MAPE-K/MAPE-SAC: An Interaction Framework for Adaptive Systems with Security Assurance Cases

Sharmin Jahan[1], Ian Riley[1], Charles Walter[1], Rose F. Gamble[1]
Matt Pasco[2], Philip K. McKinley[2], Betty H.C. Cheng[2]

1. Tandy School of Computer Science, University of Tulsa, Tulsa, Oklahoma, 74104, USA
2. Department of Computer Science & Engineering, Michigan State University, East Lansing, Michigan 48824, USA

Corresponding author(s)
Rose F. Gamble (gamble@utulsa.edu)

Abstract
Security certification establishes that a given system satisfies properties and constraints as specified in the system security profile. Mechanisms and techniques have been developed to assess if and how well the system complies with the properties, thereby providing a degree of confidence in the security certification. Generally, certification of security controls defined by NIST SP800-53 is performed at design time to provide confidence in a system's trustworthiness to achieve the organization's mission and business requirements. Assuring confidence in a self-adaptive system's security profile is challenging when both functional and security conditions may change at run time. Static security solutions are insufficient, given that dynamic application of defense mechanisms often needs to dynamically adapt security functionality at run time as part of self-protection. This security adaptation may hinder maintaining functional constraints or vice versa. In addition, adaptation capabilities may give rise to the need for dynamic certification, which can be a difficult procedure given the complexity of the security dependencies. Confidence in an information system's compliance with security constraints can be expressed using security assurance cases (SACs). NIST security controls are defined with a hierarchical structure that makes them amenable to being specified in terms of SACs. A collection of SACs for related security controls form a network that can be used to measure the confidence of security compliance through certification-based evidence. Once the system is deployed, environmental and functional uncertainties may require the coordination of functional and security adaptations. This paper introduces the MAPE-SAC, a security-focused feedback control loop, and its interaction with a MAPE-K, function and performance-focused control loop, to dynamically manage run-time adaptations in response to changes in functional and security conditions. We illustrate the use of both control loops and their interaction with an example of two independent systems that need to cooperate to facilitate autonomous search and rescue in the aftermath of a natural disaster.

Keywords: Security assurance cases, self-adaptation, security certification, MAPE loop

1. Introduction

Guaranteeing trustworthiness is a prime concern for self-adaptive systems, since they must manage uncertainties in a dynamic environment [1, 2, 3]. Uncertainty may arise due to faulty hardware and software components, malicious attacks, and unpredictable environmental and/or functional changes. Effectively maintaining the trustworthiness of the system within uncertain situations requires the capability to detect and mitigate security threats at run time and deploy autonomous, self-protecting mechanisms to adapt security functionality accordingly [2]. Like self-adaptive systems, self-protecting systems should include a MAPE (Monitor, Analyze, Plan, and Execute) control loop specifically for managing security concerns. However, adapting security functionality may hinder the system's ability to maintain its other quality concerns [4]. While a self-adaptive system automatically establishes behavioral changes to preserve its functional objectives, changing functional behavior as a part of an adaptation may result in security vulnerabilities. Thus, the adaptation decisions of self-protecting systems need to consider compliance with both functional and security concerns.

When the environment is dynamically changing, the ability to assure compliance with security constraints requires a rigorous process [1] in order to ensure system integrity. To provide this assurance, U.S. federal information systems and organizations have in place security certification and accreditation (C&A) processes that establish a baseline of security controls from NIST SP800-53 [5] (which covers the international ISO/IEC 27002 standard [6]). Assessment processes described in NIST SP800-53A include examining the system (e.g., its model, architecture, and design documentation), testing the system, and interviewing personnel related to its construction. C&A is repeated at scheduled intervals or when there is a major functional change, environmental change, or emerging threat. The creation and maintenance of *assurance cases* [7] provide a method to express and justify a measure of confidence that a system complies with requirements [8].

Assurance cases use a structured set of arguments with evidence to provide a level of confidence that a system satisfies specific claims with respect to its requirements. Typically, assurance cases are developed during design time with significant human effort given the heavy use of textual descriptions. *Security Assurance Cases* (SACs) are goal-based models that include argumentation to achieve security goals, thereby providing confidence in a system's security status [9, 10]. SACs are modeled using Goal-Structured Notation (GSN) [11], reflecting control definitions and arguments substantiating their compliance. Previously, we introduced a template for SACs [12] and demonstrated an adaptation operation on an SAC model [13].

Because protecting a system from threats can involve both security and functional concerns, maintaining a certain confidence level of the system's requirement compliance following an adaptation is challenging [14]. Specifically:

- Static solutions are insufficient since functional and security conditions can change at run time, and
- Dynamic solutions without coordination can lead to an increase in requirement violations since functional adaptations can violate security requirements and vice versa.

Any appropriate solution requires a complex configuration, management, and analysis process to detect and mitigate uncertainty across regions of concerns. Requiring a single centralized MAPE loop to handle different components or concerns can be limiting. An alternative is to employ multiple, interacting and decentralized MAPE loops [15, 16].

This paper describes a framework that supports the management of two interacting MAPE loops, MAPE-K and MAPE-SAC, to respectively govern functional and security-based concerns

of an adaptive system. In both cases, GSN models are used to capture the assurance cases for the respective compliance concerns that are used to guide the adaptation process. We introduce a MAPE loop interaction protocol to support the coordination between the MAPE-K and MAPE-SAC loops. Our MAPE-K and MAPE-SAC coordination approach selects a system-level adaptation that minimizes compliance degradation across functional and security requirements. Coordination is conducted during planning and execution to ensure adaptive system behavior with optimal utility while maintaining security compliance. We implement, apply, and evaluate our MAPE loop interaction and assurance case adaptation using a robotic-based search-and-rescue scenario, emulated on an experimental testbed, where autonomous robots and wearable health monitoring devices must interact in order to locate victims in need of immediate medical treatment.

A short, preliminary paper [17] presented the conceptual aspects of this work, with a proof of concept application to a different demonstration platform that focused on an abstraction of the wheel speed sensor of an autonomous rover. We proposed the initial separation of security and functional concerns for adaptation assessment and introduced the MAPE-SAC loop to manage security concerns. An abstraction of the MAPE loop interaction based on a modified Regional Planning pattern [15] was described, but without any implementation or evaluation. This paper leverages concepts from our preliminary work [17] and significantly extends the prior work, both conceptually and with validation on new applications and new platforms. The remainder of this paper is organized as follows. Section 2 provides background material and related work. Section 3 briefly overviews the NIST security controls that we use in this work. The MAPE-K/MAPE-SAC interaction framework is introduced in Section 4. Details of the coordination between the MAPE loops are provided in Section 5. Evaluation of the interacting MAPE-K/MAPE-SAC framework is presented in Section 6, followed by a discussion of the utility and the limitations in Section 7. Finally, we summary the work and discuss future work in Section 8.

2. Background and Related Work

The MAPE-K loop, shown in Figure 1, was designed to manage the adaptation of autonomic systems [14]. The Monitor step collects information from managed resources and detects conditions that trigger adaptations. The information is analyzed (Analyze step) to determine if change is needed to satisfy system goals. If an adaptation is needed, then the Plan step creates a procedure to realize a new target condition that satisfies the goals (including the intermediate steps that occur when adapting from one state to another). Finally, the planned procedure is executed (Execute step) on the managed resources. The Knowledge component comprises a set of requirements the MAPE-K loop uses for adaptation parameters and strategies.

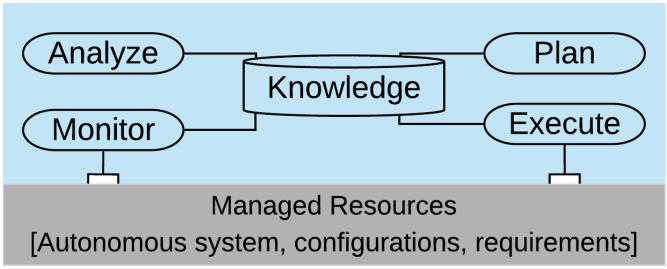

Figure 1: MAPE-K Loop

Many modern autonomous systems are distributed in nature and maintain a large, complex structure. Deploying multiple MAPE loops in the managed system and coordinating the components of the MAPE loops in a decentralized manner is crucial. Vromant et al. [16] proposed

an extension to the MAPE loop to support intra-loop and inter-loop coordination. When coordination is possible, multiple MAPE loops can interact with each other to jointly select an adaptation. Weyns et al. [15] discussed patterns of interacting MAPE loops and proposed notations to describe the patterns. The authors also suggested drivers for selecting an interaction pattern to aid future researchers in developing a systematic approach for describing different types of control in self-adaptive systems.

Industries that (1) work with U.S. federal information systems, (2) rely on controlled unclassified information, or (3) provide services to the U.S. Department of Defense supply chain are contractually required to comply with clauses that relate to the NIST SP800-53 security controls [5] or some subset thereof, such as the NIST SP800-171 [18]. The process by which compliance is determined includes certifying that the systems under consideration have executable mechanisms to maintain security control effectiveness at an acceptable confidence level. Ross et al. [19] identified four phases of the security certification and accreditation process and provides guidelines for the certification process of a federal information system. Anisetti et al. [20, 21] showed that the basic security certification process starts with building a certification model of the target system, is driven by security properties, and produces evidence that supports the properties that need to be certified. Security properties can be expressed in notations of formal logic such as LTL [22], A-LTL [23], or AutoRELAX fuzzy logic [24]. Anisetti et al. [20, 21] proposed test-based evidence to support the properties being certified, whereas Damiani et al. [25] proposed using formal proofs of abstract certification models as evidence of certification. Marshall et al. [26] restate security controls in LTL and define a formal process to verify them within a system using an automated theorem prover at design-time.

Assurance cases provide confidence of the system's trustworthiness. An assurance case model is a hierarchical construction of a claims-argument-evidence structure where a claim involves an argumentation and evidence provides support to the argumentation for the claim's acceptance [7]. Understanding assurance case structure has been challenging given the previous heavy reliance on text-based descriptions but can be facilitated through graphical argumentation notation [27, 28]. Kelly and Weaver [27] proposed Goal-Structured Notation (GSN) where each claim is stated as a goal and evidence is represented as a solution. Assurance cases are widely used in the safety domain [1, 8], but with increasing security awareness, researchers are exploring the benefits of SACs. Alexander et al. [1] demonstrated that significant differences exist between the safety and security domains and showed that the latter exhibits more uncertainty due to a lack of knowledge of potential attackers and their capabilities. SACs were first proposed by Goodenough et al. [10]. SACs provide confidence that the system is "acceptably secure." Acceptance not only requires having specific technology, but also requires people and processes to assess the system. Guidelines have been developed to structure claims asserting that a system is sufficiently secure [9, 10]. These guidelines include what should be considered to select and elaborate strategies and choose suitable evidence.

Dynamic environments or functional adaptations may result in security threats at run time, which may affect the system's security functionality, security assurance, and trustworthiness [4, 29]. Thus, having security capabilities as part of self-protection is important. Security capabilities include assessment on security requirements, features, components, and a measure of the system's trust level [5]. Self-protecting systems should be able to respond either reactively to mitigate security threats or proactively to anticipate security threats [2]. At run time, certification of adaptations must be performed quickly and effectively to ensure that system security has not been reduced below an acceptable confidence level. Without an efficient certification process, the adaptive system may be vulnerable to attack.

Previously, we proposed a runtime risk assessment approach where formal proof of requirement compliance is performed against the originally deployed code of the system, and metadata of code architecture is extracted for analysis [30]. These metadata contain information about potential state variables; their association with the system's functions, methods, and components; and their condition and impact on the original verification process. Adaptation may add/delete/modify the system's functionality, which inhibits the reuse of the original proof process. We characterized the risk of reusing the original requirement compliance verification process after an adaption introduces changes [30]. In related work [12], we identified a pattern in state security controls defined in NIST SP800-53 and defined a template to model security assurance cases for security controls as shown in Figure 2. We have also previously introduced adaptation operations that can be used to evolve SAC instances and a dynamic assessment that can be used to evaluate the confidence level of an adapted SAC [13]. The assessment process must calculate the impact of security and functional changes to determine an achievement weight of each goal to assure security compliance. But changes to one security control may propagate to other interrelated security controls. We determine satisficing levels using the achievement weight of security controls as a part of the assessment of a system's compliance [13].

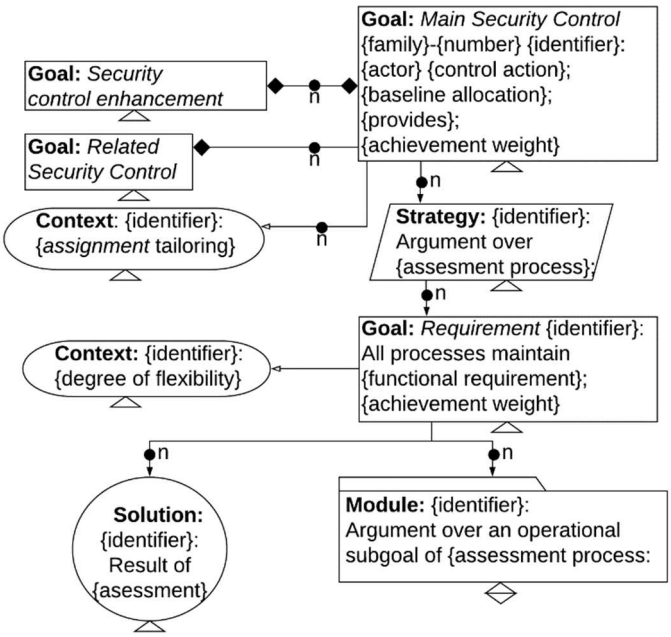

Figure 2: Template to instantiate security assurance case for NIST SP800-53 security control

3. Working with NIST SP800-53 Security Controls

Prior work defined a template for this form of SAC using GSN [12] by extracting structural patterns from security control statements, and we utilized that work to construct a SAC for NIST SP800-53 [18] security control SI-7(5) [17]. This paper extends prior work to a new set of interrelated security controls—AC-3, AC-18, AC-21, IA-3, and SC-40—covering access control (AC), information and authorization (IA), and system and communications protection (SC). Table 1 shows the organization of the certification guidelines defined by the Joint Task Force [31] for these security controls.

AC-3, which relies on security control AC-18 and AC-21 for determining access requirements, restrictions and circumstances, asserts that there must be steps to enforce authorization of logical access to information and system resources. SC-40 asserts that the information system should protect wireless links from defined signal parameter attacks. SC-40 is indirectly related to IA-3,

which asserts that the information system should identify and authenticate devices before establishing a connection. Both SC-40 and IA-3 rely on AC-18 for obtaining the guidelines to implement protection and authorization mechanisms. The design evidence for the system's trustworthiness is formulated through certification assessment methods of examination, interviewing, and testing [31]. We instantiate a security control network (SCN) according to the labels, parameter values, and formal statements in [31] as shown in Table 1.

Table 1: Interrelated Security controls as a part of information security defined by NIST SP800-53A REV. 4

AC-3: ACCESS ENFORCEMENT		
ASSESSMENT OBJECTIVE: Determine if the information system enforces approved authorizations for logical access to information and system resources in accordance with applicable access control policies		
AC-18: WIRELESS ACCESS		
ASSESSMENT OBJECTIVE: Determine if the organization:		
AC-18(a)	establishes for wireless access:	
	AC-18(a)[1]	usage restrictions;
	AC-18(a)[2]	configuration/connection requirement;
	AC-18(a)[3]	implementation guidance; and
AC-18(b)	authorizes wireless access to the information system prior to allowing such connections	
AC-21: INFORMATION SHARING		
ASSESSMENT OBJECTIVE: Determine if the organization:		
AC-21(a)	AC-21(a)[1]	defines information sharing circumstances where user discretion is required;
	AC-21(a)[2]	facilitates information sharing by enabling authorized users to determine whether access authorizations assigned to the sharing partner match the access restrictions on the information for organization-defined information sharing circumstances;
AC-21(b)	AC-21(b)[1]	defines automated mechanisms or manual processes to be employed to assist users in making information sharing/collaboration decisions; and
	AC-21(b)[2]	employs organization-defined automated mechanisms or manual processes to assist users in making information sharing/collaboration decisions.
IA-3: DEVICE IDENTIFICATION AND AUTHENTICATION		
ASSESSMENT OBJECTIVE: Determine if:		
IA-3[1]	the organization defines specific and/or types of devices that the information system uniquely identifies and authenticates before establishing one or more of the following:	
	IA-3[1][a]	a local connection;
	IA-3[1][b]	a remote connection; and/or
	IA-3[1][c]	a network connection;
IA-3[2]	the information system uniquely identifies and authenticates organization-defined devices before establishing one or more of the following:	
	IA-3[2][a]	a local connection;
	IA-3[2][b]	a remote connection; and/or
	IA-3[2][c]	a network connection;
SC-40: WIRELESS LINK PROTECTION		
ASSESSMENT OBJECTIVE: Determine if:		
SC-40[1]	the organization defines:	
	SC-40[1][a]	internal wireless links to be protected from particular types of signal parameter attacks;
	SC-40[1][b]	external wireless links to be protected from particular types of signal parameter attacks;
SC-40[2]	the organization defines types of signal parameter attacks or references to sources for such attacks that are based upon exploiting the signal parameters of organization-defined internal and external wireless links; and	
SC-40[3]	the information system protects internal and external organization-defined wireless links from organization-defined types of signal parameter attacks or references to sources for such attacks.	

Figure 3 shows the interrelated security controls of AC-3, AC-18, AC-21, IA-3, and SC-40 at a high level. Control statements are stated as main goal statements and the "provides" attribute holds the provision set of state variables' value (val) and conditions (sat) as a part of the security control's dependencies for compliance. This set flows within the network through a SupportedBy link, augmented with a diamond to designate the security control source for the provision set, which implies the interrelationship among the security controls.

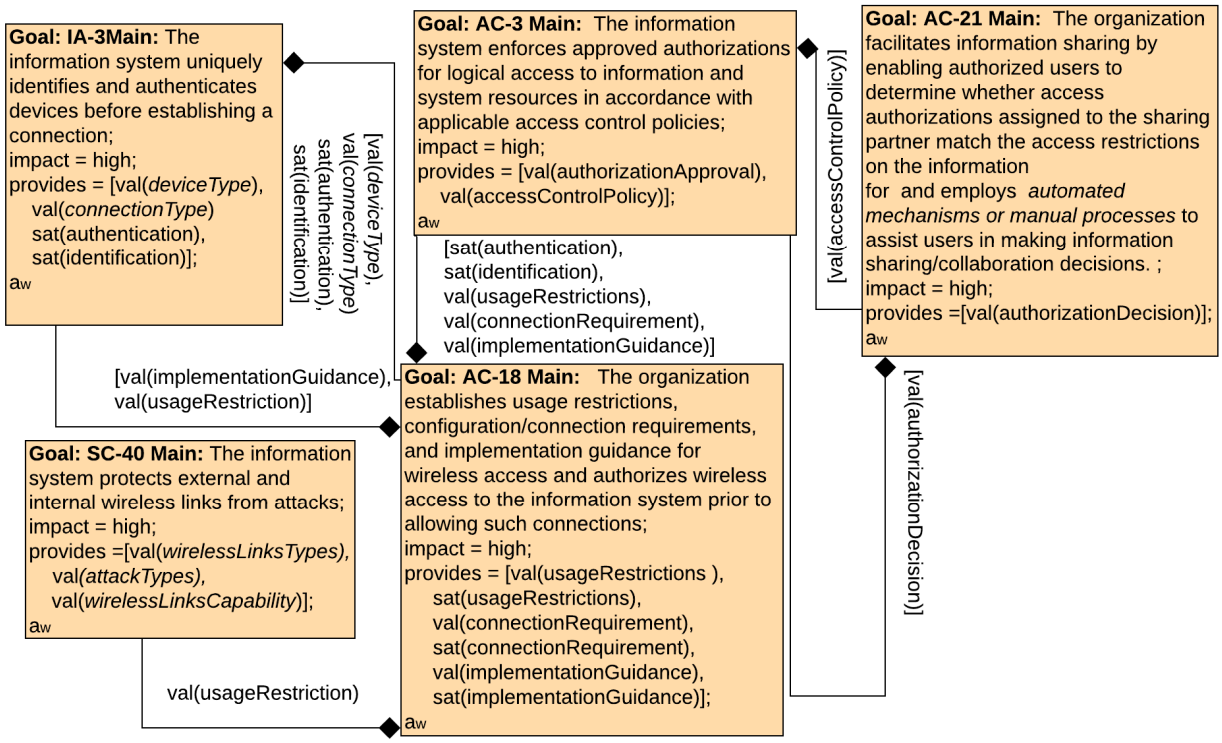

Figure 3: Instance of security control network for access control (partial)

Figure 4 shows the SAC for AC-3 with the control statement as the main goal. Organizationally defined values form the context (pink ellipses) of the applicable access control policies (AC-3), with an identity-based policy as the chosen access control policy. As a related control to AC-3, AC-21 is an Away Goal, a separate but related argument that provides or requires support from the main goal, provides specific parameters to AC-3 for its compliance strategy. AC-18 is another Away Goal which requires specific parameters from AC-3 (as does AC-21) to certify its own compliance strategy. Strategy *AC-3 S1* denotes the mechanism, *enforceAuthorization*, supporting the argumentation of AC-3's effectiveness. *AC-3 Req1* is a sub-goal where LTL or other propositional logic can be used to formally express requirements unambiguously. *AC-3 Req1* is further decomposed into modules that are expanded into supporting operational goals directly associated with system code as shown in Figure 5. Solutions, shown in the green circles, are evidence to support the certification process of security control AC-3. We have also modeled security assurance cases for security controls AC-18, AC-21, IA-3, and SC-40 and deployed them as a part of the knowledge of the MAPE-SAC control loop discussed in detail in Section 3.

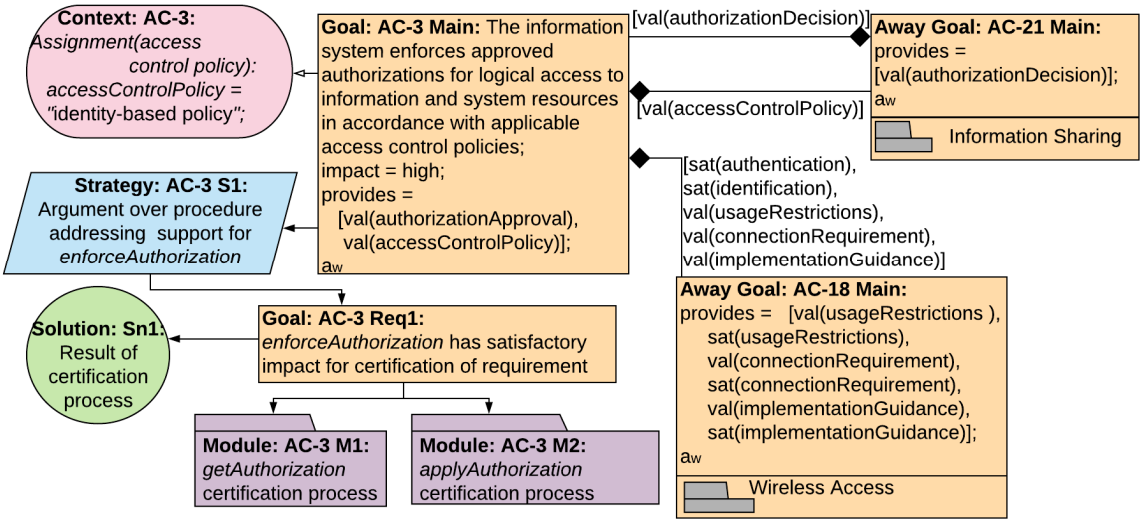

Figure 4: AC-3 instantiated within the SAC GSN Template

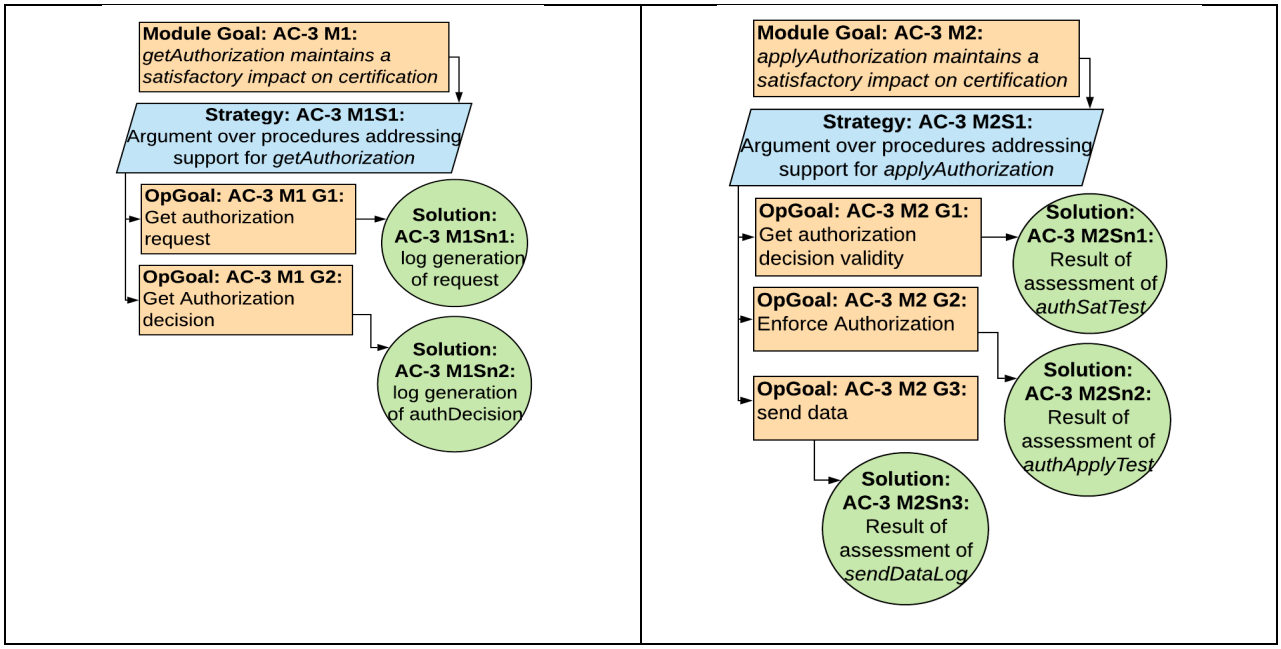

Figure 5: Expanded AC-3 Module Goals to operational goals needed to enforce authorization

4. MAPE-K/MAPE-SAC Interaction Framework

The MAPE-K/MAPE-SAC Interaction Framework is motivated by the need for security-based self-protection of autonomous systems. The objective is to monitor run-time compliance with security controls using the SACs and adapt SACs as appropriate in coordination with the MAPE-K loop, which monitors run-time compliance with functional concerns. Similar to the MAPE-K loop, the MAPE-SAC loop comprises four main steps: monitor, analyze, plan, and execute, shown in Figure 6. It relies on resources tailored to security requirements and the confidence in the compliance needed for the system to maintain a threshold of acceptable risk.

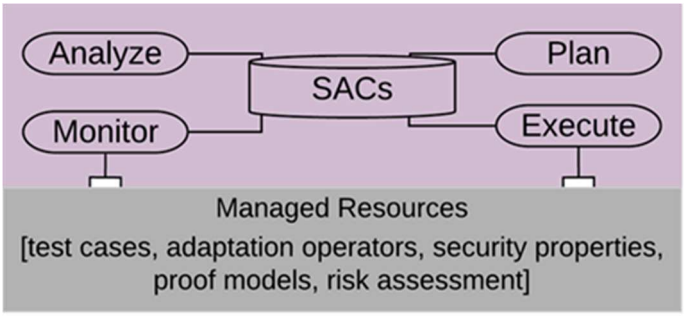

Figure 6: MAPE-SAC loop

The monitor step examines the security control effectiveness of the system, which includes functional and environmental changes. SACs used in this work reflect security controls and the certification process guidelines, which are outlined in NIST SP800-53A [31], to certify goal satisfaction. The template has been instantiated by providing the parameters' values, and the confidence level of the certification process has been encoded as a set of achievement weights for each goal in the SAC. Thus, a model of the concerned security control's assurance case is developed and, in the monitoring step of the MAPE-SAC loop, monitored against functional and environmental conditions.

The security control's effectiveness is assessed in the analyze step using information collected from the executing system pertaining to possible security threats. The analyze step must reliably identify when a system or environment change requires a response and assesses the impact of any security threat on the goals of a SAC. A security threat has the potential to propagate through a network of related security controls and cause the overall system's compliance to degrade [13]. Thus, analysis must assess each security threat's impact on the security control network and trigger an adaptation when it observes significant compliance degradation

The system reconfiguration cost and the level of security confidence maintained are the main planning considerations. Thus, the planner should dynamically evaluate potential adaptations by assessing the effectiveness of security controls over the network of SACs. Coordinated planning between MAPE-K and MAPE-SAC loops enables system-wide adaptation, which is discussed further in Section 5. Finally, during execution, the system is reconfigured to the newly adapted state, requiring one or more SACs to be updated.

We assume functional and security concerns represent two different regions of the system with their own mechanisms for evaluating an adaptation. However, a system-wide adaptation has global objectives to which it must comply by achieving an optimal global utility. As was stated above, any adaptation to one region may affect the requirements of the other region. System-wide adaptations must therefore be coordinated between each regional planner in order to plan and execute a unified adaptation plan. We adopt a modified Regional Planning Pattern [15], where MAPE loops coordinate both in the planning step and the execution step.

The interaction between the MAPE-K and MAPE-SAC loops is represented in Figure 7. The Monitor and Analyze components in both MAPE loops monitor and analyze their respective concerns in parallel. Adaptation can be triggered by either or both of the MAPE loops. The loops coordinate to plan the optimal adaptation of the system that is compliant for both functional and security requirements. When an adaptation has been triggered for either type of requirement, the coordinated planner identifies potential changes to functional and security behavior due to the adaptation, and each MAPE loop planner assesses the effect of the adaptation on its own concern. MAPE loop planners coordinate with each other to reach the optimal adaptation with respect to both types of requirements. The optimal adaptation is the adaptation which best satisfies the compliance level of the system's security requirements without risking the proof that supports the

system's compliance with its functional requirements. The chosen adaptive plan is executed in a coordinated manner to evolve regional assurance cases along with changes on the system-level code.

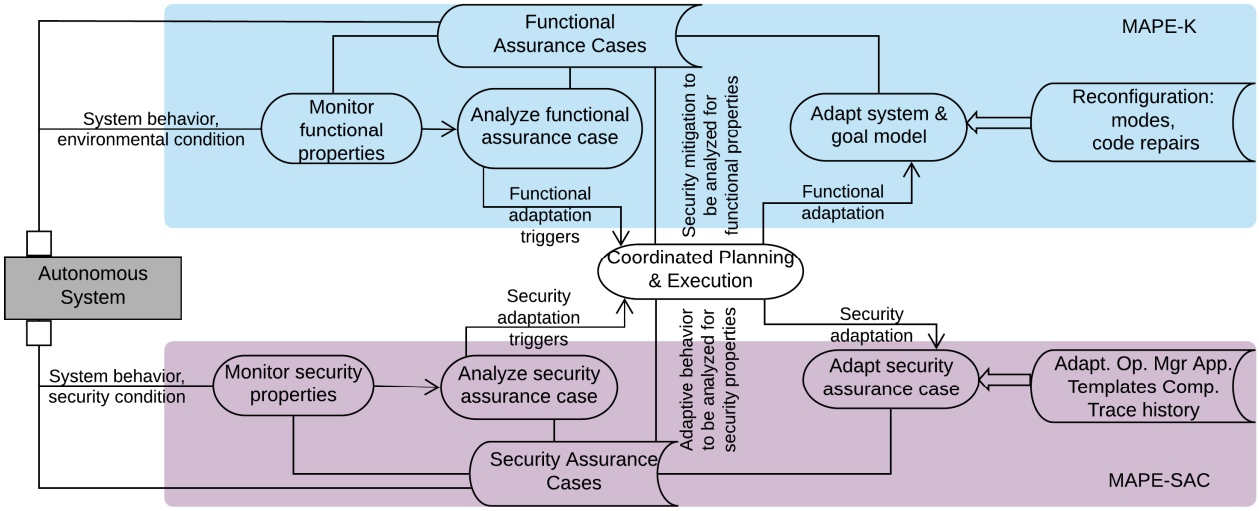

Figure 7: Interaction between MAPE-K and MAPE-SAC loops

5. Coordination of Interactive MAPE Loops

To illustrate the MAPE loop interaction, we consider a scenario in which networks of health monitoring devices and autonomous robots need to adapt and interact during search and rescue operations. Each system operates its own MAPE-K/MAPE-SAC framework tailored to its requirements. The first platform comprises an experimental testbed of wearable devices [32] that prioritize security and privacy of health-related information, including information potentially needed by the rescue robots in our scenario. The testbed wearables are represented by Raspberry Pi 3s that can simulate the Bluetooth communication of basic cell phones and high-end wearables. The wearable or base station triggers an adaptation if it decides that it is in an insecure environment. We limit the possible adaptations to include disconnecting a wearable from its base station and preventing reconnection until the base station requests a connection or remaining connected to the base station but sending only empty data packets. The second platform is an autonomous robot experimentation testbed comprise of a collection of Cozmo robots [33]. The robots can adapt to adjust the current mission to more quickly complete a global objective, such as searching a large or different area.

We consider the following scenario to show how MAPE loop interaction enables cooperation among the two types of systems given functional and security concerns. Imagine a natural disaster has occurred, requiring the deployment of search and rescue robots. These robots are equipped with visual sensors, searching for survivors, as well as Bluetooth to connect to a survivor's phone or wearable device. Because Bluetooth can be used for localization [34], the robots can discover the location of survivors accurately even when visual identification is impossible. Since it is important to prioritize survivors who may need immediate medical attention, the robots should connect to and request basic information from health-monitoring wearables, such as heart rate or stress monitors.

The two testbeds can simulate this scenario for experimentation and validation purposes. Because the adaptations available to the wearable devices focus on the security of personal data, they conflict with the needs of the robots. This conflict requires the wearables to adapt their SACs to share information with the robots (role-based adaptation). Specifically, AC-3 (in Figures 3 and 4) must change its context to allow role-based identification.

To resolve this conflict, rather than remove the ability of the wearables to prioritize security completely, we introduce two new adaptations, which are both modifications of adaptations already available to the wearables. A communication protocol called *fostering*, originally designed to pass security states between wearables quickly, can be repurposed for passing important information to the rescue robots when needed. When adapted to utilize fostering, a robot can search for Bluetooth signals and, once found, connect to the available fostering server. It can remain connected only for a short time that is long enough to request and receive information. The fostering protocol allows only a single connection request, a single request for data, and responses to these two requests before it forcibly terminates the connection.

Figure 8 shows the architecture of the combined testbed. The left shows the autonomous vehicle coordination network. Each robot is in constant communication with all nearby robots to alert them of any potential change. The right portion of the figure shows the wearable network. The dashed lines represent only occasional communication when the wearables and base station are connected to each other, but other base stations and wearables only communicate in the event of fostering.

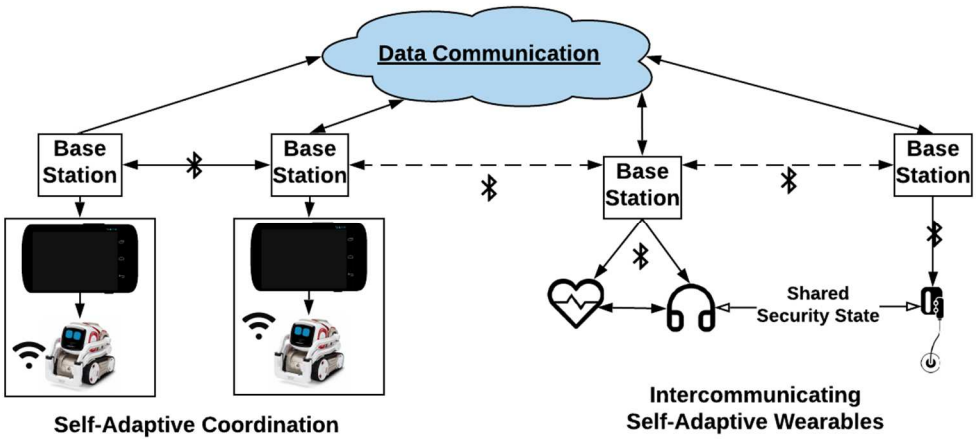

Figure 8: Testbed Integration Architecture

The combined testbeds can communicate via the cloud, but, for our purposes, we are focusing on its Bluetooth communication capability. Should a base station be discovered via Bluetooth, the base station of the robot will connect to the base station of the wearable and maintain that connection until the wearable location is discovered, at which point the location will be stored and the robot will continue searching its designated area for persons in need of assistance.

5.1. Security Assurance Cases

The SACs are implemented on the base stations. The security requirement defined by security control AC-3, which ensures authorization, can be maintained in two ways. The first approach uses default Bluetooth pairing, which has the Raspberry Pi 3s refuse connections from unpaired devices. The second approach is within the control code, where a list of approved devices is stored and checked before connections are accepted. The latter means that, even if a device is paired, it

cannot connect in order to request data without also being on the approved list. Both the request information (containing MAC addresses of the connecting devices) and the result of the request are logged into a connection log for future review. This results in a conflict preventing the robots from connecting to the wearables and requires adaptation of the SAC to allow fostering.

Security control SC-40 (Figure 3), focused on organization-defined attacks (in this case: Man-in-the-Middle attacks), is achieved through both sending empty packets and disconnection adaptations, though again results in a conflict requiring adaptation when fostering. We also check the current Bluetooth connection status and, if no data is being sent, disable Bluetooth entirely.

Security control IA-3 (Figure 3), focused on device identification, is achieved through the use of randomized PINs when pairing Bluetooth devices. We directly embed the PIN randomization into the control code, choosing a random PIN for each connection request which must be confirmed by the user initiating the connection. The PIN will be valid only for a limited time, in this case 10 seconds, before a new PIN is generated. The random PIN is stored and checked to ensure it is of sufficient length and complexity. We define sufficient length as having at least 8 characters, and a strong PIN as any PIN not having the same number repeated consecutively. IA-3 also conflicts when a robot must connect to the device, requiring an additional adaptation for fostering.

5.2. Functional Assurance Cases

Both the wearables and the robots have unique requirements, which we define as functional assurance cases (FACs) that cannot be violated under normal operation. These FACs are represented using GSN notation exactly like the SACs. The FACs are unique to each wearable device but are shared across the robots. For this paper, we focus only on the heart rate variability monitor (HRVM) FAC shown in Figure 9. The wearable main goal, to autonomously monitor both heart rate and stress level, is composed of two sub-goals. The first sub-goal is focused on ensuring that the buffer that stores the heart rate data does not overflow, as a loss of heart rate data will violate the main goal. The second is focused on ensuring that the buffer that stores the stress level data does not overflow. We note that the LTL square means “it is always the case” and the LTL diamond means “eventually.”

In this work, during the monitor phase of the MAPE-K loop, heart rate data, stress level data, buffer conditions are monitored on the heart rate variability monitor (HRVM) testbed and are analyzed to determine the user’s health condition. During the analyze phase, the FAC is instantiated with observed stress level data and buffer conditions so that the system can assess its compliance status. The *determineStressLevel* and *bufferMaintenance* procedures analyze stress level data and buffer conditions over time to determine whether the user is stressed or if the buffer has overflowed, which eventually implies that the user needs immediate medical attention. An adaptation is triggered in response to significant requirement compliance degradation.

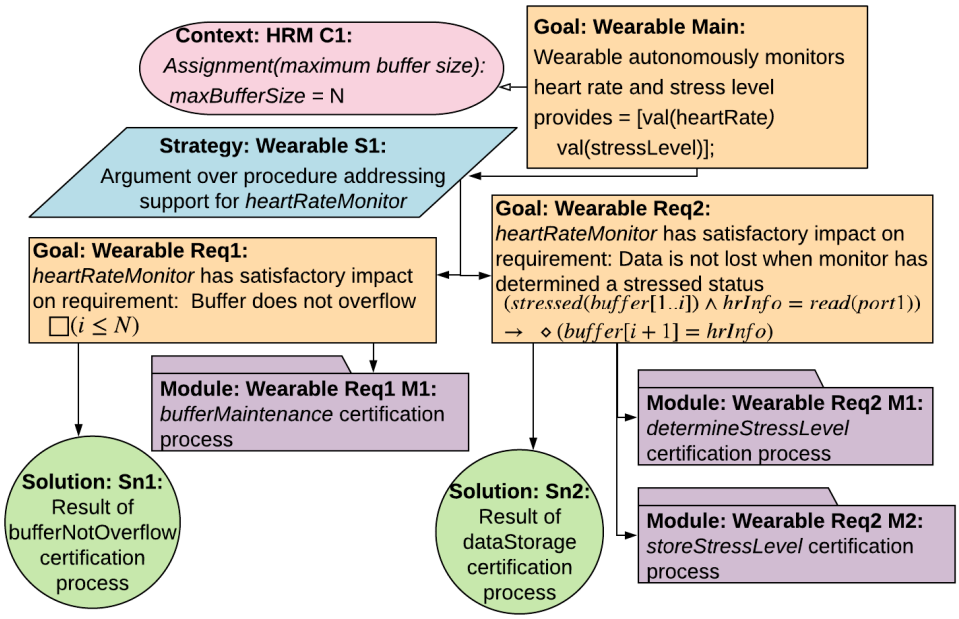

Figure 9: Heart Rate Variability Monitor Functional Assurance Case

The robot FAC is shown in Figure 10. Its primary goal is search and rescue. It contains an away goal to announce the location of people it finds and their heart rate condition, if data is available, and an away goal to integrate itself into the global goal (rescue the maximum number of people) with the other robots. It contains only one sub-goal with modules focused on detecting users and rescuing (determining the location of) users when possible. Similarly, the rescue robot testbed also maintains a MAPE-K loop for its own functional requirements. The monitor phase captures the heart rate and location of the people for rescue as environmental conditions. It captures the assigned search area and rescue mode as functional conditions. During the analyze phase, these environmental and functional conditions are analyzed to determine if the robot's integration is valid for performing the rescue mission; this is handled by *rescueProcedure*.

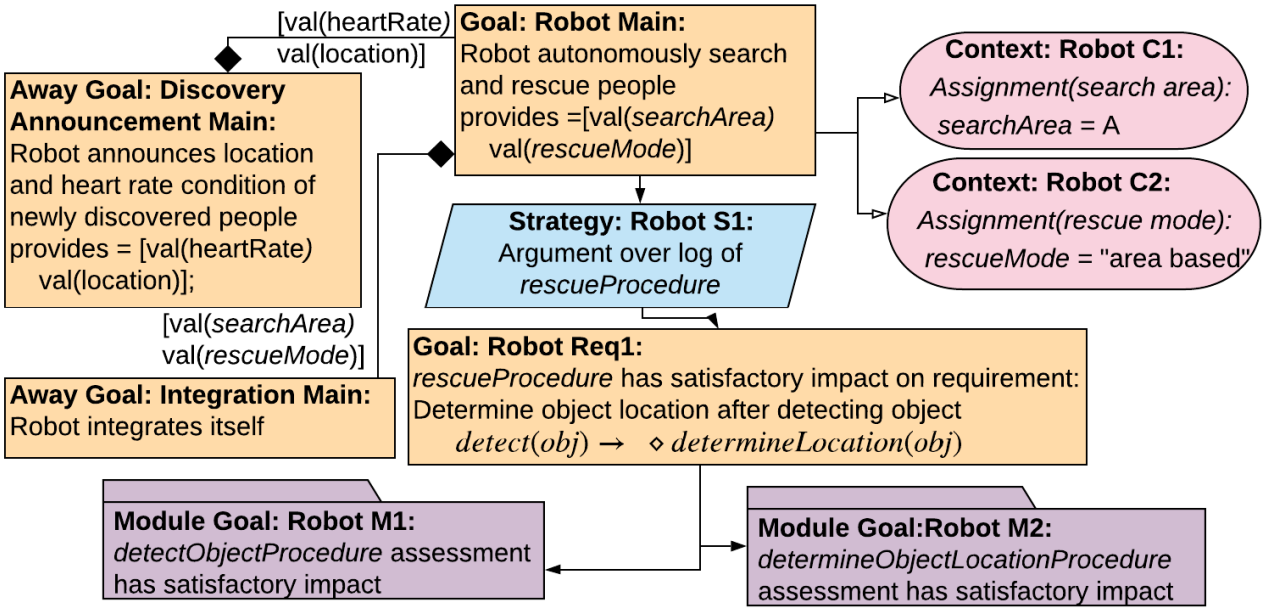

Figure 10: Autonomous Robot Functional Assurance Case

5.3. Interaction Protocol

In this paper, we introduce, design, and implement an interaction protocol based on the Regional Planning Pattern [15]. The purpose of the interaction protocol is to standardize a communication pattern that processes can use to effectively coordinate their adaptation. The Regional Planning pattern was designed so that processes with separate regions of concerns could coordinate during the planning step to have their adaptation decided by a third process, referred to as the coordination manager, which can be deployed on a separate system or as a subprocess, as is the case with our testbed. The coordination manager is responsible for selecting an adaptation that will be executed by both coordinating processes. It is not specified by the Regional Planning pattern how the coordination manager should select an adaptation, so we have taken the liberty of defining an interaction protocol that is robust enough to be deployed by a variety of processes.

According to the Regional Planning pattern, coordination takes place during the planning step. One process must initiate, and the other process must respond. One processes may not reach its respective planning step at the same time as the other process. As such, either process can wait during its planning step by blocking for a period of time, as specified by an input value, or until it receives a response. If the blocking process does not receive a response before its waiting period has expired, then the process does not coordinate. The process that initiates the coordination is termed the initiator and the other process is termed the participant.

The interaction protocol is conducted as follows. Once they reach their planning step, the MAPE-K process and the MAPE-SAC process prompt the coordination planner to generate a set of adaptation plans and block until the coordination planner is finished. Once the coordination planner has generated a set of adaptations, it sends the set of adaptations to both the MAPE-K and MAPE-SAC processes, both processes become unblocked, and the coordination planner blocks until it receives a risk assessment from the MAPE-K process and a degree of compliance confidence assessment from the MAPE-SAC process. The MAPE-K and the MAPE-SAC processes translate each adaptation to a change set, and then apply associated risk assessment methods on each change set. The results are sent to the coordination planner. It is left to each process to select and perform the assessment process according to its requirements. By separating

the logic of ranking adaptations, which is performed by the MAPE-K and MAPE-SAC processes, from the logic of selecting an adaptation, which is performed by the coordination manager, the interaction protocol remains robust. Any process that can perform any kind of assessment of the adaptation changes in accordance with the requirements of the coordination manager can utilize the interaction protocol to interact with any other process that can also perform a similar assessment; the two methods of assessment do not need to be the same.

After receiving the assessment results from both the initiator and the participant, the coordination planner decides on a plan as follows. First, it selects the highest ranked plan (i.e., least risk). If the plan selected from the initiator is in the top half of the participant plan rankings, then the plan is accepted. Otherwise, the plan from the initiator is rejected and the coordination planner selects the next best plan from the participant and the plan ranking review process continues. Similarly, if the highest ranked plan is selected from the participant and is in the top half of the initiator's ranked plans, it is accepted. Otherwise, the plan selected from the participant is rejected and the coordination planner selects the next best plan from the initiator and the plan ranking process continues. The coordination manager alternates between reviewing the next highest rank plan from the initiator and reviewing the next highest ranked plan from the participant so that both the initiator and the participant each has the opportunity to have their best plan selected. This process repeats until either a plan is selected or until there are no more plans. Once the outcome is decided, the plan is communicated to the MAPE-K and MAPE-SAC processes, unblocking both processes and enabling them to proceed to their execution steps.

The interaction protocol presented in this paper extends our prior work [17]. Previously, to select an adaptation, the initiator would send a sequence of adaptations to the participant who would evaluate each adaptation individually and either accept or reject the adaptation. The initiator would accept the first adaptation accepted by the participant and reject any adaptation rejected by the participant. If the participant rejected all adaptations, then the system would enter a default state and no adaptation would be applied. Our current work differs from our prior work in that (a) no adaptation is assessed individually as all adaptations are known by both the initiator and the participant, (b) the initiator can reject an adaptation that has been accepted by the participant, and (c) at least one adaptation must be accepted by both the initiator and the participant, i.e., the system cannot enter a default state.

5.4. Adaptation

By default, the wearable systems focus primarily on security of personal information and therefore do not allow fostering as shown in Figure 11 (left side). However, in a disaster scenario, fostering is needed by the autonomous rescue robot to help determine the condition of the victim. We simulate four adaptation plans for wearable devices to assist a rescue mission in a disastrous area.

A1: Stay connected, send empty packets, no fostering is allowed

A2: Stay connected, send empty packets, fostering is allowed

A3: Get disconnected, no fostering is allowed

A4: Get disconnected, fostering is allowed

A wearable's interactive MAPE loops evaluate the adaptation plans by coordinating themselves and allow fostering as a functional adaptation as shown in Figure 11 (right). However, fostering violates security control IA-3 as written. Other interrelated security controls realize the operational circumstance and prioritize access to rescue-relevant data over identification and authorization as defined by IA-3. Once the context of *accessControlPolicy*, defined in security control AC-3, and *connectionRequirements*, defined in AC-18 (Figure 3), have been changed, the coordination

between the MAPE loops yields an optimal system-wide adaptation. To maintain rescue mission objectives, the adaptation must prevent communication while remaining connected and allow fostering, if needed, in order to connect to a rescue robot.

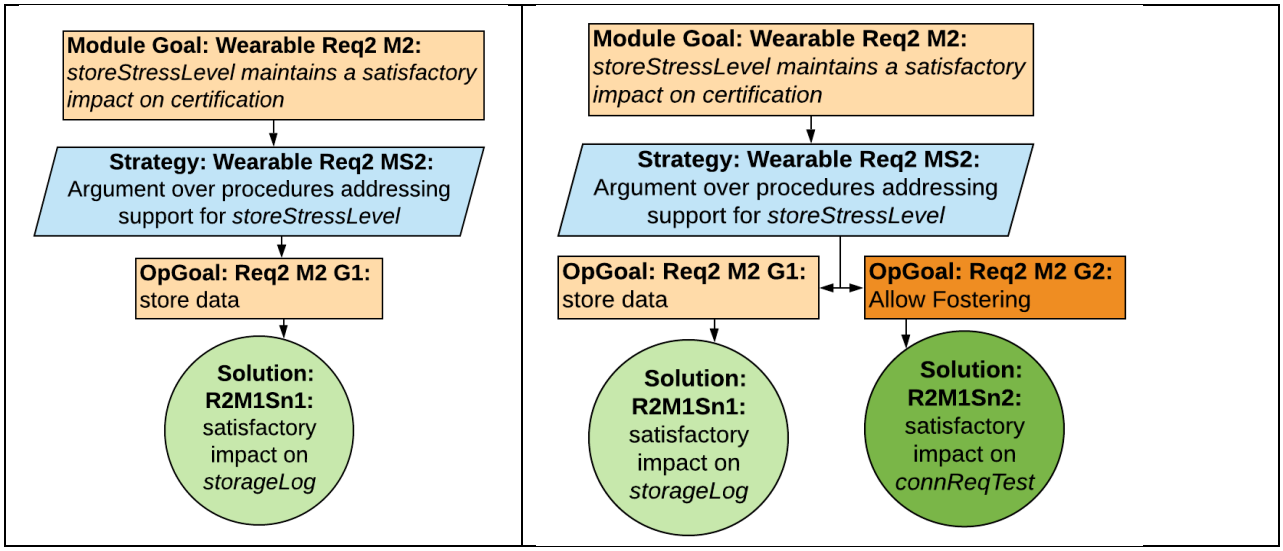

Figure 11: Expanded Modular goal: Wearable Req2 M2 (left: before adaptation and right: after adaptation)

6. Evaluation

We implemented and tested the MAPE-K loop, MAPE-SAC loop, and coordination planner on the wearable testbed. To validate the above scenario, we focus only on the heart rate variability monitor. The wearable device is connected to its base station via Bluetooth, sending data to simulate heart rate information. If the system detects the need for an adaptation, it must determine an optimal system configuration.

To evaluate the MAPE loops and their coordination, we designed a series of experiments using the four previously described adaptations: A1, A2, A3, and A4. We assume that a coordination planner implemented on a wearable device can dynamically generate adaptation plans. Further, we assume that the planners used in the MAPE-K and MAPE-SAC loops can translate those plans into a change set, respective to their region of concerns, and that they can perform assessment of changes for compliance of their requirements.

Security compliance is assessed by determining the satisficing level of the main security control goal included in SAC model. Here, a low satisficing level implies higher risk of non-compliance with a security requirement. We produce a *changeSet* to determine (1) what state variables are affected due to change, (2) what will be the new state of the state variables after changes have been applied, (3) what are conditions of the changes, (4) what evidence is needed to support the changes, and (5) what is the rationale behind the changes. We represent each main security control goal as a softgoal and interdependencies among the arguments the goals as a Soft goal Interdependency Graph (SIG) [13] The impact of the changes associated with state variables and their propagation toward the main security control goals are considered to calculate the achievement weight, a_w for each goals within the argumentation, which is used to assess compliance degradation.

$a_w(g) = I(g)$, for leaf nodes,
$= average(a_w(c))$, for all $c \in children(g)$ for non-leaf nodes

The compliance degradation of a security control main goal can affect other related security controls and cause signification compliance degradation in the overall system. After determining achievement weight for each main security control goal, we calculate the satisficing level, SL for all interrelated main security controls.

$$SL(m) = average(a_w(m) + \sum_{g \in neighbors(m)} a_w(g))$$

Functional compliance is assessed by determining the risk of reusing the original proof of verification process, as changes may inhibit reusing original proof [30]. The planners would then supply their risk assessment to the coordination planner, which then selects an adaptation plan based on the ordering of the adaptation plans across the two provided risk assessments. For our risk assessment, we produce a *changeSet* to determine (1) what state variables are affected due to change, (2) what are the conditions of the changes, and (3) how do the changes impact the state variables. The risk assessment process uses this information to identify conflicts that arise during the verification process due to the changes and to alert when there are potential conflicts. The alerts include all of the impact multipliers needed for the risk assessment on the changes, which are (1) the impact on state variables due to changes, M_{VC}, (2) the impact on component due to changes, M_{PL}, and (3) what the planner believes the impact on state variables is, $\hat{p}$. We multiply these impact multipliers together to estimate the probability of risk associated with each alert, t.

$$\boldsymbol{p(t) = M_{VC}(t)\, M_{PL}(t)\hat{p}\,(t)}$$

Then, we calculate expected utility of the adaptation plan using the following utility function.

$$\boldsymbol{E = \sum_{r \in R} w(r) \prod_{t \in T} p(t)}$$

Here, $w(r)$ is utility weight of requirements needed to maintain system compliance. Higher expected utility means adaptation is less risky, and adaptation plans are ranked based on their expected utility.

For the purposes of our experiments, the coordination planner always provides the four aforementioned adaptation plans: A1, A2, A3, and A4, to the MAPE-K and MAPE-SAC planners which, in return, provide their risk assessment and satisficing level of the four adaptation plans.

Three experiments are conducted to evaluate the MAPE-K and MAPE-SAC coordination on the wearable security testbed. The first two experiments are designed to evaluate the expected best and worst behavior of the MAPE loop coordination, as determined by the ordering of the adaptation plans provided by the MAPE-K and MAPE-SAC planners. The best case is that the orderings of the risk assessments are the same, and the worst case is that they are the reverse of one another. The final experiment is designed to provide an assessment of the MAPE loop coordination across all possible wearables. MAPE-K and MAPE-SAC planners can provide a wide variety of risk assessments based on the concerns of the wearable system on which they are implemented.

Each experiment comprises three processes. The first process manages the MAPE-K loop, the second process manages the MAPE-SAC loop, and the third process manages the coordination planner. The MAPE-K process loops through its monitor and analyze steps, as does the MAPE-SAC process, constantly checking to determine if the system is still secure. The current security

state of the system is represented as a global Boolean flag that is set to true if the wearable system is in a secure state or false if the wearable system is in an insecure state. If the monitor step determines that the wearable system is still in a secure state, then no adaptation is needed and the plan and execute steps produce no adaptation.

After all threads are started, the state of the wearable system is flipped to insecure. Both MAPE processes then observe that the wearable system is insecure in their respective monitor steps, determine that an adaptation is needed in their respective analyze steps, and interact with the coordination planner in their planning step.

For the first experiment, both the MAPE-K and the MAPE-SAC processes provide the same risk assessment to the coordination planner: A1, A2, A3, A4. The first set of plans selected by the coordination planner are A1 and A1. Since the first set of plans match, the coordination planner decides on A1.

For the second experiment, the MAPE-K process provides the following risk assessment: A1, A2, A3, A4, and the MAPE-SAC process provides the following risk assessment: A4, A3, A2, A1. The coordination planner decides on plan A3. The MAPE-K process communicated first and became the initiator. Since A1 (the highest ranked plan from the MAPE-K process) is in the bottom half excluding the middle of the risk assessment from the MAPE-SAC process, it is rejected and A2 is selected. Then, since A4 (the highest ranked plan from the MAPE-SAC process) is in the bottom half excluding the middle of the risk assessment from the MAPE-K process, it is rejected and A3 is selected. Since A2 is ranked in the top half inclusive by the MAPE-SAC, the coordination planner decides on A2. If the MAPE-SAC had been the initiator, then the coordination planner would have decided on A3 for the same reason.

In the third and final experiment, we iterate over every possible permutation of A1, A2, A3, and A4 for the MAPE-K loop. For each permutation, we iterate over every possible permutation of A1, A2, A3, and A4 for the MAPE-SAC loop. There are 24 possible permutations of each set for a total of 576 trials. Each plan decided by the coordination planner is recorded, the results are aggregated into Table 2. After each trial, the system is reset so that the initial conditions of each trial are the same.

Table 2 shows the results gathered from the evaluation conducted in the third experiment. From this table, we can see that plan A1 was selected 146 times, A2 was selected 135 times, A3 was selected 147 times, A4 was selected 148 times, and in 0 cases there was no agreement between the MAPE-K and the MAPE-SAC loops. Based on these results, we can see that each adaptation plan is selected with roughly equal likelihood, as should be expected. In addition, we can see that if no coordination takes place, then in 42% of the trials, a violation will occur in either the functional concerns or the security concerns. A violation occurs when a functional adaptation or a security adaptation causes significant degradation in the system's compliance with its security requirements or functional requirements, respectively. Whereas, if coordination does take place, then a violation will occur in 0% of the trials with the adaptations selected.

Table 2: Results gathered from the implementation of the interaction protocol on the interactive Bluetooth testbed with 576 trials

Plan	Count	Percentage
A1	**146**	**0.25**
A2	**135**	**0.23**
A3	**147**	**0.26**
A4	**148**	**0.26**
Violations w/ Interaction	**0**	**0**
Violations w/o Interaction	**240**	**0.42**

We also examined the average position of the chosen adaptation for both the MAPE-K and MAPE-SAC loops. Ideally, this average should be similar over the 576 trials and between 1 and 2, indicating that the first or second plan of the MAPE-K and MAPE-SAC are chosen more often than not. In our trials, the MAPE-K average position was 1.66, while the average position of the MAPE-SAC was 1.61. These numbers are very similar and, importantly, very close to the optimal position. It is important to observe this value as each time an adaptation is rejected, the next adaptation is less satisfactory for either the MAPE-K or MAPE-SAC process. In our experiment the MAPE-K was more often the initiator. This results in the MAPE-SAC plan being prioritized when planners disagree.

7. Discussion

Given the inherent complexity of coordinating an adaptation across separate regions of concerns, the proposed approach has some limitations. First, the interaction protocol cannot ensure that there exists a best adaptation across cooperating systems. Rather, given a set of ranked adaptations from each system, the interaction protocol can select the adaptation that best minimizes cumulative impact on both systems. Any attempt to minimize cumulative impact can only be as effective as the assessment that each system uses to rank the adaptations. In addition, our framework also assumes that both systems have equal priority. When selecting an adaptation, the interaction protocol does not attempt to minimize impact for one system over the other. Further, the risk assessment that we use to rank adaptations cannot guarantee that there is an acceptable adaptation. It can only rank adaptations in terms of their likelihood to inhibit proof re-use for functional requirements and compliance degradation for security requirements, and there is currently no metric to evaluate system trustworthiness.

The concept of interacting MAPE loops has largely been pioneered by Vromant [16] and Weyns [15] through their theoretical exploration of control patterns. But they did not consider explicitly how knowledge of an adaptation would be exchanged and shared among MAPE components, which gives rise to the need for an interaction protocol. To the authors' best knowledge, our interaction protocol is the first such explicitly defined protocol to enable two separate interactive MAPE loops to systematically share such knowledge. Consequently, there is a lack of other appropriate approaches with which we might compare our work.

We note that Vromant et al. [16] has published and evaluated a coordination manager that coordinates separate MAPE loops in each step of the control loop following the Coordinated Control pattern from Weyns [15]. Their implementation is a centralized system coordinating parallel processes. In contrast, our approach coordinates MAPE loops implemented on separate but integrated testbeds, making our solution applicable to a distributed system. Vromant [16] does not provide an evaluation section that would allow for an appropriate comparison of the two approaches. As such, we are limited to describing and validating our framework within specific application domains.

This framework can be used for systems employ dynamic graphical security models or reliability models. Security models that use attack graphs identify vulnerabilities and dependency nodes to exploit a system's vulnerabilities [35]. Our framework supports the instantiation of security assurance cases by including the security requirements (as dictated by the NIST security controls) and dependencies among them that are needed to maintain the system's security profile. Compliance degradation within the system's security profile can be considered as a potential vulnerability. Thus, our assessment of compliance confidence allows for the detection of potential

vulnerabilities. Reliability models rely on the assessment of components, i.e., functional reliability, and the probabilistic distribution of the component, i.e., functional utilization [36]. Our assessment of functional compliance relies on reasoning about functional or environmental changes. This reasoning uses impact metrics that are pre-established by the system developers given their understanding of the range of state and environment variabilities for the function to still be effective. Assessing the compliance degradation of maintaining a goal can be considered as determining the extent of system reliability overall.

8. Conclusion

This paper proposes a MAPE-K/MAPE-SAC interaction framework to dynamically maintain functional and security concerns in an adaptive, self-protecting system. We represent the interaction between the MAPE-K and MAPE-SAC loops by modifying the Regional Planning Pattern to use coordinated planning and execution, and we separate security and functional concerns to different regions. We recognize that coordination requires knowledge about code level interdependency between functional and security concerns for conflict resolution, as both adaptations apply on a single application level. We illustrate how such a framework can be used to manage system security requirements, as environmental uncertainty and functional adaptations occur, using an example of two interactive testbeds, designed for an autonomous search and rescue scenario in the event of a natural disaster.

Future work will explore several directions. First, we plan to develop a dependency profile to represent the interdependency and assign weights, so that we can determine tradeoffs between different adaptations. Second, we will apply security controls from different domains and devices and develop a catalogue for modeling security assurance cases. Finally, we are exploring how these techniques governing two interacting MAPE loops can be applied to large autonomous distributed systems that are exposed to additional sources of environmental and security uncertainty.

Acknowledgements

This material is based on research sponsored by Air Force Research Laboratory under agreement FA8750-19-2-0002. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes not withstanding any copyright notation thereon. We acknowledge Michael Langford from Michigan State University for his review of the document and internal conversations regarding assurance cases.

References

[1] R.D. Alexander, R.D. Hawkins, T. P. Kelly, Security Assurance Cases: Motivation and the State of the Art, Issue 1.1. CESG/TR/2011/1. University of York, (2015).

[2] E. Yuan , N. Esfahani , S. Malek, A Systematic Survey of Self-Protecting Software Systems, ACM Transactions on Autonomous and Adaptive Systems (TAAS), v.8 n.4, (2014), pp.1-41. DOI: 10.1145/2555611.

[3] F. Moyano, B. Baudry, J. Lopez, Towards Trust-Aware and Self-adaptive Systems, In: Fernández-Gago C., Martinelli F., Pearson S., Agudo I. (Eds.), Trust Management VII. IFIPTM 2013. IFIP Advances in Information and Communication Technology, vol. 401. Springer, Berlin, Heidelberg, (2013).

[4] D. T. Le, Quality Trade-offs in Self-Protecting System, 2015 IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops, Cambridge, MA, (2015), pp. 152-156. DOI: 10.1109/SASOW.2015.30.

[5] Joint Task Force Transformation Initiative, Security and privacy controls for Federal Information Systems and Organizations, NIST Special Publication 800-53 revision 4, (2015).

[6] IT Security Techniques Technical Committee, Information technology – Security techniques – Code of practice for information security, ISO 27002, (2013).

[7] R. Thomas , Software Assurance Using Structured Assurance Case Models, Journal of Research of the National Institute of Standards and Technology, (2010).

[8] J. Rushby, The interpretation and evaluation of assurance cases, Technical Report SRI-CSL-15-01, (2015).

[9] H.F. Lipson, C.B. Weinstock, Evidence of assurance: Laying the foundation for a credible security case, (2008). https://www.us-cert.gov/bsi/articles/knowledge/assurance-cases/evidence-assurance-laying-foundation-credible-security-case.

[10] J. Goodenough, H.F. Lipson, C.B. Weinstock, Arguing security – creating security assurance cases, (2017). https://www.us-cert.gov/bsi/articles/knowledge/assurance-cases/arguing-security-creating-security-assurance-cases.

[11] The Assurance Case Working Group, Goal Structuring Notation, Community Standard, Version 2, 2018. https://scsc.uk/r141B:1?t=1

[12] S. Jahan, A. Marshall,R. Gamble, Self-adaptation strategies to maintain security control assurance cases, In Proceedings of the 12th IEEE International Conference on Self-Adaptive and Self- Organizing Systems, (2018).

[13] S. Jahan, A. Marshall, R. Gamble, Evaluating security assurance case adaptation, In Proceedings of the 52nd Hawaii International Conference on System Sciences, (2019).

[14] J.O. Kephart, D.M. Chess, The vision of autonomic computing, Computer, vol. 1, (2003), pp. 41–50.

[15] D. Weyns et al., On patterns for decentralized control in self-adaptive systems, Lecture Notes in Computer Science, (2013), pp. 76–107

[16] P. Vromant, D. Weyns, S. Malek, and J. Andersson, On interacting control loops in self-adaptive systems, Proceedings of Soft. Eng. for Adaptive and Self-Managing Systems, (2011).

[17] S. Jahan, M. Pasco, R.F. Gamble, P. McKinley and B.H.C. Cheng, MAPE-SAC: A Framework to Dynamically Manage Security Assurance Cases, 1st International Workshop on Self-Protecting Systems (SPS 2019), Umea, Sweden, (2019), pp. 146-151.

[18] R. Ross et al., Protecting Controlled Unclassified Information in Nonfederal Systems and Organizations, (2016).

[19] R. Ross et al., Guide for the Security Certification and Accreditation of Federal Information Systems, NIST Special Publication #800-37, Gaithersburg, MD, USA, (2004).

[20] M. Anisetti , C. A. Ardagna , E. Damiani , F. Saonara, A test-based security certification scheme for web services, ACM Transactions on the Web (TWEB), v.7 n.2, (2013), pp.1-41. DOI:10.1145/2460383.2460384.

[21] M. Anisetti , C. A. Ardagna , F. Gaudenzi , E. Damiani, A certification framework for cloud-based services, Proceedings of the 31st Annual ACM Symposium on Applied Computing, Pisa, Italy, (2016). DOI:10.1145/2851613.2851628.

[22] O. Lichtenstein, A. Pnueli, Checking that finite state concurrent programs satisfy their linear specification, In Proceedings of the 12th ACM SIGACT-SIGPLAN symposium on Principles of programming languages, ACM, (1985), pp. 97–107.

[23] J. Zhang, B. H. C. Cheng, Using temporal logic to specify adaptive program semantics, Journal of Systems and Software, 79(10), (2006), pp. 1361– 1369.

[24] E. M. Fredericks, B. DeVries, B. H. C. Cheng. Autorelax: automatically relaxing a goal model to address uncertainty, Empirical Software Engineering, 19(5), (2014), pp.1466–1501

[25] E. Damiani, C. A. Ardagna, N. E. Ioini, Open Source Systems Security Certification. Springer Publishing Company, Incorporated, (2008).

[26] A. Marshall, S. Jahan, R. Gamble., Toward evaluating the impact of self-adaptation on security control certification, In Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems, ACM, (2018), pp. 149–160.

[27] T. Kelly, R.Weaver, The goal structuring notation–a safety argument notation, In Proceedings of the dependable systems and networks 2004 workshop on assurance cases, Citeseer, (2004).

[28] E. Luke, Using Claims, Arguments and Evidence: A Pragmatic View and tool support in ASCE. (2008) www.adelard.com.

[29] M. Alia, M. Lacoste, R. He, F. Eliassen, Putting together QoS and security in autonomic pervasive systems, Proceedings of the 6th ACM workshop on QoS and security for wireless and mobile networks, (2010). DOI:10.1145/1868630.1868634.

[30] A. Marshall, S. Jahan, R. Gamble, Assessing the Risk of an Adaptation using Prior Compliance Verification, In Proceedings of the 51st Hawaii International Conference on System Sciences, (2018).

[31] Joint Task Force Transformation Initiative et al., Assessing Security and Privacy Controls in Federal Information Systems and Organizations: Building Effective Assessment Plans, NIST Special Publication 800-53A Revision 4, (2014).

[32] C. Walter, I. Riley, R. Gamble, Securing wearables through the creation of a personal fog. In Proceedings of the 51st Hawaii International Conference on System Sciences, (2018).

[33] S. Jahan, C. Walter, S. Alqahtani, R. Gamble, Adaptive coordination to complete mission goals, IEEE 3rd International Workshops on Foundations and Applications of Self* Systems (FAS* W), (2018).

[34] F. S. Daniş, A. T. Cemgil, Model-Based Localization and Tracking Using Bluetooth Low-Energy Beacons, Sensors (Basel, Switzerland) vol. 17,11 2484, (2017). DOI: 10.3390/s17112484.

[35] S. Noel, S. Jajodia, L. Wang and A. Singhal, Measuring security risk of networks using attack graphs, International Journal of Next-Generation Computing, (2010), vol. 1, no. 1, pp. 135-147.

[36] R. Cheung, A User-Oriented Software Reliability Model, in IEEE Transactions on Software Engineering, (1980), vol. 6, no. 02, pp. 118-125. doi: 10.1109/TSE.(1980).234477