Security by Design in an Enterprise Architecture Framework

Wim Mees
Royal Military Academy, Department CISS
Renaissancelaan 30, 1000 Brussel, Belgium
wim.mees@rma.ac.be

1.0 INTRODUCTION

It is essential to meet the military organization's need for information assurance. This means that appropriate measures must be taken to protect critical assets. It is however equally important to accept risk where this would lead to an operational advantage. Information security must indeed be considered as an enabler of operational success, rather than an obstacle to be avoided.

Existing security controls are often preventing military personnel from deriving an operational advantage from the ability to collect, process, and disseminate freely the information that is needed for an improved situation awareness. Consider for instance the need to share information within a coalition. When one of the connected systems gets compromised, it can have an operational impact on every other nation's information security and for that reason most often the most pessimistic option is considered. However, how is it possible to achieve unity of effort when restrictive information sharing policies are in place?

It is important that the information security initiatives that are undertaken demonstrate value to the military operations, instead of being perceived as an inconvenience that obstructs effective military operations. Moreover this perception will lead to soldiers trying to bypass the security controls that are in place, introducing new risks that were most often not foreseen and therefore not addressed by the information security management system.

In section 2.0 we describe the different types of requirements that guide the development and operation of an information system, and more in particular the way in which security concerns are addressed during the full life-cycle of the system. Next in section 3.0 we briefly situate the role of Enterprise Architecture in handling security across the information management landscape of a military organization, followed by a more specific methodology in section 4.0 for addressing the information security risk. Finally in section 5.0 we offer some suggestions on how to use modern evolutions in the area of information system development and information service delivery to close the gap between the security staff and the other stakeholders, followed in section 6.0 by a few typical pitfalls and how these can be avoided.

2.0 SECURITY REQUIREMENTS

If one does not know what to protect, against whom and to what extent, it is very hard to design, build or operate a security system or make substantial statements about the security of a system. Designing a secure system therefore requires an assessment of the risks that need to be addressed.

The ISO 27000 series of standards provides a best practice framework to manage risks to the security of information managed by an organization. Its goal is to put in place an *"Information Security Management System"* (ISMS) of continuous improvement, that assesses risks, puts in place the necessary policies and controls to address those risks, evaluates the result hereof through internal and external audits, improves the whole system based on the findings of these audits and finally starts all over again. The ISMS needs to scoped, for instance to the information and the information management systems used in the context of a given mission or by a specific task force that is sent out into theatre. Within this scope the most important assets are to be identified, and the threats to these assets enumerated. The organisation will then determine how vulnerable it is to those threats should they occur and decide how it will treat these risks.

The ISO 15408 *"Common Criteria"* (CC) is another framework for expressing an end-user's need for protecting information. The user will write a *"Protection Profile"* (PP) that describes in an implementation independent way the threats he considers, his security objectives, assumptions, *"Security Functional Requirements"* (SFRs), *"Security Assurance Requirements"* (SARs) and rationales.

The security of the information is in itself never the reason a system is originally built. The motivation for investing a lot of resources in the development of a system is to answer a certain information need for the end-user. This end-user is however just one of the stakeholders that has an interest in the system. Each stakeholder can have his own specific goals for the system that are expressed in a stakeholder view.

These goals consist of functional and non-functional goals, as is illustrated in figure 1. The functional goals describe *what* the system should do, while the non-functional goals place constraints on *how* the system will do so. A functional goal could for instance be to support the targeting process, by managing objects or installations to be attacked, managing the priorities assigned to the targets, match possible actions to targets to create the desired effect, etc. A non-functional goal could be that changes to the targeting information performed at one location must be available within 2 seconds to all users at all locations accessing the information.

The security concerns of a stakeholder will typically be expressed as abstract *"security goals"*, that are part of the non-functional goals and can be subdivided into *"integrity"*, *"confidentiality"* and *"availability"* goals. For the purpose of this presentation we consider that concepts like accountability and non-repudiation can be categorised under integrity, whereas authentication is needed to ensure confidentiality and integrity, and finally anonymity or unobservability is part of confidentiality.

Security goals are very general statements about the security of an asset, in a language that the stakeholders understand. For instance a mission commander wants his operational plans to be kept confidential from the opponent, the operational picture to be always available to his commanders in the field, etc. These goals are however too vague to be verifiable requirements. The security goals therefore need to be refined in the form of more detailed *"security requirements"*.

Older security approaches focused on protecting a system and its users against external hackers

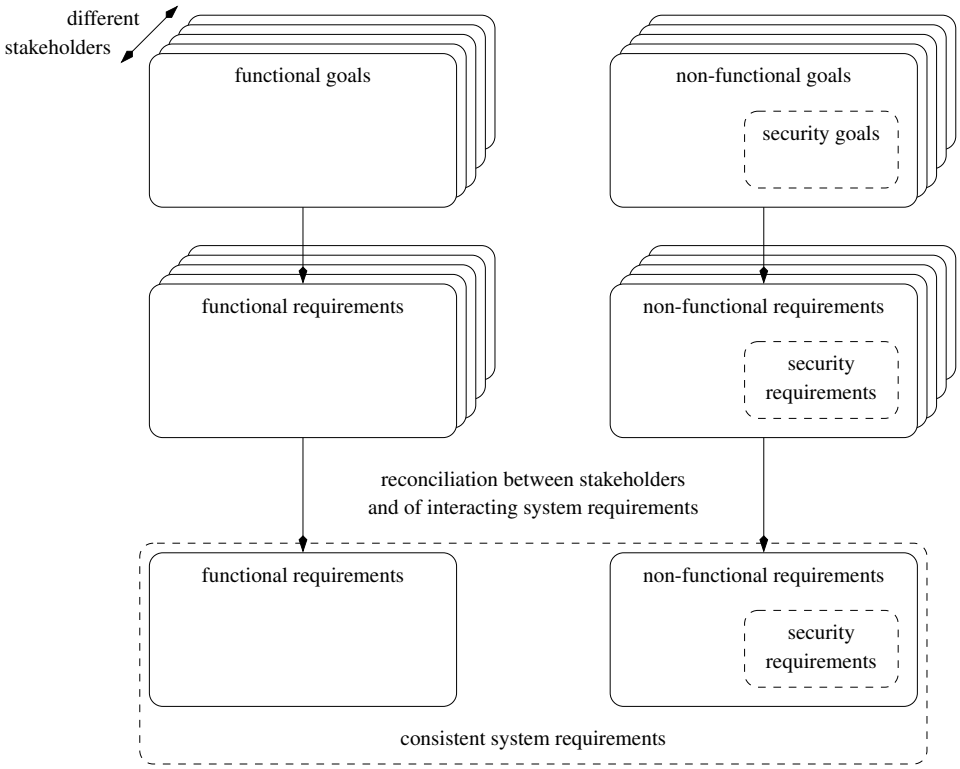

Figure 1: requirements

and misbehaving users, assuming that it was quite clear who has to be protected against whom. In practice however stakeholders often have different priorities and sometimes even conflicting interests. Therefore the requirements have to be reconciled between stakeholders. This reconciliation is not only required for the security requirements, but rather for all functional and non-functional requirements. The reconciliation leads to a single consistent set of system requirements, functional requirements on the one hand and non-functional requirements on the other, as is illustrated in figure 1.

It is important that the cybersecurity-architect recognises the operational needs of the military commander, and provides solutions that support these needs, rather than focusing only on security and thinking in term of restrictive controls and which technology to apply. Security specialists tend to focus mainly on protecting the confidentiality, integrity and availability of critical information, as well as ensuring accountability. There are however many more requirements that should be considered to ensure that the end-users across the military organization can rely on having access to the information they need in a timely and secure manner.

usability: the solution must be appropriate given the technical competence of its intended users, for instance a soldier in an infantry platoon. It must also be ergonomically acceptable to those users, given the circumstances in which they will have to operate it, for instance inside an armoured vehicle. This is also a matter of availability of the information to the soldier in the field and of integrity by avoiding data entry or manipulation errors.

inter-operability: the solution must take into account current and future requirements for inter-operability with other systems, for instance in the context of future coalitions or cooperation with non-governmental organisations. Indeed, consider for instance information about *"Improvised Explosive Devices"* (IEDs). For reasons of *"Force Protection"* (FP) it is essential that this information is rapidly exchanged between different nations that partner in a coalition and made available to the soldiers in the field.

integration: it must be possible to integrate the solution with the wide range of applications and platforms for which this might be required in the future, for instance in the context of the "Military Internet of Things". Indeed, considering the extremely long in-service life of military systems and platforms, typically between 15 and 50 years, they must be designed in such a way that they have the necessary open interfaces that will allow the future integration with new communication technologies, sensor devices, etc.

supportability: it must be possible to support the application in the environment in which it will be used, for instance at the tactical edge where a limited CIS support staff has to support a range of applications and therefore receives a generic training rather than in-depth troubleshooting skills for a specific system. When for instance a given *"High Assurance Internet Protocol Encryptor"* (HAIPE) can easily be misconfigured by an inexperienced operator to use a low-grade encryption alternative or even to pass the packets in clear-text, having this type of device operated by entry-level CIS technicians in the field would introduce unnecessary risks.

fast time to theatre: it must be possible to rapidly integrate the development of important new solutions into a development programme in order to contribute to an operational advantage over the opponent. When new technologies are available that can provide an information advantage, it is important to put them in the hands of the military end-user as fast as possible, without having excessively restrictive security regulations and processes unnecessarily slowing down the whole process.

scalability of platforms: the solution must match the range of computing platforms on which it will be implemented, for instance ranging from the handheld device of a dismounted soldier to the powerful workstation of an image analyst at a reach-back location. When a certain type of information is important to the individual foot soldier, the limited computational resources of for instance the *"Nett Warrior"* (NW) should not be a reason to stop distribution of the information at the level of the company commander.

scalability of security level: the solution must be compatible with the range of cryptographic and other security controls that will need to be implemented to provide the required security strengths and assurance levels, for instance network layer encryption that can cause problems in combination with performance enhancing proxy on satellite links, or protocols requiring acknowledgements that need to run across a data diode.

scalability of use: the solution must be able to scale to the future numbers of users or the future capacity requirements for throughput and storage, for instance taking into account future very high resolution multi-spectral sensors being operated on drones at tactical level.

re-usability: the solution must be re-usable across a range of similar operations to get the best return on investment in terms of acquisition or development, support technician and end-user training, etc.

risk-based cost/benefit effectiveness: the reduction of risk achieved by the solution should match the costs of acquisition, development, installation, administration and operation of the solution.

enabling operations: security can play an important role in generating the level of confidence required for enabling new ways of managing the operations, for instance by creating a coloured cloud of securely interconnected islands within a community of interest.

The architect needs to adopt a holistic approach. The security of a system is only as strong as the weakest link. Indeed, an attacker only needs a single vulnerability to compromise the security of a first system and will attempt to move from there to other systems. Information security should therefore be addressed at all levels, ranging from the highest-level overarching strategy and design level for the entire military organization, down to the individual information systems and services, as will be discussed in section 3.0.

3.0 ENTERPRISE ARCHITECTURE

Today's military information environment is unprecedented in terms of the richness of the opportunities it offers, as well as the always new challenges it presents us with. It is therefore essential to be able to design and implement changes to the information infrastructure in a rapid and recurring manner. This is not so different from managing a city, that is in constant evolution, where people share common resources like roads or bridges, and also face common challenges like mobility and security. For that reason *"enterprise architecture"* (EA) has often been compared to city planning.

EA models the organization's business and technology, as well as processes and governance in order to provide an overall view of the organization that supports decision making. A well known EA framework is the *"The Open Group Architecture Framework"* (TOGAF), that is characterized by an *"Architecture Development Method"* (ADM), shown in figure 2, and a meta-model, shown in figure 3.

As figure 2 illustrates, the ADM is continuously driven by a *"requirements management"* process at the centre of the diagram. Architecture by its very nature deals with uncertainty and change, and it is therefore crucial that the requirements and their changes be identified, stored, and fed into the relevant ADM phases by a dynamic process rather than using a static set of requirements,

The security requirements are addressed by the ADM in each of its phases in the following way:

- *preliminary phase*
  The enterprise organizations that will be impacted by the security architecture are identified and a list of boundary conditions is produced. A list of applicable regulatory and security policy requirements is compiled. The role of the security architect as part of the overall architecture effort is defined.

- *phase A: architecture vision*
  It is important to obtain management support for the security-related aspects of the architecture

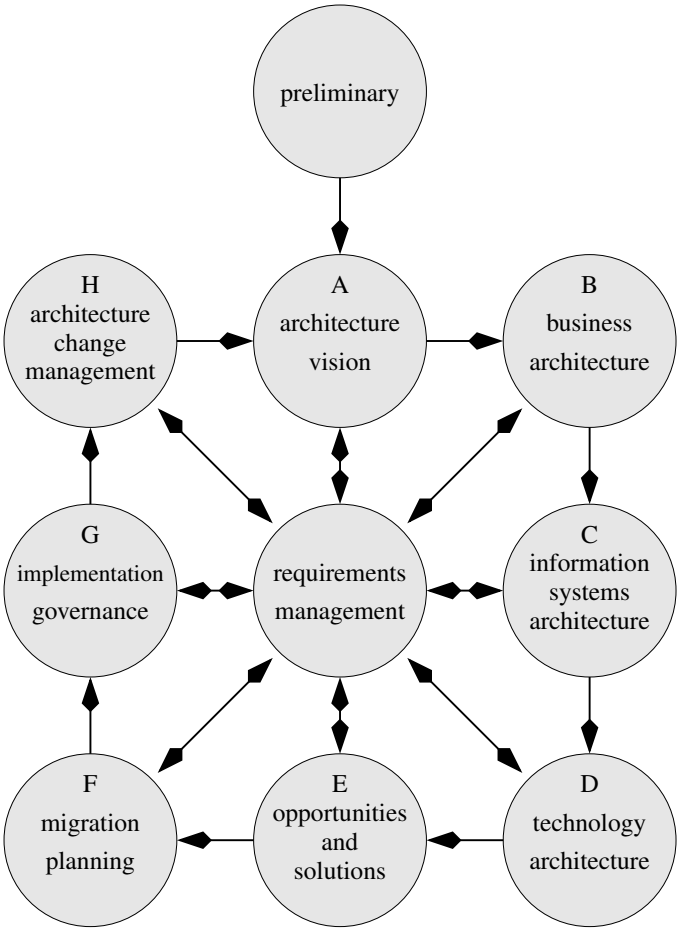

Figure 2: architecture development method

development since they can have an important impact on other aspects and as a result meet resistance.

Systems are classified as safety-critical, mission-critical, or non-critical. Disaster recovery and business continuity plans are identified and documented.

The anticipated physical, business and regulatory environment in which the systems will be deployed, is determined, for instance an armoured vehicle in a hostile environment with only a limited bandwidth tactical radio-link for a battlefield management system.

- *phase B: business architecture*
  The first step consists in determining what needs to be protected, in other words the assets potentially at risk, as well as identifying the owners of these assets and the legitimate actors who will interact with them.

  Next the criticality of the availability and correct operation of each service is assessed, and the cost or impact in case of loss or failure is determined. This leads to an estimation of how much

additional security can be justified.

- *phase C: information systems architecture*
  In this phase we are still answering the question "what can go wrong?". This means identifying the criticality of the availability and correct operation of each function, the sensitivity or classification level of the information, as well as possible attack avenues. Finally approaches for managing the identified risks are selected, such as mitigation, acceptance, transfer or avoidance.

- *phase D: technology architecture*
  At the level of the technology architecture the technical mitigating security measures are identified, as well as methods for regulating the consumption of resources, and methods for measuring the efficacy of the selected security measures. Finally the minimal privilege level that entities must have to achieve the technical or business objectives is determined.

- *phase E: opportunities & solutions*
  Existing security services that are suitable for re-use are identified. Mitigating actions for the identified risks are designed, implemented, and deployed.

- *phase F: migration planning*
  The impact of the new security measures is assessed, and solutions for measuring their efficacy are implemented. Correct secure installation parameters, initial conditions, and configurations are identified. Disaster recovery and business continuity plans are implemented.

- *phase G: implementation governance*
  Architecture artifact, design, and code reviews are organized. The necessary training is organized to ensure correct deployment, configuration, and operations of security-relevant subsystems and components. Security awareness trainings are organized.

- *phase H: architecture change management*
  Security-relevant changes to the environment are incorporated into the requirements for the following iterations.

The reference models in TOGAF identify a number of services that can be used to protect sensitive information in an information system. The *"Technical Reference Model"* (TRM) for instance proposes the following security services as building blocks: identification and authentication services, system entry control services, audit services, access control services, non-repudiation services, security management services, trusted recovery services, encryption services, trusted communication services. The *"Integrated Information Infrastructure Reference Model"* (III-RM) defines the following security services as building blocks: authentication, authorization, and access control, single sign-on, digital signature, firewall, encryption, intrusion detection, identity management, key management.

These building blocks are not sufficient. It is important to build a catalogue with additional standard security building blocks or security design patterns within the organization.

Different levels of patterns can be distinguished, for instance [2]:

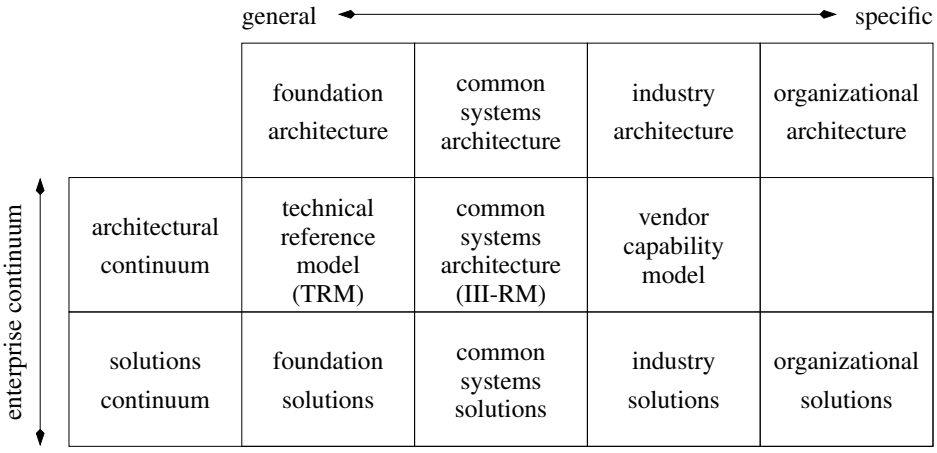

Figure 3: meta-model

- *"architecture patterns":* propose a structural organization for software systems, in the form of predefined subsystems, each with specific responsibilities, together with a set of rules and guidelines that define the relationships between them.

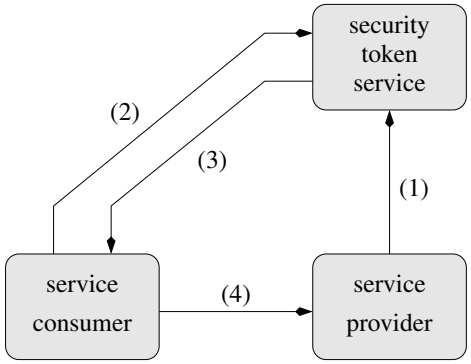

Figure 4: federated identity pattern

An example of such an architectural pattern is the federated identity pattern, shown in figure 4. The *"service provider"* trusts an *"identity provider"* (IdP) that exposes a *"security token service"* (STS) (1) to verify the identity of the *"service consumer"*. An interested consumer authenticates with the IdP, receives a token from the STS and presents this token to the service provider.

Other examples are for instance a gatekeeper pattern, a valet key pattern, but also for instance an information exchange gateway, a protected system pattern or a policy pattern.

- *"design patterns":* propose schema for defining the components of a software system and the relationships between them.

  An example of such a pattern is the *"distrustful decomposition"* pattern, illustrated in figure 5,

that breaks the server up into a number of separate processes, each one implementing a well-defined subset of the system functionality and running at minimum privilege level.

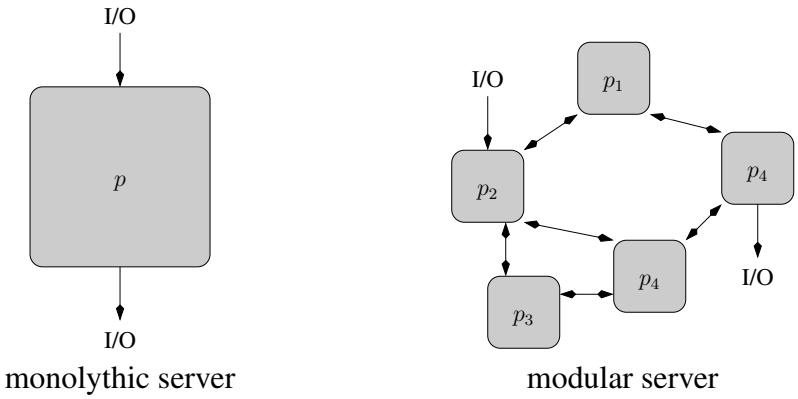

Figure 5: distrustful decomposition pattern

4.0 SABSA MODEL

An EA framework like TOGAF allows us to better align the information systems and services with the operational objectives of a military organization. Unfortunately, information security has long been considered a separate discipline, isolated from the enterprise architecture. For this reason we will combine TOGAF and its ADM cycle with the *"Sherwood Applied Business Security Architecture"* (SABSA) methodology. The SABSA model is based on Zachman's enterprise architecture and consists of six layers, as is depicted in figure 6.

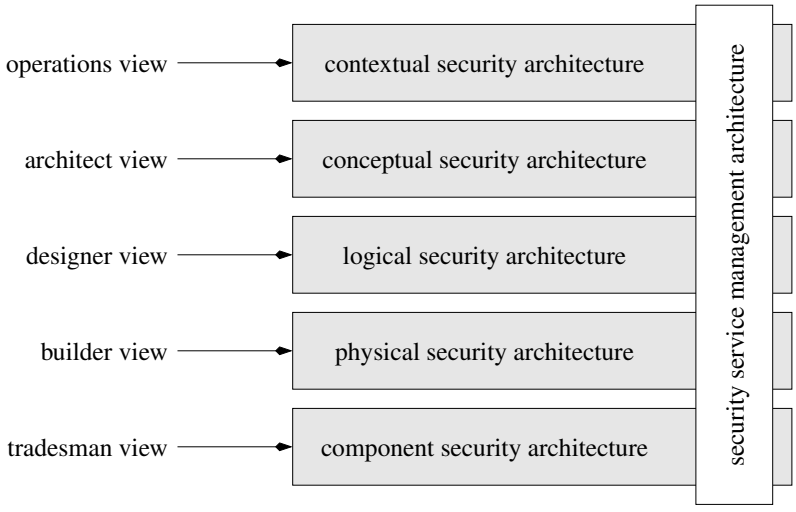

Figure 6: SABSA layered model

operations view: at the level of the "business" of military operations, the assets to be protected are listed, and the corresponding operational risks are identified. It will be these risks that drive the need for protection.

architect view: the architect is an artist who designs creative solutions at a higher level of abstraction. He defines the principles and fundamental concepts that will guide the subsequent selection and organisation of the logical and physical elements at the lower layers of abstraction.

designer view: the designer takes the architect's conceptual vision and as an engineer converts it into a logical design that can then later be implemented. This results for instance in security and risk management policy requirements (high-level security policy, registration authority policy, certification authority policy, physical domain policies, logical domain policies, etc.), the logical security services (entity authentication, confidentiality protection, integrity protection, non-repudiation, system assurance, etc.) that are required and how they fit together as common re-usable building blocks into a complex security system that meets the overall business requirements.

builder view: the builder translates the logical security services into physical security mechanisms and servers that will deliver these services. This means for instance specifying security mechanisms (encryption, access control, digital signatures, virus scanning, etc.), and the applications, middleware, operating systems, physical servers, communication lines, etc., that will implement these mechanisms.

tradesman view: In order to finally implement the solution a team of product specialists is brought together that installs, configures and integrates the products and component, which can be hardware, software or services, as specified by the physical security architecture.

Most existing security practices focus on threats against assets, with a particular attention to technical vulnerabilities in the IT infrastructure, and subsequently implement controls to mitigate the risks caused by these threats. Such a threat-based approach does however not necessarily provide the best information assurance for military operations. A commander may for instance consider that the most important asset to protect is the life of his soldiers who are engaged in an operation. A threat-based approach will typically result in complex information system and network architectures for managing and protecting command and control information, ensuring that logistic information is available, that military plans and intelligence information are kept confidential, etc. It may also ignore opportunities that information sharing, for instance sharing force protection information about *"improvised explosive devices"* (IEDs) with other nations in the context of a coalition task force, would offer since the negative impact of "not sharing" information is not being considered.

Designing an architecture begins with understanding the business, in our case the military organization and the operations it conducts, and define its specific business drivers and attributes. The drivers are determined by the organization's strategies, operational plans, and the key elements critical to its success. The organization's mission and vision statement often give an idea about the possible drivers. For a military organization a strategic objective could be *"operational excellence"*.

The drivers can then be reduced to their essence in the form of attributes that are required to satisfy the driver. In the case of the "operational excellence" driver this could for instance be the attributes "available", "safe", and "reliable". This results in so-called *"proxy assets"*, for instance in our case the driver "operational excellence" with the attribute "available", that have value to the organization and therefore must be protected. It is possible to define risks against these proxy assets; it is however difficult to assign a monetary value since they are often intangible and high level.

An inventory of proxy assets must therefore kept, which can then be used for an operational threat and risk assessment. When the risk assessment reveals competing confidentiality, integrity, availability requirements these conflicts can now be reconciled based on the operational impact thanks to the link through the proxy assets with the operational priorities.

Not only does the threat-based approach lead to more complex solutions incurring costs that could have been avoided, it furthermore does not consider potential benefits that could be realized when embracing certain additional risks. Starting from the business objectives it is indeed possible to focus on risk acceptance rather than risk avoidance. This then leads to the definition of *"key performance indicators"* (KPI) and *"key risk indicators"*. The KPI evaluate the performance with regard to the business attributes in the context of the associated business driver, whereas the KRI define a threshold or a condition that triggers a warning.

Consider for instance the availability of the *"common operational picture"* (COP) to the unit commanders in a combined task force. The KPI would measure the availability of the COP in support of the driver of "operational excellence". The KRI in this context might be an ongoing measurement of the time the COP is not available to at least one of the unit commanders, and an alert could be raised when this time exceeds a certain pre-defined threshold. In this way risk is being considered in terms of risk to the business, or in this case to the military operations, and the security architect together with the business architect can develop a security architecture that addresses these risks.

Alongside the proxy assets with KPI and KRIs, there are other models that are needed to cover the complete business risk spectrum. One additional attribute that needs to be modeled is *"trust"*, which must first and foremost be considered as a business characteristic rather than a technical one. In most military applications, in order to allow for two entities to interact and possibly exchange information, a trust relationship between the entities must first be established. The security architect will therefore identify all exchanges of information internally within the organization or with external entities where a certain level of trust must be assured. The level of assurance that is required depends on the criticality of the relationship and the sensitivity of the information that is exchanged. The security architect can then define the logical relationships and project these onto the physical IT environment.

Another model that is needed by the security architect is the *"threat model"*. It represents all known events or items that could cause harm to the organization. It can be a deliberate action, for instance originating from a hostile government agency, or an accidental unintentional realization of for instance a natural hazard. Knowledge about the threats, their likelihood of occurring and the impact they would have on the proxy assets, is essential for prioritizing the possible mitigating safeguards.

Precise knowledge of the safeguards is the last model that needs to be developed. The existing safeguards are to be modelled, as well as any known gaps with respect to the known threats. The relative maturity of the security controls and safeguards is determined using an industry framework or best practice.

Based on the proxy asset inventory, the trust relationships and the identified threats, the *"inherent risks"* are determined.

5.0 SECURITY, DEVELOPMENT AND OPERATIONS

DevOps is an agile-inspired approach that encompasses a set of cultural values together with the necessary tools and practices that enable the continuous development of software and its delivery into the production environment. One of the underlying concepts is to lower the risk associated with new releases and in this way make it possible to push more frequently smaller changes from the development to the production environment [4].

DevOps has its roots in the lean manufacturing concept, which aims at eliminating waste from the manufacturing process in a systematic way. In the context of software development the waste that is to be eliminated consists of low quality software that is produced yet ends up being not usable, unnecessary features that are developed but are never really used, etc.

One of the important outcomes of adopting the DevOps approach is that it brings the development and IT operations teams together. Developers can deploy code into production without needing an intervention from the operations staff. As a consequence however, they share the responsibility of supporting it once it is in production, making them allies rather than opponents.

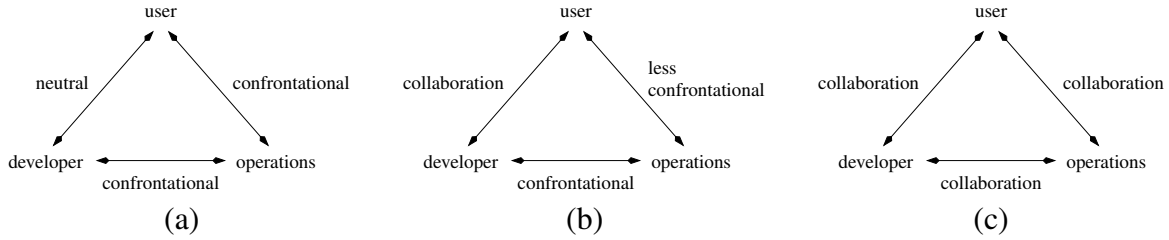

Figure 7: relationships between stakeholders

Figure 7 illustrates the evolution of the relationships between the stakeholders involved in developing, providing and consuming information services [6]. For the classic software development approach the relationships are depicted in figure 7(a). The developer interacts with the end user early in the development phase, when the user is not yet confronted with any of the problems the final operational solution will present, and therefore their relationship is neutral. The developers then design and build the system without necessarily taking into account the operational impact and complexity their design choices imply, resulting in difficulties when the system finally has to be put into production. Once the system is fully operational, the relationship between developers and operations staff becomes even more confrontational, since the developers want to update the software regularly in order to fix bugs and add new features, whereas the operational staff is reluctant to implement changes in order to preserve stability. The operations staff is furthermore confronted with the frustration of the users when new updates, produced by the development team, break the system, or cause performance problems, etc.

Figure 7(b) shows the influence of modern evolutions in the area of software engineering and service management on the relationships between stakeholders. Indeed, the adoption of service man-

agement frameworks like ITIL has reduced the conflicts between users and the operations staff, while incremental development methodologies like Agile have users and developers working closely together, with active stakeholder participation, prioritized requirements, etc. The relationship between developers and the operations people still remains confrontational however.

The aim of the DevOps approach is precisely to close this gap. It brings the operations people into the collaboration that already existed between users and developers, as is illustrated in figure 7(c). The operations people can now contribute to the design of new applications as well as to their implementation from the early start, and can support the whole process with virtualisation, configuration management and continuous integration tools.

The DevOps approach presents both challenges and opportunities with regard to security. Manual audits and formal reviews would result in an important obstacle for the DevOps way of working, and are therefore most often omitted or limited to major milestones. Furthermore, since a single developer can push changes all the way into the operational environments, he has end-to-end control. This violates the segregation of duties that can be required as a security control in certain applications or environment, in which case additional checkpoints need to be added.

DevOps does however also present a range of opportunities to the security officer. Indeed, when security teams are engaged in the design process from the earliest stages, the fast development and deployment of changes, the culture of continuous improvement, and the large extent in which testing and validation operations are automated, can clearly be used in a beneficial way for security. Small changes are easier to test completely, also for security. Security patches or improvements can incrementally be developed and pushed into production in a much faster way. The systematic use of configuration management tools makes it furthermore possible to more easily standardize secure system configurations, as well as to push the necessary tools and configurations for logging, alerting and generating security metrics. Therefore we are advocating a SecDevOps approach, where developers, operations staff and the security staff are all working together with eachother and with the user to satisfy the user's needs in the best possible way, as is illustrated in figure 8.

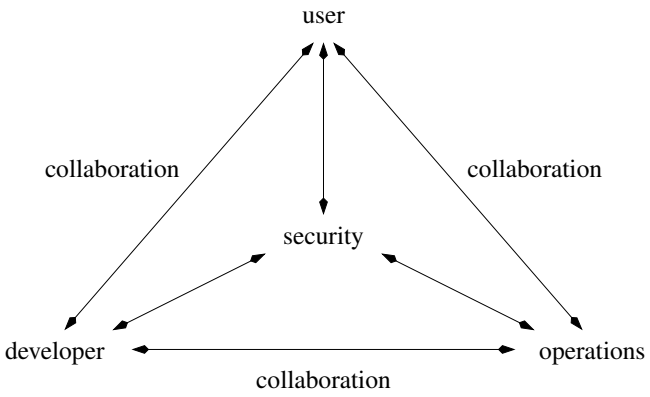

Figure 8: SecDevOps relationships between stakeholders

The Agile and DevOps teams have identified well-proven practices for managing functional requirements, for instance using the "prioritized requirements stack", also called the "product backlog' in SCRUM. Some teams create "technical stories" to capture non-functional requirements, in

a similar way as "user stories" are created to capture functional requirements. This works well for non-functional requirements that are self-contained and can be addressed in a finite period of time, such as producing documentation. Many non-functional requirements however cross-cut functional requirements, such as performance, availability, confidentiality, integrity, ...

The following approaches can help meeting the security requirements:

- *initial envisioning:* during the initial requirements envisioning the high-level security requirements will be identified by the security team member. These security requirements will then contribute in driving the architecture envisioning to ensure that it meets all the requirements.

- *model storming:* when the teams perform just-in-time modelling, as well for analysis model storming where the requirements are analysed, as for design model storming, a member of the security team with the appropriate skills shall participate. This will bring in the latest know-how regarding the evolving threat landscape, which can result in changing security requirements throughout the project. He can aso provide state of the art knowledge about the available controls, which will influence the design of the solutions, etc.

- *full-lifecycle testing:* the teams shall have members with analysis skills, design skills, programming skills, leadership skills, but also testing skills. This "whole team" strategy makes it possible to continuously test whether the requirements, and more in particular also the security requirements, are being met.

  As the members of the team work side by side, instead of having the developers develop code and then "thrown it over the wall" to the testers who then test it and report back to the developers with all the problems they found, over time the distinction between the rolls starts to blur and team-mates start picking up skills from each other. This will help in installing a basic security awareness in the minds of all the team members, resulting in a structurally more secure final product.

  In large scale complex projects, the whole team testing approach will however not be sufficient, and parallel independent testing by a dedicated test team will be performed throughout the project. This will typically be the case for verifying the security requirements of large system development projects in a military environment, which requires sophisticated skills and possibly expensive tooling. An independent security test team will then be created that provides recurrent testing support to a number of projects.

- *education & training:* educate developers so they are more aware of the security risks and learn to make better choices within their area of expertise in order to contribute to meeting the security requirements.

6.0 PITFALLS

6.1 Lack of strategic choices

Too often the information security strategy that is adopted in an organization tries to satisfy a multitude of conflicting demands and interests without making clear choices. It is often restating the obvious, hiding its lack of focus by using a language of broad goals, expressing a seemingly ambitious yet vague vision.

The strategy should provide an approach to overcome an obstacle, address a difficulty or respond to a challenge. The role of the commander is not just to motive his people to "push until they get there" but also to create as a strategist the conditions to make that push that he requests effective. Identifying the key challenges is therefore not enough, the commander must also develop an overall approach to deal with these challenges instead of simply restating the desired outcome or the challenges themselves. It is up to the commander to select a few, pivotal objectives and focus the limited available resources on achieving these objectives, that may trigger a cascade of favourable outcomes.

A good security strategy typically has the following underlying structure:

1. a *diagnosis* reducing the complexity of the real world problem to a limited number of critical challenges,
2. a *guiding policy* that defines the approach to be applied to address these challenges,
3. *coherent actions* that support the implementation of the guiding policy.

The information system landscape in an organization evolves based on the requirements of the organization. These requirements are typically made up of functional and non-functional requirements, with security requirements being a subclass of the non-functional requirements. Since resources are inevitably limited, there is a competition between the requirements, and since non-functional requirements are typically not directly visible to the end-users, they are often given too little attention during the development process.

In order to avoid this, Agile experts propose to treat functional and non-functional requirements in exactly the same way, and call them for instance simply "features". This is illustrated in figure 9 where the priority of these features is subsequently assessed and the features are added to the queue of work items without distinguishing them. Cohn [3] proposes to address both "end-user features" (the functional requirements) and "operational features" (the non-functional requirements) in the same way using "user stories".

6.2 Mismatch between ambition and available resources

In order to implement an information security strategy, people are a key resource. It is essential to effectively utilize the available know-how at the right places by allocating the people to their most useful tasks, rather than having important pieces of knowledge unused due to a wrong allocation of human resources.

Information security related skills are a scarce resource. As experts with these scarce skills tend to get over-solicited, projects take longer than planned with a knock-on effect on subsequent projects.

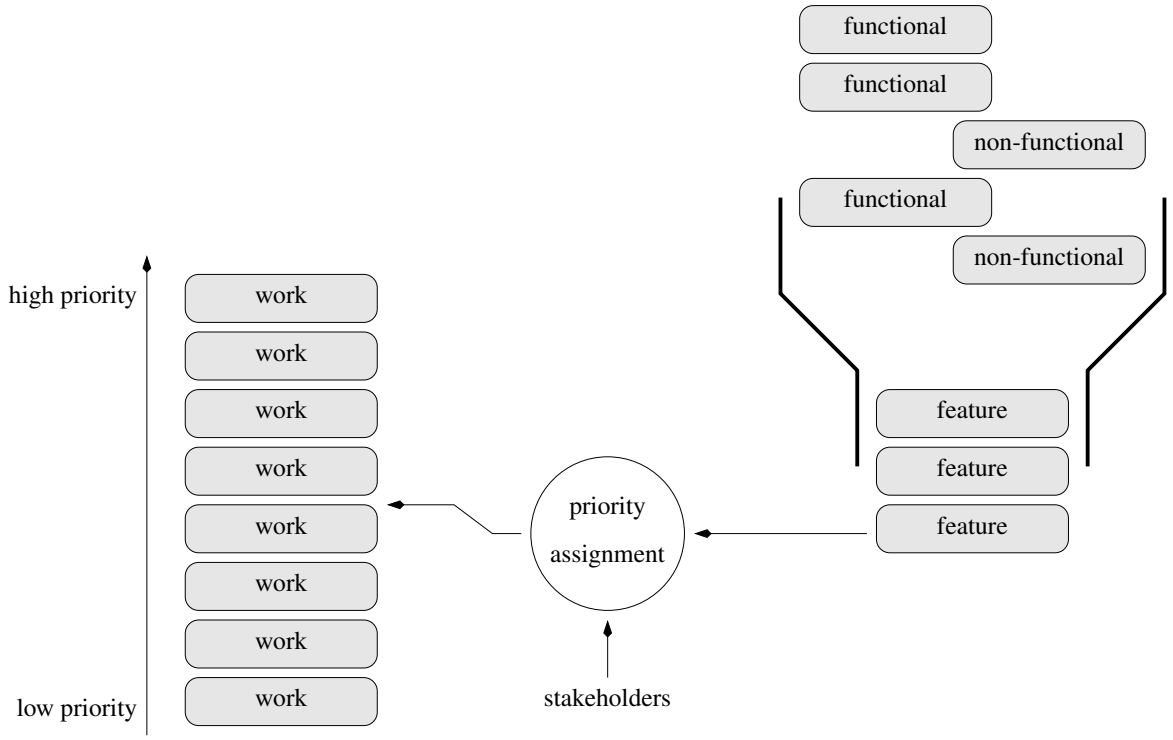

Figure 9: work item queue

Because of the unavailability of the most appropriate staff, less appropriate personnel will be assigned to subsequent projects which again increases the chance of problems, and even longer overruns for these projects.

Figure 10(a) shows the classic software development approach, with different experts playing their role sequentially. When due to a lack of resources some posts are occupied by less skilled employees, for instance because they are new and still learning the ropes, this will have an important effect on the quality of the output. In the Agile approach however, the serial processing is replaced by a team working in close collaboration. Everyone brings in the skill set he has, and in the end all the skills that are required for the project to succeed must be covered by the team as a whole. In this way, newcomers can be integrated in an existing strong team and learn on the job.

This team collaboration works even better when the members are *"generalizing specialists"* [1]. This means they have one or more technical specialities, at least a general knowledge of software development and of the business domain in which they work, and are actively seeking to gain new skills in both their existing specialities as well as in other areas.

Hodgetts [5] advocates that having specialists on the team can result in a loss of team productivity, for instance because it reduces the flexibility when dynamically assigning tasks to individuals as the work progresses. Specialization furthermore forces the break down of tasks to be finer-grained, resulting in more hand-off points between people, with each time the risk of information loss or misunderstandings and therefore increasing the overall project risk. Finally specialists tend to become bottlenecks as work queues up for them, potentially blocking progress for the whole team.

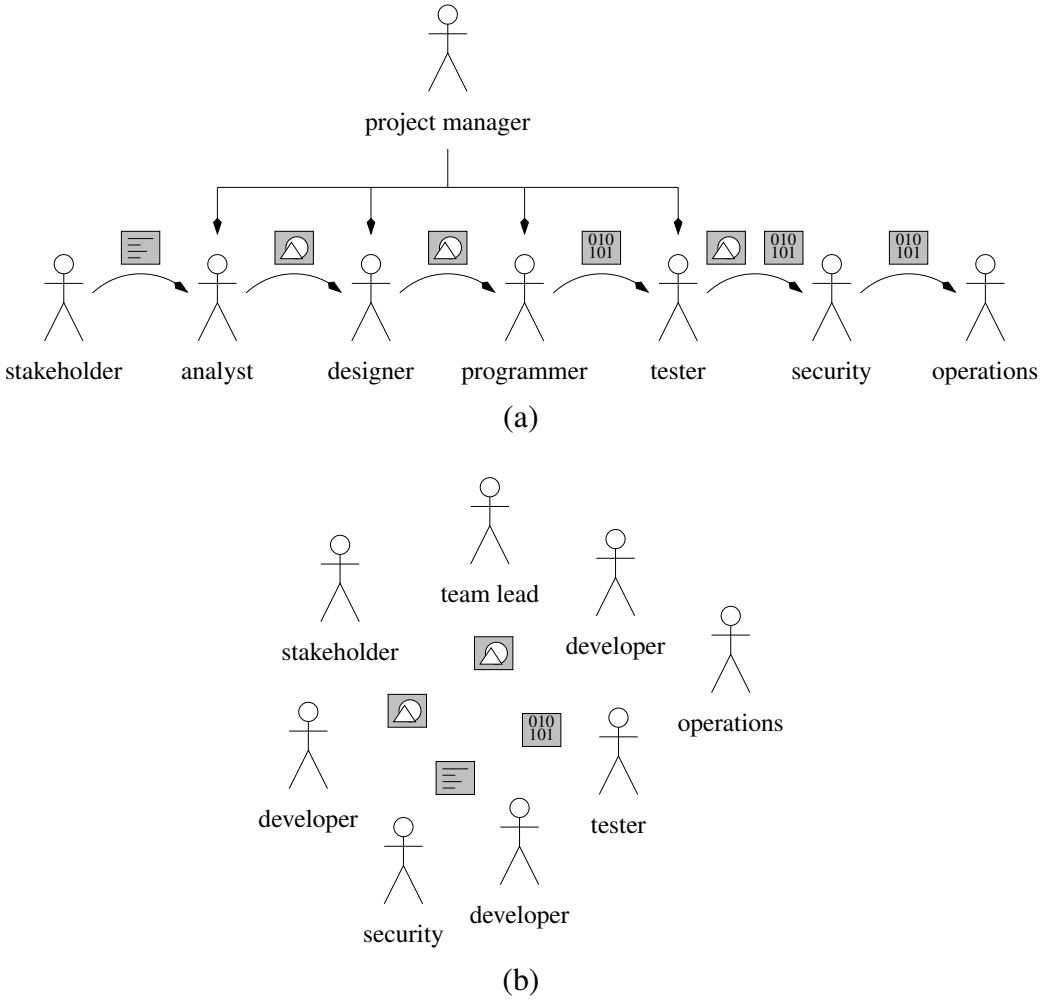

Figure 10: classic versus Agile/DevOps software development

We therefore advocate to promote the culture of becoming generalizing specialists. For new teams it is important to have a security specialist involved, but as the team advances, one or more of the team members can increase their security knowledge and the security specialist is no longer required most of the time.

6.3 Lack of clear communication

When a strategy has been adopted, its developers often don't translate it into useful information for the people to understand what they are supposed to do with it. When new initiatives are taken, they are too often not communicated throughout the organization in terms of which steps to take, in which time-frame, etc. This results in plans that silently die months after they were launched.

Once the security strategy has been translated into clear goals at the level of the organization, they must be communicated throughout the hierarchy, translating them into group and individual level plans. People must clearly see how they contribute to the broader organisational and governmental

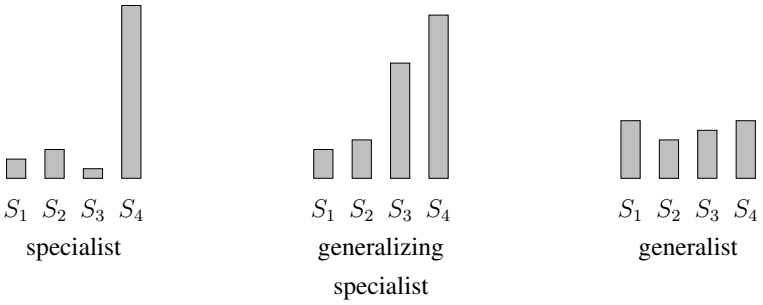

Figure 11: from specialist to generalist

goals, how they are making a difference.

In the Agile/DevOps teams the generalizing specialist typically has a good overview of what his team-mates are working on, and can bridge between them when needed. A generalizing specialist will furthermore often be intuitively inclined to play this role since it will allow him to learn something new. Specialists on the other hand have more difficulty grasping what another specialist is doing, the problems the other one is facing, and as a result sometimes even look down on the work of others, which may result in communication barriers. Yet another reason why the culture of having generalizing specialists on the team should be encouraged.

Specialists tend to communicate using the tools they know, for instance a business rule specification, a logical data model, a UML diagram, acceptance tests, etc. This results in duplication, with information represented in different ways that needs to be reviewed and maintained and in the end a good chance of having ambiguities or contradictions. Again, generalizing specialists will write less documentation than specialists because they have more options available to them and can align on one or a limited number of common formats.

6.4 Maximization instead of optimization

Often the team working on a project will try to obtain the best possible result, but spend a lot of resources in doing so, whereas if they would have just aimed at a sufficiently good result, they could have started much earlier taking on another project.

Figure 12 shows the “prioritized requirements stack”, as it is known by the Agile practitioners. The requirements are sorted based on priority. This priority is determined together with the stakeholders and evolves as the project goes forward and new insights are gained. The stack not only contain “requirements” but all types of work that is to be done, for instance addressing defects that were reported.

The team will furthermore perform look ahead modelling, which means that they do just enough modelling to understand an item on the stack, but not more. Indeed, nowadays for big software development projects, the software engineering methodology often mimics the classic civil engineering approach of designing the whole construction in detail before laying the first brick. This is sometimes referred to as “big design up front” (BDUF) and often follows after a “big requirements up front” (BRUF) phase. For software development projects there is however no need to follow this approach

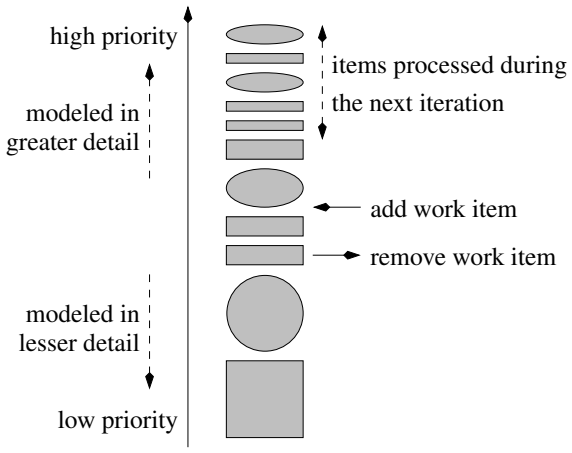

Figure 12: prioritized work item queue

since it is perfectly well possible to develop and deliver software incrementally.

Modelling as late as possible results in better modelling decisions since they are based on more mature knowledge about the problem. It furthermore increases the willingness of the developers to accept evolving requirements from the stakeholders.

REFERENCES

[1] Scott W. Ambler and Mark Lines. A practitioner's guide to agile software delivery in the enterprise,, 2012.

[2] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. Pattern-oriented software architecture: a system of patterns. 1996. *Part II*, 2001.

[3] Mike Cohn. *User stories applied: For agile software development*. Addison-Wesley Professional, 2004.

[4] Jennifer Davis and Katherine Daniels. *Effective DevOps: Building a Culture of Collaboration, Affinity, and Tooling at Scale*. " O'Reilly Media, Inc.", 2016.

[5] Paul Hodgetts. Experiences integrating sophisticated user experience design practices into agile processes. In *Agile Development Conference (ADC'05)*, pages 235–242. IEEE, 2005.

[6] Paramu Kurumathur. A devops architecture. PM Power Consulting Blog, January 2015.