Embedded Software Testing Issues and Addressing them with Software Product Lines Paradigm

Iurii Ierofeiev[1,†], Igor Sinitsyn[1,*,†] and Olha Slabospitska[1,†]

[1] *Institute of Software Systems of the NAS of Ukraine 40, University Academician Glushkov Avenue, 40, Kyiv, 03187, Ukraine*

Abstract

The paper declares a novel approach for Embedded Software (ES) development based on Embedded DevSecOps enhancing with Software Product Line paradigm. While retaining DevSecOps benefits to cope its everyday challenges an approach firstly focuses on the new ones that (non)anticipated diversity, interdependence and volatility of DevSecOps environment pose in today's rapidly evolving technological landscape, such as comprehensive Variability managing and Reuse enabling.

To this end an approach prescribes: consider end-to-end ES Continuous Testing over DevSecOps CI/CD pipeline as its Foundation mostly assuring ES quality; fix its key blocking Issues; represent it as a feedback-based series of the stages where regular ES testing procedures known as X-in-Loops should be executed with simulator/emulator; finally, each stage explicit modeling in a Test driven development style i.e. as a process of dedicated Test Product Line (TPL) engineering up to ISO/IEC 26554 while assuming Embedded DevSecOps process to be in turn modeled as a target ES Product line (SPL) engineering up to ISO/IEC 26550. Each of successive TPL modeling should be tightly aligned with all TPLs and SPL assumptions just defined and with current refinement of SPL assumptions to enable it drive adjoining TPL

To justify the approach proposed two results are presented of applying it to ES Continuous Testing sub-process being performed on host at the very beginning of Software in open Loop (CT) just producing first ES code for it. The first is sound (i. e. compliant, transparent, ES Quality-focused and automated) CT's Technological Model composing its Context, Parameters, Sub-processes. Pilot Kit for CT automation combining proven tools for the minimal set of necessary TPL processes tasks up to the Model proposed is thus the second one. Drafting the Model proposed benefits and future work to refine it concludes the paper.

Keywords

embedded software, DevSecOps, continuous testing, CI/CD pipeline, Product Line, kit, model

1. Introduction

In today's rapidly evolving technological and social landscape conventional challenges of Embedded Software (ES) Development being adequately coped with Embedded DevSecOps dedicated practices [1, 2, 3, 4] are complemented with the new ones that ever growing (non)anticipated diversity, interdependence and volatility of DevSecOps environment currently cause such as [2]:

- proper all-leveled and all-aspect Variability management over DevSecOps Embedded CI/CD Pipeline environment
- directed reuse of ES with their (non)executable components/tests and Pipeline's workflows, its infrastructure-as-a-code fragments, test harnesses elements

Workshop "Software engineering and semantic technologies" SEST, co-located with 15th International Scientific and Practical Programming Conference UkrPROGP'2025, May 13-14, 2025, Kyiv, Ukraine[*]

[*] Corresponding author.

[†] These authors contributed equally.

seo@erofeev.com.ua (I.Ierofeiev); ips2014@ukr.net (I.Sinitsyn); ols2014@ukr.net (O.Slabospitska)

0009-0006-8985-2729 (I.Ierofeiev); 0000-0002-4120-0784 (I.Sinitsyn); 0000-0002-9443-4154 (O.Slabospitska)

- safety and secure managing complex and highly distributed ES development environment.

Recognizing Embedded DevSecOps insufficiency to cope them both academia [5] and practitioners [6] just consider Software Product Line paradigm as its perspective complement.

The purpose of the paper is two-fold. First, in line with ideas of [5, 6] it presents a novel approach for Embedded DevSecOps-based ES Development enhancing with System and Software Product Line paradigm up to ISO/IEC 265xx series as a remedy to effectively and efficiently address both above-mentioned new ES DevSecOps challenges and conventional ones. To this end ES Continuous Testing within DevSecOps CI/CD pipeline is proposed to consider as a feedback-based series of the stages where regular ES testing procedures known as X-in-Loops [7] should be executed and step-by-step modeling each stage in a Test driven development style i.e. as a service process of dedicated Test Product Line (TPL) engineering up to ISO/IEC 26554 while assuming Embedded DevSecOps process to be in turn coherently modeled as a target ES Product line (SPL) engineering up to ISO/IEC 26550 that drives adjoining TPL and wraps it on .

Secondly, the paper sums up the results of the author's attempt to justify their approach. It describes draft Technological Model for ES Continuous Testing critical sub-process (CT from now on) that should be performed on host at the very beginning of Software on open Loop stage [7] to produce the first (or just changed to fix the bugs) ES code for it with appropriately high quality.. Moreover, pilot Toolkit to automate the Model's usage is also presented.

2. ES Continuous Testing: Issues and Approach to Address them

The process of end-to-end Software Continuous Testing over CI/CD pipeline is introduced with ISO/IEC/IEEE 32675 as implementation of core DevOps Principle "Left shift and continuous everything". Within Embedded domain it purposes at constantly meeting unique requirements for ES safety, security, privacy, reliability, performance and extremely long life cycles thus becoming the Foundation of DevSecOps. As such, being well-designed and implemented, its key promising features are as follows:

- Just it should ensure both ES appropriate quality and its generation efficiency (speed, security, and coordination among Dev, Sec, Ops and Test teams) enabling its usage within target system as expected [2].
- It should implement feedback among ES versions within a single build (in single Embedded CI/CD pipeline round) as well as traceability and controllability among ES versions of different builds (across multiple rounds) thus implementing Variability Management for ES and their (non)executable artifacts.
- It have to highlight defects at Continuous Integration, Continuous Deployment and Operation stages; concerning information obtained after ES careful static analysis [8], particularly Advanced Static Application Security Testing (SAST) [8, 9, 10] for (non-)directed fuzzing, followed further by automated (and appropriate manual) testing of safety and security, reliability, compatibility, performance, load, maintainability, and portability.
- It should set up and maintain appropriate conceptual and information environment for Embedded DevSecOps.

Realizing the above promises could however be blocked with key Issues. Some of them are largely originated in embedded systems overarching architecture and inherent complexities of their design and deployment [1] and aren't thus considered in the paper. But others are inherent ES Continuous Testing Issues especially addressed further concerning ES CT with Product Line Paradigm.

These are listed beneath.

1. Common conventional processes and techniques for dynamic testing as prescribed with ISO/IEC/IEEE 29119-3 and ISO/IEC/IEEE 29119-4 effective usage within Embedded domain.

2. ES Continuous Testing assets effective and efficient configuration, management and controllable reuse both over single Embedded DevSecOps rounds and among them.
3. Full-aspect traceability enablement among ES Continuous Testing assets themselves on the one hand and over DevSecOps process as a whole on the other one.
4. ES testsing on embedded targets from Software in the Loop to Hardware (or in automotive domain even vehicle) in the Loop Initiation, Monitoring and Management.
5. Due ES quality, especially Safety and Security, timely and carefully monitoring, reporting and assuring.
6. Proper responsibilities division and control among various Test and Dev, Sec, Ops teams.
7. Current maturity assessment of ES Continuous Testing process and gain insights concerning the needs, aspects and ways to increase it.

As far as proper ES Continuous Testing process should be the Foundation for Embedded DevSecOps its intrinsic above-listed Issues 1-7 become also crucial for DevSecOps too thus demanding be effectively and efficiently addressed. Analysis of Embedded domain [1, 2, 4, 7] identifies Model Driven Engineering as a top-level meta-paradigm for ES development and also two most promising paradigms within it beyond Embedded DecSecOps, namely System and Software Product Line [5, 6] (as defined with ISO/IEC 265xx series) and Microservice Architecture, preferably with Docker [1] compared in Table 1. As Table 1 shows they partially complement each other in addressing the Issues 1-7 above over different DevSecOps stages. The both could also refine it while Microservice Architecture especially increases Continuous Integration efficiency and its builds quality with proper Docker-based containers [1]. Moreover, automated tests over ES Continuous Testing could be distributed across separate containers and executed independently thereby enabling incremental ES improvement [2].

Table 1
Comparing Product line and Microservice Architecture for ES Engineering

Paradigm	Key Idea	Benefit	Limitation	Use case
Software product line	Focus on Reusability	Embedded ESe Quality and Consistency	Initial Implementation Cost Variations Maintaining Complexity	Automotive electronics featuring model variations
Micro-services architect-ture	Collection of independent micro-services	Efficient resources usage Agility, scalability, and maintainability	Not deterministic Data consistency, synchronization, integrity, Complex design	Consumer devices like weather station Control units: solar panels, gas, fuel

Product Line paradigm with appropriate containerization should thus be effective remedy to enhance Embedded DevSecOps with addressing ES Continuous Testing intrinsic Issues 1-7 above.

3. ES Continuous Testing Process: Sound Technological Model

With above-mentioned ultimate purpose in mind of Embedded DevSecOps enhancing with Product Line and Microservice Architecture, proper ES CT process is proposed to introduce in Test driven development style [1] but under PL context. More specifically, this process is modeled as a couple of Testing sub-processes of target SPL Domain and, respectively, Application Verification and Validation, on the one hand, and simultaneously as SPL— driven process for TPL engineering. Common concepts necessary to define the process at hand in such a way are delineated beneath to use hereafter.

The first such concept is Embedded DevSecOps. It could be considered as two-stepped refinement of conventional DevOps defined with ISO/IEC/IEEE 32675-2022 as a set of principles and practices which enable better communication and collaboration between relevant stakeholders for the purpose of specifying, developing, and operating software and systems products and services, and continuous improvements in all aspects of the life cycle. — for further delivering and maintaining higher-quality ES overcoming the challenges above. It brings together three core fundamentals: development, operations, quality assurance [1].

ES safety and security become a next-gen challenge that transforms Embedded DevOps [1] into Embedded DevSecOps. Secure ES requires early potential threats identification to define appropriate security requirements that guide ES development from architecture to coding to operation within target system. In other words, security must be embedded into ES from the very beginning, as an integral component of overall target embedded system quality.

Embedded CI/CD pipeline is thus the second

The third one is PL being defined with ISO/IEC 26550 as a set of products and/or services sharing explicitly defined and managed common and variable features and relying on the same domain architecture to meet the common and variable needs of specific markets. ISO/IEC 26550 also fixes PL engineering peculiarities useful hereafter such as:

- Problem and Solution spaces being formed with anticipated common and variable domain products' features (that are referred to as commonality and variability) together with the rules of composing them for a product and, respectively, with reusable assets (both non-executable and software) together with corresponding rules of configuring assets for terminal or compound features implementing within a sample product.
- Problem space de-facto standardized representation with Feature Model (FM) being defined as a tree of commonalities and variabilities with their additional dedicated interrelations.
- In turn, Solution space de-facto standardized representation with Platform being defined as a PL architecture, a configuration management plan and domain assets enabling Application engineering to effectively reuse and produce a set of derivative products.
- Domain and Application engineering lifecycles where the above reusable resources are engineered (for reuse) and, respectively, iteratively configured within sample products with commonalities and anticipated variabilities (with their respective Verification and Validation and also Realization processes where ES development specificity will be primarily implemented).
- Both Organizational and Technical management process groups with the process of PL Variability management within the second one where Variability is considered as the ability of PL's product or artefact to be extended, changed, customized or configured for use in a specific context.

To cope ES Continuous Testing Issues 1—7 from Section 2 let's define CT (being considered as earliest and simplest sub-stage) with its dedicated Technological Model trough three successive steps. First, assume ES is being developed for various hardware systems with explicitly define common and variable characteristics as an extensible set of software products possessing hardware data—driven common and variable features with appropriate PL (SPL) where:

1. SPL engineering process is constituted with the series of PL environment initial Set-up and further Reinvention rounds with its FM and Platform initial defining and then evolving (where PL products creating is prohibited) that interchange with unified production rounds (where FM and Platform are conversely fixed and options are especially provided to create ES up to them). Just this round is considered further in the article as a context of TPL defining within process perspective.
2. Unlike ISO/IEC 26550, SPL Domain Engineering and Application Engineering life cycles are implemented by means of appropriate Domain and Application Embedded DevSecOps

cyclic processes being synchronized with Organizational and Technical Management processes that are however left conventional. In particular, Domain and Application Realization processes are implemented by means of Embedded CI/CD pipelines from corresponding DevSecOps processes.

Secondly, list requirements for the Model to make it constructive. These are inspired with the requirements [8, 9] or reliable analysis of ES code data and control flows as follows.

1. Compliance — applicability for any ES target PL and CI/CD pipelines implementation technologies, test automation strategy and Dev, Sec, Ops, Test teams responsibilities and skills.
2. Transparency — clear representation of CT's tasks, tests and artifacts being produced as their solutions, teams' roles producing them and all the traceability links between and target ES PL artifacts.
3. Focus on ES Quality — enablement tests for compliance checking with critical ES Quality characteristics, foremost Safety and Security, Performance efficiency, Reliability, Compatibility, Maintainability and, not compromising, Functional suitability prescribed with ES quality model adopted (ISO/IEC 25010 by default or its refinements e. g. Z.Tamrabet's ESQuMO).
4. Lean automatability — acceptability any test automation strategy (from classical R. Martin's Test Pyramid to DevOps Hourglass to Spotify's Honeycomb) and tools (from online and opensource tools to proprietary software).
5. CT estimability — enablement its sound, informed and consistent maturity profile assessment in accordance with TPI Next [11] industrial testing maturity model.

Lastly, to meet the requirements listed, seamlessly combine industrially proven best practices and techniques from ES Engineering and business-driven industrial testing domains, namely:

- Embedded DevSecOps process and its CI/CD pipeline effective patterns [1]
- ISO/IEC 26554 Product line testing reference model
- Advanced static analysis of (non) executable DevSecOps artifacts [8, 9, 10]
- Embedded (non)directed fuzzing, both forward and backward [4, 12]
- Common conventional techniques for software testing levels from unit to system being defined with ISO/IEC/IEEE 29119-4 and their model-based, foremost with state charts enhancements for exactly ES testing [3]
- Performing and Organizing Topics of TMap Body of knowledge covering Industrial testing best practices to reconcile the rest highly divergent constructs [13].

Resulted high-level CT's Model is defined from the process perspective to be three-leveled structured triple:

$$CTM = \langle CNT; PAR; PRC \rangle;\ CNT = \langle RL, DO, TT \rangle; PAR = \langle ST, Ql, MT, TL, RR, AL \rangle; \quad (1)$$

$$PRC = \langle \langle DT, AT \rangle; \langle TM, AM, VM \rangle; EN \rangle; DT = \langle DD, DS, DE, DR \rangle; AT = \langle AD, AS, AE, AR \rangle; \quad (2)$$

$$TM = \langle SD, PD, PL, MC \rangle; VM = \langle VC, VA, TV \rangle; EN = DAS \cup AAS, \quad (3)$$

where CNT is CT's context — organizational RL, developmental DO and methodical TT;

PAR denotes CT's parameters proposed to tailor it up to CNT, namely CT's Strategy ST and extensible sets of quality characteristics QL to be necessarily tested, adopted testing techniques MT and automated tools TL, TPL reinvention rules RR and CT's $TL-$based automation levels AL;

PRC is a structured sextet (2) of TPL generic processes and TPL information environment EN where they should execute that includes Domain Testing DT and Application Testing AT, Test

Management TM, Asset Management in Testing AM and Variability Management in Testing VM as defined with ISO/IEC 26554 while being enhanced with appropriate reference CT workflows using advanced static analysis and embedded fuzzing for CI/CD pipelines within Domain and Engineering DevSecOps being assumed [14];

EN composes domain assets DAS and application assets AAS that DT and AT (2) processes produce with own internal structure (presented further in Figure 1);

$$DAS = TF \cup PS \cup PD \cup CS \cup CD \cup DRP; AAS = RC \cup AS \cup AD \cup ARP \cup TMP, \quad (4)$$

where TF fixes TPL features with their interrelations thus capturing its variability;

PS and PD contains patterns for static analysis of SPL Platform elements and respectively autonomous tests and scenarios for dynamically testing them thus constituting TPL Platform;

CS, CD are the same as PS, PD but for admissible configurations of SPL Platfom elements;

RC fixes rules for reuse of DAS, AS, AD elements within Application Testing;

DRP and ARP store reports being produced with DT and AT;

TMP contains CT maturity profiles up to TPI Next [11].

At the second TM's level its constituents from (1) are further detailed as follows:

RL is an extensible set of paired personal and team roles of CT's actors from Dev (tester, programmer, analyst, Tec writer, manager by default), Quality Assurance (leader, analyst, technician by default), Security (CISO, analyst, technician by default) and Operations (CIO, analyst, technician by default);

DO fixes an extensible set of Embedded DevSecOps with CI/CD pipeline proven patterns where J. Beningo's "ideal" CI/CD pipeline [1] is a core element by default;

TT combines Organizing and Performing topics from Sogety's TMap Body of knowledge [11];

ST is proposed to define as a four-dimensional relation that enables explicit responsibility distribution among CT's actors;

$$ST \subseteq RL \times M \times \{u, i, s\} \times \{d, a\}, M \supseteq \{sa, sr, ir, ia\}, \quad (5)$$

where extensible M set includes reference modes of testing being performed by actors with roles from RL at unit (u), integration (i) and system (s) levels within Domain engineering(d) or Application one (a) that by default are: product by product testing with tests immediately generating (sa); product by product testing with tests proactively generating as a Domain engineering reusable assets (sr); incremental testing where the first product is tested with sa or sr mode but for the next only tests for new features are generated while for the features yet previously tested existing tests are used as reusable assets of Domain engineering (ir) or Application one (ia);

AL contains initial level (where only open source, free trial and as-a-service solutions should be used along with initial requirements for automation kit eliciting), interim level (additional subscription solutions and requirements refining) and the last level of full CT's automation with proprietary and custom tools meeting elicited requirements that altogether are the core.

In turn, for CT's sub-processes P from (2), (3) unified view is proposed focusing at P's tasks:

$$P = \langle TP, WP, EP \rangle; P \in \{DD, DS, DE, DR, AD, AS, AE, AR, SD, PD, PL, MC, VC, VA, TV\}, \quad (6)$$

where TP just lists these P's tasks;

WP includes workflows of P's operations being executing over P's sub-environment $EP \subseteq EN$ to perform its tasks TP.

Table 2 presents all the tasks from TP obtained through CT-based refinement of conventional PL testing sub-processes' tasks as prescribed with ISO/IEC 26554.

Table 2
Tasks of CT sub-processes to TPL set-up and evolve for it

Sub-Process	Task description
	Domain Testing
Test Initiation and Design *DD*	Domain tests for SPL domain artefacts initiation including requirements and conditions for them and design for unit, integration and system testing (above *u*, *i*, s levels from (5)) while accounting TPL variability
Environment Set-up and Maintenance *DS*	Implementing or setting up TPL domain test environment including test data and tools, enabling interoperability with SPL information environment in order to access domain artefacts to test and proper feedback, changing TPL domain test environments and sharing its status with the relevant stakeholders
Test Execution *DE*	Preparing review, inspection or static analysis to conduct static testing of SPL domain artefacts, executing domain static tests and recording their results while differentiating those for commonality from for variability ones
	Running an ordered set of test cases, determining the pass/fail of them, document test execution and results related to variability
Test Reporting *DR*	Trace the status of defects in both SPL domain artefacts commonality and variability reported during domain and application testing, produce reports on domain test status and results
	Application Testing
Test Initiation and Design *AD*	Setting up the readiness to Application Testing, deriving application-specific test cases and defining test procedures based on Domain Testing assets
Environment Set-up and Maintenance *AS*	Setting up and maintaining Application Testing environments based on Domain Testing environment established with *DS*
Test Execution *AE*	Performing application static and dynamic testing on Application testing environments established with *AS*
Test Reporting *AR*	Reporting application-specific and domain test incidents and how they will be managed, analyzing the failures and locations where they have occurred
	Test Management
Test Strategy defining *SD*	Providing technical guidelines for performing domain and application testing. Elaborating Test Strategy (primarily as (5)) that provides the scope of domain testing (of SPL domain artefacts) and application testing, where these tested artefacts will be deferred to
Test Process defining *PD*	Selecting and tailoring the rest TPL processes from (2) to prevent and solve TPL-specific test problems
Planning *PL*	Providing test plans for performing TPL tests while accounting variability

	model of both domain and application testing
Monitoring and Control *MC*	Accounting domain and application test progress and sharing it with the relevant stakeholders, monitoring and controlling its compliance with test plans
	Variability Management in Testing
Variability Mechanism Category *VC*	Maintaining a set of TPL variability implementation mechanisms that can be used for expressing or realizing variability in Domain Testing artefacts
Variability in Test Artefacts *VA*	Defining TPL variability types and ways to express variability included in TPL artefacts and products such as test plans, test cases and scenarios within both Domain and Application Testing
Traceability of Variability in Test *TV*	Establishing and maintaining trace links between variability in test artefacts and variability models in test defined separately to support the reuse of test artefacts in Application Testing (foremost with the authors' dedicated Integrated three-dimensional variability assessment sub-model [14])
Asset Management in Testing *AM*	Managing Domain Testing artefacts that will be reused there and in Application Testing such as test cases for commonalities, variabilities and their interactions Managing TPL Application Testing artefacts that will be referred in regression testing due to the application evolution or as inputs in other applications

It's worthwhile to note that besides process-oriented view (1)-(6) it should also be useful to consider CT within staged perspective as series beginning with Set-up round and then interchanging Production and Reinvention rounds inspired and tightly driven with SPL evolution.

Internal structure of CT's Model outlined above is depicted with Figure 1 where notation of (2)-(4) is kept

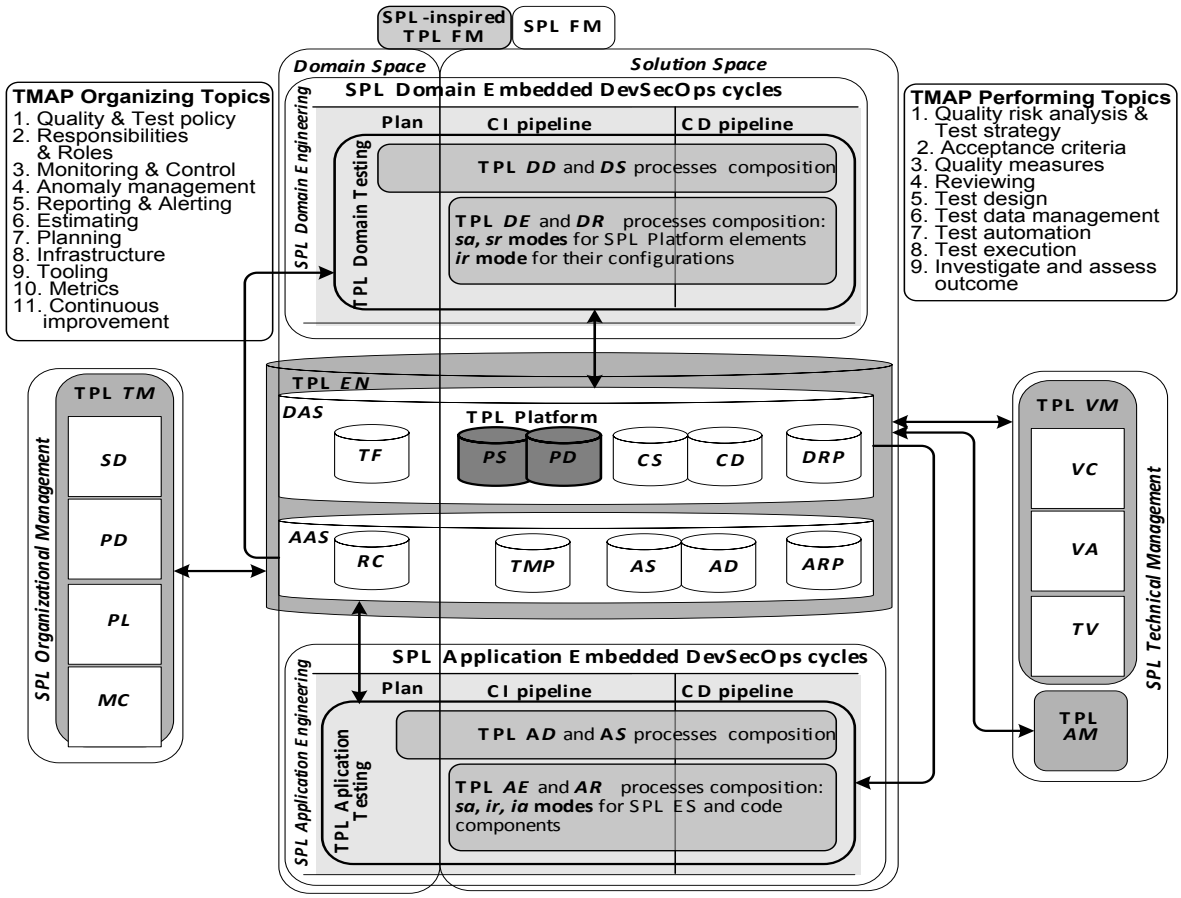

Figure 1: Inner Structure of CT Technological Model

As Figure 1 shows, SPL, when defined, in turn defines and then drives TPL by means of its FM in Problem space and through integration CT workflows as DE and DRcomposition into Domain CI/CD pipelines as well as AE and AR ones — into Application CI/CD pipelines.

It's worthwhile to note that besides process-oriented view (1)-(6) it should also be useful to consider CT within staged perspective as series beginning with Set-up round and then interchanging Production and Reinvention rounds inspired and tightly driven with SPL evolution.

While be formally enabled with (1) — (6) expressions above expected benefits of CT being modeled accordingly to them couldn't be achieved without its proper automation.

As the first step to this end pilot CT automation Toolkit is proposed based on both the maximal shortening the list of tasks to be automated from Table 2 and generic requirements for necessary tools in foremost ISO/IEC 23643 and 30130 as well as ISO/IEC 20741, 23396, 23531, 24766, 33060.

Resulting Toolkit aligned with the Model (1)-(6) above is presented with Figure 2.

4. Conclusion and Future work

The Technological Model and Prototype Toolkit for ES Continuous Testing being now considered only on host at the very beginning of Software in the open Loop stage provides Dev, Sec and Ops teams with significant benefits such as:

- Ensures readiness of tested software for Software in-the-Loop testing
- Provide a background for comprehensive embedded testing process model elaborating
- Leverages and coordinates various Test and Dev, Sec, Ops teams efforts at all the testing levels among embedded CI/CD stages over SPL Domain and Application engineering.

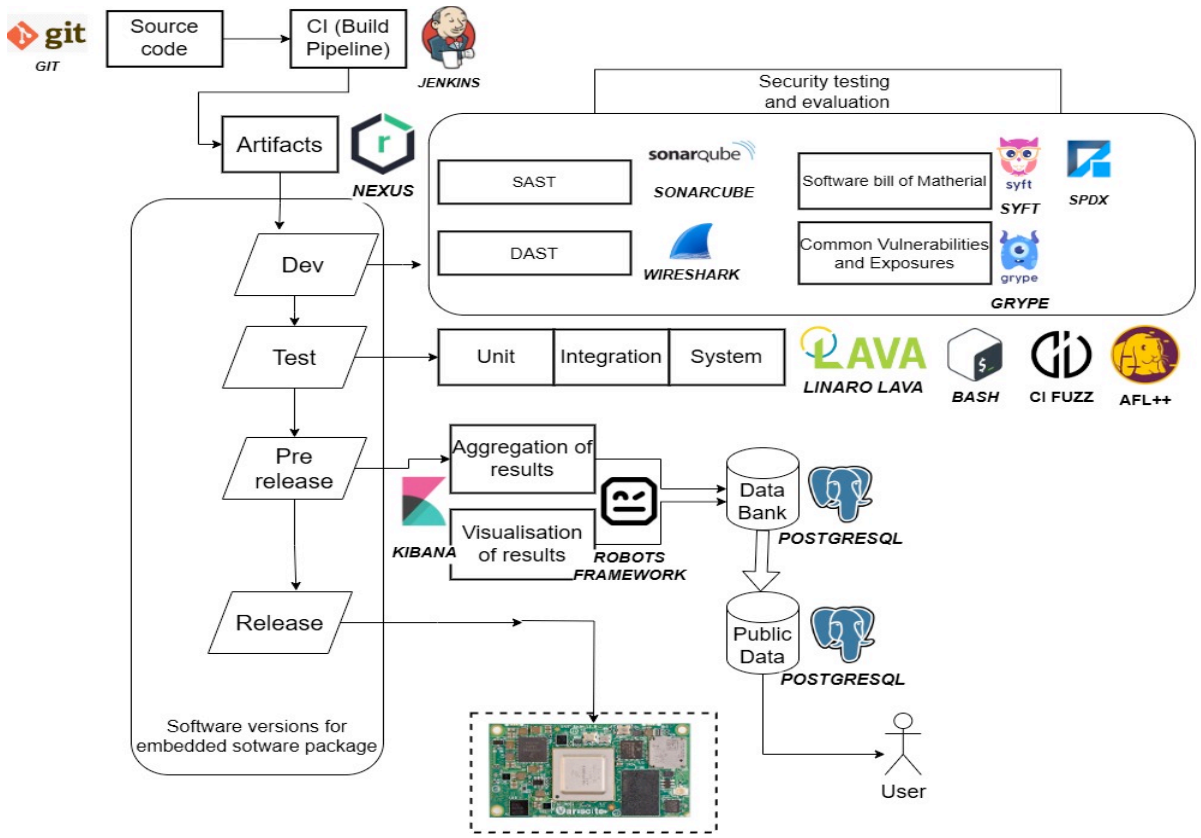

Figure 2: Author's view of Pilot Toolkit for CT automation

Their additional bonus is the possibility to further enhance CT with GenAI potential related to TMap Organizing and Performing topics [13] included in the Model.

Extending the outlined Model and its automation Kit for Embedded Testing process as a whole for further implementing an approach proposed is the authors' future work.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] J. Beningo, Embedded Software Design: A Practical Approach to Architecture, Processes, and Coding Techniques, Apress, Linden, MI, USA, 2022

[2] The Ultimate Guide to CI/CD for Embedded Software Systems. Parasoft Whitepaper. URL: https://alm.parasoft.com/hubfs/Whitepaper-CI-CD-for-Embedded-Systems.pdf

[3] Y. Yongfeng, J. Bo, Embedded Software System TestingAutomatic Testing Solution Based on Formal Method, CRC Press, Press2 Park Square, Milton Park, Abingdon, Oxon,, 2024

[4] Eisele et al, Embedded fuzzing: a review of challenges, tools, and solutions Cybersecurity (2022) 5:18. doi: https://doi.org/10.1186/s42400-022-00123-y

[5] .W. Böhm, M. Broy, C. Klein, K. Pohl, B. Rumpe, S. Schröck (Eds.): Model-Based Engineering of Collaborative Embedded Systems. ISBN 978-3-030-62135-3. Springer, Jan. 2021. https://doi.org/10.1007/978-3-030-62136-0

[6] J. Park, S. Han, Product Line Engineering for Basic Software of Automotive Embedded Systems, SAE Technical Paper (2018). URL: https://www.sae.org/publications/technical-papers/content/2018-01-1457/

[7] Clausen et al. A scoping review of In the loop paradigms in the energy sector focusing on software in the loop, Energy Informatics (2024) 7:12. doi:org/10.1186/s42162-024-00312-8

[8] M. Becker, J. Palczynski, Automatic Verification of (un)intended Data and Control Flows in Embedded Software (2024). URL: https://www.mathworks.com/content/dam/mathworks/conference-or-academic-paper/automatic-verification-of-unintended-data-and-control-flows-in-embedded-software-ew24.pdf

[9] M. Becker, J. Palczynski, Increasing Resilience to Cyberattacks through Advanced Use of Static Code Analysis (2021). URL: https://www.mathworks.com/company/technical-articles/increasing-resilience-to-cyberattacks-through-advanced-use-of-static-code-analysis.html

[10] M. Becker, J. Palczynski, Reconciling Software Development Speed and Robustness with Optimally Balanced Static Application Security Testing (2023). URL: https://www.mathworks.com/company/technical-articles/reconciling-software-development-speed-and-robustness-with-optimally-balanced-static-application-security-testing.html

[11] A. van Ewijk et al., TPI® NEXT Business Driven Test Process Improvement, Sogeti, 2013

[12] M. Huang, C. Lemieux, Directed or Undirected: Investigating Fuzzing Strategies in a CI/CD Setup (Registered Report), in; Proceedings of the 3rd ACM International Fuzzing Workshop (FUZZING '24), September 16, Vienna, Austria. ACM, New York, NY, USA, 2024, pp. 33-41. doi: org/10.1145/3678722.3685532

[13] Topics plotted on the High-performance IT delivery models, 2025. URL: https://www.tmap.net/page/topics-plotted-high-performance-it-delivery-models

[14] O. Slabospitskaya, A. Kolesnik, The Model for Enhanced Variability Management Process in Software Product Line, in: Mayr H.C., Kop C., Liddle S., Ginige A. Information Systems: Methods, Models and Applications. Revised selected papers of 4-th International United Information Systems Conference (UNISCON 2012). Yalta, Ukraine, 2012, pp. 162–171. doi: https://doi.org/10.1007/978-3-642-38370-0_15