Abstract—Nowadays, software development is a more complex process than ever was and it faces the challenges, where security became one of the most crucial. The security issues became an essential part of software engineers and understanding the vulnerabilities, risks and others became the everyday bread. The needs of security in software development resulted in the creation of the so-called Secure Software Development Life Cycle (SSDLC). This is a methodological concept included in classical Software Development Life-Cycle, which is described by five main phases - analysis, design, implementation (building), testing, and evaluation (deployment and maintenance). The SSDLC adds another dimension ensuring the security. We introduce our same named tool "Secure Software Development Life-cycle", which follows the general idea and goes beyond it. Our tool helps to create security, hardening, testing, and validation reporting guidelines for selected use-cases. This tool is an environment for defining the current and future security requirements based on the collection of standards, recommendations, best practice, and many others. Connecting the SSDLC with other tools improves the general level of automation of the Product Life Cycle (PLC). The SSDLC gives a connection and context among security, safety and performance parameters. Compared with static security requirements definition, the SSDLC provides simple future extension and straight integration to the PLC process with non-or nearly-non personal (human) interaction.

Index Terms—Security, Software development lify cycle, Development, Software engineering, Management

I. INTRODUCTION

The product development evolved continuously over time together with the new technological phenomena. Nowadays, the product life cycle (PLC) or product life cycle (PLC) management is a business activity of managing, in the most efficient way, company’s products all the way across their life cycles; from the very first idea for a product all the way through until it is retired and disposed off [1]. One of today’s most important parts of PLC processes is without doubts the software development. Over the past years, many different languages including multi-language programming [2], cloud computing [3], micro-services [4], new hybrid programming techniques [5] as well as inclusion of new phenomena such as artificial intelligence [6], machine learning [7], advanced modeling [8] and simulation [9] have changed the area of software development. The process became very complex including multiple stages, which ensures the required quality of final product [10] (see Fig. 1). Requirement analysis often contains also requirement gathering, analysis and it is focused on answering basic questions such as who will use the system, how the system will be used, what are the inputs/outputs and others [11]. Design phase includes definition of specific hardware or software requirements together with overall system architecture design [12]. Development and coding or often also called as implementation phase contains divided modules or units, which are one by one implemented to meet the requirements [13]. Testing phase is actual check whether or not the requirements are met [14]. Deployment phase starts after successful testing in previous stage and it is already delivered to the customers [15]. Operation and maintenance includes actions, which comes from real usage of the system, which was not possible to discover during the test phase [16].

Fig. 1. Multiple stages recognized in Software Development Life Cycle (SDLC).

Nowadays, the software products serve our everyday tasks, but also ensure the functionality of the most crucial applications such as hospital systems [17], critical infrastructure [18], banking systems [19], communication and data infrastructure [20], transportation [21], and many others. Therefore, the security became one of the crucial parts of developing reliable software products [22]. Moreover, the more complex environment increases also the possibility of software bugs, mistakes, vulnerabilities, malfunctions and others [23]. In this paper, we focus on the security of SDLC together with related challenges and issues. Moreover, we introduce our tool, which helps in the software product development stages to improve cooperation and provide clear guidance on security requirements necessary for creating reliable applications.

The rest of this paper is organized as follows: Section II introduces the current state of the art for secure software development life cycle, including the different approaches and tools. Further, Section III provides description of our tool dealing with secure software development life cycle. Its main characteristic follows in Section IV, which summarizes via examples several use-cases, where our software was used and how it helps to deal with product life cycle stages. Finally, Section IV summarizes our conclusions, findings, and suggests future work.
II. STATE OF THE ART

There has been already high number of software engineering surveys as reported in [24], which summarized more than 25 survey works connected with software development worldwide as well as Turkey region in particular. The most interesting findings are: (i) Most commonly used standards for SDLC are ISO 9000 and CMMI (Capability Maturity Model Integration); and (ii) most effort is focused on development. However, a standardized approach to software engineering is still not a full warranty of reliable product. Therefore, the Secure SDLC (SSDLC) come-up and currently there are several different standards and guidance, which are applicable to it such as [28]–[32]: OWASP, ISO/IEC (such as 13335, 13849, 21434, 26262, 27002, or 27034), NIST, IEC standards (such as 61580 or 62443), and many others. The Secure SDLC has many different approaches, but we can summarize it as another layer to standard SDLC (see Fig. 2).

The SSDLC adds other tasks to the already current stages of SDLC. There are two different approaches [37]–[39]: (i) Proactive approach (Preventing all possible flows and breaches at the very beginning of the project and implementing solutions in a secure way on required level) and (ii) Reactive Approach (Ensure security before release and maintain it through the product’s existence). Proactive approach should be always considered as it brings several advantages over reactive approach. The highest benefit of proactive approach is finding a issues or bugs early in the SDLC processes (up to development stage) before release. This of course save valuable resources and time to fix the founded issue. Moreover, it is also much easier to correct bugs when the software is not yet implemented. Last but not least, the considered training of employees or staffs on security practices is valuable investment, which might solve many issues even before the project starts. On the other hand, the reactive approach includes additional testing due to the later stages of SDLC or even after everything has already been built (when security needs to be implemented retroactively). This approach adds another tasks to the developers and testers such as penetration testing, dynamic code analysis, vulnerability scanning, incident response plan and others. This approach adds over the proactive approach possibility to prevent all possible flows and breaches, which were not discovered or was overseen in the early stage of SDLC. However, already complex SDLC became with these even more complex SSDLC with more tasks, more decision issues, and more management load. Therefore, the management approach for SSDLC is required.

III. MANAGING THE SECURE SDLC

The management approach brings another level to the SDLC and improves it. In particular, our Management SSDLC tool (mSSDLC) brings:

- general security controls for information systems for effective risk management;
- flexible catalogue of security controls to meet the security threats, requirements and technologies;
- measurement metrics for security control effectiveness.

The most important from these points are:

- Integration of security activities into the application development process (creation of security requirements, security design analysis, guidelines for developers, code review, penetration tests and others).
- Depending on the business criticality of the developed application, the appropriate security engagement in development is made to ensure that security activities do not unnecessarily disturb the development of the application.
- Based on generally accepted standards and best practice (NIST FIPS, STIG, Microsoft SDL, CIS Benchmarks) to treat the various attack vectors with even quality.

- Ability to integrate security activities into different developmental models (especially Waterfall, Agile development).

Using the results from the interviews and documents analysis, we have developed a secure SDLC integration program, mSSDLC, including recommended policies, guidelines, and knowledge transfer. Secure software development has three elements best practices, process improvements, and metrics of software development process that focuses on these elements. The goal is to minimize security-related vulnerabilities in the design, code, and documentation and to detect and eliminate vulnerabilities as early as possible in the development life cycle. The simplified block diagram of mSSDLC is shown in Fig. 3.

The main added values for SSDLC stages are:
• Requirements and Analysis - Over functional, non-functional and technological requirements it is also added security objectives (or requirements).
• Architecture and Design - adding security requirements (security is part of the architectural design) and creating security requirements check-list. Moreover, promoting the visibility of security - Review designs for possible security issues that means to develop mitigation of all threats. Finally, adding baseground for risk analysis and threat modelling, security requirements, privacy requirements, architecture and design review for security, design guidelines for security and threat modelling.
• Development - It helps to minimize the security issues and vulnerabilities, by adding clear roles, tasks and controls defined for unit tests and Code-review.
• Testing - During the Verification phase, the tool helps to ensure that the code meets the security and privacy tenets established in the previous phases. This is done through security and privacy testing, and a security push, which is a team-wide focus on threat model updates, code review, testing, and thorough documentation reviewing and editing. A public release privacy review is also completed during the Verification phase.
• Deployment - After implementation, there is supposed to be a verification phase for the real devices. Security audit is an expensive approach and should be made after the final tests. This might be done by fast security evaluation based on the quantitative analysis of the final product, which should be provided by the TP methodology.

Last but not least, the main advantages to the general SSDLC approach, which the tool offers, are:
• Identifying Security Objectives - To understand key security objectives and scenarios within the SDLC stages.
• Security Design Guidelines - Create guidelines, specify security requirements, architecture and design review for security purposes.
• Threat Models - Identifying threats, attacks and vulnerabilities for specified information system, with countermeasures.
• Assessment - Penetration tests, reviews and configuration audits; simulation of attacks and scenarios to uncover weaknesses and vulnerabilities during the development and in deployment.

IV. APPLICATION AREAS

The mSSDLC helps in many different areas with giving the management context to SSDLC. We are bringing one particular use-case, where mSSDLC was used with a lesson learned. The use-case might be described as follows: Early requirements/design PLC stage of a critical control system for asynchronous motor, where the most important parameter was the performance (i.e., fast stop command delivery etc.). However, the cyber-security standards IEC 62443 brought the security questions and new requirements, which must be faced in the PLC stages. The gathered security requirements from the specification and best practice were included in the mSSDLC and used for another tasks of SSDLC such as final minimum security requirements for design phase, trade-off analysis and decision process. The simplified block-diagram is shown in Fig. 4.

![Fig. 4. Simplified block-diagram for selected use-case.](image)

The WEFACT\(^1\) was a static database in the project, which was automatically actualized with defined requirements from SSDLC and validated. Moreover, the uniform UML/XML format was used as an input in the simulation and modeling tools - TTool\(^2\) and CHESS\(^3\). The minimal security requirement provided from the mSSDLC was used to simulate the impact on the performance (particularly to get the delay growth). Thanks to the early-stage analysis via mSSDLC and management approach the early-stage gets sufficient information for pre-simulations, which gave enough valid information for the decision making process and saved time as well as costs. Without the mSSDLC, the whole early-stage pre-simulation would not be possible and the right decision would be very hard to make with high probability to make wrong decision, which would increase future costs and time for application development.

V. CONCLUSION

We bring the very current state of the art for SDLC (Software Development Life Cycle) as well as for SSDLC (Secure Software Development Life Cycle). We introduce both approaches, highlight the challenges and bring-up the new topic of management approach in the SSDLC together with introduction of our mSSDLC tool. The whole development is unfortunately not anymore just an engineering work. The project management plays crucial role and helps to move forward in the right direction through the development life-cycle. As mentioned, there are different teams involved with many decisions, which need to be made thought the whole SDLC, before the product hits the market [25]. The early stage of SDLC is about creating requirements, which must be fulfilled by the final product and even at this early stage we must fight the different areas of requirements, which

---

\(^1\)WEFACT is Workflow Engine for Analysis, Certification and Test, which is commonly used in the industrial use-cases.

\(^2\)TTool is SysML modeling tool, which implements Security and Performance analysis for many different applications.

\(^3\)CHESS is Composition with Guarantees for High-integrity Embedded Software Components Assembly and it provides a model-driven, component-based methodology.
often ends in performance versus security trade-off [27]. However, the industrial applications, medical applications, air-space applications, transportation applications include another dimension of the requirements - safety, which must be also taken into account, and which makes the SDLC even more complex [26]. In this paper, we introduce the two-dimensional approach, where performance is not anymore the only issue, but also how the security requirements enter the early stage of SSDLC. However, the issue is much more complex and it is unfortunately beyond the scope of this paper. Therefore, this is also a discovered challenge, which should be focused on in the future research.

VI. ACKNOWLEDGEMENT

This article has received funding within the National Sustainability Program under grant L01401 and Technology Agency of the Czech Republic under grant no. TJ01000381. Our research and the idea of the paper is coming from the research conducted and supported by research project Aggregated Quality Assurance for Systems (AQUAS H2020-EU.2.1.1.7 ID: 737475). For the research, the infrastructure of the SIX Center was used.

REFERENCES