Secure and Agile Connected Things

D2.1 - IoT platform

Franklin Selgert (AnyWi), franklin.selgert@anywi.com

Till S. Witt (NXP Semiconductors Germany GmbH), till.witt@nxp.com

Arne Ehrlich (consider it GmbH), ehrlich@consider-it.de

2021-08-30
DE/NL
ITEA3, 17005
BMBF, 01IS18062(A-E)

Abstract

Development and operation of secure, large-scale IoT systems is hard. While there are Software Development Environments (SDE), platforms aimed at providing tools and methods to control software development, testing and integration, they do not solve the major concerns of today's software-intensive systems: security, agility and a need for fast deployment of IoT system updates. Tooling is just one element of an SDE, evenly important are the processes that control the workflow of all developer involved, and which include the necessary security steps. SDEs are one part of the puzzle, companies apply different methods to guide their IoT development. This mix of methods, practices and tools do not have a common standard or way of working*(ref method wars Jacobson).

The main challenge in the SCRATCh project is to describe a method, way of working that improves the overall security of IoT systems, without inventing yet another method or way of working. Our approach to this is to use available practice and methods, abstract commonalities and describe what specific actions and tooling could contribute to more security awareness in IoT development.

In this paper the DevOps cycle is used as a common workflow process and combined with elements from the Essence method SDE of the OMG group, with

the intention to provide a kind of frame work for the tools developed and a working method with a specific attention to security.

1. Contents

2. Introduction

What is a method? According to the Cambridge dictionary “a particular way of doing something”. This covers almost anything including throwing a dice to decide on the most important security requirement. That also explains the multitude of available methods; a comparison of methods for IoT development is given in (Merzouk, Cherkaoui, Marzak, & Nawal, 2020). Each method tends to have its own focus point, e.g. Xtreme programming focuses on writing “good” code, DevOps, focuses on the relation between operation and development. SecDevOps focuses on the relation between security development and operation.

Most of DevOps discussions are about continuous integration and delivery. Less attention is paid to the Plan Phase of DevOps. In SCRATCh we identified the Plan Phase as an important step to improve on security. In search for a method to apply within the context of SCRATCh we took the generic method the Essence, Kernel and language for software engineering methods (OMG, 2018). Figure 1 shows the approach. The Essence method is composed of practices and we add two elements to this model: a Kernel existing of standard practice elements and a common language to describe those. A metamodel is constructed that could be used to describe most methods in a common language. (Jacobson & Stimson, 2018). Assuming most development processes have a similar structure, taking the Essence model as a start and describe the attention points that SCRATCh considers important to improve on the security standpoint of IoT system, led to the 3C method of SCRATCh, not a new method but more a clarification, as an addition to two existing ones: DevOps and Essence.

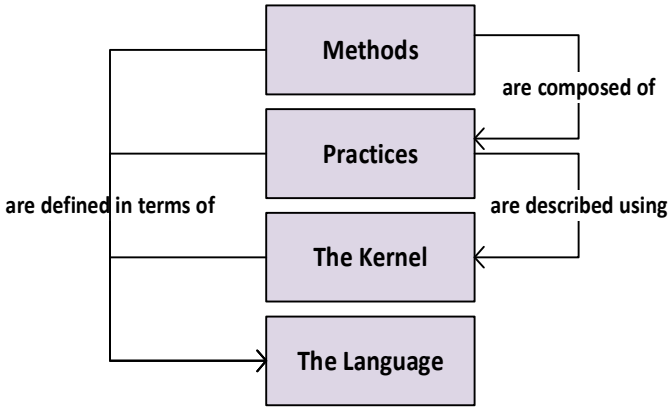


Figure 1 Method architecture Essence

3. The Essence Method Introduction

In this chapter we will explain the 3C method using the Essence method and the DevOps process. The assumption is that the reader has some knowledge of DevOps and a generic understanding of software development.

Crash course of the Essence method

The main concept in the Essence theory is the existence of a Kernel that is common to all software development methods, the main element in this Kernel is what is called Alpha's (Abstract Level Progress Health Attribute) a complicated name that captures the versatile nature of an Alpha, depending on the area of concern that it belongs to.

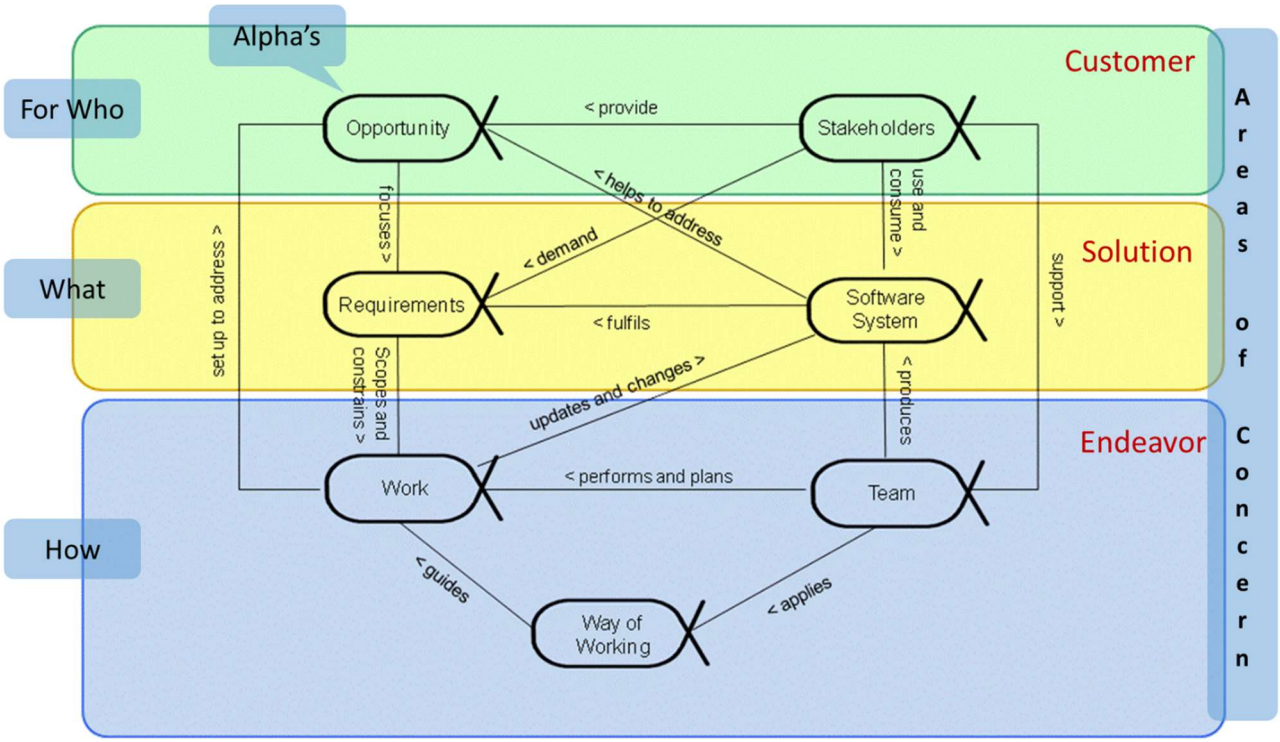


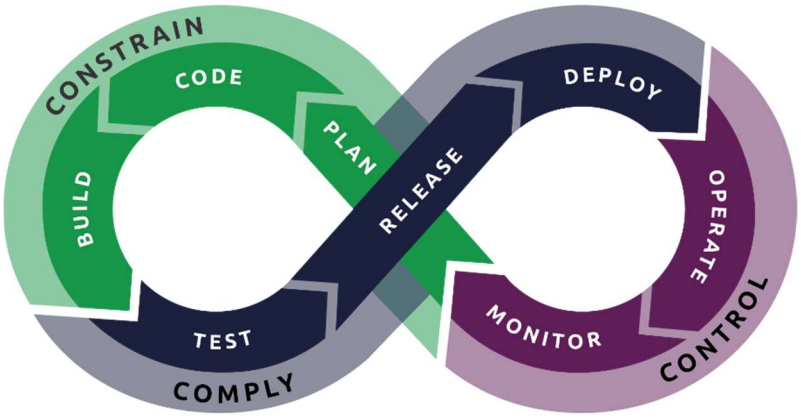
Figure 2 Essence Method

The essence method has three main concepts:

1. Things to work with, these are called Alphas. Alphas undergo states, e.g. for the Work alpha this is initiates-prepared-started-under control-concluded-closed. Each alpha has its own state transitions.
2. Things to do, activities that combine certain states of Alphas as input to realize a state transition of one or more Alpha's, as an example see Figure 6: Essence method tasks to do within "Test the System" activity.
3. Competencies relate to the skills needed to fulfil the task of state transitions. This concept relates to the DevOps philosophy that certain competencies need to be combined in one team.

The essence method organizes alphas in areas of concern, concepts that are common to any development cycle, a **customer** with needs, a **solution** to comply with those needs and an **Endeavour** to manage and organize the work that creates the solution. Security is not a separate competence nor an integrated item in one of the Alphas; it is inherently addressed in the Alpha "Requirements". The paper (Syynimaa., 2018) gives an overview and explanation of the Essence standard.

4. DevOps introduction

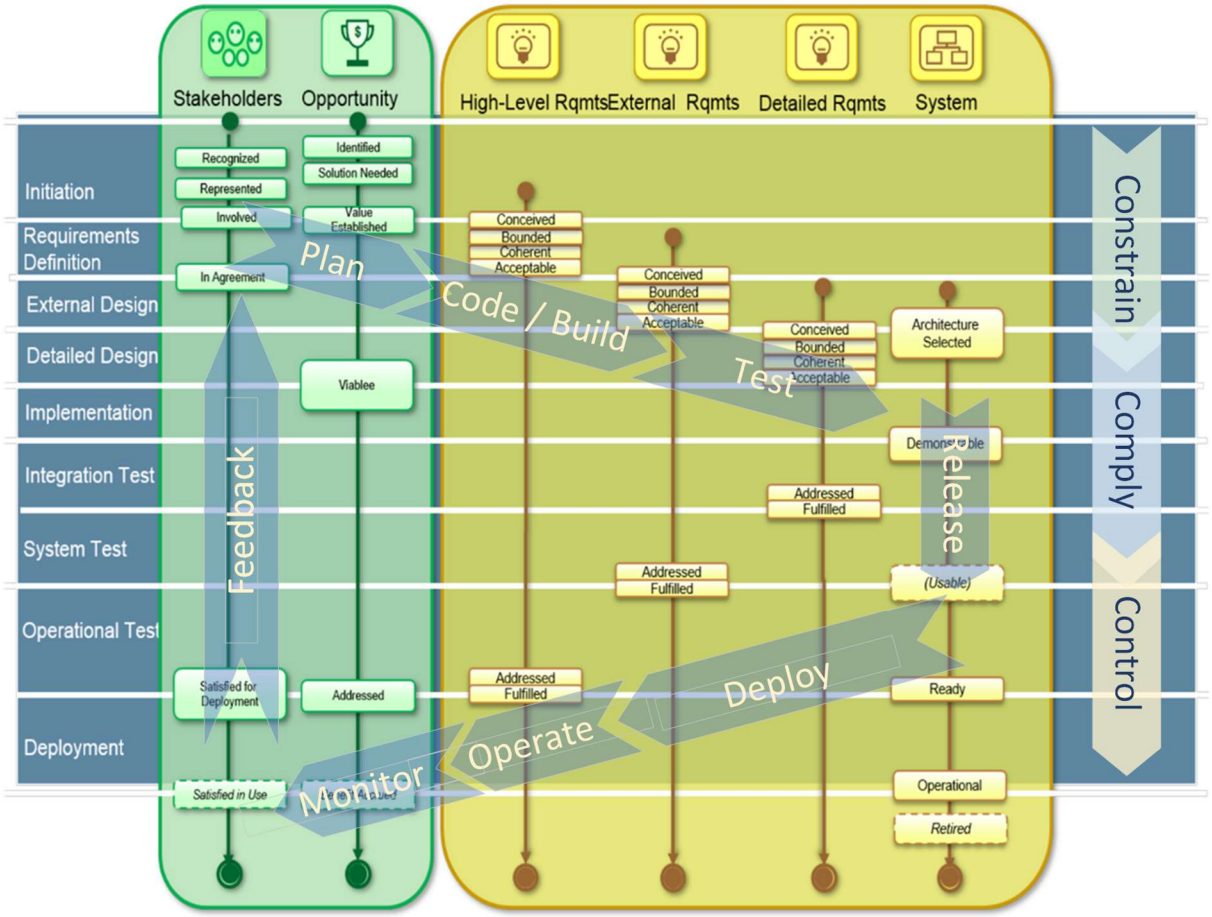


One of the core ideas of DevOps is to unify both loops development and operation into one, and to create teams that can do the full iteration. Still both halves have different methodologies, tools and skillsets applied to them, a remanence of the split of software developers and system administrators. DevOps is a process approach on top of a software Development and maintenance methodologies. In the development Phase methods like agile or V-model approaches and in the maintenance Phase ITIL methods and processes may still apply. (DevOps)

The main focus of DevOps is the continuity of release and deploy, very much in alignment with the Agile paradigm. The general thought is quickly release new features and quickly repair errors, as such also a fast repair of security vulnerabilities. Most DevOps companies focus on Continuous deployment or CD involving plan/Code/Build/Test/release and deploy. Most implementation realize Continuous integration (Mann, Stahnke, Brown, & Kersten, 2019)

Development and operation of large-scale IoT systems is hard just because of the lack of continuous deployment. Systems are scattered out and not always continuously connected making DevOps CD difficult. While there exist technological platforms aimed at providing the necessary building blocks to integrate devices and backbone logic, they do not address the major concerns of today's software-intensive systems: security, agility and a need for continuous deployment.

5. The SCRATCh 3C Method



SCRATCh proposes a integrative approach to IoT, security, development and DevOps practices through combining the Essence and DevOps method into a simple methodology the 3C Method. The emphasis for security is reflected in 3C's Constrain, Comply and Control. The 3C method is supported by a set of interoperable tools (toolkit) based on a common conceptual architecture and consisting of the following elements

Security foundation for strong device identity – use of secure elements protecting secrets providing guarantees on device identity, communication confidentiality, tamper resistance and evidence as well as collecting security metrics as part of the continuous secure deployment chain.

A SecDevOps-inspired development process combining two methodologies and is supported by tools that actively enable continuous deployment of incremental system upgrades that facilitate security and reliability, based on real-world operational metrics.

3C methodology IoT tools integrating process and technology that accelerate development and continuous deployment of IoT solutions. This is based on the DevOps and Essence principles and includes security controls, tests and feedback loops, built on top of a secure-by-design architecture.

CONSTRAIN - the first C of the method

In SCRATCh we identified the Plan Phase of DevOps or the Design Phase of a software development process as the starting point for security.

The first C covers Plan Code and Build from the DevOps cycle

As a proven concept that costs incurred by changing a product are low in the design phase, this also applies to security-related adaptations of a product, page 18 (Fraunhofer, 2014). Security in most models is an attention point across all phases and items. In case of Essence, this would span the areas of concern of customer, solution, and endeavor, (Figure 3: Security and areas of concern). Although true this does not provide much guidance.

Areas of concern SCRATCh, DevOps and Essence

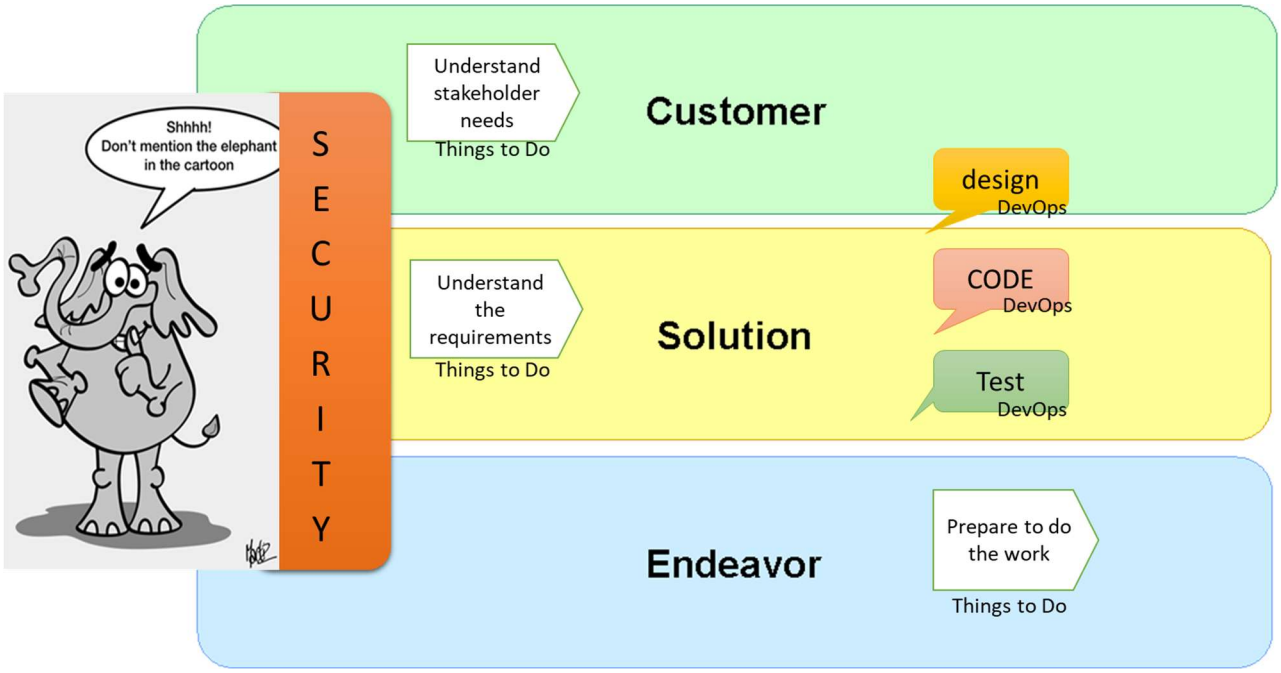


Figure 3: Security and areas of concern

The DevOps viewpoint, has a focus on processes, security viewpoint would be: “involve the security discipline in the process at start”. From the customer perspective it means a security-minded customer or stakeholder to guard the security aspect. From the development part have security “experts” in the development team, etc..

The essence, however, is that the solution should be as secure as possible or needed. The question to answer is: what guides the solution development? Most likely answer, seen from different methodologies: stakeholder needs and requirements.

Getting security on top in the beginning of the process stakeholder needs or requirements are the items to look at. SCRATCh proposes “constrains” or essential security requirements to achieve security attention from the start. They can act as a design-constrain throughout the development process.

In SecDevOps it would mean that in the Plan Phase a security requirement from a standard or best practices is inserted as a design constraint. As an example In the SCRATCh whitepaper (Selgert, 2020), one constraint is identified as no.-1-important type of constraint: the capability to update the IoT device securely, keeping the system safe. In all major standards like ETSI, ENISA, OWASP IoTSF, requirements are mentioned taking this constraint seriously.

In the Essence method **Constraints** are part of the Kernel Alphas, *Restrictions, policies, or regulatory requirements* the team must comply with see Figure 4: Constraints as security requirements.

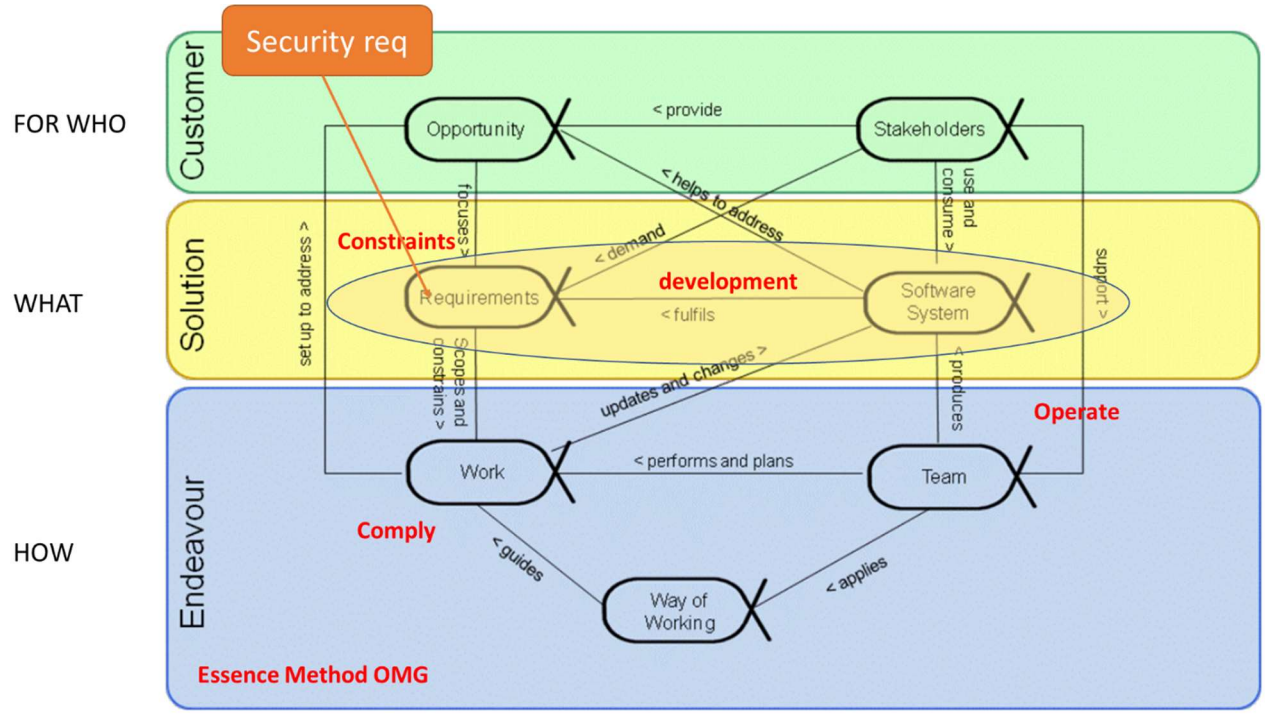


Figure 4: Constraints as security requirements

HOW TO implement the constraints. The first C, constrain means that a set of security needs should be available matching the type of development at hand, and matching the granularity of the stakeholders’ need. Choosing the best practices / regulations to start the development is time-consuming and depends on the sector where the IoT device/software is supposed to work in. Scratch proposes a minimal basis set of relevant essential security requirements to incorporate in the plan phase or requirements management process. Its then up to each company to add specific sector requirements.

The SCRATCh knowledge base provides several starting points.

(change to Github or public source)

1. https://trusttab.com/req_tool/dcms_code, 13 high level security requirements that are mapped to 47 regulatory, standards and industry consortia.
2. https://trusttab.com/req_tool/owasp_top10 , 10 high-level security requirements and related with these https://trusttab.com/req_tool/owasp_isvs_1_0rc 125 test requirements.
3. https://trusttab.com/req_tool/enisa_req ENISA baseline security recommendations.

Example: If security from a stakeholder requires basic security requirements like Confidentiality, Integrity, Authentication searching on these keywords in ENISA Best Practices gives a starting list of the following constraints:

Table 1: ENISA, authentication constraints

Description
Ensure password recovery or reset mechanism is robust and does not supply an attacker with information indicating a valid account. The same applies to key update and recovery mechanisms.
Protect against 'brute force' and/or other abusive login attempts. This protection should also consider keys stored in devices.
Authentication credentials including but not limited to user passwords shall be salted, hashed and/or encrypted.
Authentication mechanisms must use strong passwords or personal identification numbers (PINs), and should consider using two-factor authentication (2FA) or multi-factor authentication (MFA) like Smartphones, Biometrics, etc., and certificates.
Ensure default passwords and even default user names are changed during the initial setup, and that weak, null or blank passwords are not allowed.
Design the authentication and authorization schemes (unique per device) based on the system-level threat models.

Table 2: ENISA, software and firmware update constraints

Description
For control systems which cannot be updated (e.g. legacy systems), apply compensating measures, such as network segmentation, micro segmentation, system relocation or additional real-time monitoring tools. Perform risk analysis to determine if it is possible and sufficient to improve security of existing system or if the replacement of the system is necessary.
Allow Third Parties to perform patching only if they guarantee and are able to prove that the patch has been tested and will not have any negative consequences on the device or if the Third Party assumes the liability for the update according to an applicable agreement. In addition, require Third Parties to report any executed actions related to the patching process and inform about them in advance. Update procedures shall be documented, known and controlled by the organization.
Perform deployment of patches for the IoT devices only after proving that no negative consequences exist. Test the patches in a test environment before implementing them in production. If this is not possible, begin with deploying patches only on a segment of a system, ensuring that other zones will continue to operate normally in case a patch exerts any negative impact on a chosen segment.
Execute automatic update procedures only if they are based on the risk analysis and if the devices for which the automatic update can be allowed are identified. Verify the source of the update.
Verify endpoints' software/firmware authenticity and integrity and ensure tight control over the update. Signing code updates (to be able to authenticate the code before it is loaded) and maintaining the authenticity is advisable.

As a practical starting set for minimal security SCRATCh defined a limited set of

7 essential security requirements to start with as a bare minimum (ref d1,1)

Table 3: Minimal Set

short_d	description	origin
Securely store credentials and security-sensitive data	Any credentials shall be stored securely within services and on devices. Hard-coded credentials in device software are not acceptable.	DCMS nr 4 OWASP nr 7
Minimize exposed attack surfaces	All devices and services should operate on the principle of least privilege. unused ports should be closed, hardware should not unnecessarily expose access, services should not be available if they are not used and code should be minimized to the functionality necessary for the service to operate. Software should run with appropriate privileges, taking account of both security and functionality.	DCMS nr 6 OWASP nr 10
No default passwords	All IoT device passwords shall be unique and not resettable to any universal factory default value.	DCMS code nr 1 OWASP nr 1
Ensure software integrity	Software on IoT devices should be verified using secure boot mechanisms. If an unauthorized change is detected, the device should alert the consumer/administrator to an issue and should not connect to wider networks than those necessary to perform the alerting function.	DCMS code nr 7
Make systems resilient e.g. to outages or firmware update failures	Resilience should be built in to IoT devices and services where required by their usage or by other relying systems,	DCMS code nr 9
Monitoring telemetry data	If telemetry data is collected from IoT devices and services, such as usage and measurement data, it should be monitored for security anomalies.	DCMS code nr 10
validate input data	Data input via user interfaces and transferred via application programming interfaces (APIs) or between networks in services and devices shall be validated.	DCMS Code nr 13
Firmware update mechanism	Lack of Secure Update Mechanism Lack of ability to securely update the device. This includes lack of firmware validation on device, lack of secure delivery (un-encrypted in transit), lack of anti-rollback mechanisms, and lack of notifications of security changes due to updates.	OWASP top 10 nr 4 DCMS nr 3

Additional Design Constrains for use of SCRATCh Tools

Design requirements Firmware / software update tool

Having identified **firmware and software update** as important, a tool is developed in SCRATCh to provide a method to perform updates securely, a tool that is used in the Deployment Phase of DevOps. This tool, however, poses constraints this phase of the development, if a component of an IoT system does not have an update capability, there is no way to perform it securely or insecurely.

And in case a component is lacking the update capability for whatever reason another counter measure must be designed as pointed out in ENISA best practice:

For control systems which cannot be updated (e.g. legacy systems), apply compensating measures, such as network segmentation, micro segmentation, system relocation or additional real-time monitoring tools.

Design constrains needed for use of firmware update tools

1. Device support for openssl, to generate private, public keypair
2. Availability of signing server (e.g redwax)
3. Provisioned "edge" device" with smart element or other ID
4. Minimum Manifest definition for firmware update, stating the update definition, (describes the policy for update)

Secure firmware update tool: https://github.com/SCRATCh-ITEA3/AnyWiRedWaxFirmwareUpdate

Other tools to be used in the constrain Phase of the SCRACTCh Process:

https://github.com/SCRATCh-ITEA3/SCRATCh-Tools-Repo/tree/master/C1_Constrain

COMPLY - the second C of the Method

The second C covers /Test/release and deploy from the DevOps cycle

The second C "Comply" refers to testing, a major part of the effort for creating a new system. The assumption is that if the constraints also contain the essential security-related requirements, then testing has to prove compliance with them as is with all the other requirements. Testing itself is a complex activity and occurs at different moments of the development (ref whitepaper SoTA).

From a DevOps perspective Comply covers the Code, Build, Test and Release Phase

From the DevOps perspective (see Figure 5; Testing cycles in DevOps), it is clear that there are multiple methods for testing a system, from code testing to load testing. Testing of the essential security constraints is done in the complete testing chain depending on the type of requirement.

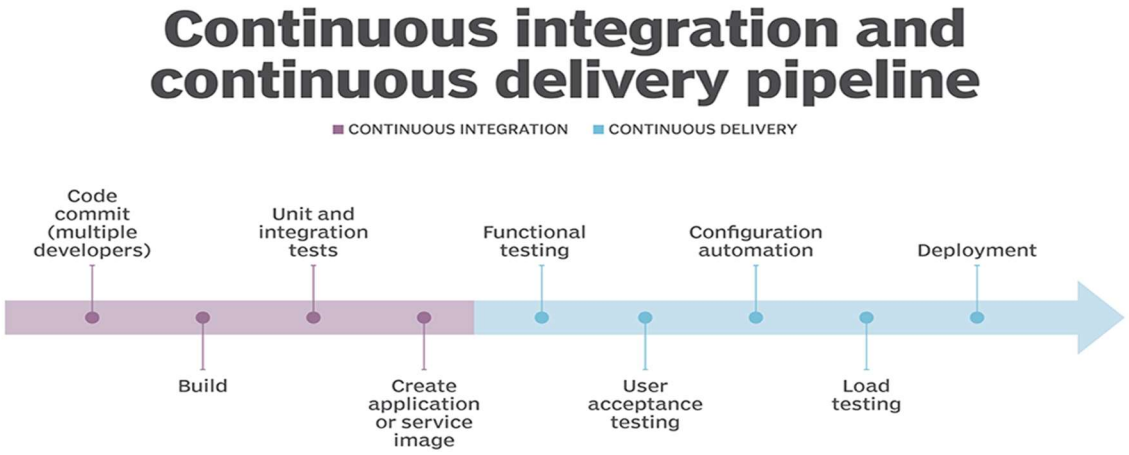


Figure 5; Testing cycles in DevOps

As an example, the requirement from ENISA: *Verify endpoints' software/firmware authenticity and integrity and ensure tight control over the update. Signing code updates (to be able to authenticate the code before it is loaded) and maintaining the authenticity is advisable.* can only be tested on an already deployed system at point of loading new firmware. In DevOps "Load testing" of an IoT system means to have an identical copy of parts of the life system or the test is conducted on the live system. Mirroring an IoT environment is a real identified bottleneck for end system testing. Potential solutions for this might be a digital twin system emulation or a software copy of a hardware system. Currently these two solutions are not available.

A second example of a security requirement from ENISA: *Authentication credentials including but not limited to user passwords shall be salted, hashed and/or encrypted. c*an be tested at code level and/or at unit integration.

A third example reflected in OWASP ISVS: *Verify that compilers, version control clients, development utilities, and software development kits are analyzed and monitored for tampering, trojans, or malicious code.* is a requirement that is more on a process level for software development. It requires to check for known threats/vulnerabilities in the libraries and tools used to create the system.

Taking a look at the Essence method Figure 6: Essence method tasks to do within "Test the System" activity, it indicates the same process as with DevOps, although its presentation is different and more abstract. The essence method also sees testing as verifying the compliance to requirements. However,

the different stages of testing are spread out over different states of the system and its requirements. The activity “Test the System” is equivalent to functional testing, user acceptance testing and load testing.

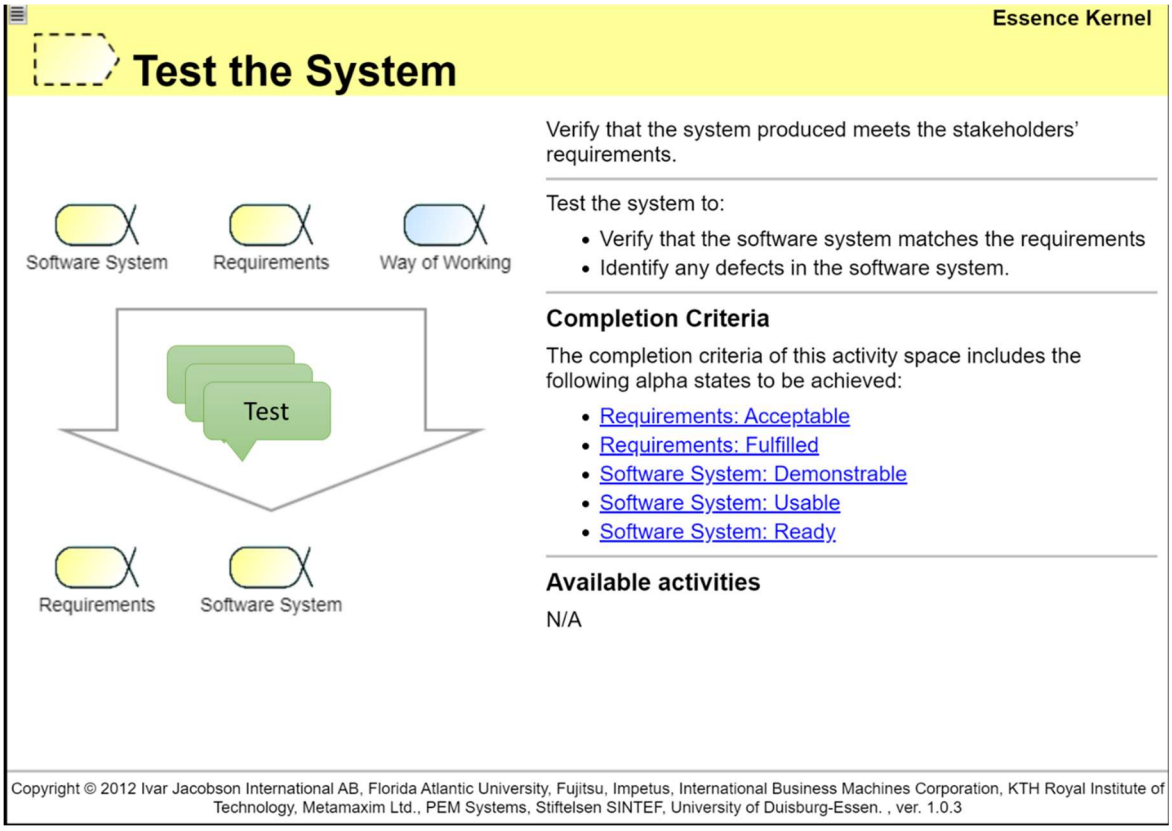


Figure 6: Essence method tasks to do within “Test the System” activity

Other testing is done in an activity labelled “Implement the System” where bug fixing and unit testing can be positioned. There is no specific mentioning of the check for known threats/vulnerabilities in the libraries and tools used to create the system. In Essence terms this would be handled by the item “Way of Working” as it is a more process-related requirement.

We can conclude that this step of the 3C method Comply is well covered in existing methods, the emphasis is done in the Constrain step by identifying essential security requirements.

Tools

For this step there are multiple tools commercially available (see D1.2 SCRATCH), In SCRATCh we developed a few specific tools for testing:

https://github.com/SCRATCh-ITEA3/SCRATCh-Tools-Repo/tree/master/C2_Comply

CONTROL - the third C of the Method

The third C covers Operate and Monitor from the DevOps cycle

Controlling a system is all about keeping it safe, the level of control possible depends on what part of a collection of essential security requirements is implemented in the system and what part of the security policies are embedded in the support organisation. This is the last C of the proposed 3C method but actually the most important one as it is to be expected that any system will fail one time during its lifecycle, as explained in Cynefin Framework, DevOps and Secure IoT (Selgert, 2020). In the 3C method Control includes the deployment step of Essence and DevOps, as this step is about the control over updates to the system.

From a DevOps perspective Control covers the deploy, operate and monitor Phase

DevOps is, amongst other things, about shifting operational knowledge to the design Phase e.g. by stating certain requirements are important to maintain a system, SecDevOps is about security involvement from start to finish e.g. by inserting essential security related requirements in the design phase (the first C)

The Essence Method is less explicit about this phase and so is the DevOps method, Operation is the part where methodology explicitly enters the domain of a specific implementation and where a complete new area of knowledge is introduced, e.g. ITIL. Operation tends to be more process- or policy-focused and tailored towards the capabilities of the company that maintains a system. The inclusion of “constrains” to make a level of control possible is laid out by the first 2 C’s and is in essence covered by both methods DevOps and Essence (see Figure 7: Essence Things to Do, Operate the System).

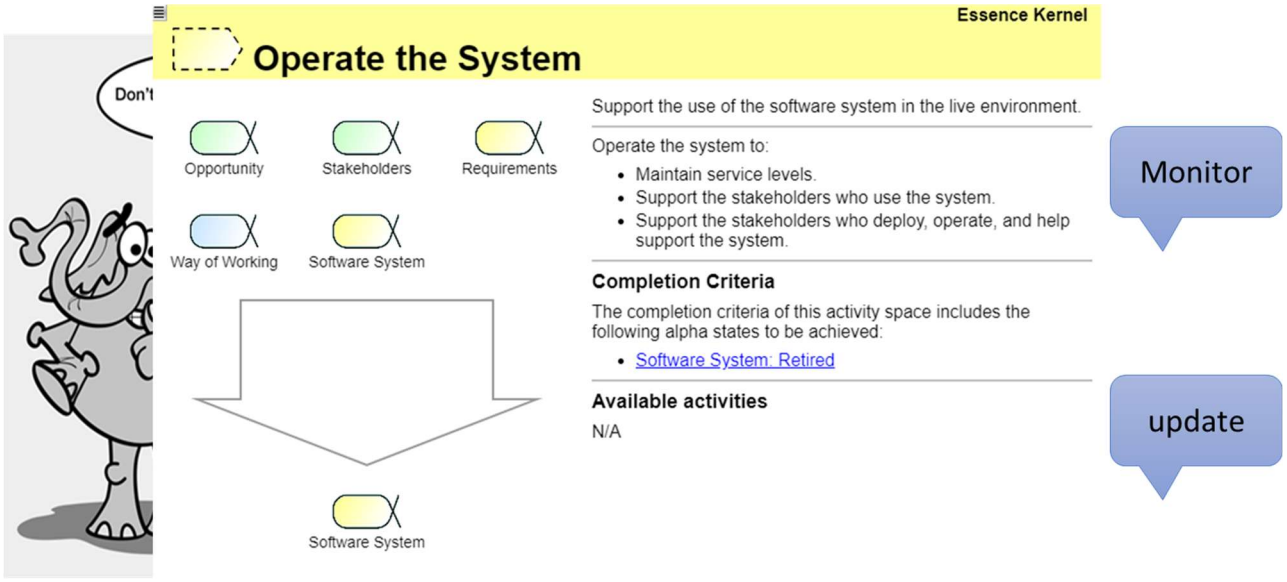


Figure 7: Essence Things to Do, Operate the System

Eliciting new essential security requirements as feedback from the operations phase is explicitly covered by the DevOps process approach. What is missing is an organisational-level of constraints posed by the restrictions of a specific organisation that is maintaining a solution. For this issue, there

is not a clear solution, but a recommendation is available in the form of the policy best practices by ENISA.

E.G ENISA: Clearly define all relevant aspects of the partnership with Third Parties, including security, within the appropriate agreements and contracts (e.g. SLA - service level agreement, NDA - Non-Disclosure Agreements). Sign these agreements and contracts before the start of cooperation.

Establish and maintain asset management procedures and configuration controls for key network and information systems.

There are more requirements, but it is clear that these requirements should translate to features in the system. Asset management could need a strict and secure identification method of devices, partnership type of relations could mean a segmented authorisation method to the system.

As part of Control also the DevOps step “Deployment“ is considered,

Tools

For the control phase tools can be found in :

https://github.com/SCRATCh-ITEA3/SCRATCh-Tools-Repo/tree/master/C3_Control

6. Project Learnings:

During the development of the tools and maturing the use-cases and use of the tools, the project team identified a minimum set of essential security requirements that can be helpful as a low effort start for any IoT development to reach a minimum level of security.

In applying the 3C method as described in this deliverable the following observations were made:

1. The Essence Method lacks a description of the research Phase.
2. The role of essential security requirements or practices as a standard set for development was not clearly described in the essence method.
3. General remark standing organisations have already an implemented method of working, the Work part of the essence Method was not used or useful in the retail use case.
4. Simplified DevOps view of the 3C method proofed to be useful.

The Essence Method lacks a description of the research Phase.

The research phase is ad-hoc, unstructured information is taken in without clear reference to opportunity or stake holder. It makes sense to have a demarcation between a research phase and a development Phase. The research phase is loosely coupled with stakeholder wishes or foreseen market demands. A handover to a development phase can mend the shortcomings and make development a more controlled process.

The role of essential security requirements or practices as a standard set for development was not clearly described in the essence method.

Security was part of the research phase and delivered a list of security requirements that can be handed over to a development phase. As part of the research the essential security requirements of SCRATCh were taken into account and can be an input for the development phase.

General remark standing organizations have already an implemented method of working, the Work part of the essence Method was not used or useful in the retail use case.

There was no greenfield situation for the retail use case. Adapting work methods to a new methodology like essence is not wanted nor efficient. New start-ups without a working method could have a quick start using essence and set a first step in preparation for certification.

Simplified DevOps view of the 3C method proofed to be useful.

DevOps is a popular method, having a lot of community support and tooling. The popularity however contributed to confusion as many interpretations of the method popup. For starters and employees not involv3ed in process improvement The 3C method gave a simple handle on the relation between Security and DevOps.

7. Conclusion

Getting to Secure DevOps for SMEs is about awareness, process and activities. To get to the point on security: Three main aspects can be identified to contribute to more secure products* Constrain, Comply, Control, aspects that are mainstream in the last five decades of software development in different wording.

In the Essence model the wording “constraints” is related to requirements. Comply is the activity to construct a (software) system that fulfils these requirements. Control is an activity related to the use of the (software) system. In more detail, the Essence model works with the transition of states of an object, called an alpha; things to be performed by the work alpha guide in a certain way of working to progress from one state to another. Certain combinations of states for the alphas, e.g. requirements alpha or software system alpha, correlate with the process-oriented phases of DevOps.

The 3C method is a rough simplification using both models and is focused on security, to serve SCRATCh’s intent to guide SMEs towards more secure IoT-system development. The complex nature of both DevOps and software development is simplified to implement basic security with a minimal effort. This 3C method is supported by the aforementioned tooling developed in the SCRATCH project.

Tools developed to support SecDEvOps can be found on:

https://github.com/SCRATCh-ITEA3/SCRATCh-Tools-Repo

8. References

(sd).DevOps. *https://biteofnews.com/what-is-devops-its-introduction-and-tools/*.

Mann, A., Stahnke, M., Brown, A., & Kersten, N. (2019). *2019-state-of-devops-report.html.* Retrieved from https://www2.circleci.com/: https://puppet.com/resources/report/state-of-devops-report/

OMG. (2018). *OMG (2018). Essence - Kernel and Language for Software v 1.2.* OMG.

