Configurable Workflow Models. International Journal of Cooperative Information Systems (2007)
13. Rosa, M.L., Dumas, M., ter Hofstede, A.H.M., Mendling, J., Gottschalk, F.: Beyond control-flow: Extending business process configuration to roles and objects. In Li, Q., Spaccapietra, S., Yu, E., eds.: 27th International Conference on Conceptual Modeling (ER 2008), Barcelona, Spain, Springer (2008) 199–215
14. Haugen, O., Wasowski, A., Czarnecki, K.: Cvl: Common variability language. In: Proceedings of the 17th International Software Product Line Conference. SPLC '13 (2013)
15. Zhao, X., Zou, Y.: A business process-driven approach for generating software modules. Software: Practice and Experience **41**(10) (2011) 1049–1071
16. Österle, H.: Business Engineering - Prozess- und Systementwicklung. Springer-Verlag (1995)
17. Sinnhofer, A.D., Pühringer, P., Potzmader, K., Orthacker, C., Steger, C., Kreiner, C.: A framework for process driven software configuration. In: Proceedings of the Sixth International Symposium on Business Modeling and Software Design. (2016) 196–203
18. Hammer, M., Champy, J.: Reengineering the Corporation - A Manifesto For Business Revolution. Harper Business (1993)
19. (OMG), O.M.G.: Business process model and notation (bpmn). version 2.0. (2011) 1–538 available at http://www.omg.org/spec/BPMN/2.0/.
20. Strnadl, C.F.: Aligning business and it: The process-driven architecture model. Information Systems Management **23**(4) (2006) 67–77
21. Kang, K., Cohen, S., Hess, J., Novak, W., Peterson, A.: Feature-oriented domain analysis (foda) feasibility study (1990)
22. Pohl, K., Böckle, G., Linden, F.J.v.d.: Software Product Line Engineering: Foundations, Principles and Techniques. Springer (2005)
23. Weiss, D.M., Lai, C.T.R.: Software product-line engineering: a family-based software development process. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1999)

Combined Variability Management of Business Processes and Software Architectures

Andreas Daniel Sinnhofer[1], Andrea Höller[1], Peter Pühringer, Klaus Potzmader[2], Clemens Orthacker[2], Christian Steger[1] and Christian Kreiner[1]

[1]*Institute of Technical Informatics, Graz University of Technology, Austria*

[2]*NXP Semiconductors, Gratkorn, Austria*

{*a.sinnhofer,andrea.hoeller,christian.kreiner,steger*}*@tugraz.at, p.puehringer@inode.at,* {*klaus.potzmader, clemens.orthacker*}*@nxp.com*

Keywords: Software Product Lines, Feature Oriented Modelling, Business Process Variability Management, Software Configuration

Abstract: Nowadays, organizations are faced with the challenge of surviving in a highly flexible and competitive market. Especially the domain of Internet of Things is affected by short product cycles and high pricing pressure. Business Process oriented organizations have proven to perform better regarding highly flexible markets and fast production cycles. However, especially industries focused on low cost IoT systems are facing big problems if the according business processes are not aligned with the business processes. Consequently, a lot of development effort is spent for features which are never addressed by any business goal. With this work, we propose to use a combined variability management in order to efficiently address product variability on the organizational level as well as on the technical level.

1 INTRODUCTION

We are living in an ever changing and interconnected world. The dawn of the Internet of Things (IoT) further increases the trend for organizations to delivery feature rich systems in high quantities and at low costs. Due to this pricing pressure, methods have to be investigated which allows modular and highly configurable systems such that the products can be adapted to the current requirements of the market.

Business Process (BP) oriented organizations are known to perform better regarding highly flexible demands of the market and fast production cycles (McCormack and Johnson (2000); Hammer and Champy (1993); Valena et al. (2013); Willaert et al. (2007)). This is achieved by introducing a management process during which business processes are modeled, analyzed and optimized in iterative improvement processes. During recent years, business process management is further coupled with a workflow management in order to monitor the correct execution of the process and to integrate responsibilities to the process models. In order to react to changing requirements, context aware business process modeling techniques were introduced Saidani and Nurcan (2007): Flexibility is gained through the analysis of the context states of the environment which are mapped to the according business processes and their related software systems. The problem with such approaches is that the used software systems are often developed independently from each other, although they share a similar software architecture.

Software Product Lines (SPL) have proven to be essential for the development of flexible product architectures which can be adapted to the current requirements (Pohl et al. (2005)). Through the use of a common architecture and reusable product features, SPL promises to deliver high quality products while simultaneously maintaining low development costs. The most critical phase during the design and the implementation of a product line is the identification of the variable parts and the common parts of the product family (Pohl et al. (2005)). Consequently, a lot of effort is invested to identify the domain requirements of the final product portfolio. Equally important is the selection of the according features during the application engineering: It has to be guaranteed, that the customer requirements are fully met; further, all unnecessary features need to be excluded in order to ensure low productions costs of the final product. Since the identification of the domain requirements is usually carried out from developers, an integrated view

of the organizational goals is often missing. Thus, the efficiency of the product line is reduced since additional effort needs to be invested to configure the product according to the current requirements.

This work focuses on the development of a framework which aims to enforce a link between the variability of the business processes and the variability of the product platform. As such, we propose a combined variability modeling in which the requirements for the organization as well as for the development of the product platform are identified together. After identifying the requirements, order processes are designed which reflect the possible product configurations that can be ordered by a customer. These variable order processes are further used to automatically trigger the product customization process in order to reduce the production costs of the final product. This work is based on our previous works in which we already defined systems for the modeling variability of business process models (Sinnhofer et al. (2015)) as well as a framework for generating software configurations based on order processes (Sinnhofer et al. (2016, 2017)).

This work is structure in the following way: Section 2 summarizes basic concepts about business process modeling as well as software product line engineering. Section 3 summarizes out approach to link variable order process models to variable software architectures in an automatic way. In Section 4 we describe how we applied the introduced concepts in an industrial use case and present a simplified example for illustration purposes. Since the identification of business drivers is essential for an organization to survive in a competitive market, we show in Section 5 how we were able to identify improvement opportunities by analyzing the results of our framework. We conclude this work by presenting related work in Section 6 and a summary in Section 7.

2 BACKGROUND

This Section summarizes the basic concepts of Business Process Modeling and Software Product Line engineering which are applied in this work. Further, our previous publications – which are forming the foundation of this work – are briefly summarized.

2.1 Software Product Line Engineering

Software Product Line Engineering (SPLE) applies the concept of product lines to software products. As a consequence, SPLE promises to create diverse and high quality software products of a product family in short time and at low costs Pohl et al. (2005). Instead of writing software for every individual system, software products are automatically generated by combining the required domain artifacts. The principal concept can be split into two main phases: the Domain Engineering and the Application Engineering (Pohl et al. (2005); Weiss and Lai (1999)).

The Domain Engineering is the phase in which the variabilities and the commonalities of the according domain are identified and the reusable domain artifacts are implemented. Domain artifacts are development artifacts like the software architecture or the software components. One essential phase during the domain engineering is the requirements engineering process, in which a domain analysis has to be performed in order to identify the requirements of the final product. Based on the identified requirements, the domain is usually modeled by using a Feature Oriented Design Modeling (Kang et al. (1990)) approach. During this process, Feature Models are used to explicitly describe all features of a product, their relationships, dependencies and additional restrictions.

The Application Engineering is the phase during which the final products are created by combining the domain artifacts in a meaningful manner. This is enforced by the use of domain constraints which were modeled during the Domain Engineering phase. In difference to the Domain Engineering, the Application Engineering is mainly focused on reusing artifacts rather than the implementation of new artifacts. In the ideal case, this phase makes use of software generators to automatically derive product variants without the need of implementing any new logic. The amount of reused domain artifacts heavily depends on the application requirements and gives an estimate on the efficiency of the product line. Hence a major concern of the application engineering is the detection of deltas between the application requirements and the available capabilities of the product line.

2.2 Business Process Modeling

Business Processes (BP) are a specific sequence of activities or (sub-) processes which are executed in a certain order to create an amount of value to the customer Hammer and Champy (1993). In this work, we use the concept defined by Österle (1995) to model BPs: BPs are modeled in different layers, where the top level (macroscopic level) is a highly abstract description of the overall process and the lower-levels (microscopic level) are more detailed descriptions of the sub-processes. A reasonable level of detail is reached, if the process description on the lowest levels can be used as work-instructions for the responsible

employees. This leads to the fact that the higher levels of the process description are usually independent of the production facility and the supply chains; while the lower levels are highly dependent on the production facility and its capabilities. As a consequence, the macroscopic level is more stable with respect to changes and can be reused in different contexts and production environments. The microscopic levels need to be updated in order to reuse them in different contexts. Variability of such process structures can be modeled through a variable process structure (i.e. by adding/removing activities in a process) or by replacing process refinements with different subprocesses. In general, three main types of business processes can be distinguished (see Association of Business Process Management Professionals (2009)):

- **Primary Processes**: Each of the process activities adds a specific amount of value to the value chain. Consequently, such processes are also often referred to as Core Processes since the customer value is directly reflected in these processes.
- **Support Processes**: Are processes which are designed to support the Primary Processes like managing resources or infrastructure. Such processes do not directly add value to the customer but are essential to ensure the proper execution of the Primary Processes.
- **Management Processes**: Are designed to monitor and schedule business activities like the execution of Primary Processes or Support Processes. While Management Processes do not directly add value to the customer, they are designed to increase the efficiency of the business activities.

Domain specific modeling languages are usually used to model all the activities, resources and responsibilities within a Business Process. In the scope of this work, the Business Process Model and Notation (BPMN, Object Management Group (2011)) is used to model processes, but the general concept of this work is not limited to this notation. The key concepts which are used in this work, are summarized below (Object Management Group (2011); Sinnhofer et al. (2017)):

Events: Occurs during the execution of a process and may affect the flow of the process. For example, the start or the completion of an Activity are typical events that occur in every process. According to the BPMN specification Object Management Group (2011), events are used only for those types, which affect the sequence or timing of activities of a process.

Activities: An Activity is a specific amount of work that has to be performed by the company – or another organization – during the execution of a process. Two different types of activities can be distinguished: Atomic activities (i.e. a task) and non-atomic activities (e.g. sub-processes).

Gateways: Are used to control how the process flows through different sequences of activities. Each gateway can have multiple input and/or output paths. One example is a decision, where out of many possibilities, only one path is selected. The selection of the paths can be coupled to conditions or events which are triggered during the execution of the process.

Data: Data objects represents the information flow through the process. Two types of Data objects can be distinguished: Input Data that is required to start a specific activity and Output Data which is produced after the completion of an Activity.

Pool and **Lanes**: Are used to model responsibilities for specific activities in a process. Responsibilities can be usually assigned to an organization, to specific roles or even dedicated employees.

It is common practice for organizations to maintain multiple variants of business processes which are based on a common template (Rosa et al. (2017)). This leads to the situation that similar process variants are created through a copy and clone strategy. As a consequence, maintaining these process variants is a time consuming tasks since every single process variant has to be manually updated by the according process designer. Besides the additional maintenance effort, using copy and clone strategies also have a negative influence on the process documentation. To solve these issues, we proposed a Software Product Line approach for the derivation of process variants from business process models (see Sinnhofer et al. (2015, 2016, 2017)). The concept can be split into four different phases:

Process modeling: During the process modeling, process designers are responsible to design process templates. The process templates are designed using the BPMN notation and additional artifacts are integrated like documentation templates, responsible roles, resource allocations, etc. The process templates are designed in an appropriate BPM Tool to fully support the process designers during the design process. The process of designing the process templates and the process of creating the according domain model goes hand in hand to ensure that the created templates can be reused in many different contexts.

Domain modeling: During this process, the created templates are imported into a Software Product Line tool and translated into a so called feature model (see Section 2.1). During the creation of the feature model, it has to be decided which parts of the process are designed to be variable and which parts are static.

For illustration purposes, the following example is given: A company creates car parts for two major car manufacturers. While the overall process for creating the car parts is identical for both customers, different production planning strategies are used to optimize the material usage (e.g. stock size, etc.). As a consequence, the production planning strategy has to be designed variable such that the overall process model can be reused for both customers. The definition of variable parts and static parts happens in close cooperation with the according process designers and may even lead to a re-iteration of the first phase if some process templates need to be adapted. Not every combination of variants may create meaningful process variants. As a result, a comprehensive list of restrictions and rules has to be designed as well to guarantee that only valid and meaningful process variants can be created by the product line. The list of rules and restrictions has to be defined flexible as well, since not every restriction may be identified when the process model is created. Consequently, re-iterations of the restriction model are common after collecting evaluation data from the execution of the process.

Feature selection: Based on the current requirement of the organization, process variants are created using the created feature model. This is done be selecting the required features from the model and by translating this feature selection to a valid business process structure. To ensure an automatic transformation, generators have to be developed which are able to translate between the business process model and the feature model. The defined rules and restrictions are enforced during this process to guide the domain expert in selecting a meaningful set of options. To continue the example from above, two process variants may be created for the two customers. The only difference between the processes is the production planning strategy.

Maintenance and Evolution: One of the most important phases is the maintenance and evolution phase: To be highly flexible and adaptive to the current requirements of the market, processes and their according models have to be continuously improved and adapted. As such, the derived processes are monitored by production experts during the time in use and evaluated against the requirements. Based on the collected data, feature selections can be improved or process improvement processes can be scheduled. During a process improvement process, process templates are updated or created from the process designers and integrated into the existing feature model. Through the capabilities of the Software Product Line tool, it is possible to automatically propagate the changes of the process templates to every instance. As a consequence, no time consuming and error prone manual maintenance process is necessary to adapt all the existing process variants. Since it may happen that some of the process variants shall not be updated in case of changes, version control can be used to explicitly state which version of a template shall be used.

Our today's business environment is focused on creating sustainable value by increasing the revenue of business drivers. The identification of such business drivers, or the identification of the drivers which are able to destroy value is an essential step for an organization Strnadl (2006). Otherwise, staying competitive or even survive on a flexible market is not possible. The combination of business variability and software variability is a promising way to improve the identification of such drivers. Further, having a combined view of the requirements helps to increase the overall efficiency of the product line.

3 COMBINED VARIABILITY MODELING

The goal of this work is an automatic generation of software products based on the product order. In order to achieve this goal, an integrated view is necessary in which the variability of the software product is reflected in a variable order process. Since only a few configuration options are usually exposed to the end-customer, also all internal processes need to be covered in the variability management process. The overall concept of the resulting process is highlighted in Figure 1. As illustrated, the Process Variability Framework – which was described in Section 2.2 – is used as a foundation to model variable order process models. Based on this order process models, order entry forms are automatically generated which need to be filled by internal or external customers. Based on the provided data, a product line is triggered which maps the provided order data to the customization options of the software product. Based on the generated feature mapping, the final product can be automatically derived without any manual step beside verification steps which may be required by certification requirements. In order to achieve a binding between the order process models and the generated order entry forms, the following type model was introduced (Sinnhofer et al. (2017)):

- **None**: No special data needs to be submitted. Thus, a process node marked with none will not appear as a setting in the order entry form.
- **Inputs**: Is the abstract concept for different input types which are described below. Each In-

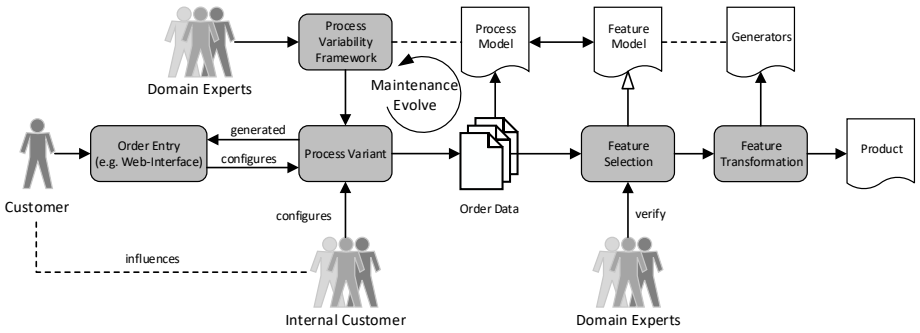


Figure 1: Overview of the concept for combining order process variability and software variability to automatically derive software products.

put is mapped to a specific input type, defining the format of the input. For example, input data could be delivered in form of a file, or configuration settings could be delivered in form of strings or integer values. Depending on the applied domain, also non functional properties may need to be modeled in the Input type. For example, if a security critical product is developed, a customer may be asked to provide a cryptographic key which is used to authenticate the customer to the device. Besides providing this key, also some kind of specification is required in which format this data was provided (e.g. pgp encrypted, etc).

- **Customer Input**: Specific data that has to be added from an external customer. A process activity marked with this type will generate an entry in the order entry form of a specific type. For example, drop down list will appear if a customer can select between different options.
- **Internal Input**: Specific data needs to be added from an internal stakeholder. A process activity marked with this type will not generate an entry in the external customer order interface, but will create a separate order entry for the according internal stakeholder.
- **Input Group**: A set of inputs which are logically linked together. As a consequence, all of these inputs will be highlighted as a group in the generated order entry and all of them are required for a single customization feature of the final software product.

The type information has to be added to the process feature model of the SPLE tool. To support the domain experts in creating the according mappings, the following rules are automatically applied by the SPLE tool based on the BPMN types (Sinnhofer et al. (2017)):

- **Activities**: Non-atomic activities are used to group specific sets of input parameter to a single feature. For example, a process designed for customizing an application may require several input parameter (like user name, password, license files, etc.). As a consequence, non-atomic activities will appear as an Input Group for all inputs defined by the according sub-process(es). Any atomic activity will be automatically tagged as input type ”None”. The input type ”None” is also automatically applied if a non-atomic activity does not contain any data. Consequently, ”empty” non-atomic activities will also not appear in the generated order entry form.
- **Gateways**: Are used to define the structure of the generated form. For example, for a decision node, a drop down selection will appear such that the customer can choose between different customization paths. For decisions it is further enforced that the customer can only select and submit the data for one single path.
- **Data**: Data to be provided by any entity involved in the process(es). With respect to our case study, ”String” turned out to be a meaningful default value.
- **Pool** and **Lanes**: Are used to define the source of the input data. For example, a data node which is part of a company internal lane will automatically be tagged as an ”Internal Input”, while Data in an external lane will be marked as ”External Input”. ”Internal Input” should be used as a default value to circumvent accidental exposure of internal configuration settings to the end-customer if Pool and Lanes are not used.

All default mapping rules can be manually overwritten by the Domain Expert during the creation of the process model. Changes to the process model

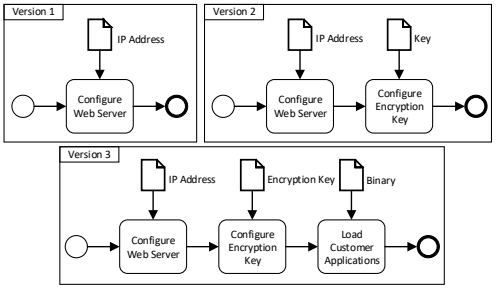


Figure 2: Exemplary order processes for the three different versions of the IoT device, based on (Sinnhofer et al. (2017)).

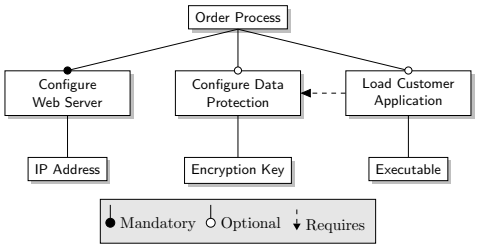


Figure 3: Exemplary feature model for the customization of the IoT device in three different flavors.

(e.g. adding/removing/changing activities) are traced via unique identifiers and illustrated as a diff-model such that changes can be reviewed by the Domain Experts. After the order process model was successfully tagged, the according order entry forms can be generated. With respect to this work, we have chosen web-based forms since they are commonly used in practice.

As illustrated in Figure 1, the provided customer data is used to create the feature selection of the final product. A manual verification step is advisable in order to ensure that no mistake was made during the development of the translation logic. Additionally, for certification purposes it may also need to be proven that a verification was done to ensure that no customer related data is confused with other products. In order to automatically select the features, the grouping information of the order entry is used to select the required features. After the selection was approved by the Domain Experts, it is automatically processed by the product specific code generators of the product line which utilizes the provided order data to actually generate the according product. The result of this process strongly depends on the use case: It could be a binary file that is loaded to the Integrated Circuits during the production, a configuration script which is executed on the final product, or any other approach. We will discuss a script based approach in Section 4 in more detail.

Especially for new types of products it is very likely that new knowledge is gained on how to increase the efficiency of the whole process(es). Only if changes to the generated order entry forms are necessary, a maintenance process for the product customization system is required. The maintenance costs for the product line can be kept as low as possible, since the code generators and model transformation logic only has to be updated once, but can be reused for the whole product line.

4 INDUSTRIAL CASE STUDY

For illustration purposes, we will discuss our industrial case study in more detail, showing how process models are translated and the final product is automatically derived. The implemented business processes of our industrial partner are controlled by an SAP infrastructure and are designed with the BPM-Tool Aeneis. Further, we are using the SPLE tool pure::variants to manage to variability of the business processes as well as the variability of the final product configurations. A more detailed description of the developed tools plugins can be found in our previous publications (see Sinnhofer et al. (2015, 2016)). For illustration purposes, we will consider the following – simplified – example: A company is developing small embedded systems which are used as sensing devices for the Internet of Things (IoT). The devices are sold to distributors (refered to as customer in the following) in high quantities which mean that the customization of the devices cannot be done by the customer in a feasible way. The device is offered in three different variants with the following features (based on Sinnhofer et al. (2017)):

Version 1: Senses the data in a given time interval and sends the recorded signal to a customer operated web-server which is used for post-processing the data. In the first version, the communication channel between the web-server and the device is unprotected. The customer is responsible for providing the connection string of the web-server to the company.

Version 2: Additionally to the basic features of the first version, this version allows encryption of the communication channel between the server and the node using symmetric encryption algorithms. For simplicity of this example, it can be assumed that the encryption key is provided by the customer, which has to be loaded to every single device. For simplicity, we assume that the key is sent in plain by the customer.

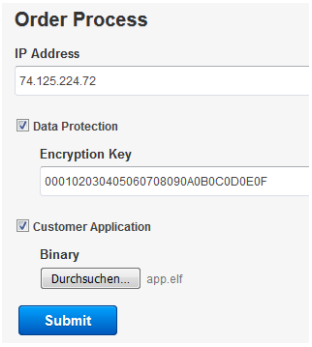


Figure 4: Exemplary order entry form that is automatically generated from the Feature Model highlighted in Figure 3

Version 3: Additionally to the basic features of the first version, this version allows customer applications to be run on the system. This requires that the customer submits a binary file which is loaded to the device during the production.

Traditionally, this would result in three different order processes which are formed via a copy and clone strategy (see Figure 2): The order process of the first version is copied and extended for the second version, while the third version is an extended copy of the second version. This means that changes to the basic version would result in the maintenance of two other processes as well. Using our developed framework leads to the situation that all three process variants are derived from one common process model. As such, the same result is achieved like using a manual preparation, but the maintenance costs can be reduced essentially since all variants are automatically updated. The according Feature Model is illustrated in Figure 3. For illustration purposes, a ”requires” relationship between the web-server configuration and the data protection configuration is not highlighted since the ”Configure Web Server” feature is a mandatory feature. Consequently, the configuration is always part of the final product and does not need to be explicitly modeled.

To automatically generate the order entry form based on the feature model, the input data ”IP Address”, ”Encryption Key”, and ”Binary” has to be set to the according type. As such, rules can be defined to ensure that the given IP Address is formatted as a valid IPv4 or IPv6 IP Address, or that the encryption key has a meaningful length, etc. If no additional checks are implemented, the Domain Expert would only need to specify the input type of the ”Binary” to be a binary file. Doing so, the order entry form illustrated in Figure 4 can be generated, assuming that all input parameter are provided by the customer. After clicking

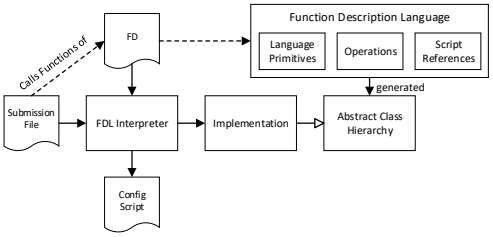


Figure 5: Framework for a flexible, runtime-configurable script generation system

the submit button on the order entry form, the provided data is converted into an XML file and zipped together with all the provided files to a zip archive. The XML file is necessary to ensure that the product configuration product line is able to automatically interpret the given zip file. Further, having an XML file has also the positive side effect that it is human readable which is essential for manual verification steps. Additional data can be included into the archive like identifiers and time-stamps to have a traceable link from placing an order to the actual manufacturing of the product.

Based on the provided data, the final product can be generated using dedicated code generators. To allow a flexible system without the need of re-releasing the product line if new products are supported, we decided to define run-time configurable generators. For this purpose we defined a Domain Specific Language (DSL) which is designed to be used by product experts. This DSL is called Function Description Language (FDL) and is used to create customization scripts for the final product. This means that during the production process, a script is loaded to each device which is triggered automatically to customize the according devices. For every supported product family, a Function Description (FD) is written which basically lists all the possible features of the platform (i.e. the customization options of the order process model) and how the provided order data is translated into the final script. The overall concept is illustrated in Figure 5. A script library is used which contains common scripts that can be referenced by the function description. For example, a ’loadApplication’ script may be developed for the product line in order to install the customer provided application to the devices.

5 EVALUATION

First results were already compiled in our previous works (Sinnhofer et al. (2016, 2017)) in which we compared the development efforts using ”traditio-

nal software development" techniques and compared it with the overhead of the developed framework. We use the term "traditional software development" techniques for a software development with ad-hoc (or near to ad-hoc) software architecture which means that multiple different systems are designed almost independent, but make use of copy and adapt strategies. Consequently, the maintenance efforts for such systems are rather high. We investigated, that the economical breakeven point of the developed framework is at around 3 to 4 systems. Further, the robustness of the customization process was increased since automatic methods were used for the feature selection and thus, configuration errors could be reduced significantly. Through the use of automatic methods, it was also possible to generate log-files for certification purposes which are used to ensure that the provided customer data was loaded and not confused or manipulated.

In this work, we will investigate other aspects of the developed framework, namely the identification of business drivers: We analyzed the development efforts of individual product features and contrasted them with the revenue that was earned by selling the according product configuration. The development efforts were extracted from the time-recordings of the responsible workers and should give a reasonable estimate about the real development efforts. In total, 106 different product orders were analyzed which provided 30 individual configuration settings. The results are illustrated in Figure 6. The first 10 product configuration options are internal system specific configuration settings which are mandatory for every product. A business decision was taken to reduce the costs for the base system to a minimum level to ensure a low cost base product. As a consequence, the revenue earned by the basic product configuration is rather low compared to other customization options. An interesting finding was that a lot of development effort was invested in complex features which were never used for any customer or are only rarely used. Due to this finding, it was decided that some of the features will be removed from the product in a future release, in order to reduce the overall costs. As illustrated in the Figure, feature 12 required a lot of development effort, but is also frequently ordered by customers. Consequently, an improvement process was triggered in order to reduce the costs of this feature.

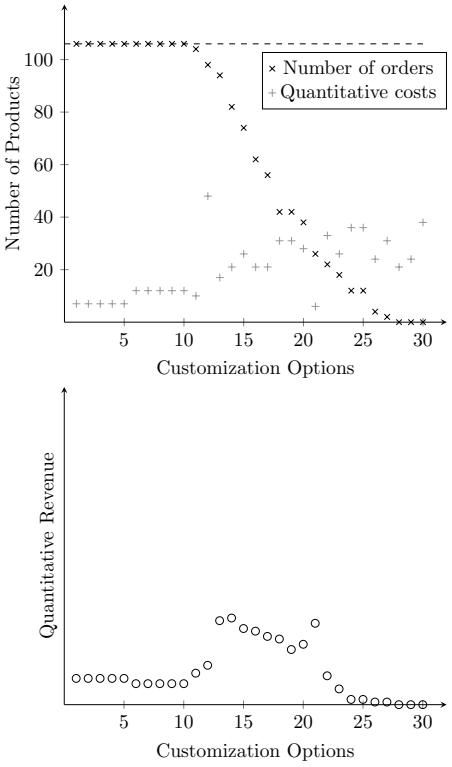


Figure 6: Analysis of the development efforts and revenue to identify business drivers. The revenue and the costs are illustrated in a quantitative manner.

After having discussed the positive aspects of the developed framework, we also want to address some limitations of the current implementation: While we were able to fully generate the required customization scripts for simple product configurations, we were able to only partially generate the scripts for complex product configurations due to the high number of inter-feature constraints of the product features. This is not a technical problem of the approach, but having a complete coverage of all inter-feature constraints is a time-consuming and iterative process. Further, modeling all the constraints in advance is usually not possible for complex systems. As a result, we decided to model only basic constraints in advance and to update the constraint model with every product order. Based on this semi-automatic generation, we managed to reduce the time to release a complex product by 50 percent.

6 RELATED WORK

Traditionally, business process modeling languages do not explicitely support the representation of families of process variants (Rosa et al. (2017)). As a consequence, a lot of work can be found which tries to

extend traditional process modeling languages with notations to build adaptable process models. As such, adaptable process models can be customized according to domain requirements by adding or removing fragments to the model and by explicitly transforming this model to dedicated process variants which can be executed in the field. This promises to increase the flexibility of business process oriented organizations with respect to highly flexible requirements of the market. Having such a variability modeling for business process models builds the foundation of this work. Thus, related work which is utilizing similar modeling concepts are presented in the following:

Derguech (2010) presents a framework for the systematic reuse of process models. In contrast to this work, it captures the variability of the process model at the business goal level and describes how to integrate new goals/sub-goals into the existing data structure. The variability of the process is not addressed in his work.

Gimenes et al. (2008) presents a feature based approach to support e-contract negotiation based on web-services (WS). A meta-model for WS-contract representation is given and a way is shown how to integrate the variability of these contracts into the business processes to enable process automation. It does not address the variability of the process itself but enables the ability to reuse business processes for different e-contract negotiations.

While our used framework to model process variability reduces the overall process complexity by splitting up the process into layers with increasing details, the PROVOP project (Hallerbach et al. (2009a,b) and Reichert et al. (2014)) focuses on the concept, that variants are derived from a basic process definition through well-defined change operations (ranging from the deletion, addition, moving of model elements or the adaptation of an element attribute). In fact, the basic process expresses all possible variants at once, leading to a big process model. Their approach could be beneficial considering that cross functional requirements can be located in a single process description, but having one huge process is also contra productive (e.g. the exchange of parts of the process is difficult).

The work of Gottschalk et al. (2007) presents an approach for the automated configuration of workflow models within a workflow modelling language. The term workflow model is used for the specification of a business process which enables the execution in an enterprise and workflow management system. The approach focuses on the activation or deactivation of actions and thus is comparable to the PROVOP project for the workflow model domain.

Rosa et al. (2008) extends the configurable process modelling notation developed from Gottschalk et al. (2007) with notions of roles and objects providing a way to address not only the variability of the control-flow of a workflow model but also of the related resources and responsibilities.

The Common Variability Language (CVL Haugen et al. (2013)) is a language for specifying and resolving variability independent from the domain of the application. It facilitates the specification and resolution of variability over any instance of any language defined using a MOF-based meta-model. A CVL based variability modelling and a BPM model with an appropriate model transformation could lead to similar results as presented in this paper.

The work of Zhao and Zou (2011) shows a framework for the generation of software modules based on business processes. They use clustering algorithms to analyse dependencies among data and tasks, captured in business processes. Further, they group the strongly dependent tasks and data into a software component.

7 CONCLUSION

The reuse of business process models is an important step for process driven organizations to survive in a competitive market. Through an integrated view of the according IT, it is possible to raise the efficiency of the overall business. With this and our previous works, we proposed a way to use software product line engineering techniques for the modeling of business process models. Further, we developed a framework which enables to combine the variability models of order processes with the variability models of software customization product lines. This enables an automatic customization process which is triggered by the according order processes. As a result, the development costs and the required time to react to changes of the market can be reduced significantly. Moreover, using the proposed techniques supports Domain Experts to identify business drivers and thus, raise the overall efficiency of the organization.

In the current state, the presented framework is focused on covering the variability of order processes for similar type of products. Consequently, future work will address the extension of the developed framework to other processes. Further, we are currently investigating methods on how to bind non-functional requirements like security requirements to the variability models in order to enforce specific properties throughout the whole process in an automatic and systematic way.

ACKNOWLEDGEMENTS

The project is funded by the Austrian Research Promotion Agency (FFG). We want to gratefully thank Danilo Beuche from pure::systems for his support.

REFERENCES

Association of Business Process Management Professionals (2009). *Guide to the Business Process Management Common Body of Knowledge: ABPMP BPM CBOK®*. Association of Business Process Management Professionals.

Derguech, W. (2010). Towards a Framework for Business Process Models Reuse. *In The CAiSE Doctoral Consortium*.

Gimenes, I., Fantinato, M., and Toledo, M. (2008). A Product Line for Business Process Management. *Software Product Line Conference, International*, pages 265–274.

Gottschalk, F., van der Aalst, W. M. P., Jansen-Vullers, M. H., and Rosa, M. L. (2007). Configurable Workflow Models. *International Journal of Cooperative Information Systems*.

Hallerbach, A., Bauer, T., and Reichert, M. (2009a). Guaranteeing Soundness of Configurable Process Variants in Provop. In *Commerce and Enterprise Computing, 2009. CEC '09. IEEE Conference on*, pages 98–105. IEEE.

Hallerbach, A., Bauer, T., and Reichert, M. (2009b). Issues in modeling process variants with Provop. In Ardagna, D., Mecella, M., and Yang, J., editors, *Business Process Management Workshops*, volume 17 of *Lecture Notes in Business Information Processing*, pages 56–67. Springer Berlin Heidelberg.

Hammer, M. and Champy, J. (1993). *Reengineering the Corporation - A Manifesto For Business Revolution*. Harper Business.

Haugen, O., Wasowski, A., and Czarnecki, K. (2013). Cvl: Common variability language. In *Proceedings of the 17th International Software Product Line Conference*, SPLC '13.

Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E., and Peterson, A. S. (1990). Feature-oriented domain analysis (foda) feasibility study. Technical report, Carnegie-Mellon University Software Engineering Institute.

McCormack, K. P. and Johnson, W. C. (2000). *Business Process Orientation: Gaining the E-Business Competitive Advantage*. Saint Lucie Press.

Object Management Group, O. (2011). Business process model and notation (bpmn). version 2.0. pages 1–538. available at http://www.omg.org/spec/BPMN/2.0/.

Österle, H. (1995). *Business Engineering - Prozess- und Systementwicklung*. Springer-Verlag.

Pohl, K., Böckle, G., and Linden, F. J. v. d. (2005). *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Reichert, M., Hallerbach, A., and Bauer, T. (2014). Lifecycle Support for Business Process Variants. In Jan vom Brocke and Michael Rosemann, editor, *Handbook on Business Process Management 1*. Springer.

Rosa, M. L., Aalst, W. M. P. V. D., Dumas, M., and Milani, F. P. (2017). Business process variability modeling: A survey. *ACM Comput. Surv.*, 50(1):2:1–2:45.

Rosa, M. L., Dumas, M., ter Hofstede, A. H. M., Mendling, J., and Gottschalk, F. (2008). Beyond control-flow: Extending business process configuration to roles and objects. In Li, Q., Spaccapietra, S., and Yu, E., editors, *27th International Conference on Conceptual Modeling (ER 2008)*, pages 199–215, Barcelona, Spain. Springer.

Saidani, O. and Nurcan, S. (2007). Towards context aware business process modelling. In *8th Workshop on Business Process Modeling, Development, and Support (BPMDS07), CAiSE*, volume 7, page 1.

Sinnhofer, A. D., Pühringer, P., and Kreiner, C. (2015). varbpm - a product line for creating business process model variants. In *Proceedings of the Fifth International Symposium on Business Modeling and Software Design - Volume 1: BMSD,*, pages 184–191.

Sinnhofer, A. D., Pühringer, P., Potzmader, K., Orthacker, C., Steger, C., and Kreiner, C. (2016). A framework for process driven software configuration. In *Proceedings of the Sixth International Symposium on Business Modeling and Software Design - Volume 1: BMSD,*, pages 196–203.

Sinnhofer, A. D., Pühringer, P., Potzmader, K., Orthacker, C., Steger, C., and Kreiner, C. (2017). *Software Configuration Based on Order Processes*, pages 200–220. Springer International Publishing, Cham.

Strnadl, C. F. (2006). Aligning business and it: The process-driven architecture model. *Information Systems Management*, 23(4):67–77.

Valena, G., Alves, C., Alves, V., and Niu, N. (2013). A Systematic Mapping Study on Business Process Variability. *International Journal of Computer Science & Information Technology (IJCSIT)*.

Weiss, D. M. and Lai, C. T. R. (1999). *Software Product-line Engineering: A Family-based Software Development Process*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Willaert, P., Van Den Bergh, J., Willems, J., and Deschoolmeester, D. (2007). *The Process-Oriented Organisation: A Holistic View - Developing a Framework for Business Process Orientation Maturity*. Springer.

Zhao, X. and Zou, Y. (2011). A business process-driven approach for generating software modules. *Software: Practice and Experience*, 41(10):1049–1071.

Where do all my Keys come from?

Andreas Daniel Sinnhofer
Christian Kreiner
Christian Steger
Institute of Technical Informatics – Graz University of Technology, Austria

Felix Jonathan Oppermann
Klaus Potzmader
Clemens Orthacker
NXP Semiconductors, Austria

ABSTRACT

Nowadays, cyber-physical systems (CPS) are omnipresent in our daily lives and are increasingly used to process confidential data. While the variety of portable devices we use excessively at home and at work is steadily increasing, their security vulnerabilities are often not noticed by the user. Therefore, portable devices such as wearables are becoming more and more interesting for adversaries. Thus, a robust and secure software design is required for the implementation of cryptographic communication protocols and encryption algorithms. While these topics are well discussed and subject to further research activities, the issue of provisioning the initial device setup is widely uncovered. However, the protection of the initial setup is as important as the protection of the confidential data during the time in use. In this work, the authors will present different solutions for a secure initialization of security critical integrated circuits (ICs).

Keywords: Cryptographic Keys, Key Management, Secure Integrated Circuits, Initialization, Secure Data Generation

INTRODUCTION

Cyber-physical systems (CPS) and Internet of Things (IoT) devices are increasingly used in our daily lives. Generally speaking, IoT refers to the connection of our everyday objects with a network like the internet. Each of these devices is usually equipped with different kind of sensors to observe its environment, making the device a smart object. In combination with embedded systems, IoT promises to increase the quality of our daily lives by taking over simple tasks like controlling the room temperature and cooking coffee (Nest Labs, 2016). On the other hand, smart objects like wearables are becoming more and more interesting for adversaries due to their increasing functionalities like internet capabilities, cameras, microphones, GPS trackers and other senor devices.
Furthermore, such smart objects are often deployed in unsupervised and untrusted environments raising the question about privacy and security to a crucial topic. Thus, a robust and secure software design is required for the implementation of cryptographic communication protocols and encryption algorithms. Moreover, tamper-proof solutions like secure elements and trusted platform modules are necessary to securely calculate cryptographic functions and to store confidential data or cryptographic keys. While cryptographic protocols and secure hardware architectures are well discussed and subject to further research activities, the issue of provisioning the initial confidential device setup is widely uncovered. However, the protection of this initial setup is as important as the protection of the confidential data during the time in use. Especially the protection of master keys is essential, because otherwise all security measures, which are based on such keys, are futile.

Due to the high quantity of produced chips – e.g. 8.8 billion secure elements for smartcard chips in 2014 (IHS Markit, 2014) – it is obvious that automatic methods are required to generate the trusted data needed for each chip. Otherwise an economical and practical production is infeasible. On the one hand, the system creating this data has to be designed flexible since every product can support different cryptographic protocols and thus, require different keys. On the other hand, the personalization system needs to fulfil high security requirements to prevent the risk that the generated data leaks during the production process to an operator – or even worse – to an adversary.
As revealed in 2015 by Edward Snowden, the secret master key of SIM cards, securing the 3G and 4G mobile communication channels was subject to such an attack (Begley & Scahill, 2015; Scahill, 2015):

The American "National Security Agency" (NSA) and the British spy agency "Government Communications Headquarters" (GCHQ) perpetrated an attack and hacked into the network of Gemalto. Gemalto produces, amongst other things, Smart Cards in the form of SIM cards and EMV (Europay, MasterCard, and Visa) chip cards. More precisely, the company generates and inserts an individual cryptographic key (a symmetric encryption key) into each SIM card during the personalization process of the manufacturing process. The inserted cryptographic key is used to secure the communication between the mobile phone and the cell tower of the mobile network operator. Mobile network operators purchase SIM cards in bulks with pre-loaded keys by Gemalto. Additionally, the mobile network operators get a copy of each key from Gemalto in order to allow their networks to recognize an individual's phone. For this purpose, Gemalto provided a file containing the cryptographic keys for each of the new SIM cards to the mobile network operator. The primary goal of this hack was to steal millions of such symmetric encryption keys to wiretap and decipher the encrypted mobile phone communication. By using the stolen symmetric encryption keys, the national agencies can decrypt any mobile phone conversation or text message sent by a mobile phone having a Gemalto SIM card. With this heist, no assistance from the mobile operators, or permission from the legal official court was necessary. To get inside Gemalto's network, social engineering attacks like phishing and scouring Facebook posts where used to take over the employees PCs (Begley & Scahill, 2015). Once inside the network, the agencies were able to retrieve the cryptographic keys because Gemalto sent them via unencrypted FTP to the Smart Card manufacturing factories. According to Begley & Scahill (Begley & Scahill, 2015), millions of keys where stolen by GCHQ in a three month period in 2010. This is a good example showing the impact of insecure data handling and how many users can be affected by hacking the personalization system of secure integrated circuits.

In this chapter, the authors will present state of the art solutions for the secure generation and distribution of security critical data during the production of secure integrated circuits. The presented solutions are based on techniques form the Smart Card industry, since those processes have already evolved over the past decades. The reminder of this chapter is structured in the following way Section "**BACKGROUND**" gives an overview about the lifecycle of secure CPS devices and more generally of secure integrated circuits. Security Certification is an essential topic in the domain of consumer products. Thus, concepts for security certification are summarized as well. Section "**PERSONALIZATION OF SECURE INTEGRATED CIRCUITS**" describes the whole personalization process of secure integrated circuits during the production process. The Section gives information about how to generate the data, how the data is loaded and defines generic protection mechanisms that can be used to protect the process. Further, the authors summarize which data is generally personalized during the production and who is responsible for generating / protecting it. The Section is concluded with potential attack vectors and attackers which may be interested in hacking the personalization system. Finally, this Chapter is summarized and concluded with future research trends.

BACKGROUND

From a high level perspective, a secure CPS or IoT device – or more general a secure system – consists of two separated system components. The first system component consists of all non-security critical components like peripherals or processing units. The second component is the security critical equipment which consists in most cases of a secure microcontroller and a tamper-proof memory which are usually combined in a single chip. This ensures that cryptographic functions can be securely processed and confidential data is stored in a secure manner. The second component can consist of multiple microcontrollers or co-processors, optimized for special purposes. The authors will use the term secure Integrated Circuit (IC) in the following as an abbreviation for all security critical modules, controllers, and memory.

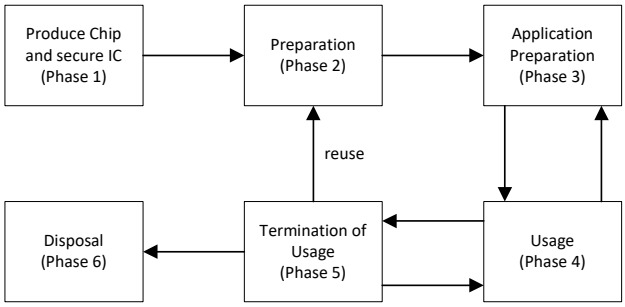
The aforementioned two component structure is similar to the structure of secure Smart Cards (Rankl & Effing, 2010) where the Smart Cards consists of two independent components; the card body including the printing and the external interface, and the card module for cryptographic operations and secure storage. Since there is no standard which defines the lifecycle for secure ICs, the authors will adapt the lifecycle of Smart Cards, which is defined in ISO-10202-1 (ISO 10202-1, 1991). The domain of Smart Cards is an ideal example for requirements of future IoT devices: Smart Cards need to fulfill high security requirements, are produced in mass production processes, and are cheap for the end-customer.
The focus of this chapter is the secure personalization process of secure ICs. In case of consumer products, it is likely that those secure ICs are evaluated in a security certification. During such security certification audits, the countermeasures of the IC are evaluated against possible attacks. Further, the production process is investigated to ensure, that the countermeasures of the IC are not circumvented during the production. Thus, the personalization process is subject of investigation during security certification audits. In the domain of secure information technology, the Common Criteria (Common Criteria, 2012) standard is widely used to evaluate the security mechanisms of IT products. To avoid costly re-evaluation of the personalization process for different products, the authors will present state of the art concepts for an incremental and modular certification process at the end of this Section.

Lifecycle of secure Integrated Circuits

Over the last decades, the development and production processes of secure Smart Cards were continuously improved with respect to security and costs. Since the market of IoT devices is expected to grow substantial over the next years (Manyika, et al., 2015), the findings of the Smart Cards domain build a good foundation for the processes of securing IoT devices. For Smart Cards, the ISO-10202-1 standard attempts to define a lifecycle which is valid independent of the manufacturing process and the application of the card. It is strongly biased by financial transaction applications, but the lifecycle is described generic and independent of the use-cases. As a consequence, the defined lifecycle is used as a basis and adopted to be used for secure CPS or IoT devices. The lifecycle can be separated into 6 phases which are illustrated in Figure 1 and shortly described in the following enumeration (Rankl & Effing, 2010; Markantonakis, Mayes, Sauveron, & Tunstall, 2010):

- **Production of Components (Phase 1):** During this phase, all components of the secure CPS or IoT device are designed and manufactured. For example, in case of a Smart Card, usually components like the Operating System, the Secure Element and the card body are manufactured. In many cases, the development and production of the individual components is done by different vendors. With respect to security, the production is an essential step since no matter how high the quality of the system and its cryptographic protection mechanisms are they are of little help if all the confidential data is leaked during the production process. Besides security consideration, also functional testing is an important topic, since the production yields of chips can be rather low for new processes. Thus, excessive testing is used on chip level to ensure the proper operation of the IC with respect to the electrical operation as well as its functionality.

Figure 1: Lifecycle of secure CPS or IoT devices adopted from the Smart Card standard (ISO 10202-1, 1991)



- **Preparation (Phase 2):** Here, all manufactured components are assembled together to the final device and the common device data is loaded. This includes configuration settings, files, and secret keys which are shared between all devices in a batch. In contrast to Phase 1, this phase is usually carried out by a single company (see (Rankl & Effing, 2010) or (Markantonakis, Mayes, Sauveron, & Tunstall, 2010)) called personalization company. In the case of Smart Cards, usually the Operating System and the applications (e.g. a banking app) are loaded to the device during this phase. For security reasons, the manufacturing of the hardware must be kept completely separated from loading the device data. Since the device data for secure devices usually always contains sensitive data, it has to be protected properly. This means that the data is encrypted and the encryption key is loaded to the device via a separate process. The authors will discuss this issue in more detail in Section "**PERSONALIZATION OF SECURE INTEGRATED CIRCUITS**"
- **Application Preparation (Phase 3):** In contrast to Phase 2, Phase 3 deals with the personalization of the device individual data. This can also include the visual personalization of the device if for example unique identifiers are engraved to the body housing. During this process, device individual secret keys are generated and loaded. As in Phase 2, this data has to be encrypted such that it is not exposed to any operator or third party during the production. In case of symmetric key material, the personalization company is further responsible for a secure distribution of the generated data to the according stakeholder. For example, in case of SIM cards (which was briefly discussed in the introduction), symmetric master keys are generated for each individual SIM card and loaded to the chips as well as distributed to the mobile network operator. In general, the process of loading additional applications require authentication to the device to ensure that only authorized persons can load the according data. This phase can be carried out from an external company or even from the customer in the field (see Phase 4) if the authentication tokens are shared with the customers.
- **Usage (Phase 4):** During this phase, the secure CPS or IoT devices are used by the end-customer. Depending on the use-case this phase also includes loading of additional applications, or the deactivation or deletion of applications. Thus, confidential data may be generated and loaded by the customer to the system. To ensure that this mechanism is protected, the customer needs to be aware of cryptographic keys which are used to securely communicate with the device. In case of symmetric channel encryption, there are two possible ways to personalize the according key. The first way is that a key is generated by the personalization company and securely shipped to the product issuer who further distributes it to the end-customer. The second approach is that the encryption key is sent by the product issuer to the personalization company. Either way, it has to

be ensured that no single operator gets knowledge about this key during the personalization process.

- **Termination of Usage (Phase 5):** Usually, the devices are thrown away by the end-customer if they are no longer needed. Nevertheless, a reuse of the device is possible if it is returned to the vendor after deactivating all applications and erasing the sensitive confidential data. The latter should always be carried out even though the device is disposed, to ensure that malicious third parties cannot gain knowledge about any secret device data.
- **Disposal (Phase 6):** As already summarizes in Phase 5, the secure CPS device should be completely erased before disposal to ensure that the contained confidential data cannot be leaked. As stated by Rankl and Effing (Rankl & Effing, 2010), recycling of such devices is an interesting topic since rare components like gold and others are used for such devices. With the increasing number of built CPS and IoT devices, recycling will even get a bigger role.

Additionally, secure CPS or IoT devices need to be developed and produced by using appropriate quality assurance methods. In the domain of security critical devices, the ISO 9000 family is usually applied to guarantee the traceability of the manufacturing process. To ensure this traceability, each individual process step is recorded. In most cases, unique identifiers (IDs) are used to track the personalizer or manufacturer of the device and every individual process step. In case of failures during the production process, the IDs are used to identify the cause and the possible impacts to other devices.

Security Certification of secure CPS or IoT devices

Security Certification is an essential step during the development of secure CPS or IoT devices. This ensures that customers gain trust in the developed security measures of the device. Further, this security evidence may be an important selling point for customers. In the domain of secure information technology, the Common Criteria (CC) standard (Common Criteria, 2012) is widely used to evaluate the security measures. The CC defines a common set of requirements that need to be implemented by the security functionality of the Target of Evaluation (ToE). As a consequence, the evaluation process creates a level of confidence into the security functionality which is implemented in hardware, software, or both. The CC evaluation also considers the maturity of the development processes, the production processes and the used toolchains in case of high security levels. This includes also the personalization process of secure systems, because every cryptographic protection mechanism is futile if a secret key is leaked during the personalization process. The assurance level of the CC is rated from a scale from EAL 1 to EAL 7 (Evaluation Assurance Level), where EAL 1 is the lowest level and EAL 7 is the highest level. A higher level does not necessarily mean a higher security level, but indicates that more effort was invested during the development and evaluation of the implemented security mechanisms. The key components and stakeholder of such an evaluation are listed in the following enumeration:

- **Evaluation Facility:** Is responsible for testing and evaluating the implemented security functionalities of a Target of Evaluation. Tests and evaluation methods are derived based on a Security Target (see below). The results of the evaluation are collected in form of an Evaluation Technical Report (ETR). Based on this report, a certification body will issue the CC certificate.
- **Target of Evaluation (ToE):** Is defined *as a set of software, firmware and/or hardware* (Common Criteria, 2012) that is target of a CC evaluation. The ToE describes the whole system and the according configuration. This configuration freedom may lead to problems during the evaluation process, since all possible configurations of the ToE must meet the defined security requirements. As a consequence, it is often the case that this configuration freedom is limited to “meaningful” configurations. These limitations are documented within the ToE in form of

guidance documents. Using the ToE beyond the defined guidance, leads to a loss of the certificate. The ToE is compiled from the vendor(s) of the system.

- **Security Target (ST):** The ST is a description of the *implementation-specific statement of security needs for a specific identified Target of Evaluation* (Common Criteria, 2012). It describes the assets, their threats and the implemented countermeasures. During the evaluation process, it is determined if the stated countermeasures are sufficient to counter the threats. Countermeasures can be divided into two separate groups (Common Criteria, 2012):
    - Security Objectives for the Target of Evaluation: These countermeasure(s) are directly implemented by the system. The correctness with respect to the threats and risks will be determined during the evaluation process.
    - Security Objectives for the Operational Environment: These countermeasure(s) are not implemented by the system, but need to be provided by the operational environment. The correctness of these countermeasures is not determined during the evaluation process.

  The ST is written by the vendor(s) of the system.
- **Protection Profile (PP)**: To allow groups and communities of interest to express their security requirements, the CC defines the concept of Protection Profiles (Common Criteria, 2012). While the ST always describes a specific ToE, a PP is designed to describe a group of ToE such that it can be reused in different STs. Thus, being compliant to a PP does not necessarily mean that a specific EAL is reached. A ToE is either fully compliant to a PP or non-compliant.
- **Evaluation Technical Report (ETR):** It is a document which is assembled by the evaluation facility during the evaluation process. It documents the overall verdict of the evaluation facility and justifies this decision based on the collected evidence of the implemented security mechanisms. It is submitted to a certification body which issues the certificate in case of a positive attestation.
- **Evaluation Processes:** The CC standard and its supporting documents defines formal and informal evaluation processes. Formal processes are explicitly defined in the standard, whilst informal processes are not defined explicitly. Thus, informal processes strongly depend on the instructed evaluation facility.

With respect to the personalization process, evaluation facilities will investigate the tools and the process flow. The used tools are usually developed by the personalization company and are designed to be compatible with a wide variety of different products. Since security requirements changes over the time, agile and modular product development techniques are widely used for a rapid and steady progression of incremental product developments, based on common parts and a modular architecture (Anderson D. , 1997). On the one hand, this leads to a faster time to market, but on the other hand it raises big challenges in terms of security certification. At present, a security certification is usually started in a late phase of the development, which can lead to a big delay between the release of the product and the date the certificate is issued (Sinnhofer A. D., Raschke, Steger, & Kreiner, 2015). Further, the personalization processes should be flexible such that it can be easily integrated into the production process of secure systems independent from the involved companies. As a consequence, a modular certification scheme is required such that the personalization process can be easily integrated into the production process of the device. This is important to ensure, that costly re-evaluations can be avoided to guarantee low production costs. The following formal or informal evaluation processes are defined in the CC standard to support modular certification processes (Sinnhofer A. D., Raschke, Steger, & Kreiner, 2015):

- **Delta Evaluation (Informal process):** The delta evaluation is a CC certification process used to maximize the reuse of previously compiled evidences. To do so, the standard specifies that the

following documents need to be shared between the evaluation facilities (Common Criteria, 2002):

- Product and supporting documentation
- New security target(s)
- Original evaluation technical report(s)
- Original certification/validation report(s)
- Original Common Criteria certificate(s)
- Original evaluation work packages (if available)

Providing these data enables that the current evaluation facility should not have to repeat the analysis of parts of the system, where the requirements have not changed nor been impacted by any other changes. Such changes are identified by performing a delta analysis. The delta analysis is an analysis between the new security target and the original security targets(s) to identify the impact of changes. A drawback of this approach is that the evaluation technical reports contain information about the evaluation process and the applied measure to proof the security objectives of the target of evaluation. As a consequence, these reports are usually considered as proprietary to the evaluation facility, which are usually not interested in sharing this knowledge with other – competing – facilities. Thus, performing a delta evaluation is a tough challenge if the evaluation facility was changed between different certifications.

- **Composite evaluation (Formal process):** Although the composite evaluation was designed for Smart Card products, it can be applied to a wide variety of products which fulfill the condition that *an independently evaluated product is part of a final composite product to be evaluated* (Common Criteria, 2012). The only restriction is that the already certified product builds the underlying platform of the composite product. This means that a layers pattern is used for the overall system architecture. Thus, it is applicable for example for an embedded system, whereas an application is running on a certified operating system which is running on a certified hardware platform. Compositional evaluations are technically similar to delta evaluations but with additional restriction to the overall structure of the product. Due to these additions, it is not necessary to share the evaluation technical reports. The lowest EAL of all components is the limiting factor of the composite evaluation. Further, the validity of the overall product certificate depends on the validity of the each individual component. As a consequence, if the validity of one component expired, the whole product certificate is invalidated.
- **Composed evaluation (Formal process):** The composed evaluation is used to certify products which consists of independently certified (or going through an independent certification process) products/modules which are assembled to a new final product. It is similar to the composite evaluation but with the difference that the overall system architecture is not limited. Further, it is applicable for situation in which a delta evaluation is not possible since the evaluation technical reports are not shared. Since the individual components are already certified, the composed evaluation mainly focuses on the interface between the components and their according interaction(s). As a consequence, new evaluation assurance levels were introduced. These levels range from CAP-A to CAP-C (Composed Assurance Packages), where A is the lowest level and C is the highest level. The assurance level CAP-C stands for *Attack potential "Enhanced Basic"* which is approximately comparable with EAL-4 (see (Common Criteria, 2012); Part 3 pages 38 and 47). Due to this limitation, the composed evaluation has been performed much less successful than other modular CC certification processes.

The above certification processes all rely on delta analysis to identify the impact of changes. As such, traceability is required to explicitly state the dependencies of security requirements and the actual implementation artifacts and tests. To incorporate these requirements, Raschke et al. (Raschke, et al., 2014) introduced two processes to support an agile and modular development and certification process. The impact analysis is based on a change detection analysis in combination with a traceability impact

analysis. A security model is used to describe the properties and dependencies. This model is based on the CC security target, the design documentation, the implementation artefacts and the tests.
To summarize, in the context of personalization processes, usually the Delta – or Composite evaluation is used to evaluate the tools, the process, and the configuration space.

PERSONALIZATION OF SECURE INTEGRATED CIRCUITS

The personalization process covers the phases 2 and 3 of the secure IC lifecycle which was described in Section "**Lifecycle of secure Integrated Circuits**". These phases are usually carried out by the same company which is usually referred to as the personalization company (Markantonakis, Mayes, Sauveron, & Tunstall, 2010). During Phase 2, the application and data is loaded which is common for every single device. In contrast, Phase 3 is dedicated for loading device individual applications or settings. This separation is required, since loading individual data to the devices is much more complex and time intensive than loading the common data (Rankl & Effing, 2010). As a consequence, the personalization company will try to minimize the size of the loaded data during Phase 3 as much as possible.
For secure applications, confidential data needs to be generated and loaded to the cards. Therefore, several security requirements need to be fulfilled by the process: On the one hand it requires encryption such that confidential data is not leaked to any third party, including also the operator of the personalization equipment; on the other hand, it needs to be ensured that only authorized data or applications can be loaded to the system. In case that symmetric key material is generated that needs to be shared with the product issuer, the personalization company needs to take care of securely distributing the keys to the issuer.
Since the personalization process is operated in an automated high volume production process, all implemented security requirements need to be able to be operated in – at least – near real time.

Generation of the personalization data

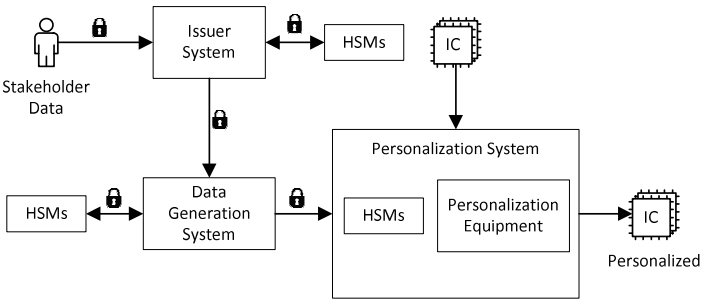
For key management purposes, various standardization organizations (like (NIST FIPS PUB 140-2, 2001; NIST SP 800-57-1, 2016; PCI Security Standards Council, 2016)) recommend the use of dedicated hardware security modules (HSM) to protect the data. Hardware security modules are tamper-resistant hardware modules, designed for generating random numbers and calculating cryptographic functions. Furthermore, the hardware is optimized to accelerate the computation of the cryptographic functions, to ensure that a large amount of data can be processed efficiently by such systems. Thus, HSMs are a natural consequence for generating the required confidential personalization data in a secure and efficient way.

Figure 2 illustrates the basic components of a personalization system (adapted from (Chan & Ho, 2003)). The "Issuer System" contains the product and order specific data that is required by the personalization company to generate the data. This may also include static (confidential) data that is provided by a stakeholder (e.g. a mobile network operator). As illustrated in the Figure, external Stakeholder data is sent via a secure channel between the Stakeholder(s) and the "Issuer System". Usually, HSMs are used to secure the communication channel. This means that the sent data is encrypted with a key that is only available within the HSM. Typically, protocols using asymmetric cryptography are used to protect the communication channel (like TLS), but also other domain specific protocols may be used for this purpose (see (Sinnhofer, et al., 2016)). In case of symmetric encryption, the corresponding protection key needs to be separately transported from the data. Recommendations foresee, that symmetric keys are split onto multiple shares which are sent on multiple independent ways (PCI Security Standards Council, 2016). Only after successful reception of one share, the next share is sent.

The HSMs of the "Data Generation System" are used to generate the required device data. This includes for example the generation of random keys, or the derivation of keys based on master key data that was sent by a Stakeholder. But also non confidential data like unique chip identifiers may be generated by the

system. To be able to generate data that is based on Stakeholder data, a secure communication channel between the HSMs of the "Issuer System" and the "Data Generation System" is required.

Figure 2 Principal Structure of a personalization system (adapted from (Chan & Ho, 2003))



The "Personalization Equipment" in general consists of special hardware which ensures that the data is sent cryptographically protected and authenticated to the system (Hautier, Macchetti, & Perrine, 2012; Chen, 2007). The data which is exchanged between the "Data Generation System" and the "Personalization Equipment" is again protected through a secure communication channel between the HSMs. For performance reasons, this channel is usually based on symmetric cryptography. Since such key exchange processes take up to several working days, it is usually executed several days or weeks in advance of the production. Re-keying is recommended for every separate production, even if the same products are manufactured.

The personalization equipment is a special device, to which the secure ICs, or the whole devices, are automatically mounted. In practice, many personalization devices are used in parallel to increase the throughput of the product line (Rankl & Effing, 2010). The Personalization equipment is usually executing scripts (Graham, Jr., Nablo, & Haeuser, 2002) that are prepared in advance from the personalization company based on the order. The script contains placeholder for the data that is either generated from the "Data Generation System", or directly provided by the issuer. In case that the data is directly provided by the issuer, the data needs to be re-encrypted by the "Data Generation System" for the "Personalization System".

The bottlenecks for this process are transferring the data to the chips and writing it into the non-volatile memory. The data transfer-rate heavily depends on the device interface. For example, if the personalization data is required to be loaded via NFC (ISO 18000-3, 2010), the data rate is limited to a maximum of 423,75 kBit per second. Further, in case of slow memories (EEPROM) the time needed to write the data can be high as well. Consider the following example: if 16 kBytes of data shall be loaded to a single device, the time needed to transfer the data is about 300 milliseconds. This may not seem much, but if one million devices shall be personalized, the communication alone will require about three and a half day! Considering the additional overhead of changing devices and processing, this number is even higher. As a consequence, the common device data is usually loaded in a previous production step (Rankl & Effing, 2010), using an interface which allows high data-rates. One common used practice to ensure short personalization times is 'hard-wiring' applications such that they are already programmed during the lithography process in Phase 1. As a consequence of this, updating such 'hard-wired' applications may require additional effort since the ROM-Code cannot be updated during the time in use. Another drawback of ROM based products is, that the ROM mask itself has to be prepared several weeks before the start of the production. An alternative option to ROM-Code based products is the trend towards Flash based products: a Flash memory is used to store the applications that were previously loaded to the ROM

code. As such, 'hard-wiring' an application does not require a time consuming preparation of a ROM-mask.

Usually, the "Data Generation System" (as illustrated in Figure 2) is using a fixed and certified HSM firmware to generate the actual personalization data. This is required by security certifications to ensure, that the confidential data is not leaked during the data generation process. As a consequence, this firmware is in most cases suited for one dedicated product. If other products need to be supported, a new firmware needs to be developed and certified. In case of complex multi-application products, this is a considerable amount of development and certification effort. In the domain of secure ICs, the required data is similar for different use-cases or applications. Having a mature, modular development and certification processes can significantly reduce the required effort and costs. As described in Section "**Security Certification of secure CPS or IoT devices**", a security model can be used to trace the requirements of the personalization system to the implementation artifacts of the HSM firmware implementation. Thus, changes can be easily tracked and evaluated by the according evaluation facility. For traditional software development, the problem of flexibility is usually solved by introducing a dedicated API that can be called arbitrarily. As identified by Anderson (Anderson R. J., API Attacks, 2001), the most common API failure mode of secure systems is that functions / transactions that are secure in isolation become insecure in combination. The cause of such errors is usually application syntax, feature interactions, slow leakage, or concurrency problems. Developers of such system have to be aware, that an attacker could use any unexpected combination of function calls in order to break the trusted system. To solve the issue of leaking information, security APIs are usually designed simple such that they can be formally analyzed during the evaluation process. This is an open research challenge and thus, the authors will discuss this topic in Section "**FUTURE RESEARCH DIRECTIONS**" in more detail.

Authentication Methods

During the personalization process, random session keys are usually used to communicate with the individual chips. To start this process, the personalization system needs to be aware of an authentication token. This is necessary to ensure a unilateral or mutual authentication between the personalization equipment and the secure IC. There are many different methods that can be used, but the following describes a simple but common approach from the Smart Card domain (Rankl & Effing, 2010):

During the production of the secure IC, the producer of the operating system (OS) incorporates a secret value to the ROM (Read Only Memory) code of the IC. The ROM usually contains the static functions and data which are already programmed during the manufacturing of the chip. This is done by preparing a ROM mask from the program code which is applied to the chip using lithographic processes. Thus, the code is "hard-wired" to the memory. To prevent attacks on the ROM code, it is usually protected against optical attacks, including etching attacks.

The secret ROM value is combined with a value which is written to the non-volatile memory (EEPROM or Flash) of the chip by the semiconductor manufacturer to create an authentication token. The idea is that neither the OS producer, nor the semiconductor manufacturer has enough knowledge to generate this authentication token by his own. The parts of this token are sent encrypted to the personalization company. The described method can be used to generate authentication tokens which are valid for one production batch. The method can be refined if a chip individual number is combined with the secret values. Thus, the authentication tokens are only valid for a single specific chip. This increases the security in case that the chips are compromised before being completed. This method – or a similar method – is commonly used in practice.

Another attack that has to be prevented during the personalization process is the manipulation of the software of the IC during the production process. An attacker with sufficient resources (e.g. a national agency or a competitor) may be able to smuggle a manipulated IC to the personalization process. This

manipulated IC basically acts like the real product, but has additional functionality like dump routines which can be used to retrieve the plain confidential data that was loaded during the personalization process. Again, a large variety of methods can be used to ensure the integrity and authenticity of the secure IC, but the following method is commonly used in practice (Rankl & Effing, 2010):

The semiconductor manufacturer stores a chip individual key into the non-volatile memory during the production process. The chip individual secret key and the unique chip identifier are securely shared with the personalization company. Before sending the personalization data to the card, the personalization system sends a challenge consisting of a random number to the chip. The chip answers this request by computing a keyed-hash message authentication code (HMAC) using the secret key. The message that is hashed consists of the ROM code and the received random number. The random number is necessary to prevent replay attacks. Further the unique chip identifier is retrieved from the card, such that the personalization system can identify the secret HMAC key. With the knowledge of the key and the contents of the ROM code, the personalization system is able to verify the retrieved HMAC. Thus, manipulation to the ROM code – and respectively to the operating system – can be detected. The verification is done in the HSM of the personalization system and thus, the secret HMAC key is never exposed to any operator.

Exemplary personalization process

To illustrate the application of the introduced personalization System (see Figure 2), the authors will elaborate the process in an exemplary case study showing the production of SIM cards for 3G. This requires the personalization of two unique identifiers and a symmetric encryption key: the International Mobile Subscriber Identity (IMSI), the International Mobile Station Equipment Identity (IMEI) and the Authentication and Key Agreement (AKA) key. In the following, the authors will use the term “shared” for data that is shared between the different stakeholders in an encrypted way. The plain data is only accessible by the HSMs of the Stakeholder.

SIM cards are usually ordered by a mobile network operator from a SIM card manufacturer in batches of several thousand devices. For simplicity, it is assumed that the SIM card manufacturer is responsible for the Hardware and Software development, while a dedicated Personalization company is conducted for the personalization. For illustration purposes, it is assumed that the AKA key is derived from a Master Key that is generated by the personalization company and that the IMSI and IMEI are provided by the mobile network operator.

To start the process, a secure communication channel has to be established between the mobile network operator and the SIM card manufacturer (e.g. using TLS) via which the IMSI and the IMEI is shared. Since the IMSI and IMEI are in general not confidential, the data can be sent in plain, but it has to be authenticated. The authentication is required to ensure that the provided data is not manipulated, or sent from an untrusted entity.
The SIM card manufacturer orders the personalization of the SIM cards from a dedicated personalization company. As such, another secure channel has to be established between the SIM card manufacturer and the personalization company. Via this channel, the data of the mobile network operator is forwarded as well as an authentication token such that the personalization company is able to authenticate to the manufactured SIM cards. The “Data Generation System” of the personalization company is used to generate the master key for the AKA key hierarchy, as well as to derive the actual keys for the individual chips. During the personalization process itself, the personalization equipment makes use of the generated data. Usually, symmetric encryption is used to protect the generated data to reduce the computational overhead.

After the personalization, the personalization company shares the generated master key as well as a list containing the derivation parameters of the individual chips with the SIM card manufacturer. This data is forwarded to the mobile network operator. The list of derivation parameter is required such that the

mobile network operator is able to compute the device individual key if a SIM card is trying to access the services of the network. The derivation data usually contains the device individual unique identifier as well as other data.

As a consequence of the described system, the personalization company is fully transparent to the mobile network operator (and vice versa) since the whole communication is carried out through the SIM card manufacturer.

Classification of personalization data

A wide variety of different data is usually personalized. This Section tries to summarize and classify the most common types of data that is loaded during this process. Generally speaking, keys which are described in the following as "known" to a production system (e.g. the personalization equipment) means, that the according key is located in a Hardware Security Module. Operators or other involved persons must not be able to retrieve such keys in plain. As illustrated in Figure 2, a secure channel exists between the issuer system and the data generation system. Thus, data or keys which are shared with the issuer can be forwarded by re-encrypting the data for the personalization system.

Operating System: The operating system of the secure IC is usually provided as ROM code during the production of the IC. Since the ROM code needs to be prepared several months before production, it may happen that updates need to be applied during the personalization process. These updates are generally loaded to the non-volatile memory (EEPROM or Flash) of the chip during Phase 2 of the lifecycle.

- **Type of Data:** Updated OS routines or code is usually provided as a binary which is signed from the OS producer. Further, encryption can be used to prevent reverse engineering or to protect hard-coded confidential data**.**
- **Owner of the involved Keys:** The signature creation key (private part of an asymmetric key pair) is owned by the OS producer and is not shared with any other party. The signature verification key (public part of the asymmetric key pair) is either hard coded into the ROM-code of the OS, or personalized by the personalization company. In the latter case, the public key needs to be sent at least authenticated to the issuer. If the OS code is encrypted, the protection key needs to be known by the issuers HSM and shared with the OS producer (or vice-versa).

Applications: Like for the operating system, applications may also be part of the ROM code which was burned to the IC during the production process (Phase 1). For Flash based products, this can also include the applications which are loaded to the Flash during the production process (Phase 1). The advantage of having such romized or pre-loaded applications is that less data needs to be sent during the personalization process, which saves time and money. As before, updates or additional application(s) may need to be loaded during the process since the final application(s) were not available when the ROM-code was created.

- **Type of Data**: Applications are usually provided as binary. Dependent on the use case, the operating system may require that the applications are signed. This may also be required from a security certification to ensure that only valid applications are loaded. Optionally, encryption can be used to prevent reverse engineering or to protect hard-coded confidential data.
- **Owner of the involved Keys**: In case of signed binaries, the signature creation key is usually owned by the OS producer. Certificate hierarchies may also be used to separate the individual application provider. The signature verification key (public key) is either hard coded into the ROM-code of the OS, or personalized by the personalization company. If the application is encrypted, the protection key needs to be known by the issuers HSM and shared with the application developer (or vice-versa). For security reasons, the protection key should be different for every individual application developer and also different of the OS protection key.

Application Keys: For security relevant applications, different cryptographic keys are required to ensure the proper functionality. For example, if the application is sending confidential data to a web-server in the internet, an encryption key is required that protects the communication. This can be achieved by using symmetric or asymmetric cryptography.

- **Type of Data**: The generated key material can be either symmetric or asymmetric. Usually, symmetric encryption keys are used if the encryption operation needs to be calculated within a limited time frame. Such symmetric encryption keys can be categorized in three groups:
    - Static Keys: Every chip will use the same static key.
    - Derived Key Hierarchy: State of the art key derivation functions (KDFs) are applied during the data generation process to generate chip individual keys based on a derivation master key. This process typically uses the unique identifier of the chip to derive the device key which is loaded during the personalization process. KDFs are cryptographic one way functions to ensure that an attacker cannot break the secret derivation master key by knowing the derived keys. In general, the data that is used during the derivation process has to be shared with the involved stakeholder. The derivation master key that is used for the key derivation may be provided by a Stakeholder, or generated in behalf of the Stakeholder from the personalization company.
    - Random Keys: As the name suggest, the keys are purely random for every individual device. Depending on the use case, this key material can be generated by the issuer or by the personalization system.

  Asymmetric Keys can be categorized into two groups:

    - Encryption Keys: The device is communicating with an external system using asymmetric encryption for a secure channel establishment. As a consequence, the encryption key (public key) of the destination needs to be personalized. This is usually done using certificates. The key-pair of the device can either be static or randomly generated.
    - Signature Keys: Either the device needs to verify the signature of message or data, or an external system needs to verify that the data was sent from a trusted device. In case that the device needs to verify a signature, the signature verification key (public key) needs to be personalized. This is usually done using certificates. If the device needs to compute the signature.

  Generally, from a security perspective static keys are the most critical use-case, since breaking one device automatically breaks all other devices.

- **Owner of the involved Keys**:
    - Symmetric Keys: Independent of the category (Static, Derivation Master Key, or Random), the key can either be provided by the application developer or generated by the personalization company on behalf of the application developer. In the latter case, the key(s) may need to be shared with the application developer or other system components in a secure manner. In any case, the protection key needs to be shared with the issuer. Whether to generate or import keys needs to be decided on a case by case basis considering all product and security requirements.
    - Asymmetric Keys: Public keys are usually provided via a certificate which is signed by a CA that is trusted by the device (i.e. the signature verification key was personalized or hard coded). The device key pair is usually randomly generated by the personalization company and loaded to the device. In rare cases, the key-pair was provided by the

application developer. In case that the key-pair was generated, the public key is usually exported as a certificate and is shared with the application developer, or any other system component that needs to be aware of the public key. The resulting certificate can be self-signed or trusted-root signed. In case that a trusted-root is used, a secure process needs to be established, such that operators are not able to issue valid certificates for misuse. Further, fake devices have to be reliably detected. The protection key used for importing static key-pairs needs to be shared with the issuer.

Device and/or Application Settings: Every device and/or application may require additional configuration data to be loaded. For example, an application which communicates with a different application or process need to know the identifier of the application or process.

- **Type of Data**: In general, proprietary binary formatted data is preferred since it can be efficiently compressed and loaded on small embedded systems. But of course, any arbitrary data can be loaded like strings, files and others. In most cases, this data is not confidential and thus, not encrypted. The data can be signed if the origin of the data needs to be verified, which could be a requirement of the security certification process.
- **Owner of the involved Keys**: In case of signed data, the signature creation key (private part of an asymmetric key pair) is owned by the according provider of the data (e.g. Customer, OS provider, etc.). The public key can either be hard coded into the ROM code, or personalized from the personalization company. In the latter case, the public key needs to be sent at least authenticated to the issuer. If the data is sent encrypted, the protection key needs to be known by the issuer's HSM and shared with the data provider (or vice-versa). For security reasons it is important to use different encryption keys for the individual provider of the data.

Loading personalization data to the chips

In principal, there are various ways for loading the personalization data to the secure ICs. The methods can be categorized into two main groups: The first category requires that the secure IC is able to perform basic file and data management commands such as Create, Install and Update. Thus, the personalization process makes use of these commands to send the data to the chip. The second category requires that the secure IC is able to load the personalization data from a defined memory region which was written during the production of the IC. The following methods are currently state of the art:

Category 1: Loading personalization data using logical addresses (Rankl & Effing, 2010) : This method tries to avoid physical addresses using basic file and data management commands. The individual datasets are identified with a symbolic name such that the secure IC is able to determine the according physical address. The sent data can be independent from the used micro-controller if the API is written in a platform independent way. The drawback of this method is that the personalization process will take more time since the secure IC needs to resolve the symbolic identifier before the data is written. This overhead is only small but in a high volume production process this may be to cost intensive.

Category 1: Loading personalization data using physical addresses (Rankl & Effing, 2010; Atsumi, Kondo, & Shona, 1995): In opposite to the first method, this method makes use of the real physical addresses of the individual datasets. Consequently, the overhead for writing the data can be kept as small as possible. In order to write this data to the real physical addresses, the personalization system needs to be aware of how the data is expected by the secure IC. This is a huge drawback since it is an additional source of error which can render all produced chips unworkable. To be able to generate the data as expected, usually a sample device is used which contains a dump memory functionality. The sample device is configured as the final product, but with the difference, that specific pattern values are used for the device individual and/or confidential data. After the initialization was done, the memory content is

dumped and the physical addresses are identified by searching the according pattern values. The memory dump is used as a template during the production of the secure ICs. Only the placeholder data is replaced during the personalization process with the real data. The critical aspect is that the process needs to assure, that it is not possible to manufacture a production device having the dump memory functionality inside. Otherwise, an attacker is able to read the confidential data from the dumped memory.

Category 2: Loading personalization data using a dedicated personalization memory region: Another possibility for loading the personalization data is using a dedicated memory region to which the personalization data is written as a whole block. As such, no specific device API is required which is able to load / update individual data. The personalization data could be loaded to the secure IC during the last stages of the production. To actually load the data, the personalization system authenticates to the IC and sends the decryption key, necessary to decrypt the personalization data. Additional meta-information is necessary such that the IC can determine the purpose of the data. Usually, the meta-information uses symbolic names which can be interpreted by the secure IC, but also real physical addresses could be used in this process. The personalization company needs to agree with the OS producer on the format of the data which is one of the main drawbacks of this approach. An additional drawback is the computational overhead which leads to higher costs during the production process. Further, the memory region for the personalization data needs to be allocated by the secure IC which can be a problem for small low cost ICs. Of course, this additional memory can be used for customer data during the in-field use, but during the production process this may be a limiting factor.

If certified products are manufactured, it is further necessary to log every production step and to validate that the correct data was loaded. Independent from the method which was used during the personalization process, the following data is usually logged in practice:

- Unique chip identifier
- Checksum or hash values of non-confidential data like applications or configuration settings
- Key Check Values (KCVs) or secure cryptographic one way hashes of confidential data

The important point is that one way functions have to be used for logging verification values of confidential data. This ensures that an attacker is not able to obtain the plain key values by observing the logged verification values. The verification values can be verified using the HSMs of the personalization system. Thus, the plain confidential data is never exposed to any operator.

Classification of attacks and attackers

In general, information technology systems are vulnerable to multiple possible threat vectors. On the one hand, design weaknesses of the hardware, as well as the software, can lead to a break of the system. On the other hand, cryptographic protocols which were consider as secure today may be broken in the future. One such example is the cryptographic strength of different key types. With the increasing computational power of personal computers, brute force attacks on short keys are getting more interesting. While several years ago, RSA-768 was considered as being sufficiently secure, nowadays RSA-2048 is recommended. With respect to the scope of this chapter, the authors will focus on possible attack vectors which aim to break the system due to weaknesses in the personalization process. This is an essential topic, since breaking the personalization system leads to breaking every single produced IC! Possible attacks can be classified as follows:

- **Hardware attacks**: As described in Section "**Generation of the personalization data**", the system for generating and importing confidential data consists of hardware security modules and the personalization equipment. Since these hardware components are only storing temporary data, which is used for the processing of the current batch, invasive attacks can be neglected due to operational measures. Nevertheless, these hardware components are still vulnerable to side-

channel attacks like timing analysis or power analysis (Demaertelaere, 2010; Sanchez-Reillo, Sanchez-Avila, Lopez-Ongil, & Entrena-Arrontes, 2002). Furthermore, manipulated hardware devices could be used to leak generated device data.

- **Software attacks**: As already mentioned in Section "**Generation of the personalization data**", the API of the hardware security modules may have vulnerabilities which can be used to leak confidential data (Anderson R. J., API Attacks, 2001) by combining specific functions.
- **Social Engineering attacks**: This sort of attack mainly aims at the people who are involved in operating the production equipment, or in developing software for the production equipment. In case that symmetric keys are shared via paper forms (as described in (PCI Security Standards Council, 2016)) between the personalization company and any other involved stakeholder, the companies responsible for dispatching the mail are also of interest to such kind of attacks.

In order to estimate the strength of attackers, the type of attackers has to be classified as well. The basic motivation behind attackers is usually the gain of financial benefit or knowledge (Rankl & Effing, 2010). National agencies for instance, are not interested in gaining financial profit, but are interested in gaining knowledge about the users, or the environment to which the system is deployed to. If details of such attacks become public, the reputation of the involved companies is damaged dramatically. Thus, another potential attack vector includes a competitor, who either wants to gain knowledge about the processes, or wants to harm the reputation.

Similar damage to the reputation can be caused by scientific researchers, which aim to break the system to gain reputation in their specific scientific field. As such, these attackers are only satisfied if their successful hacking attempt is published in scientific publication. But also individual hackers may only be motivated in being the first, breaking the system.

In the domain of the personalization process, a few potential attackers and attack vectors can be neglected: Since the production environment is secured, it is not possible for hackers to place specific hardware devices into the production environment. This may only be possible by competitors who are using the same production facilities, or organizations with a high amount of resources they could spent to manipulate the production equipment. Further, the control logic of the personalization equipment and the equipment itself is in general disconnected from any external network. As a consequence, malicious software cannot be loaded to such devices via the internet. Thus, attacks during the personalization process are mainly carried out either by insiders or organizations like competitors, national agencies, or research institutions.
In practice, the following techniques are used to minimize the risk of hacks from the above mentioned attackers:

Authentication: As already discussed in Section "**Generation of the personalization data**", every single personalization operation needs to be authenticated, to ensure that only valid data is loaded to the chips. Further, authentication of the secure IC is required such that it is not possible to replace the chip with a dummy IC. As already explained, such dummy ICs could contain code which can be used to export the personalized confidential data in plain. This is especially critical with respect to static confidential data.

Encryption: Only if data is stored in dedicated Hardware Security Modules – or the final secure IC – the data can be stored in plain. Whenever data is transferred between system components, the data has to be encrypted with reasonable strength. Recommendations for the required key length are usually published by standardization organizations like the NIST (NIST SP 800-57-1, 2016) and frequently updated according to the state of the art.

Split Knowledge: During the production process, at least a two-man rule should be used to ensure that no single operation can be performed by a single person. Otherwise, this single individual may be subject of attacks like social engineering. The two-man rule is also reflected in the security measures described in Section "**Generation of the personalization data**", where the authentication token for the

personalization company is split into two parts. One is delivered by the OS producer and one is delivered by the semiconductor manufacturer. Further, split knowledge is also used during development of the software that is deployed onto the hardware security modules, or developed for other production equipment. In general multiple people are involved in the development of the software; code reviews are mandatory for code releases, which are performed by people that are not involved into the development process. Further, it is required that multi-person authentication mechanisms are used to deploy new software onto the production equipment.

Table 1 summarizes typical attacks during the personalization process and their according countermeasures:

Table 1 Non-exhaustive list of typical attacks during the personalization process

Attack	Attacker	Description / Countermeasure
Tapping Data Communication	Insider, Organizations, Hackers	**Description**: During the whole process, confidential data is transferred between multiple stakeholders. This includes also the communication channels between the personalization equipment and the actual device during the production. **Countermeasure**: Secure messaging has to be used throughout the whole personalization process.
Manipulation of Data transfers	Insider, Organizations, Hackers	**Description**: Related to the previous attack. Proper origin of data streams has to be ensured throughout the whole process since otherwise, devices may be compromised (e.g. by loading dummy key data). **Countermeasures**: Authentication methods have to be used throughout the whole process to ensure the proper origin of the data, as well as message authentication methods to ensure that the data was not manipulated during the transfer.
"Fake" devices	Organizations, Insiders	**Descriptions**: Fake devices could be used to leak data that is loaded during the personalization process. In case that static keys are used, all devices can be broken if a fake device is integrated into the personalization process. Even worse, if faked HSMs are used to generate the data, all confidential customer data could be leaked. **Countermeasures**: Bi-Directional authentication of the personalization equipment and the actual devices. Further, multiple eye principal for the deployment and the maintenance of the personalization equipment as well as the HSMs.
Manipulation of the Personalization Equipment	Organizations, Insiders	**Description**: Manipulated software could be loaded to the personalization equipment – including also the HSMs, to leak confidential data. **Countermeasure**: Operational measures are required to restrict the access to the personalization equipment. Further, multiple eye principals have to be applied to ensure that only multiple operators are able to load software to the equipment. Authentication methods have to be used such that only authorized operators can execute functions of the personalization equipment.
Social Engineering	Organizations, Hackers	**Description**: Attacks on the social level can be carried out to gain access to company networks, or to establish backdoors for other attacks. **Countermeasures**: Split knowledge such that no single person is able to break the whole system. Secure messaging and secure storage of confidential data, as well as separation of networks. Increase the security awareness of all involved employees.

FUTURE RESEARCH DIRECTIONS

As mentioned in Section "**Generation of the personalization data**", the API of the HSMs, that are responsible for generating the personalization data, is still an open research question; although the topic of API security is subject to research since the 1990s. There is a growing literature about formal verification tools which are used to analyze the security API of a system and to detect security flaws. The problem with these tools is that they are usually designed to analyze rather simple APIs. Additionally, they require a lot of resources in terms of computational power and memory consumption. Thus, formal analysis tools are in general not suitable to be run on HSMs to ensure runtime protection against API attacks. It is an open research challenge to adopt formal analysis tools for the requirements of a personalization system.

The current alternative approach is to come up with robustness principles to guide the designers of such systems (Anderson R. J., API Attacks, 2001). Instead of providing a sequential API of the HSMs, which requires a steady interaction with the system, a set of function calls could be grouped to a single execution unit. This execution unit can be verified in advance to ensure that this sequence of function is valid with respect to the defined robustness principles. In case that the sequence of functions was verified as secure, the developer can sign the execution unit such that the HSMs are only executing verified sequences of functions. Besides this, security principals which were considered to be secure in the past are now identified to be a big security problem. For example, as Anderson pointed out (Anderson, Bond, Clulow, & Skorobogatov, 2006), the use of exclusive-or to combine a key with a PIN leads to a huge security flaw, although it was used by every VISA security module. It is up to now an open research challenge on how to systematically identify such vulnerabilities in existing systems.

Our today's organizations are driven by fast changing markets and requirements. Thus, agile and incremental development techniques are used to cope with these problems. As of today, security certification is contradicting to such development strategies. As a consequence, certification is usually carried out at a late stage of the development. Current research aims on integrating the security certification process into the development process such that the evaluation facility is able to gain trust already during the development of the system. Further attempts are made to ease the certification process by reusing previously collected evidences. But still, a lot of research and standardization activities are required to develop mature development processes that integrate security requirements. The aim of this integration should be a bilateral traceability of the security requirements to the according implementation artifacts and tests to ensure an automatic detection of the impact of changes.

CONCLUSION

The personalization of secure CPS or IoT devices is a crucial step during the production process. Every security measure which is employed during the in-field use is futile, if data is leaked during this personalization process. Especially in the domain of IoT devices, which are observing our daily activities, it is essential to protect the devices against malicious manipulations. The past has shown that hacks of such IoT devices are valuable aims for hackers. As the Mirai malware showed, one of the main problems was the careless handling of the user name and password of the webcams manufactured by Hangzhou Xiongmai Trading (Hautala, 2016). These credentials were static for every hacked webcam. Thus, every manufactured webcam of this producer can be taken over via the internet without any hint to the customer that his device was hacked by trying the default user name and password. Even worse, after changing the user name and password, the devices were still vulnerable via the remote web based administration panel (Krebs, 2016). With a mature an established personalization process, these vulnerabilities could be mitigated since every produced device could use dedicated user credentials.
As a consequence, mature personalization processes need to be established, which enables a low-cost but secure personalization for CPS or IoT devices. Thus, the developed processes need to be flexible such that they can be used for a wide variety of different products, independent of the use-case of the devices. Flexibility is also required to ensure that the personalization process can be integrated into the production

process of secure systems independent of the manufacturer of the devices. As a result, the costs for such processes can be kept as low as possible. This chapter summarized generic and state of the art concepts for secure personalization processes to get a step closer to a low cost personalization process for secure systems.

REFERENCES

Anderson, D. (1997). Agile Product Development for Mass Customization: How to Develop and Deliver Products for Mass Customization, Niche Markets, Jit, Build-To-Order and Flexible Manufacturing. Irwin Professional Pub.

Anderson, R. J. (2001). API Attacks. In *Security Engineering: A Guide to Building Dependable Distributed Systems.* New York, NY, USA: John Wiley & Sons, Inc.

Anderson, R. J., Bond, M., Clulow, J., & Skorobogatov, S. (2006). Cryptographic Processors—A Survey. (pp. 357-369). IEEE.

Atsumi, S., Kondo, T., & Shona, Y. (1995). *Patent No. 5,442,165.* US.

Begley, J., & Scahill, J. (2015, February 19). Retrieved from The Intercept - The Great SIM Heist: https://theintercept.com/2015/02/19/great-sim-heist/

Chan, V., & Ho, F. (2003). *Patent No. 6588673 B1.* United States of America.

Chen, X. (2007). *Patent No. 1983466 A2.* European Union.

Common Criteria. (2002). Common Criteria Information Statement. Reuse of Evaluation Results and Evidence. Common Criteria.

Common Criteria. (2012). Common Criteria for Information Technology Security Evaluation Part 1 - 3; Version 3.1. Common Criteria.

Common Criteria. (2012, April). Common Criteria Supporting Document Mandatory Technical Document - Composite product evaluation for Smart Version 1.2. Common Criteria.

Demaertelaere, F. (2010). *Hardware Security Modules.* SecAppDev.org - Secure Application Development.

Graham, H. E., Jr., M. B., Nablo, R. J., & Haeuser, W. W. (2002). *Patent No. 6402028 B1.* United States of America.

Hautala, L. (2016, October 24). *Why it was so easy to hack the cameras that took down the web.* Retrieved from https://www.cnet.com/how-to/ddos-iot-connected-devices-easily-hacked-internet-outage-webcam-dvr/

Hautier, R., Macchetti, M., & Perrine, J. (2012). *Patent No. 2720167 A1.* European union.

IHS Markit. (2014, August). *Smart Card Shipments to Rise by 2.1 Billion Units by 2019.* (IHS Markit Newsroom) Retrieved October 2016, from http://press.ihs.com/press-release/design-supply-chain-media/smart-card-shipments-rise-21-billion-units-2019

ISO 10202-1. (1991). Financial transaction cards -- Security architecture of financial transaction systems using integrated circuit cards -- Part 1: Card life cycle. International Organization for Standardization.

ISO 18000-3. (2010). Information technology — Radio frequency identification for item management — Part 3: Parameters for air interface communications at 13,56 MHz. International Organization for Standardization.

Krebs, B. (2016, October 21). *kresbonsecurity.com*. Retrieved January 4, 2017, from https://krebsonsecurity.com/2016/10/hacked-cameras-dvrs-powered-todays-massive-internet-outage/

Manyika, J., Chui, M., Bisson, P., Woetzel, J., Dobbs, R., Bughin, J., & Aharon, D. (2015, June). *Unlocking the potential of the Internet of Things.* Retrieved from

http://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/the-internet-of-things-the-value-of-digitizing-the-physical-world

Markantonakis, K., Mayes, K., Sauveron, D., & Tunstall, M. (2010). Smart Cards. In H. Bidgoli (Ed.), *Handbook of Technology Management* (Vols. II: Supply Chain Management, Marketing and Advertising, and Global Management, pp. 248-264). Wiley.

Nest Labs. (2016). *Nest*. Retrieved 11 7, 2016, from https://nest.com

NIST FIPS PUB 140-2. (2001). *Security Requirements for Cryptographic Modules.* National Institute of Standards and Technology Federal Information Processing Standards Publications 140-2.

NIST SP 800-57-1. (2016). *Recommendation for Key Management - Part 1: General.* National Institute of Standards and Technology Special Publication 800-57-1.

PCI Security Standards Council. (2016). *Data Security Standard - Requirements and Security Assessment Procedures.* Payment Card Industry Security Standards Council.

Rankl, W., & Effing, W. (2010). *"Smart Card Handbook 4th Edition"*. Wiley.

Raschke, W., Massimiliano, Z., Baumgartner, P., Loinig, J., Steger, C., & Kreiner, C. (2014). Supporting evolving security models for an agile security evaluation. IEEE.

Sanchez-Reillo, R., Sanchez-Avila, C., Lopez-Ongil, C., & Entrena-Arrontes, L. (2002). Improving Security in Information Technology using Cryptographic Hardware Modules. *International Carnahan Conference on Security Technology* (pp. 120-123). IEEE.

Scahill, J. (2015, February 25). Retrieved from The Intercept - What Gemalto doesn't know what it doesn't know: https://theintercept.com/2015/02/25/gemalto-doesnt-know-doesnt-know/

Sinnhofer, A. D., Raschke, W., Steger, C., & Kreiner, C. (2015). Evaluation paradigm selection according to Common Criteria for an incremental product development.

Sinnhofer, A. D., Raschke, W., Steger, C., & Kreiner, C. (2015). Patterns for Common Criteria Certification. ACM.

Sinnhofer, A., Oppermann, F., Potzmader, K., Orthacker, C., Steger, C., & Kreiner, C. (2016). Patterns to Establish a Secure Communication Channel. *Proceedings of the 21st European Conference on Pattern Languages of Programs*, pp. 13:1-13:21.

Bibliography

[1] David J. Anderson. *Agile Management for Software Engineering: Applying the Theory of Constraints for Business Results*. Prentice Hall, Sept. 2003. ISBN: 0131424602.

[2] David M. Anderson. *Agile product development for mass customization. how to develop and deliver products for mass customization, niche markets, JIT, build-to-order, and flexible manufacturing*. Ed. by Joseph Pine II. McGraw-Hill [u.a.], 1997. ISBN: 0786311754.

[3] Ross J. Anderson. "API Attacks." In: *Security Engineering: A Guide to Building Dependable Distributed Systems*. 2nd ed. Wiley Publishing, 2008, pp. 547–557. ISBN: 9780470068526.

[4] Ross J. Anderson. "The Correctness of Crypto Transaction Sets (Discussion)." In: *Revised Papers from the 8th International Workshop on Security Protocols*. Springer-Verlag, 2001, pp. 128–141. ISBN: 3-540-42566-7. URL: http://dl.acm.org/citation.cfm?id=647218.720851.

[5] Felix Bachmann; Michael Goedicke; Julio Leite; Robert Nord; Klaus Pohl; Balasubramaniam Ramesh; and Alexander Vilbig. "A Meta-model for Representing Variability in Product Family Development." In: *Software Product-Family Engineering: 5th International Workshop, PFE 2003, Siena, Italy, November 4-6, 2003. Revised Papers*. Ed. by Frank J. van der Linden. Springer Berlin Heidelberg, 2004, pp. 66–80. ISBN: 978-3-540-24667-1.

[6] Elaine Barker. *NIST Special Publication 800-57 Part 1 Revision 4 – Recommendation for Key Management*. 2016.

[7] Per Bjesse. "What is Formal Verification?" In: *SIGDA Newsl.* 35.24 (Dec. 2005). ISSN: 0163-5743. DOI: 10.1145/1113792.1113794. URL: http://doi.acm.org/10.1145/1113792.1113794.

[8] Florian Böhl; Klaus Potzmader; Clemens Orthacker; Andreas Daniel Sinnhofer; and Christian Steger. "Method for Symbolic Execution on Constrained Devices." Patent 82086638 (82019770US01). Status: Application; App. No. 15/482462. Apr. 2017.

[9] Mike Bond and Ross Anderson. "API-Level Attacks on Embedded Systems." In: *Computer* 34.10 (Oct. 2001), pp. 67–75. ISSN: 0018-9162. DOI: 10.1109/2.955101. URL: http://dx.doi.org/10.1109/2.955101.

[10] Frank Buschmann; Regine Meunier; Hans Rohnert; Peter Sommerlad; and Michael Stal. *Pattern-Oriented Software Architecture - Volume 1: A System of Patterns*. Wiley Publishing, 1996. ISBN: 0471958697, 9780471958697.

[11] V. Chan and F. Ho. *Method and system providing in-line pre-production data preparation and personalization solutions for smart cards*. US Patent 6,588,673. July 2003. URL: https://www.google.com/patents/US6588673.

[12] X.S. Chen. *Method and apparatus of secure authentication for system-on-chip (SoC)*. EP Patent App. EP20,080,005,228. Aug. 2011. URL: https://www.google.com/patents/EP1983466A3?cl=en.

[13] Paul Clements and Lina M. Northrop. *Software Product Lines: Practices and Patterns*. Addison-Wesley Longman Publishing Co., Inc., 2001. ISBN: 0-201-70332-7.

[14] Payment Card Industry Security Council. *Payment Card Industry (PCI) Data Security Standard – Requirements and Security Assessment Procedures – Version 3.2*. PCI. 2016.

[15] Patrick Cousot and Radhia Cousot. "A gentle introduction to formal verification of computer systems by abstract interpretation." In: *Logics and Languages for Reliability and Security*. Ed. by Javier Esparza; Bernd Spanfelner; and Orna Grumberg. Vol. 25. NATO Science for Peace and Security Series - D: Information and Communication Security. IOS Press, 2010, pp. 1–29. ISBN: 978-1-60750-099-5.

[16] Patrick Cousot and Radhia Cousot. "Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints." In: *Proceedings of the 4th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*. POPL '77. ACM, 1977, pp. 238–252. DOI: 10.1145/512950.512973. URL: http://doi.acm.org/10.1145/512950.512973.

[17] Common Criteria. *Common Criteria for Information Technology Security Evaluation Part 1 - 3. Version 3.1 Revision 4*. CC. Sept. 2012.

[18] Common Criteria. *Common Criteria Information Statement – Reuse of Evaluation Results and Evidence*. CC. Oct. 2002.

[19] Common Criteria. *Supporting Document – Mandatory Technical Document – Composite product evaluation for Smart Cards and similar devices – Version 1.2*. CC. Apr. 2012.

[20] Marcelo Fantinato; Maria Beatriz Felgar de Toledo; Lucinéia Heloisa Thom; Itana Maria de Souza Gimenes; Roberto dos Santos Rocha; and Diego Zuquim Guimarães Garcia. "A survey on reuse in the business process management domain." In: *International Journal of Business Process Integration and Management* (2012).

[21] Dániel Fey; Róbert Fajta; and András Boros. "Feature Modeling: A Meta-Model to Enhance Usability and Usefulness." In: *Software Product Lines: Second International Conference, SPLC 2 San Diego, CA, USA, August 19–22, 2002 Proceedings*. Ed. by Gary J. Chastek. Springer Berlin Heidelberg, 2002, pp. 198–216. ISBN: 978-3-540-45652-0.

[22] Robert W. Floyd. "Assigning Meanings to Programs." In: *Program Verification: Fundamental Issues in Computer Science*. Ed. by Timothy R. Colburn; James H. Fetzer; and Terry L. Rankin. Springer Netherlands, 1993, pp. 65–81. ISBN: 978-94-011-1793-7. DOI: 10.1007/978-94-011-1793-7_4.

[23] Erich Gamma; Richard Helm; Ralph Johnson; and John Vlissides. *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Longman Publishing Co., Inc., 1995.

[24] Florian Gottschalk; Wil M. P. van der Aalst; Monique H. Jansen-Vullers; and Marcello La Rosa. "Configurable Workflow Models." In: *International Journal of Cooperative Information Systems* (2007).

[25] H.E. Graham; M.B. Kekicheff; R.J. Nablo; and W.W. Haeuser. *Integrated production of smart cards*. US Patent 6,402,028. Nov. 2002. URL: https://www.google.com/patents/US6402028.

[26] Ibrahim Mustafa Habli. PhD thesis. University of York - Department of Computer Science, 2009.

[27] Alena Hallerbach; Thomas Bauer; and Manfred Reichert. "Guaranteeing Soundness of Configurable Process Variants in Provop." In: *Commerce and Enterprise Computing, 2009. CEC '09. IEEE Conference on*. IEEE, 2009, pp. 98–105.

[28] Alena Hallerbach; Thomas Bauer; and Manfred Reichert. "Issues in modeling process variants with Provop." In: *Business Process Management Workshops*. Ed. by Danilo Ardagna; Massimo Mecella; and Jian Yang. Vol. 17. Lecture Notes in Business Information Processing. Springer Berlin Heidelberg, 2009, pp. 56–67.

[29] Helmut Hammer. *Towards an automated Golden Sample configuration process*. Bac.-Thesis. 2017.

[30] Michael Hammer and James Champy. *Reengineering the Corporation - A Manifesto For Business Revolution*. Harper Business, 1993.

[31] R. Hautier; M. Macchetti; and J. PERRINE. *Method and system for smart card chip personalization*. EP Patent App. EP20,120,188,097. Aug. 2014. URL: https://www.google.com/patents/EP2720167A1?cl=en.

[32] K. C. Kang; S. G. Cohen; J. A. Hess; W. E. Novak; and A. S. Peterson. *Feature-Oriented Domain Analysis (FODA) Feasibility Study*. Tech. rep. Carnegie-Mellon University Software Engineering Institute, Nov. 1990.

[33] Kyo C. Kang; Jaejoon Lee; and Patrick Donohoe. "Feature-Oriented Project Line Engineering." In: *IEEE Softw.* 19.4 (July 2002), pp. 58–65. ISSN: 0740-7459.

[34] James C. King. "Symbolic Execution and Program Testing." In: *Commun. ACM* 19.7 (July 1976), pp. 385–394. ISSN: 0001-0782. DOI: 10.1145/360248.360252.

[35] W. Maconachy; C. Schou; D. Ragsdale; and D. Welch. "A Model for Information Assurance: An Integrated Approach." In: *Proceedings of the 2001 IEEE Workshop on Information Assurance and Security*. 2001.

[36] James Manyika; Michael Chul; Peter Bisson; Jonathan Woetzel; Richard Dobbs; Jacques Bughin; and Dan Aharon. *Unlocking the potential of the Internet of Things*. 2015. URL: http://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/the-internet-of-things-the-value-of-digitizing-the-physical-world (visited on 2016-12-27).

[37] Kevin P. McCormack and William C. Johnson. *Business Process Orientation: Gaining the E-Business Competitive Advantage*. Saint Lucie Press, 2000.

[38] Daniel Mellado; Eduardo Fernández-Medina; and Mario Piattini. "A Common Criteria Based Security Requirements Engineering Process for the Development of Secure Information Systems." In: *Comput. Stand. Interfaces* 29.2 (Feb. 2007), pp. 244–253. ISSN: 0920-5489.

[39] Daniel Mellado; Eduardo Fernández-Medina; and Mario Piattini. "A Comparative Study of Proposals for Establishing Security Requirements for the Development of Secure Information Systems." In: *Computational Science and Its Applications - ICCSA 2006: International Conference, Glasgow, UK, May 8-11, 2006, Proceedings, Part III*. Ed. by Marina Gavrilova; Osvaldo Gervasi; Vipin Kumar; C. J. Kenneth Tan; David Taniar; Antonio Laganá; Youngsong Mun; and Hyunseung Choo. Springer Berlin Heidelberg, 2006, pp. 1044–1053. ISBN: 978-3-540-34076-8.

[40] Daniel Mellado; Eduardo Fernández-Medina; and Mario Piattini. "Towards security requirements management for software product lines: A security domain requirements engineering process." In: *Computer Standards & Interfaces* 30.6 (2008). Special Issue: State of standards in the information systems security area, pp. 361 –371. ISSN: 0920-5489.

[41] Committee on National Security Systems. *National Information Assurance (IA) Glossary*. CNSS. Apr. 2010.

[42] Object Management Group (OMG). "Business Process Model and Notation (BPMN). Version 2.0." In: (2011). available at http://www.omg.org/spec/BPMN/2.0/, pp. 1–538.

[43] Hubert Österle. *Business Engineering - Prozess- und Systementwicklung*. Springer-Verlag, 1995.

[44] Donn B. Parker. *Fighting Computer Crime: A New Framework for Protecting Information*. John Wiley & Sons, Inc., 1998. ISBN: 0-471-16378-3.

[45] Donn B. Parker. "Our Excessively Simplistic Information Security Model and How to Fix It." In: *Information Systems Security Association (ISSA) Journal* (July 2010), pp. 12–21.

[46] Klaus Pohl; Günter Böckle; and Frank J. van der Linden. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer-Verlag New York, Inc., 2005. ISBN: 3540243720.

[47] Postscapes. *2013/14 IoT Awards*. 2014. URL: http://www.postscapes.com/internet-of-things-award/ (visited on 2016-12-27).

[48] Klaus Potzmader; Clemens Orthacker; Andreas Daniel Sinnhofer; Christian Steger; and Christian Kreiner. "A runtime-configurable in-HSM secure IC intialization process." Patent –. Currently in a company internal review process. 2017.

[49] Wolfgang Rankl and Wolfgang Effing. *Smart Card Handbook*. 4th. Wiley Publishing, 2010. ISBN: 0470743670, 9780470743676.

[50] Srivaths Ravi; Anand Raghunathan; and Srimat Chakradhar. "Tamper Resistance Mechanisms for Secure, Embedded Systems." In: *Proceedings of the 17th International Conference on VLSI Design*. VLSID '04. IEEE Computer Society, 2004, pp. 605–611. ISBN: 0-7695-2072-3. URL: http://dl.acm.org/citation.cfm?id=962758.963491.

[51] Manfred Reichert; Alena Hallerbach; and Thomas Bauer. "Lifecycle Support for Business Process Variants." In: *Handbook on Business Process Management 1*. Ed. by Jan vom Brocke and Michael Rosemann. Springer, 2014.

[52] Marcello La Rosa; Marlon Dumas; Arthur H. M. ter Hofstede; Jan Mendling; and Florian Gottschalk. "Beyond Control-Flow: Extending Business Process Configuration to Roles and Objects." In: *27th International Conference on Conceptual Modeling (ER 2008)*. Ed. by Qing Li; Stefano Spaccapietra; and Eric Yu. Springer, 2008, pp. 199–215.

[53] Jerome H. Saltzer and Michael D. Schroeder. "The Protection of Information in Computer Systems." In: *Proceedings of the IEEE 63-9*. 1975.

[54] Jeremy Scahill. *Gemalto doesn't know what it doesn't know*. Feb. 2015. URL: https://theintercept.com/2015/02/25/gemalto-doesnt-know-doesnt-know/ (visited on 2017-02-18).

[55] Jeremy Scahill and Josh Begley. *The Great SIM Heist - How Spies Stole the Keys to the Encryption Castle*. Feb. 2015. URL: https://theintercept.com/2015/02/19/great-sim-heist/ (visited on 2017-02-18).

[56] Klaus Schmid; Karsten Krennrich; and Michael Eisenbarth. *Requirements Management for Product Lines: A Prototype*. Tech. rep. Fraunhofer - IESE, July 2005.

[57] Douglas C. Schmidt. *Why Software Reuse has Failed and How to Make It Work for You*. originally published in C++ Report magazine. Jan. 1999. URL: http://www.cse.wustl.edu/\~{}schmidt/reuse-lessons.html.

[58] Andreas Daniel Sinnhofer; Peter Pühringer; and Christian Kreiner. "varBPM — A Product Line for Creating Business Process Model Variants." In: *Proceedings of the Fifth International Symposium on Business Modeling and Software Design - Volume 1: BMSD 2015*. 2015, pp. 184–191. ISBN: 978-989-758-111-3. DOI: 10.5220/0005886901840191.

[59] Andreas Daniel Sinnhofer; Peter Pühringer; Klaus Potzmader; Clemens Orthacker; Christian Steger; and Christian Kreiner. "A Framework for Process Driven Software Configuration." In: *Proceedings of the Sixth International Symposium on Business Modeling and Software Design - Volume 1: BMSD 2016*. 2016, pp. 196–203. ISBN: 978-989-758-190-8. DOI: 10.5220/0006223701960203.

[60] Andreas Daniel Sinnhofer; Andrea Höller; Peter Pühringer; Klaus Potzmader; Clemens Orthacker; Christian Steger; and Christian Kreiner. "Combined Variability Management of Business Processes and Software Architectures." In: *Proceedings of the Seventh International Symposium on Business Modeling and Software Design - Volume 1: BMSD,* INSTICC. SciTePress, 2017, pp. 36–45. ISBN: 978-989-758-238-7. DOI: 10.5220/0006527300000000.

[61] Andreas Daniel Sinnhofer; Wolfgang Raschke; Christian Steger; and Christian Kreiner. "Evaluation paradigm selection according to Common Criteria for an incremental product development." In: *International Workshop on MILS: Architecture and Assurance for Secure Systems* 1 (2015). Available at http://mils-workshop-2015.euromils.eu/, pp. 1–5.

[62] Andreas Daniel Sinnhofer; Peter Pühringer; Klaus Potzmader; Clemens Orthacker; Christian Steger; and Christian Kreiner. *Identification of business drivers using traceable links between process models and software architectures*. Working title; Accepted for publication in a LNBIP series in 2018.

[63] Andreas Daniel Sinnhofer; Wolfgang Raschke; Christian Steger; and Christian Kreiner. "Patterns for Common Criteria Certification." In: *Proceedings of the 20th European Conference on Pattern Languages of Programs*. EuroPLoP '15. ACM, 2015, 33:1–33:15. ISBN: 978-1-4503-3847-9. DOI: 10.1145/2855321.2855355. URL: http://doi.acm.org/10.1145/2855321.2855355.

[64] Andreas Daniel Sinnhofer; Felix Jonathan Oppermann; Klaus Potzmader; Clemens Orthacker; Christian Steger; and Christian Kreiner. "Patterns to establish a secure communication channel." In: *Proceedings of the 21st European Conference on Pattern Languages of Programs*. EuroPLoP '16. ACM, 2016, 33:1–33:20. ISBN: 978-1-4503-4074-8. DOI: 10.1145/3011784.3011797. URL: http://doi.acm.org/10.1145/2855321.2855355.

[65] Andreas Daniel Sinnhofer; Peter Pühringer; Klaus Potzmader; Clemens Orthacker; Christian Steger; and Christian Kreiner. "Software Configuration Based on Order Processes." In: *Business Modeling and Software Design: 6th International Symposium, BMSD 2016, Rhodes, Greece, June 20–22, 2016, Revised Selected Papers*. Ed. by Boris Shishkov. Springer International Publishing, 2017, pp. 200–220. ISBN: 978-3-319-57222-2. DOI: 10.1007/978-3-319-57222-2_10. URL: http://dx.doi.org/10.1007/978-3-319-57222-2_10.

[66] Andreas Daniel Sinnhofer; Felix Jonathan Oppermann; Klaus Potzmader; Clemens Orthacker; Christian Steger; and Christian Kreiner. "Where do all my keys come from." In: *Handbook of Research on Solutions for Cyber-Physical Systems Ubiquity*. Ed. by Norber Druml; Andreas Genser; Armin Krieg; Manuel Menghin; and Andrea Hoeller. IGI Global, 2017, pp. 278–300. DOI: 10.4018/978-1-5225-2845-6.ch011.

[67] Paolo Spagnoletti and Andrea Resca. "The duality of Information Security Management: fighting against predictable and unpredictable threats." In: *Journal of Information System Security* 4.3 (2008), pp. 46–62.

[68] International Organization for Standardization. *Financial transaction cards – Security architecture of financial transaction systems using integrated circuit cards – Part 1: Card life cycle*. ISO. 1991.

[69] Ambrosio Toval; Joaquín Nicolás; Begoña Moros; and Fernando García. "Requirements Reuse for Improving Information Systems Security: A Practitioner's Approach." In: *Requirements Engineering* 6.4 (2002), pp. 205–219. ISSN: 1432-010X.

[70] Michael Tunstall; Kostas Markantonakis; Damien Sauveron; and Keith Mayes. "Smart Cards." In: *Handbook of Technology Management*. Ed. by H Bidgoli. John Wiley & Sons, 2009.

[71] George Valença; Carina Alves; Vander Alves; and Nan Niu. "A Systematic Mapping Study on Business Process Variability." In: *International Journal of Computer Science & Information Technology (IJCSIT)* (2013).

[72] Armin Wasicek. *Security Considerations in Embedded Systems*. Research Report 108/2006. Technische Universität Wien, Institut für Technische Informatik, 2006.

[73] David M. Weiss and Chi Tau Robert Lai. *Software Product-line Engineering: A Family-based Software Development Process*. Addison-Wesley Longman Publishing Co., Inc., 1999. ISBN: 0-201-69438-7.

[74] Peter Willaert; Joachim Van Den Bergh; Jurgen Willems; and Dirk Deschoolmeester. *The Process-Oriented Organisation: A Holistic View - Developing a Framework for Business Process Orientation Maturity*. Springer, 2007.

[75] Feng Xia; Laurence T. Yang; Lizhe Wang; and Alexey Vinel. "Internet of Things." In: *International Journal of Communication Systems* 25.9 (2012), pp. 1101–1102. ISSN: 1099-1131. DOI: 10.1002/dac.2417. URL: http://dx.doi.org/10.1002/dac.2417.