FP7-SEC-2011-1 Project 285647

Cyber-security on SCADA: risk prediction, analysis and reaction tools for Critical Infrastructures

D4.2 - Automatic Reaction Strategies and RTU smart Policies-Final

General information	
Submission date	26/03/2014
Dissemination Level	Confidential
State	Final Version
Work package	WP4000
Task	Task 4003-4004
Delivery Date	31/01/2014

Editors

Name	Organization
Prof. Stefano Panzieri	Roma3
Prof. Jianmin Jiang	SURREY

Authors

Name	Organisation
Prof Stefano Panzieri	Roma3
Giovanni Corbò	Roma3
Prof. Jianmin Jiang	SURREY
Lasith Yasakethu	SURREY
Moussa Ouedraogo	CRPHT
Christophe Feltus	CRPHT

Reviewers

Name	Organisation	Approval Date
Leonid Lev	IEC	25/02/2014
Tiago Cruz	FCTUC	27/02/2014
Jorge Proença	FCTUC	27/02/2014

Table of contents

List of figures

List of table

Introduction

1.1 Context

Initially, ICS systems were isolated by nature, being limited to the process network – in those times, security was guaranteed by both obscurity and isolation. Protocols were proprietary and its documentation was undisclosed, creating a false sense of security. Only manufacturers and attackers knew of failures and vulnerabilities, with both parts having no interest in their divulgation. Still, “modern” SCADA architectures are, in general, much similar to the ones used in the ‘80s and ‘90s, even if some technologies suffered a clear evolution. This has to do with several reasons such as maturity (these architectures are tried and tested) and cost of migrating to a modern solution.

Unfortunately, when migrating from an “isolated” to an open environment, from serial communication to TCP/IP communication, conventional SCADA architectures started to show all their limits. The move to more open scenarios with network interconnection together with the use of ICT technologies and the increasing adoption of open, documented protocols, exposed serious weaknesses in SCADA architectures. By itself, the growing trend towards the interconnection of the ICS network with the ICT infrastructure, and even with the exterior created a new wave of security problems and attacks. In fact, there is a growing trend in the number of externally initiated attacks on ICS systems, when compared with internal attacks.

Also, the adoption of commercial operating systems brought its own share of problems. Albeit reducing development and lifecycle management costs, the adoption of these operating systems made SCADA infrastructures implicitly vulnerable to a vast array of issues that traditionally plague them. There are several security incidents and undirected attacks to SCADA infrastructures that were the result of operating system vulnerabilities.

The security by obscurity philosophy (which is not a good security practice, anyway) became unfeasible. However, the problem of security in SCADA systems was ignored for several years, and even now serious issues persist. For instance, unsafe protocols such as Modbus are being widely used in production systems. But even new features, such as the auto-configuration capabilities of certain equipment (plug-and-play) only got things worse, since attackers found it to be a valuable resource for attack planning and execution. Also, the old-school mind-set still persists, up to the point that some process managers still think of ICS systems as isolated and implicitly secure, disregarding the need for regular security updates or software patching procedures, increasing the probability of a successful attack.

As a result, SCADA systems have always been susceptible to cyber-attacks. Different types of cyber-attacks could occur depending on the architecture and configurations used in the SCADA system. Thus, today protection of the national infrastructures from cyber-attacks is one of the main issues for national and international security. CockpitCI will introduce intelligent intrusion detection, analysis and response techniques for Critical Infrastructure protection (CIP). The paradox is that CIs massively rely on the newest interconnected and vulnerable, Information and Communication Technology (ICT), whilst the control equipment, legacy software/hardware, is typically old. Such a combination of factors may lead to very dangerous situations, exposing systems to a wide variety of attacks. To overcome such threats, the CockpitCI project combines intelligent computing techniques with ICT technologies to produce intrusion detection, analysis and reaction tools to provide intelligence to field equipment for critical infrastructure protection.

1.2 Document Structure

The remainder of the document is organized as follows: subsections 1.3 and 1.4 present, respectively, a glossary of relevant terms and a list of acronyms and symbols that recur in the document.

The rest of the document consists of two main sections, Part I: Strategies for automatic reactions and Part II: RTUs smart policies, and it is organized as follows:

Section 2 gives an overview on intrusion response describing the fundamental concepts related to different intrusion response approaches and techniques for critical infrastructure protection. Advantages and disadvantages of different intrusion detection approaches such as notification systems, manual and automatic response systems and passive and active response systems are presented. Section 3 highlights state-of-the-art automatic response strategies. In this section state-of-the-art intrusion response strategies based on machine learning/modelling and signal processing approaches are described. Section 4 presents the proposed automatic response strategies of task 4004. A detailed description of a combined rule and fusion based intrusion reaction approach and a reputation based system for automatic reaction are presented in this chapter. Section 5 discusses the state of the art on RTU reaction/response strategies, the existing technologies and the references in the current literature. Section 6 shows the design process of the Smart RTU. The concepts of Smart Control, Smart Industrial Control Systems, Smart SCADA are presented. In the last part a definition of the Smart RTU is provided. Finally, Section 7 contains the conclusions of this work, while Section 8 lists the references cited throughout the document.

1.3 Glossary

Terminology	Description
Adverse event	Any event which may cause a degradation of the capability of the CI to provide its services.
Critical Infrastructure	A national or transnational asset which is deemed essential for the maintenance of vital societal functions. It could be in the field of health, safety, security, economic or social well-being of people.
Cyber attack	A global intrusion plan that enables the intruder to achieve his malicious objective.
Industrial control system	Industrial control system is a general term that encompasses several types of control systems used in the industrial sector, including supervisory control and data acquisition (SCADA) systems used to control Critical Infrastructures.
Potential cyber attack	Simple and/or composite security event which represent *symptoms* of possible attacks
Risk	A combination of the probability/likelihood for an accident to occur and the resulting negative consequences if the accident occurs.
SCADA operator	Personnel in charge of managing a CI in order to deliver the requested services.
SCADA system	The set of elements which perform supervision and control of an industrial process or a Critical Infrastructure, including the proprietary communication network which links the field devices to the control centre.

Security alarm	Alarm released in presence of a potential cyber attack with variable degree of confidence
Security event	Event that might be potentially relevant, from a cyber security point of view
Security operator/staff	Personnel in charge of the security of the CI.
System of Systems	An interdependent network of Critical Infrastructures
Service	It is what an infrastructure produces and makes available to its customers or other infrastructures.

1.4 Acronym and symbols

Acronym or symbols	Explanation
ACE	Alert Correction Engine
ART	Attack Response Trees
CI	Critical Infrastructure
CIP	Critical Infrastructure Protection
CRM	Context Right Management
DB	Data Base
HMI	Human Machine Interface
ICS	Industrial Control System
IDS	Intrusion Detection System
IRS	Intrusion Response System
IS	Information System
LAN	Local Area Network
MAP	Maximum-a-Posteriori
MAS	Multi-Agent System
MBP	Message Broadcasting Point
MSP	Message Supervising Point
PA	Policy Analysis
PIE	Policy Instantiation Engine
PMU	Phasor Measurement Unit
ReD	Reaction after Detection
RTU	Remote Terminal Unit
SCADA	Supervisory Control and Data Acquisition
SCOPF	Security Constrained Optimal Power Flow
SMS	Short Message Service
SE	State Estimation

TCP	Transport Control Protocol
TRM	Trust and Reputation Model
UDP	User Datagram Protocol
UML	Unified Modelling Language
USB	Universal Serial Bus
WS	Web Server

PART I - Strategies for Automatic Reaction

2 Intrusion Response

During the last decade the research community paid a lot of attention to intrusion detection due to the rapid surge of sophisticated attacks on computer systems. In general, intrusion detection refers to a variety of methods for detecting possible attacks in the form of malicious and unauthorized activity. In the event that susceptible behaviour is detected, it is essential to take necessary actions to prevent attacks and ensure safety of the targeted resources. Such actions are known as intrusion response. In many intrusion prevention systems, the intrusion response module is often integrated with the intrusion detection system (IDS). However, the intrusion response module receives much less attention than intrusion detection system research due to the inherent complexity in designing and deploying response in an automated manner. Traditionally, it was part of administrator's job to manually trigger an intrusion response to a detected attack. However during the recent past, some commercial intrusion prevention systems have shown limited set of automated response mechanisms. These mainly include blocking and logging actions [1][2] to detected intrusions. But, with the rapid growth of sophisticated attacks and its associated level of complexity the requirement for intelligent automated response strategies, as counter-measures, have become obvious for critical infrastructure protection.

It is necessary for system administrators to select an appropriate response to a detected attack by the monitoring system. The conclusion of the appropriate solution heavily depends on the properties and the deployment objective of the system components. Usually, a system administrator selects a suitable response from a selection of the available response measures together with the appropriate parameters and triggers it. This could be activated at the console of the compromised systems or even remotely over the network. When selecting the response measures and their parameters, system administrators often take in to account the following factors as reported in [3]:

• Expected Response Success

Clearly, the most important aspect is the expected success of a measure. Negative side effects (e.g. unwantedunavailability) need to be considered here. As long as a reaction does not likely have a positive effect (whatever this means in the according application scenario) on the network, it will not be chosen. This also holds for the response parameters.

• Expected Response Effort

Maybe the second most important aspect is the estimated effort (or cost) that is needed for performing response measures. If two sets of possible responses have the same expected success, the easier applied set will be selected.

• Expected Response Error-Proneness

The (subjective) probability of failing when performing a response measure is also very important. It cannot ultimately be precluded that a wrong selection of response measures and their parameters will not put the system in a state worse than caused by the attack itself. So, in most cases, the less complicated alternative would be selected by a network security officer.

• Expected Response Durability

The expected duration of the response effects is an aspect that is less important than the other three mentioned above. If two alternative sets of responses promise comparable values for the other aspects, most likely the one with the longer expected durability will be chosen, i.e. the expected time period after which additional actions will get necessary for keeping the system healthy.

Apart from the above-mentioned aspects, there could be more aspects, which need to be considered by the system administrator. However, these aspects strongly depend on the corresponding deployment scenario.

When certain attacks are detected by the intrusion detection system, some intrusion prevention systems allow automatic or semi-automatic selection of response mechanisms, if connected with access control enforcement points such as firewalls and packet filters. In some cases the reaction strategy is encoded in the detection signature, which has been defined prior to the deployment of the system [4]. As a result, this could be merely observed as a recommendation of the signature writer. Nevertheless, in these cases, there is no real-time assessment of the response involved. In complex environments, this approach of selecting static response measures is understandably not sufficient.

2.1 Intrusion response approaches

In this section we attempt to provide classification of intrusion response approaches. By this we aim to provide a foundation for the future research efforts of this task. A classification of intrusion response approaches is illustrated in Figure 1.

Details on each of the categories in the above classification are provided below.

Intrusion response systems (IRS) can be classified according to the following characteristics:

Degree of automation

Previous studies in literature have reported classification of intrusion response systems according to the degree of automation [5,6,7]. However, these studies only present a very broad view of the response strategy. The classification presented here, comprises additional principles that highlight the differences between various existing approaches.

– **Notification systems:** The task of notification systems is to alert the system administrator with information about the detected intrusion. The system administrator then uses this information in-order to select an appropriate intrusion response. Most of the existing intrusion response systems fall under this category where alarms and reports are generated to notify the system administrator. Daily and monthly periodic reports record anomalous user activity for the system administrator to further investigate the situation. Generation of periodic reports is the earliest form of intrusion response strategy. It is important to note that the frequency of reporting bounds the window of opportunity that an intruder can exploit. However, in today's environment, this window of opportunity could be too large for serious damage on the targeted resources. As a result, while generation of reports still remains as an important component of any intrusion response system, is not a sufficient intrusion response mechanism by itself.

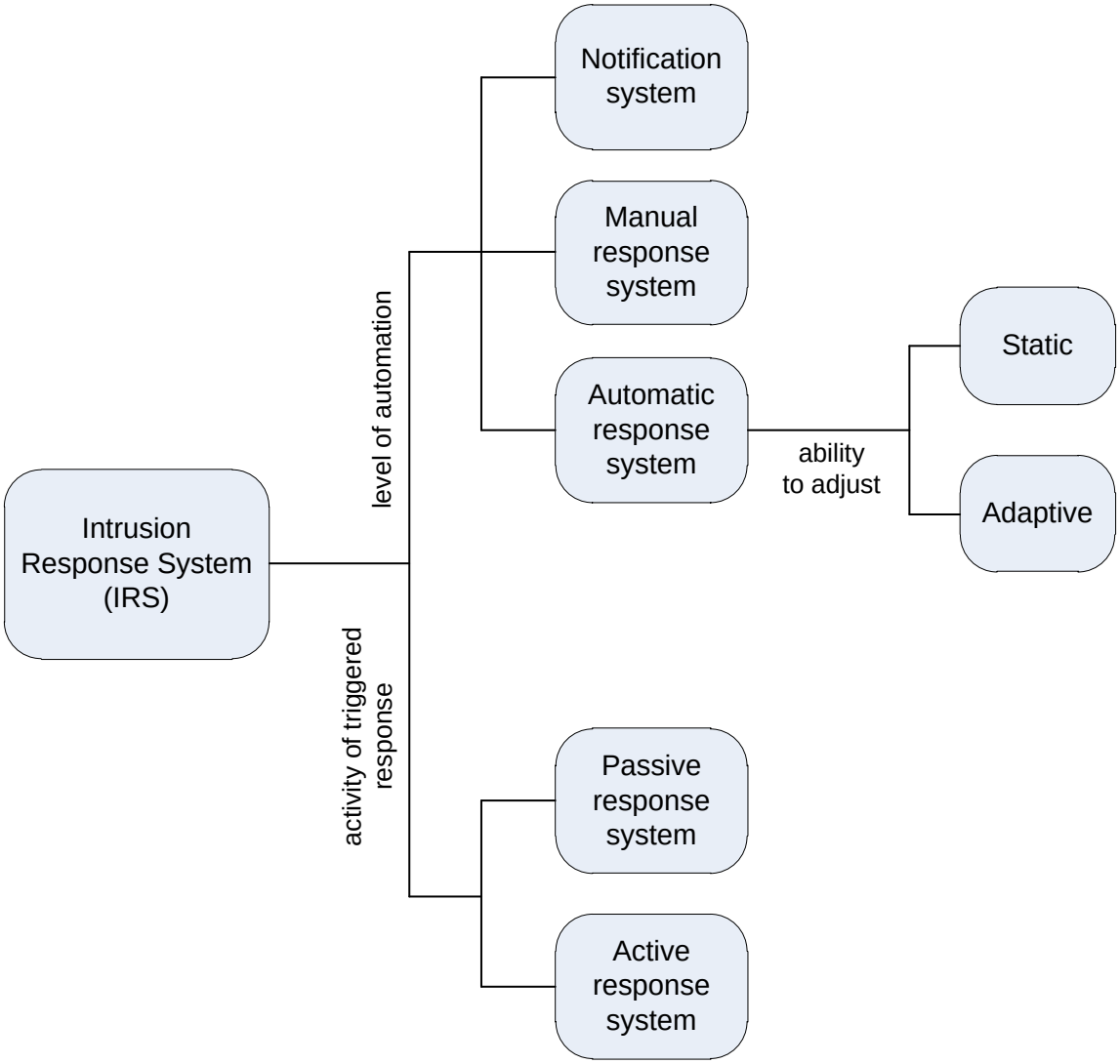

Figure 1: Classification of intrusion response systems

Alarms are generated to alert the system administrator immediately after identification of a potential attack. Alarms can be presented in a variety of formats including email messages, console alerts, and pager activations. After notification, it is system administrator's responsibility to further investigate the situation.

– **Manual response systems:** Compared to notification only systems, these systems provide greater degree automation and provide the additional capability for the system administrator to launch a manual response from a predetermined set of responses based on the reported attack information. Manual response systems often direct the user over the selection of correct reactions while permitting the system administrator to make the final judgment on suitable responses. This will allow the system administrator to respond more quickly to detected intrusions and for less experienced system administrators to receive assistance in choosing the correct reaction. Although this higher degree of automation is more useful than notification only response systems, there is still a window of opportunity between the time of intrusion detection and the time when the system administrator initiates

a response. This time gap is still could be large enough into today's environment and manual response systems. As result, today there is research interest to develop automatic response systems to replace manual response mechanisms.

– **Automatic response systems:** Unlike manual response and notification systems automatic response systems tend to provide immediate reactions to detected intrusions. These systems automatically respond to the intrusive behaviour through an automated decision making process and does not delay the process until the situation is analysed by the system administrator. Even though today intrusion response systems are automated automatic intrusion response support is still very limited. Automatic response can be classified into two: Static and Adaptive, based on the ability to adjust to the detected intrusions.

· ***Static:*** If the response selection process remains the same during the attack period it is classified as a static response. Majority of the intrusion response systems are static response systems. In order to enhance the integrated knowledge of the decision making process, these systems are periodically updated by the system administrator. Nevertheless, care should be taken to upgrade such systems regularly before attacks exploit the insufficiency of the current response strategy. The advantage of this approach is that it is easy to maintain even though it takes a conservative view of the system.

· ***Adaptive:*** If the response system is capable of dynamically adjusting to the changing environment during the attack time it is known as an adaptive intrusion response system. Automatic response system could adapt to an on-going attack in two ways:

- could adjust the system resources dedicated to intrusion response, for example additional intrusion detection systems could be activated
- could adjust the response mechanisms based on the failure and success of the previously made responses. Failure of a response could be due to activation of an incorrect reaction to a detected intrusion or the intrusion detection system falsely detecting a normal system behavior as an intrusion (false positive). Thus here the adjustment of response could be either switching to the correct reaction or rerunning the detection process.

Activity of triggered response

A discussion on how intrusion response mechanisms could be classified based on the activity performed is given below.

– **Passive response systems:** The objective of the passive response system is to alert the system administrator about the detected intrusion and to provide attack information. These systems do not try to prevent further intrusions or minimize the damage already caused by the intrusion.

– **Active response systems:** In comparison to passive response systems active systems intend to take precautions to minimize the damage caused by the intruder and try to locate or warn/harm the intruder.

The majority of the existing intrusion prevention systems provide passive responses [8].

Table 1 gives an overview of some of the passive and active approaches that could be used in intrusion detection response systems.

Table 1: Passive and active intrusion responses

Passive	Active
1. Administrator notification Generate alarm *(through email, pager notification, etc.)* Generate report (information about an intrusion: attack target, time, criticality, etc.) 2. Enable intrusion analysis tools 3. Enable additional IDS 4. Trace connection for information gathering purposes 5. Back up tempered files 6. Enable local/remote activity logging	<u>Host based response actions:</u> 1. Deny full/selective access 2. Shutdown compromised service/host 3. Restrict user activity 4. Disable user account 5. terminate/restart suspicious process 6. Disable compromised services 7. Abort/delay suspicious system calls <u>Network based response actions:</u> 1. Restart targeted system 2. Block incoming/outgoing network connections 3. Enable/disable additional firewall rules 4. Block port/IP addresses

2.2 Intrusion response techniques

The protection of the national infrastructures from cyber-attacks is one of the main issues for national and international security. Malicious attacks on critical infrastructures have raised research interest in protection of protection of critical infrastructures such as energy grids and telecommunication networks. Although much research has addressed the issue of increasing security of networked computer systems, problems and malicious acts are on the rise, rather than getting less.

There are a variety of techniques for responding to an intrusion. These techniques range from generating a report to launching a counterattack against the attacker. Following paragraphs describe these techniques as reported in [9]. These techniques are classified into either Passive or Active response techniques as described in section 4.1.

2.2.1 Passive responses

2.2.1.1 Generate a Report

All intrusive behaviour should be logged so it can be reviewed by a system administrator. These reports provide critical information for the resolution of on-going incidents and facilitate long-term analysis of security attacks.

2.2.1.2 Generate an Alarm

The success of an attack is dependent on the time between detection and response. Alarms, implemented through email messages, console messages, pagers, or even loudspeaker announcements, notify the system administrator that an attack is underway. Not all intrusive behaviour, however, should generate an alarm. Different alarms should be triggered corresponding to the severities of the detected intrusions as explained in deliverable 3.2: Real time intrusion detection strategies.

2.2.1.3 Enable Additional Logging

Some user behaviour cannot be unambiguously characterized as intrusive behaviour but is nonetheless indicative of possible intrusive behaviour. In such cases, enabling additional logging allows for the gathering of additional information that may help in classifying the user's behaviour.

2.2.1.4 Enable Remote Logging

Additional logging may not be sufficient against certain types of attacks or attackers and instead, remotely logging to another system or a non-changeable media (such as CD-ROM or a printer) may be a better technique for gathering additional information on the attacker.

2.2.1.5 Enable additional intrusion detection tools

Because intrusion detection tools are imperfect and consume system resources, intrusion response systems may enable additional intrusion detection tools, as the degree of suspicion increases that an intrusion is on-going. More robust and costly (in terms of resource utilization) detection tools are employed as additional indicators of intrusive behaviour are found.

2.2.1.6 Trace connection

Criminal prosecution of computer attackers, while a viable response to intrusions, is outside the scope of intrusion response systems. However, tracing by the network connection of an attacker so that the attacker can be positively identified is a viable response. As a side effect, the attempt to trace back a connection can be detected by the attacker. For less experienced attackers, the fact that someone is actively trying to trace them will often result in the termination of the attack.

2.2.1.7 Create Backups

Attacks against the integrity of a system can be thwarted by creating up-to-date system backups for system restoration and file comparison. While it is often impractical to maintain real-time backups of all modified files, as the degree of suspicion that the system is being

attacked increases, the time interval between backups should be decreased so as to limit lost or corrupted data.

2.2.1.8 Employ Temporary Shadow Files

A temporary shadow file is a duplicate file created and encrypted to protect the original file. When an intruder attempts to modify a critical system file, all modifications are saved in a second file and the original file remains unchanged. Additional modification attempts result in changes to the temporary shadow file and not the original file. Fisch proposed temporary shadow files as a mechanism for protecting the integrity of system while under active attack [10].

2.2.2 Active responses

2.2.2.1 Lock User Account

If a user account has been compromised, an appropriate response would be to lock that user's account so that it cannot be used to launch future attacks.

2.2.2.2 Suspend User Jobs

If there are indications of intrusive behaviour as well as normal user operations, the suspension of user jobs and termination of user sessions allows the system administrator the opportunity to terminate any intrusive jobs while not corrupting valid user tasks. While termination of user sessions without suspension of user jobs would be a more common response, there are circumstances when it would be desirable to suspend user jobs.

2.2.2.3 Terminate User Session

If a user is involved in intrusive behaviours, the user's session should be terminated and the user's account locked to prevent future damage.

2.2.2.4 Block IP Address

If the IP address of an attacking system can be identified, some network attacks can be neutralized by blocking, at a router, all traffic from that address. While this protection is often temporary if the attacker can change their IP address, it will slow the attacker and allow the intrusion response system or system administrator more time to respond to an attack.

2.2.2.5 Shutdown Host

Sometimes the only mechanism for protecting against further system compromise is to shut down the machine. While this is a draconian measure, it is sometimes the only mechanism for protecting a host under an active attack.

2.2.2.6 Disconnect from the Network

For network-based attacks, disconnecting from the network is less draconian than shutting down the host but has the same effect - network-based attacks can no longer affect the system allowing the system administrator time to response to an attack and repair any damage to the attacked system.

2.2.2.7 Disabling the Attacked Ports or Services

If a single service or well-known port is being used as the basis for the attack, that port or service can be disabled effectively stopping the attack without affecting any of the other services offered by the system.

2.2.2.8 Warn the Intruder

Most attackers operate with the assumption that they are not being actively monitored or that they can evade intrusion detection systems. Telling the intruder that they are actively being monitored is generally all that is required for them to abandon the attack.

2.2.2.9 Force Additional Authentication

Forcing additional authentication slows down or stops an attack while allowing authorized users to continue to use the affected system. The suspected intruder must provide additional proof of their identity before they can execute commands.

2.2.2.10 Restrict User Activity

Suspicious users may be restricted to a special user shell that allows some functionality while limiting the ability of the user to execute certain commands. This will slow the user's ability to damage the system without terminating a user session, suspending user jobs, or requiring additional authentication.

Following chapters discuss state-of-the-art automatic intrusion response strategies and proposed intrusion response strategies of task 4004.

3 Automatic Reaction Strategies

In this chapter state-of-the-art intrusion detection strategies based on a) Machine learning and modelling and b) Signal (information) processing are discussed. In section 3.1 machine learning and modelling approaches related to graph based modelling, information fusion and response strategy modelling are presented. Section 3.2 discusses intrusion reaction strategies based on attack trees, rules and risk assessment, centred on signal (information) processing.

3.1 Machine learning and modelling based intrusion reaction strategies

3.1.1 Graph based modelling for attack response

Underlying components/devices of critical infrastructures are controlled by communication networks. This underlying physical infrastructure could be modelled by graphs. As a result it is reasonable to assume that various problems related to these infrastructures could have solution spaces in areas that use graphs as common models, e.g. graph or scheduling theory [11]. In an attempt to increase accuracy in attack response problems, we will investigate the possibility of transforming this problem to other disciplines. Problem transformation is a well exploited practice in research where a solution for a difficult/complex problem is investigated in a different solution space where a solution could be found at lesser cost. Once the solution is found in the new solution space, it will be translated back to the original problem space using a reverse transformation.

The section below presents the concept of transformation model to analyse attack response in critical infrastructures. The transformation model allows solutions to be based on graph modelling concepts.

3.1.1.1 Model overview

The basic concept of a transformation model is shown in Figure 2. For the description of the model overview it should be noted that the application under consideration is associated with critical infrastructures such as electric power grids and telecommunication networks as considered in the CockpitCI project. Descriptions of the different blocks in Figure 2 are given below.

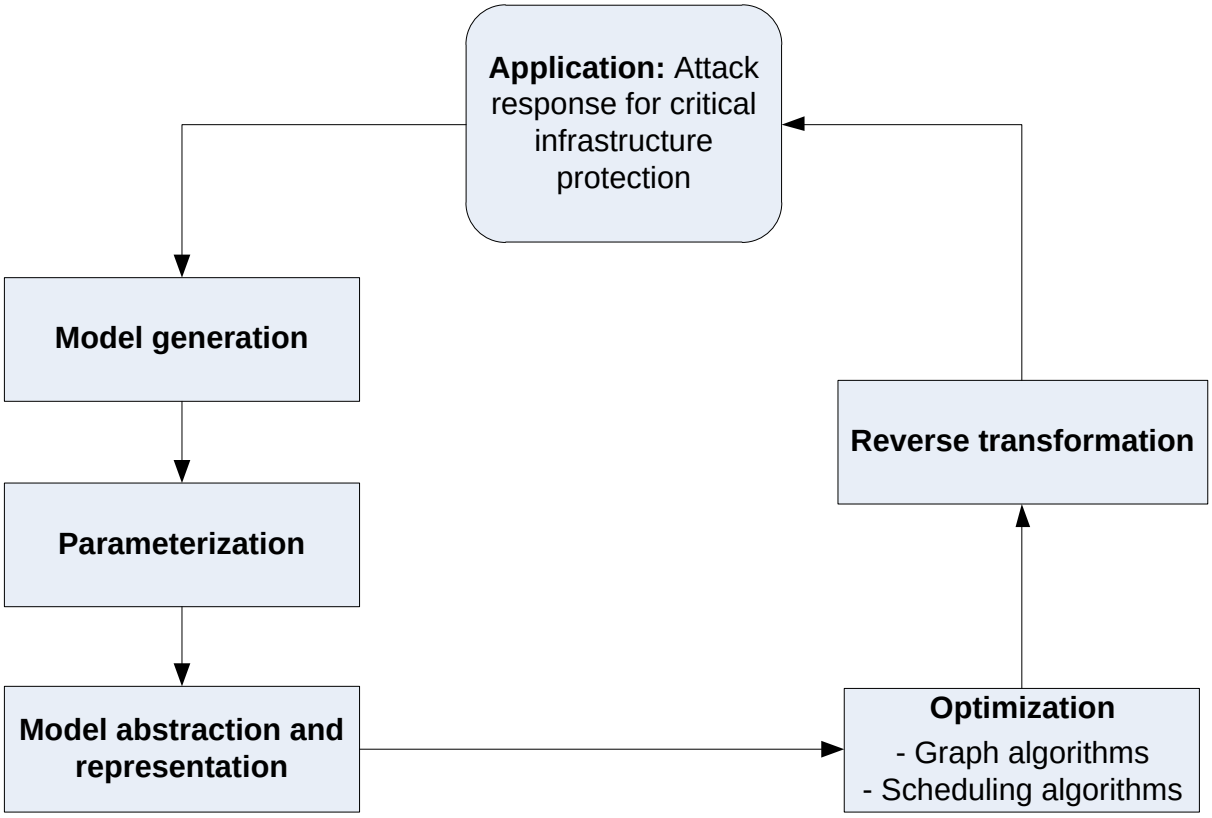

Figure 2: Modelling overview

Model Generation

The application is first transformed into a task-graph together with the task model specification. The model consists of a directed graph G = (V;E), where V is a finite set of vertices v_i and E is a set of edges e_{ij} , $i \neq j$, representing precedence relations between v_i; $v_j \in$ V. Critical infrastructure protection problems have topology maps that can be represented by directed or undirected graphs, G. Typical examples are electrical power grids and the underlying communication networks controlling these infrastructures. In a probabilistic graphical model, vertices represents a random variable (or group of random variables), and the edges express probabilistic relationships between these variables. The graph then captures the way in which the joint distribution over all of the random variables can be decomposed into a product of factors each depending only on a subset of the variables. Directed graphs are useful for expressing causal relationships between random variables. Bayesian network is a directed graphical modelling technique, which could be used for this purpose. A brief description of Bayesian network modelling is given below.

Bayesian Networks

The Bayesian network also called the Belief network, uses factored joint probability distribution in a graphical model for decisions about uncertain variables. The Bayesian network classifier is based on the Bayes rule. Given a hypothesis *H* of classes and a data *x* we have, then

$$P(H|x) = \frac{P(x|H)P(H)}{P(x)} \qquad (1)$$

Where,

- *P(H)* denotes prior probability of each class without information about a variable *x*
- *P(H|x)* denotes posterior probability of possible classes
- *P(x|H)* denotes the conditional probability of *x* given *H*

As shown in Figure 3 Bayesian networks are presented with nodes representing random variables and arcs representing probabilistic dependencies between variables and conditional probabilities encoding the strength of the dependencies, while unconnected nodes refer to variables that are independent of each other. Each node is associated with a probability function corresponding to the node's parent variables. The node always computes posterior given proof of the parents for the selected nodes. For example, in the figure factored joint probability of the network is computed as:

$$\boldsymbol{p}(x_1,x_2,x_3,x_4,x_5,x_6) = \boldsymbol{p}(x_6|x_5)\boldsymbol{p}(x_5|x_3,x_2)\boldsymbol{p}(x_4|x_2,x_1)\boldsymbol{p}(x_3|x_1)\boldsymbol{p}(x_2|x_1)\boldsymbol{p}(x_1) \qquad (2)$$

Where,

- *P(.)* denotes probability of the variable
- *P(.|.)* denotes conditional probability of variables

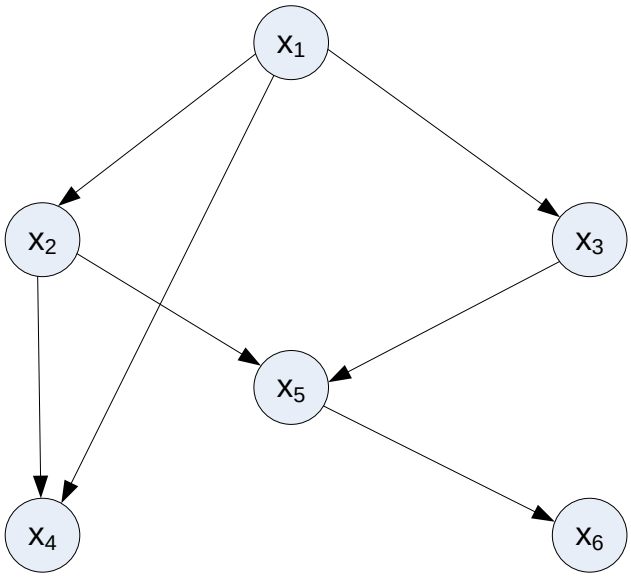

Figure 3: Example of a Bayesian network

Naïve Bayes is a simple Bayesian network model that assumes all variables are independent. Using the Bayes rule for Naïve Bayes classification, we need to find the maximum likelihood hypothesis, which determines the class label, for each testing data *x*. Given observed data *x* and a group of class labels $C=\{c_j\}$, a Naïve Bayes classifier can be solved by maximum posterior (MAP) hypothesis for the data as follows:

$$\arg\max_{c_j \in C} P(x|c_j)P(c_j) \qquad (3)$$

Naïve Bayes is effective for inference tasks. However, Naïve Bayes is based on a strong independence assumption of the variables involved. Surprisingly, the method gives good results even if the independence assumption is violated

Parameterization

Once the application is mapped to edges (E) and vertices (V) of G then it is necessary to map system specific parameters such as power transmission, sensitivity or confidentiality, communication cost, network throughput, relative importance based on the cost of loss of services etc. to generic parameters. Weights need to be assigned to edges and/or vertices of the generated graph to represent their characteristics. Therefore, edge and vertex weights are defined respectively for each edge E and vertex V. For example, let w^e_{ij} denote the weight of edge e_{ij} and w_i^v denote the vertex weight of v_i, where v_i, $v_j \in V$ and $i \neq j$. Depending on the application if multiple parameters are needed multiple weights may be defined for edges and/or vertices. In such a scenario w^e_{ij} [*m*] and w^v_i[*n*] represent the m^{th} and n^{th} parameter respectively of weight vectors w^e_{ij} and w^v_i .

Model Abstraction and Optimization

The graph G will reflect a particular scheduling problem once weights have been assigned. A graph theoretical presentation could be characterized by the graph itself along with the controlling objectives. However, a scheduling theoretical presentation involves requirements of the scheduling model (i.e. the processing environment, and the optimization criteria). The key feature of the model designing procedure is the matching of the intrusion response requirements and objectives with the graph and scheduling model and objectives.

Graph G and schedule model S are subjected to graph and scheduling theoretical algorithms respectively. During the optimization process the aim is to find optimal (or suboptimal) solutions for the required attack response criteria, applying the suitable algorithm(s). For this suitable algorithms need to be investigated that suit the optimization criteria, i.e. attack response criteria, considering response time or computation requirements. Then after the application of graph or scheduling algorithms, optimal or sub-optimal solutions will be available.

Reverse Transformation

Once the solutions for graph and scheduling algorithms have been derived they must be transformed back to the original problem space (application). This involves reverse transformation equivalent to the transformation utilized in the model generation process. Basically this represents the backward transformation from solution space to application space where the problem exists.

This above presented approach could be used to derive solutions from graph based modelling to solve problems occurring in security applications via problem transformation.

3.1.2 Reaction support with multi-source information fusion

In this chapter information fusion and prioritization for threat assessment is discussed in the following sections. An information fusion based reaction support framework is presented in order to quantify the risk(s) associated with the attack and the cost for the reaction(s) to be taken. This will provide intelligence to the field equipment to reliably identify the threat and to take correct actions to prevent it automatically.

3.1.2.1 Information Fusion for Threat Assessment

For each individual alarm triggered by the IDS (Intrusion Detection System), the decision making process needs to understand how likely it is that the alarm corresponds to an actual attack. Using Bayes's theorem, the probability of a sensor alarm meaning an actual attack could be expressed as follows [12]:

$$P(attack|alarm) = \frac{P_d P_a}{P_d P_a + P_{fa}[1 - P_a]} \qquad (4)$$

Where P_d is the probability of detecting an attack, P_a is the probability of an attack occurring and P_{fa} is the probability of a false alarm. For the context of this report we define the term $P(attack|alarm)$ as the positive predictive value. Figure 4 shows the relationship between the positive predictive value and probability of a false alarm (P_{fa}) for different P_a values. For this illustration we have used a detection probability (P_d) of 0.9.

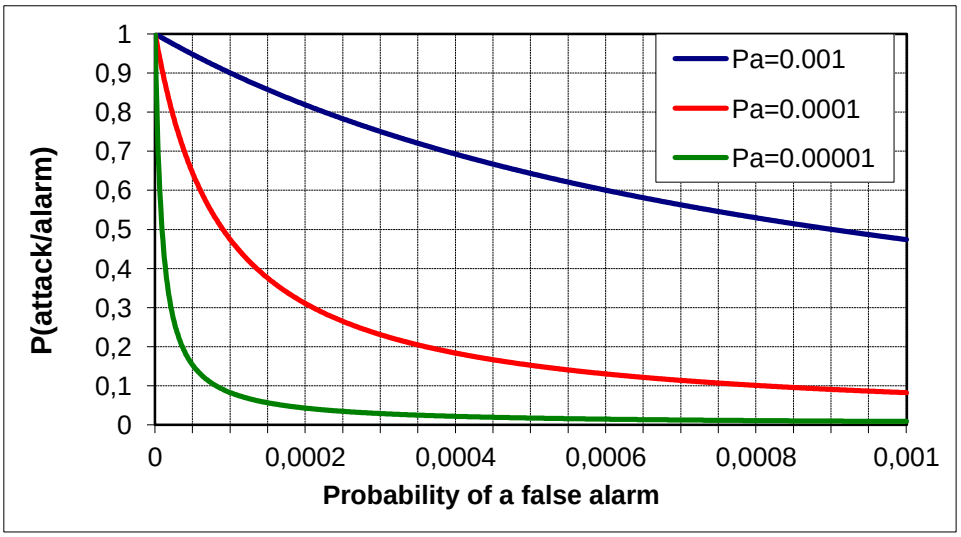

Figure 4: Relationship between positive predictive value and probability of false alarm

It is noted that when the probability of the attack decreases (as we hope in the case of cyber-attacks) the positive predictive value will also decrease for a given false alarm rate. In other words, the confidence of a sensor alarm actually reflecting a true attack will reduce. Furthermore, the relations indicate that for a high positive predictive value it requires that P_{fa} be much lower than P_a. For instance, when P_a =0.0001, P_{fa} needs to be in the range of $1x10^{-5}$, in order to have a 90% chance of that alarm corresponding to a true attack. However, developing an intrusion detection sensor that has such a low false-positive rate is an extremely difficult task. Thus, it is highly likely that the decision making process will have to rely on moderate positive predictive values to make decisions. However, this does not imply

that no action should be taken. It implies that the decision to react and the type of reaction should take into account factors such as cost of the reaction, associated risk level and the frequency of the alerts. For instance, a high cost action should not be taken if the positive predictive value is not close to 1, unless the associated risk level is extremely high. If a high cost action is taken with a low positive prediction level and later on if the action proves to be unnecessary, then the confidence in the cyber-security infrastructure could be questioned. As a result, for an efficient cyber-security infrastructure it requires an automatic threat assessment module to be incorporated to the IDS.

The objective of the threat assessment module is to quantify the risk(s) associated with the attack and the cost for the action(s) to be taken. This will provide intelligence to the field equipment to reliably identify the threat and to take correct actions to prevent it automatically. However, due to the moderate positive prediction value of an abnormal event, additional information is required by the threat assessment module in order to take correct reactions to the alerts raised. This raises the necessity of an information fusion framework. Information fusion is the process of intelligently combining information from different sources to enhance understanding on the data and its implications in order to provide an outcome that is superior to any provided by an individual source. Following a triggered alarm indicating a potential attack, there are number of information that the decision making process would like to be aware of. In addition to the positive prediction value of the attack, information such as the time of attack, extent and the time of contamination, whether it was intentional or not and the cost of the reaction to prevent it are valuable information in order to make a reliable decision. This information will come from different sources in different formats. Moreover, this information could arrive through uncertain sources and may conflict with one another. Also, information may be collected at different times and locations. Thus the information fusion framework should have an approach to analyse the different types of data and to merge the information in order to present a reliable and informative decision support to the operator. This would require huge amount of data processing which will mainly involve the following tasks:

- data cleaning: noise and irrelevant data will be removed
- data selection techniques: only useful features are extracted from the data to obtain a reduced data set while keeping the integrity of the original information
- data transformation: different data is transformed to a suitable format(s) in order to combine the knowledge of each data source
- pattern recognition: useful patterns of the data are identified

Due to the amount of data mining involved in the above tasks, machine learning techniques are necessary to address the challenges of the information fusion framework. Pattern recognition, artificial intelligence and statistics could be used to analyse, group and extract features from the entities to perform the above tasks. Thus, the processes will exploit analysis tools from machine learning methods (both supervised and unsupervised depending on the nature of the information), mathematical algorithms and statistical tools to discover and merge the relationships among different information. Figure 5 illustrates an information fusion driven automatic threat assessment architecture base on the above principles discussed. The output of the threat assessment module will provide intelligence to the field equipment to take corrective actions to prevent cyber-attacks.

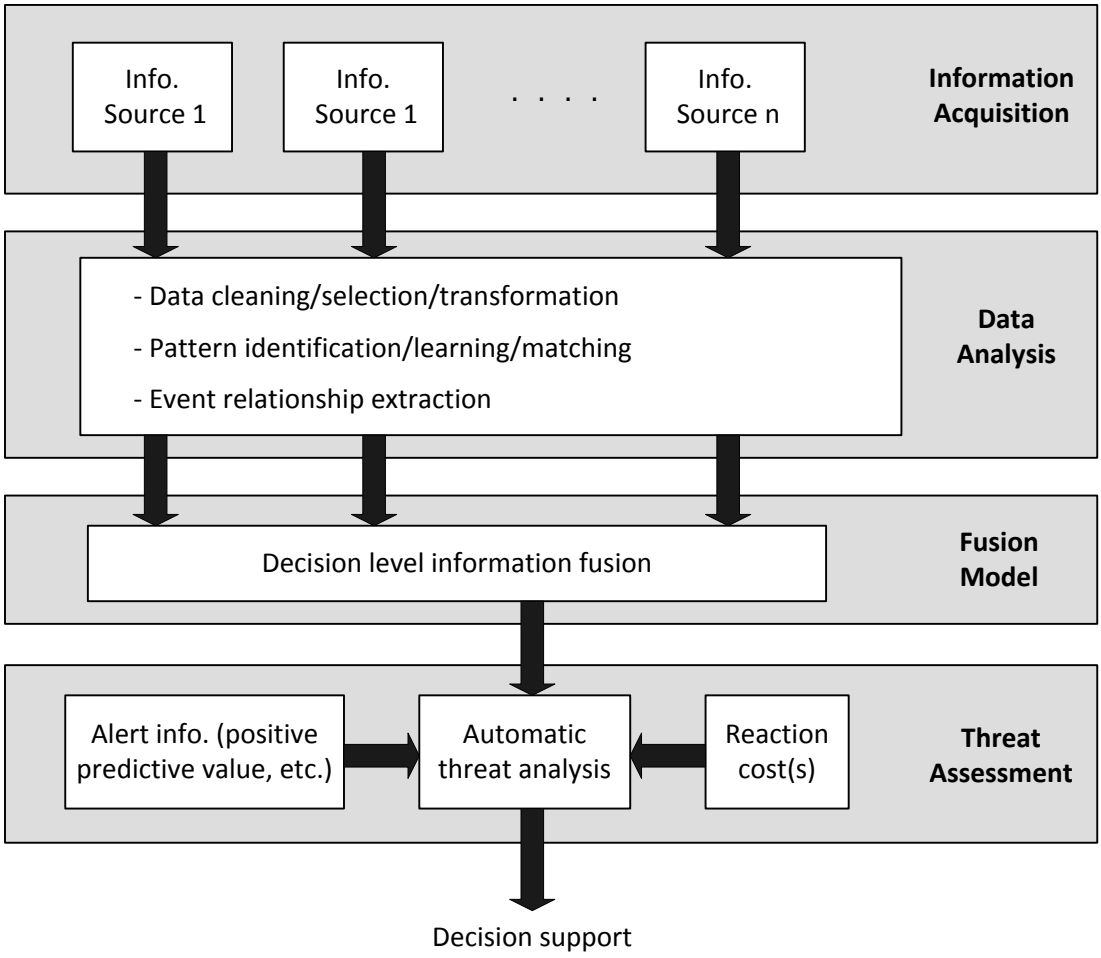

Figure 5: Information fusion model for threat assessment

3.1.2.2 Information Prioritization

The basis of the information prioritization is the assessment of the accuracy of a given alarm along with the severity of the alert triggered i.e. the impact of the related attack on the network. Several devices are very often embedded within networks for probing and detecting security concerns that may result in the system mission being compromised. Such devices often generate a large body of data that cannot be manually processed by the administrator. Indeed, the caveat often lies on how to distinguish and address alerts that require more pressing actions. One solution to this has been to improve the detection mechanism and detection signature of such devices as IDS. Another solution focuses on the outputs generated by these devices through such a technique as alert prioritization.

Alerts prioritization is particularly essential in ranking alerts based on their criticality for the network and subsequently enables the SCADA system administrator to allocate the required security controls proportionally to the security concern that could result from such an alert. In general, alerts prioritization account for domain knowledge such as the security policy in place, the network topology, vulnerability analysis of the network services and installed software [13].

For instance the M-Correlator [14] is an algorithm developed to consolidate and rank a number of incidents taking account of the topology and mission of a network as well as the administrator's interest in that type of incident. The algorithm is supported by an internal database *Incident Handling Fact Base* that provides a description of the different alerts code and the dependencies of incident types to their required OS versions, hardware platform, network services, and applications. It is possible to map and maintain the protected network through Nmap. Then it possible to obtain information relating the assets on the network, IP address to hostname mappings, OS type and version information, active TCP and UDP network services per host, and hardware type. Subsequently, a priority level for an alert will reflect its potential impact for a resource thought to be critical and the amount of interest the user has registered for this class of security alert. A second ranking concerns the actual assessment of each incident with respect to its impact on the overall system mission as reflected by the priority calculation, and the probability that the security incident reported by the network security device(s) has succeeded.

Alsubhi et al. [15] adopted a fuzzy-logic approach for scoring and prioritizing alerts generated by IDS. The metric adopted by the authors relate to:

- the applicability of the attack (a process that checks whether an attack that raises an alert is applicable in the current environment.),
- the importance of victim, which depict network critical components, application, services, etc.
- the relationship between the alert under evaluation and previous alerts,
- and the social activities between the attackers and the victims.

In the context of CockpitCI, we intend to address one of the limitations of current prioritization methods, which do not account for the accuracy of the network device. We strongly argue that prioritization should cumulate both, the information relating the criticality of the alert for the network security, but also the accuracy of the triggered alerts. Indeed, as the percentage of false positive in traditional IDS is known to be relatively high, the definition of mechanism aiming at estimating the confidence level in a network security system such as IDS is paramount. Confidence metrics on a particular network security system would take account of historical data including alerts triggered by the tool and the network security status at the time to judge on its accuracy. Assuming one is expecting a reliability value of *r* for a network security system (S). If the historical record shows that, during a given period of time (S) has triggered N alerts, with K being the number of accurate alerts and P the number of false alerts. Assuming that the reliability value *r* is uniformly distributed over the set of network security systems; the confidence that (S) will exhibit a reliability value greater or equal to α can be expressed by the probability (P): 1-α^k [16, 17]. Displaying such a value of confidence along with an alert triggered by a network security system would help the administrator make an inform decision when it comes to prioritizing certain alert.

3.1.3 Response Strategy Model

A Response Strategy Model (RSM), as explained in [18], which creates a relationship between incidents and different types of response option with different levels of priority is described below. This approach maps an appropriate reaction strategy with an appropriate incident by considering risk response planning and a time management concept in addressing the importance of response time. Moreover, the proposed approach in [18] groups incidents into a similar group based on their priority and allows for a simultaneous response.

In the above RSM approach, the time management concept intends to create efficient reactions to critical incidents. Four categories of tasks are defined with regards to the time management concept and are represented four different quadrants as described below:

- Q1: Critical and Urgent - This quadrant is for the top and high priority incidents. This category allocates immediate response options, which aim to minimize and prevent adverse impacts from any future incidents. For example, an incident with a high severity score is detected in a very critical asset

- Q2: Critical but Not urgent – This quadrant is less urgent compared to the 1st quadrant but still considered as the top priority quadrant. It allocates any planned response options where a proper action is confidently taken to minimize the incident's impact. For example, an incident with a low severity score and detected in a similar asset to the previous quadrants

- Q3: Not critical but Urgent – This quadrant is the third priority and considered a low priority quadrant and it allocates any action that needs additional time to analyze incidents in order to increase the confidence level of the planned responses. Almost the same as the 2nd quadrant in minimizing incidents' impact, this quadrant slowly collects information about incidents as well as minimizing the future impacts of incidents, for example a similar incident to the first quadrant, but detected in a noncritical asset such as a personal computer

- Q4: Not critical and Not urgent - This quadrant is the lowest priority and for a non-urgent incident and noncritical asset. This category includes passive responses. For example, a similar incident to the second quadrant, but detected in a similar asset to the third quadrant, such as a personal computer

To establish a strategic RSM, the proposed approach uses the risk response planning concept. It consists of four different reaction strategies: avoidance, transfer, mitigation and acceptance. Figure 6 illustrates the risk response planning with time management concept and Table 2 [18] shows the relationship map between them and their correspondent quadrants as well as some related examples for their response options.

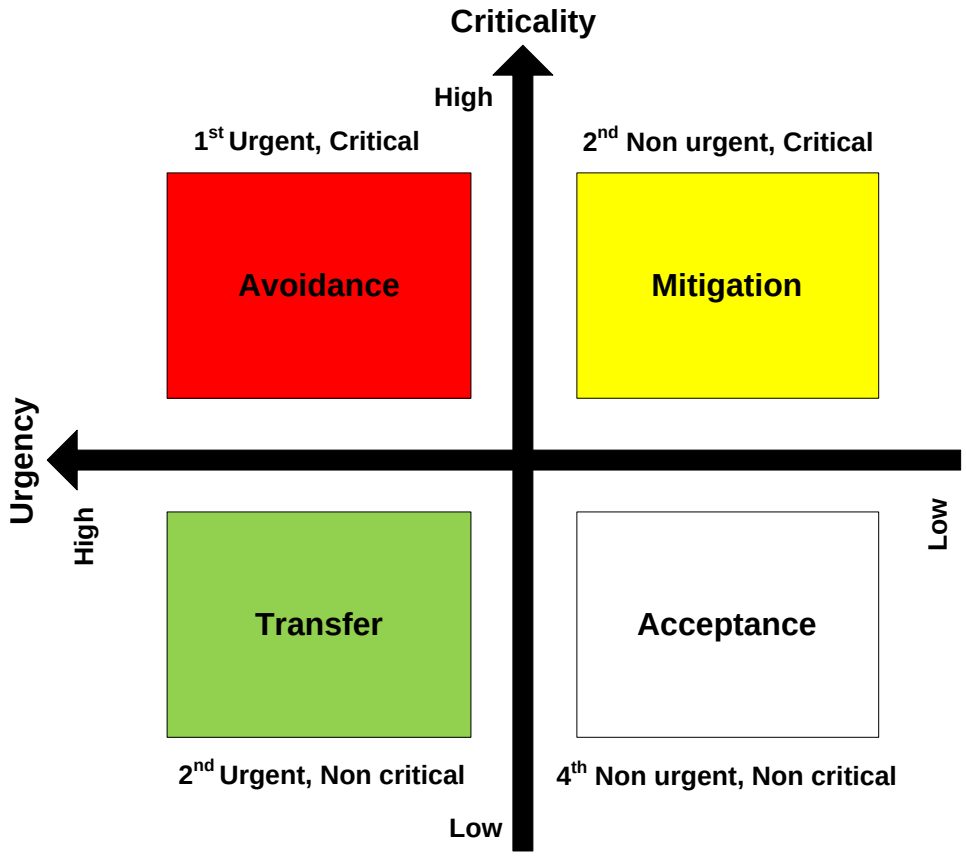

Figure 6: RSM: Risk response planning with time management concept

Table 2: Response strategy planning

Risk Response Planning	Quadrants	Response options
Avoidance	1st Quadrant: Urgent incident and for a critical asset	• Block users, processes or network traffic in preventing future attacks. • Adjust users, processes or network traffic configuration in minimising impacts but maintain system's performances
Transfer	2nd Quadrant: Not an urgent incident but for a critical asset	• Collaborate with other appliances by limiting users, processes or network traffic for delaying the process of attacks (Example: using access control, firewall, enabling other countermeasures or antivirus). • Terminate users, processes or network traffic in preventing continuous attacks (Example: locking OS, resetting connection, dropping user and killing process)
Mitigation	3rd Quadrant: Urgent incident but for a noncritical asset	• Collect information about incidents for passive responses, proactive responses as well as forensic evidence (Example: trace connections, decoy systems, honeypots, forensic evidence, recovery, incidents' blacklisting and white listing) • Escalate to administrator for a further investigation (Example: attack verification, damage recovery and assessment)
Acceptance	4th Quadrant: Not an urgent incident and not for a critical asset	• Establish passive responses like enabling a notification via syslog, console alert, email, pager, PDA or mobile

3.2 Signal (information) processing based intrusion reaction strategies

3.2.1 Reactions using attack response trees

Attack trees [19] [20] present a convenient means to methodically categorize the different ways a system can be attacked. Intrusion reaction systems use an attack tree structure called an attack-response tree (ART) to make it possible to integrate possible response actions against attack. The attack-response trees are designed offline on each computing system located within a host computer. In contrast to attack tree that is designed considering to all possible attack scenarios, the ART model is constructed based on the attack consequences, thus the designer does not have to consider all possible attack scenarios that might cause those consequences. An attack-response tree is used to define and evaluate possible combinations of attack consequences that lead to breach of a security property of the considered system. This security property is allocated to the top-event node which is the root node of the tree. An attack-response tree's is stated in the node hierarchy, such that one can split an abstract attack consequence to a number of sub-consequences which are more concrete. Node breakdown could be based either on an AND gate, where all of the sub-consequences must occur for the abstract consequence to take place, or an OR gate, where occurrence of any one sub-consequence will result in the abstract consequence to occur. The principal sub-consequences and the resulting abstract consequence are known as inputs and output for a gate. Alerts from IDS are mapped to related leaf node of the ART. These indicate possible attempts of an intruder to damage the targeted system. Response boxes are connected to some of the nodes in the ART and define the appropriate counter measure if that node is flagged with an attack.

Decomposition of abstract consequence node (output), i.e., unavailable power supply into two sub-consequences (inputs) using an OR gate is illustrated in Figure 7. The power supply is cut off if either the controlling server or the controlling agent is compromised. Moreover, the intrusion response system is able to switch to the secondary controlling server if the power supply is unavailable due to the compromised controlling server.

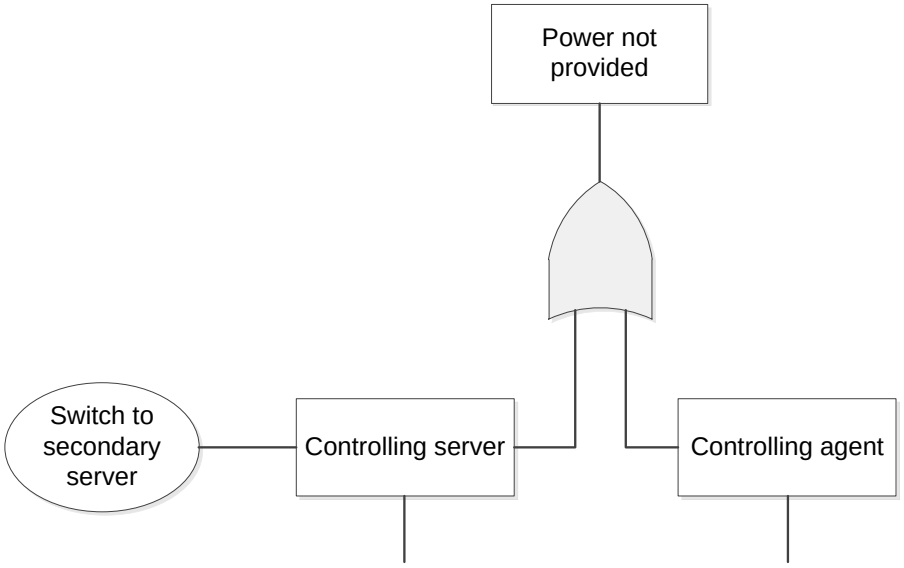

Figure 7: Decomposition of nodes in ART

One of the major goals of an ART, is to verify probabilistically, if the security property concerning ART's root node has been violated, for a given sequence of the received alerts, and the successful response actions. Boolean values are allocated to all the nodes in the attack-response tree. Initially each leaf node consequence is set to 0, and if an intrusion alert is received from the IDS it is set to 1. For other consequence nodes, together with the root node, these values are worked out bottom-up in line with values of the leaf nodes' in the sub-tree whose root is the consequence node under consideration. When response boxes are successfully occupied by the response engine, they are triggered. Consequently, all nodes in their sub-tree are reset to zero when the response boxes are activated, and the corresponding alerts are cleared. For example, all nodes in the ART are reset to zero if the response box that is attached to ART's root node is triggered,

In the section below, a case study on how response selection based on ART is applied to SCADA system is discussed.

3.2.1.1 Case Study: ART applied to SCADA system

This section explains a case study [20] and describes the response selection process for a supervisory control and data acquisition (SCADA) system. It should be noted that since SCADA controls physical assets, timely response is expected when attacks are occurring. Also, unlike general IT computer networks, SCADA systems consist of computing assets with predefined communication patterns and specific responsibilities.

Figure 8 shows a sample 3-bus power grid and its SCADA network, which is responsible for monitoring and controlling the underlying power system. There are three generators, and any one of them can provide the necessary power to the customers, i.e. the electricity load. To monitor the power system, each bus is attached to a sensor, i.e., a phasor measurement unit (PMU). The sensor transmits voltage phasors (i.e., magnitudes and phase angles) of the bus and current phasors of transmission lines connected to that particular bus to SCADA. Furthermore, after receiving sensory data, in order to control power generation, SCADA computes optimal generation set points for individual generators. As illustrated in
Figure 8, SCADA system comprises of different components (HMI, WS, SCOPF, SE and DB). The state of the whole power system is estimated by the state estimation server (SE). These stares and other information are recorded at database (DB) for future reference through the web server (WS). Control commands are computed by the security constrained optimal power flow (SCOPF) and the human machine interface (HMI) using those estimated states.

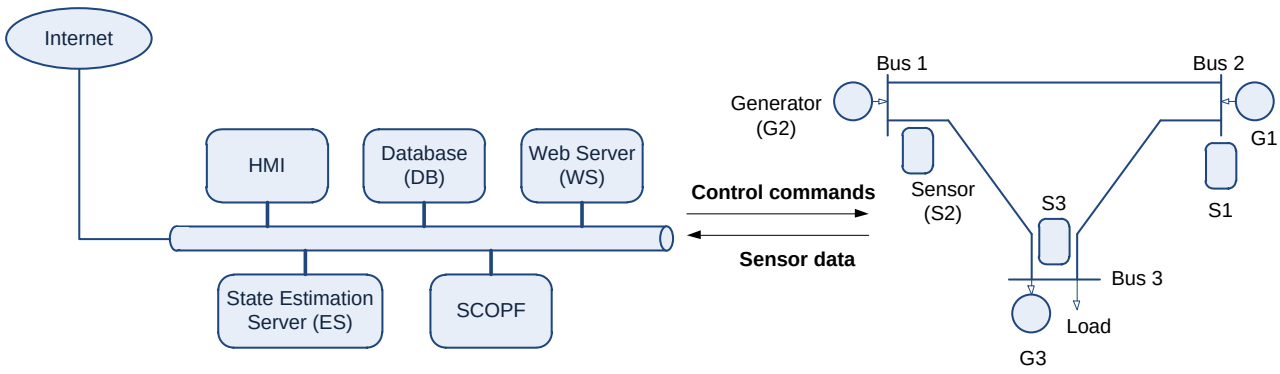

Figure 8: Example of a simple power grid and SCADA network

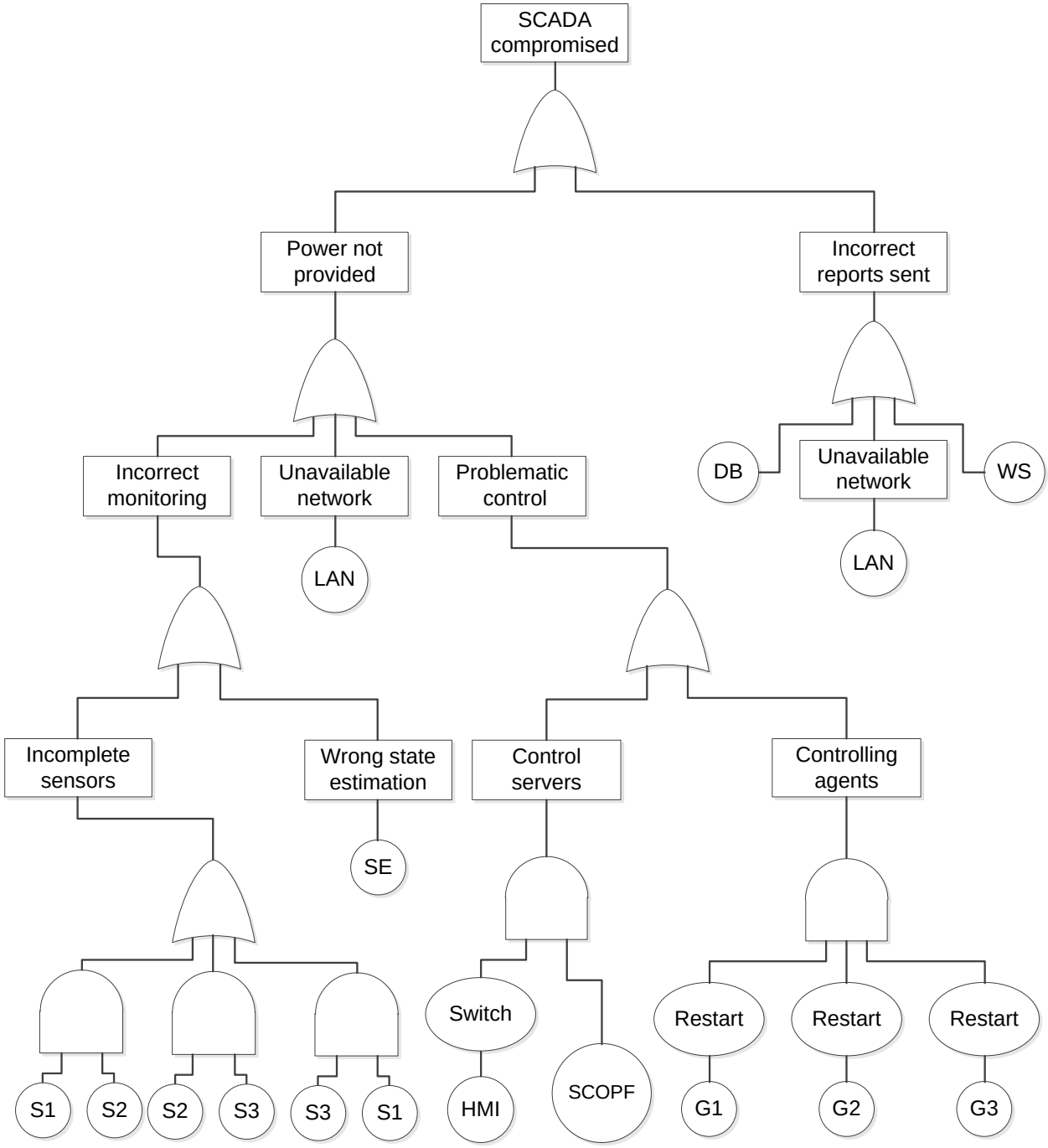

Figure 9: Example ART for SCADA system

For the process control network described above the corresponding network-level ART is shown in Figure 9. The top event or root node is chosen to be “SCADA is compromised”, which is considered as the goal of the SCADA attacker. Shortages in providing loads and report generation (which could be considered as main goals of the supervisory network) are represented by its children nodes. For simplicity, leaf nodes (the bottom nodes which does not have child nodes) here denote compromise of individual host systems, and are updated

by local engines. For example, G1, if set to 1, implies that the controller device for the generator on bus 1 is compromised, and the upper response node “restart” implies that a countermeasure (reaction) for the compromised controller is to reinstall the control software and restart the device. Nonetheless details such as action costs, rates, and probabilities are not shown in this example.

Figure 10 illustrates the binary vector state space definition for the above ART. Each individual bit of the binary vector is set based on reports from the local engines. For example, according to reports from their corresponding local engines the sample state vector in the figure indicates that HMI and G1 are compromised, while other hosts are in their usual functioning mode. According to this current state i.e., S = (100000000010) there are three response actions are available:

1. restart G1
2. switch HMI
3. or remain with No operation (NOP)

According ART in Figure 9, the optimum solution for this is to switch HMI. If any other options were selected, the attacker could have caused a vast amount of harm to the system subsequently by risking the SCOPF server (Figure 9) which could lead to entire control subsystem to fail. Consequently this would affect the way power loads were distributed and ultimately give rise to the top event “SCADA compromised.” For that reason, the automatic reaction system selects the response action that minimizes the maximum damage that the attacker can cause later. With the availability of ART information this optimum solution could be calculated by a Markov decision process.

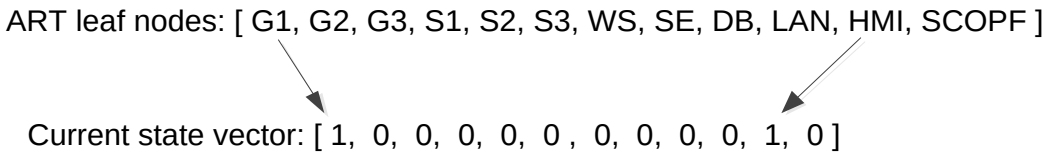

Figure 10: A sample cyber-security state

3.2.2 Rule based approach for attack response

In Rule-based expert systems a set of programmed rules are applied to existing information for problem solving. Usually these rules consist of a set of conditional sentences, which can be used by the computer to check data logically before reaching a conclusion (i.e. response to the attack). Programming such systems involves the integration of a large knowledge base. Conclusions attained here can give information about the statistical probability of the decision for the reference of technicians and operators.

Automatic rule-based response systems are intended to function similar to human experts who use their judgment to detect intrusions. The system utilizes the knowledge base to generate a set of rules in the form of if-then statements. When rule-based response systems come across likely intrusions, they apply these rules to restrict the causes and create solutions to react or counter attack. For example, a system that monitors an electrical grid would have an intrusion detection system to detect possible intrusions. If an intrusion is detected, depending on the characteristics of the detected intrusion (i.e. severity, targeted source, time of attack, extent and the time of contamination, cost of the reaction to prevent it, etc.) there could be several rules to establish the optimum reaction, to minimize the damage or to prevent any further damage or to recover from the damage. A rule based reaction system will follow step by step the corresponding if-then conditions, based on the detected characteristics of the attack (this could be achieved with a information fusion framework), to automatically determine the optimum response to the attack. These rule-based expert systems use logic that can be familiar to human experts who use similar treed decision making in the evaluation of problems.

This form of artificial intelligence is not perfect. Rule-based expert systems could fail to handle situations that fall outside their knowledge base and experience. For instance for a detected attack, depending on the characteristics the attack, a system that does not have a predefined reaction strategy could fail to achieve an optimum reaction. However, there could be a general rule to reset or shut down the compromised units in the case of an unknown scenario or to alert the system administrator about the situation. The system can accumulate information over time to improve the knowledge of the system to minimize the failure of not reaching an optimum solution. Since the rule-based response system operates under if-then conditions the probability of incorrect decisions being taken is less compared to other automatic reaction strategies.

An example framework of a rule based system is described in the flow chart illustrated in Figure 11. The selection of a particular strategy depends on the meeting of certain condition(s), similar to the if-then approach described earlier. In this example, the objective is to make the selection of a particular response strategy based on three important parameters, namely the IDS efficiency (IE), the alarm frequency (AF), and the assessed risk level (RL). Threshold levels X, Y, Z for the above parameters, IE, AF and RL are respectively assigned by the system administrator. As seen by the flow chart, based on the values of the three parameters IE, AF and RL, a different reaction strategy is adapted to the detected intrusion.

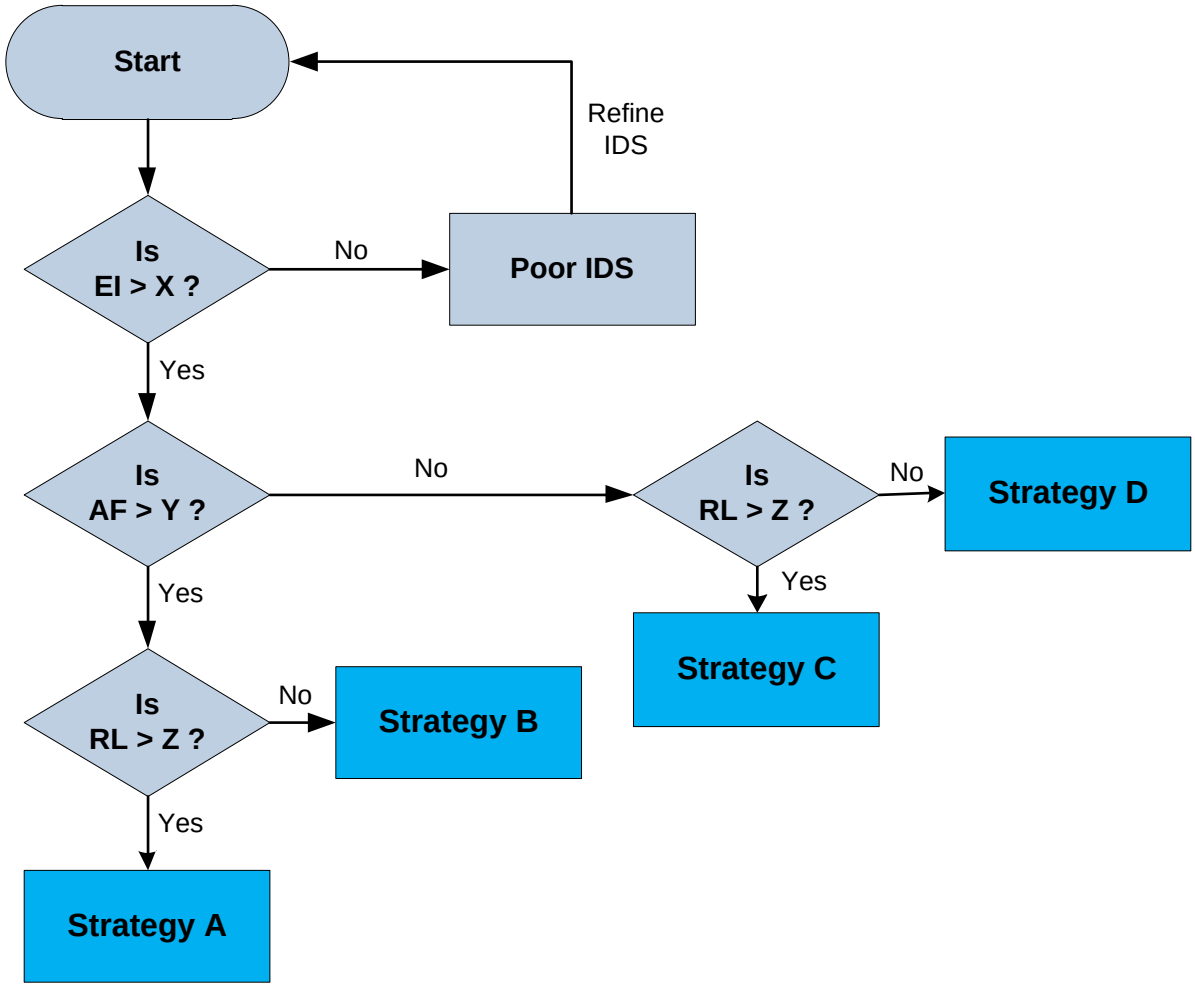

Figure 11: Example of rule based intrusion response selection strategy

3.2.3 Risk assessment based intrusion response

The risk assessment centred response strategy described below is based on a cost-efficient, alarm/defence matrix framework as described by the authors in [ref]. This is an extension of the cost sensitive intrusion detection response (IDR) model proposed by the authors of [21]. A brief description about the risk assessment based intrusion response strategy proposed in [21] in given below.

This strategy introduces new concepts of alarm and defence matrices, residue risks, and wasted response to specify a dynamic intrusion response system. Figure 12 illustrates the framework of the intrusion response system. The framework consists of 3-funtional blocks: Intrusion Detection System (IDS), Risk Assessment Systems (RAS), and Intrusion Response System (IRS). The objective of the IDS is to detect possible intrusion. Once an intrusion is detected it will raise an alarm. When the IDS triggers an alarm it is necessary to estimate the potential risk (or the damage) of the trigged intrusion. Thus the objective of the RAS block is used to evaluate the risk levels and assess the potential damages. Moreover, the RAS should be able to differentiate various forms of false alarms or miss detections. In this approach, the risk level(s) from an attack is defined in terms of the potential damage the attack may cause to the network hosts or to other resources, such as servers and database, within the network domain. All possible losses are combined to a single cost figure, for the sake of clarity and simplicity to convey the main ideas of the RAS and IRS development. Finally, the IRS module will trigger the appropriate reaction strategy based on the decision of the RAS module.

The above scheme is designed to analyse the network IDS reports and to assess the potential damages or residue risk after countermeasures are enabled. Several response strategies are suggested, based on the alarm confidence, attack frequency, potential damages, and response costs assessed. Measured false alarms and detection rates are used in the armed response decision process. For more details on the above risk assessment centred response strategy approach, please refer to [21].

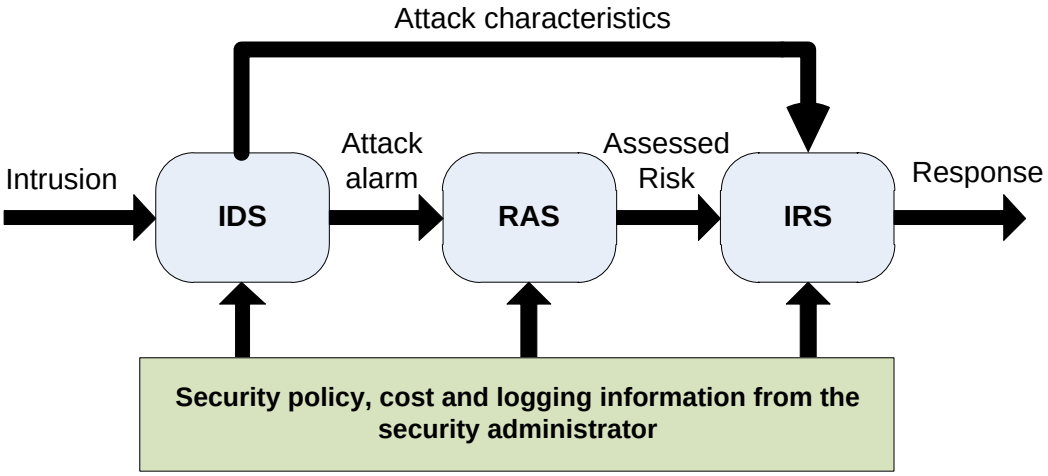

Figure 12: Risk assessment based intrusion response framework

4 Proposed Automatic Intrusion Response Strategies

In this chapter proposed intrusion reaction strategies of Task 4004 are discussed. Section 4.1 presents a combined rule and fusion based approach for intrusion reaction and a reputation based system for automatic reaction is discussed in section 4.2.

4.1 Combined rule and fusion based automatic reaction

An automatic intrusion response strategy based on a combined rule and fusion based approach is discussed in this section. Figure 13 illustrates an overview of this proposed automatic intrusion response strategy, in which the part on the left represents the existing state of the arts technologies as highlighted by the rule-based responses, and the part on the right the proposed technology development and algorithm design highlighted as fusion-based responses.

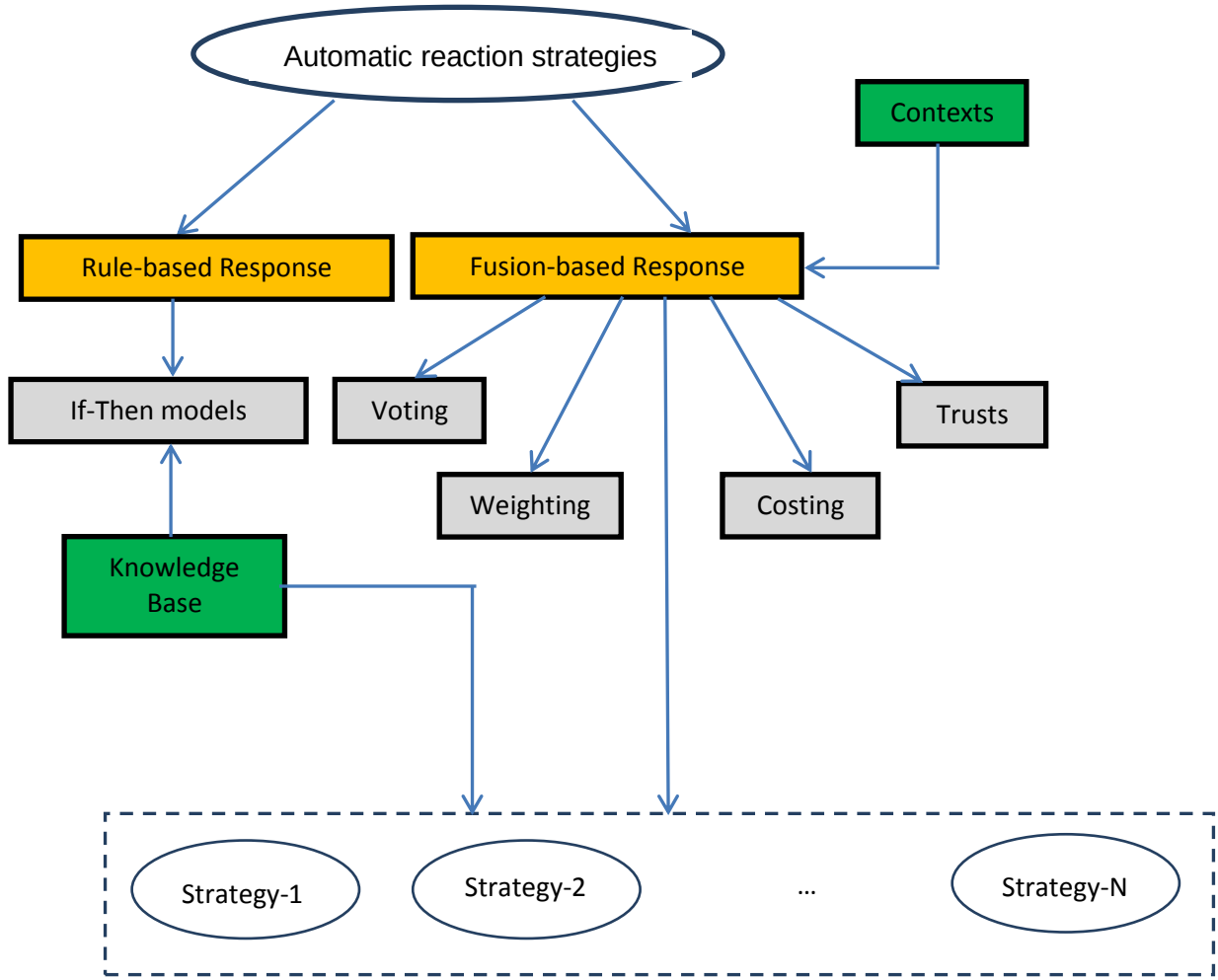

Figure 13 : Illustration of automated reaction strategies

As seen, the existing work on automated intrusion response strategy can be essentially categorized as rule-based, where a range of protocols based on various intrusion detection components at different layers are taken into consideration to formulate rules and controlling mechanism to activate corresponding actions. Under DARPA funding in the US, researchers from an American consortium consisting of Boeing's Phantom Works, University of California Davis' Computer Security Lab, Network Associates' NAI Labs etc. has reported their work [22] on developing an Intruder Detection and Isolation Protocol (IDIP) to implement an infrastructure automatic intrusion response strategy. The system integrates a number of components with a variety of boundary control devices, hosts, and intrusion detection sub-systems. In terms of functionalities, the automated capabilities include: (i) cooperative tracing of intrusions across network boundaries and blocking of intrusions at boundary controllers near attack sources; (ii) use of device independent tracing and blocking directives; and (iii) centralized reporting and coordination of intrusion responses.

In [23], a Risk Index Model (RIM) has been reported, which prioritises incidents based upon two decision factors namely impact on assets and likelihood of threat and vulnerability. The reported research also extends RIM by using it as the basis for mapping incidents with various response options, and then further develops a mapping model, Response Strategy Model (RSM). It is based on risk response planning and time management concepts (rules) and it is evaluated using the DARPA 2000 dataset. A case study analysis is also conducted upon the dataset, which shows a significant result in mapping incident into different quadrants. In particular, the results have shown a significant relationship between the incident classification with incident priorities where false incidents are likely to be categorised as low priority incidents and true incidents are likely to be categorised as the high priority incident. In [24], a taxonomy of intrusion response systems has been established and analysed based on surveying of existing work in relevant fields, from which a conclusive view of the existing efforts can be highlighted as follows to formulate the foundation for our proposed algorithm design and possible technology innovations.

(i) Over recent years, increasingly interests are focused on developing *cost-sensitive modelling* of response selection. The primary aim for applying such a model is to ensure adequate response without sacrificing the normal functionality of the system under attack. While a number of response frameworks are reported to offer facilities responsible for these mechanisms, very little work is done in providing the detailed algorithms.

(ii) In terms of *response-deployment* time, the majority of the proposed frameworks conservatively invoke responses once the existence of intrusion is a certainty. Though this reduces false-positive responses, delayed responses can potentially expose systems to a higher level of risk from intrusions with no mechanism for restoring system to its pre-attacked state. Therefore, a few research efforts developed proactive response mechanisms to enable early response to intrusions, notably, most of them appeared just recently. It should be also mentioned that developing an optimal proactive response mechanism is difficult as it can prohibitively increase false positives.

(iii) *Adaptiveness* is a powerful feature required to ensure normal functionality while still providing effective defence against intrusive behaviour, and to automatically deploy different responses on the basis of the current system state. At the same time, adaptiveness brings a system into a higher level of complexity and poses new questions such as *How can we automatically classify a response as a success or a failure? If the response has failed, how can we determine whether the system state changed due to a triggered (failed) response or a continuance of the attack? How can we separate the beginning of a new intrusion and continuance of the old attack?*'. As such, very few of the existing response mechanisms addressed such issues.

Following the investigation of the existing efforts in relevant fields, we identify that important factors to be considered as: (i) comprehensive coverage of existing strategies among the infrastructures identified in the CockpitCI project; (ii) establishment of further strategies out of the research and innovation among the project, and these two factors enable us to establish a comprehensive list of strategies corresponding various circumstances and contexts, as shown in Figure 13 at the bottom. (iii) Response and reactive timing, where delays are to be balanced among risks of damages and correctness of decisions or strategy selections; (iv) categories of intrusions (level of alarms), via classification of detected intrusions and alarms generated; (v) adaptiveness to contexts and system status. As a result, the entire automated response strategy design can be viewed as a decision support process, where identified intrusions need to be classified, risk assessed, and then selection/decision of strategy-i will be completed based on contexts, circumstances and knowledge acquired etc. A follow-up action is also needed to monitor the consequence of the selected response strategy, and corrections or further actions may be activated to make the system adaptive to system status. Therefore, a multi-source fusion is important to complete the automatic response strategy algorithm design as shown in the part on the right of Figure 13.

Essentially, the fusion-based approach is to consider multi-source information and knowledge, such as factors as identified earlier, consequences, classification results of intrusion detection or level of alarms etc. and hence the optimal decision can be made based on all available information across different protocols, fields, and sectors. For the fusion technology itself, enormous research has been reported in the published literature, and its basic classification can be viewed as four directions as summarized in Figure 13, i.e. voting, weighting, costing, and trusts. Details of the four fusion approaches are highlighted as follows:

(i) Voting involves a parallel structure, in which a number of decision-making units, such as SVM and neural networks, are arranged, and each decision is counted as votes to activate the mechanism that every unit is treated equally and majority of votes is adopted as the final decision. This is similar to $\max\{e_1, e_2, ...e_M\}$ or $\min\{e_1, e_2, ...e_M\}$.

(ii) Weighting is a compromising concept upon voting, where local decisions made by individual units are weighed according to the importance of their contributions in the past or the role of operations inside the infrastructure, or the priorities pre-identified by the maintenance engineers. As a result, each unit will be allocated a

weighting parameter ranging from 0 to 1 to adjust its contribution to the final decision-making. Such weighting mechanism can be implemented in an adaptive manner when those weighting parameters are considered to be controlled by circumstances and contexts.

(iii) Costing involves significant research on mathematical analysis, where a range of costing models can be established together with various conditions. As a result, to minimize the cost, Lagrange multiplier approach can be used to take all conditions into consideration as shown below:

$$f\left\{O_1^i, O_2^i, ...O_k^i \middle| C_1^i, C_2^i, ...C_c^i\right\} = \underset{\substack{\alpha_k^i \in [0,1] \\ C_l^i \in [O_1^i, O_2^i, ...O_k^i]}}{\arg\min} \left(\frac{\sum_k \alpha_k^i P\left(O_k^i\right) + \sum_{l=1}^{c} \lambda_l^i C_l^i}{\overline{k^i\left(\sum P\left(O_l^i\right)\right)} + \overline{C^i\left(\sum \lambda_l^i C_l^i\right)}} \right)$$

Where $\left\{O_1^i, O_2^i, ...O_k^i\right\}$ represents decision outcome, $\left\{C_1^i, C_2^i, ...C_k^i\right\}$ the conditions upon which decisions are made, λ_l^i are Lagrange multipliers, $P\left(O_k^i\right)$ are models for response strategies, and finally, the denominator represents a balancing factor to ensure that the modelling approach is constrained within the range specified by all response modelling functions as well as local conditions.

(iv) Trusts represent a further extension of weighting approach, where weighting parameters are allocated based on reputation of each individual unit. It involves a feedback system, in which each decision making process is closed monitored and performances are analyzed to establish the reputation, which is then used to influence its weighting in the final decision making process.

Figure 14 illustrates the overall algorithm design.

In comparison with the existing approaches, our analysis of automatic intrusion response at the algorithm design level has the following features:

(i) Adaptive to effectiveness analysis and evaluations as well as a range of contexts;

(ii) Multi-source information/channel fusion to support the decision making process.

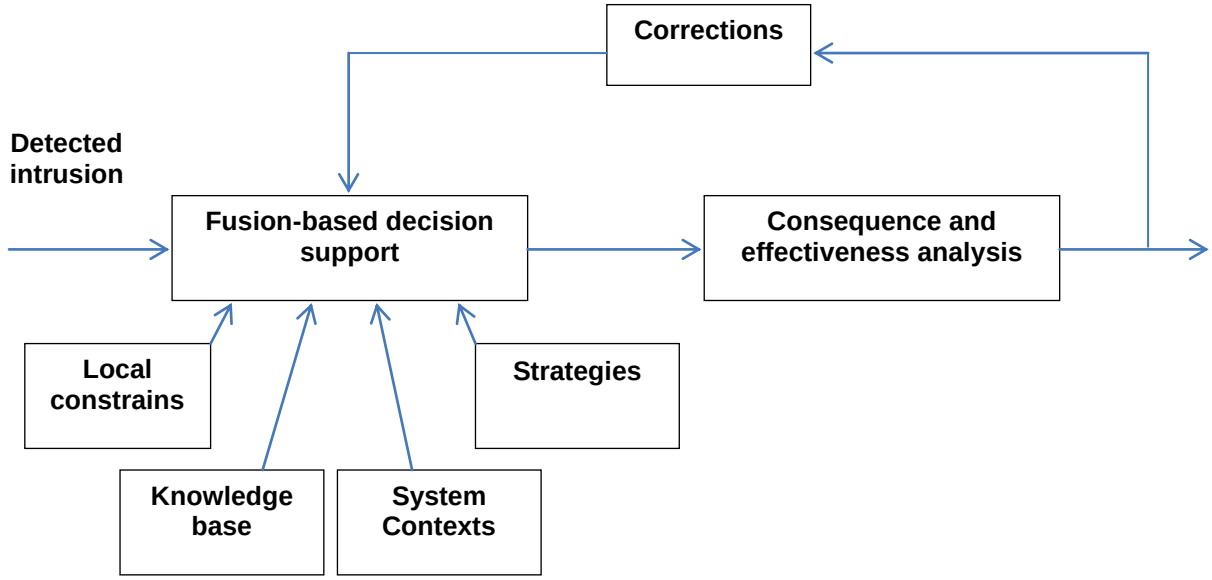

Figure 14: Illustration of algorithm design

4.2 Reputation based system for automatic reaction

4.2.1 Introduction

The current research stream in SCADA systems is to foster the smartness of the field equipment and actuators. This field is governed by policies that while dictating behaviour, depending on the equipment's roles and the context of evolution; also confer to the latter the latitude to automatic reaction based on their own perception of the evolving environment. This component ability is referring to as the component smartness and is strongly determined by, and depending on, the trust perceived by the component of its environment. Actual work related to CI component tends to consider that the latter evolve and are organized in systems. There exist some models for representing how these systems are organized at a high level, models for representing how they are spread in the networks, models to represent how they communicate to each other, and so forth. However, as far as we know, no model exists that integrates all of the above models. Therefore, we do believe that such an integrated model could have many advantages like e.g. to know the impact from the action from one layer to another, to decide which action on a component has the most important impact on a set of other components, to identify the most critical component for an infrastructure, to align the CI system with the corporate objective and to tailor it accordingly, and to support the deployment of automatic reaction strategies. Therefore, we have decided to frame an innovative version of ArchiMate® for the CI system purpose with the objective to enrich the component collaborations and, more particularly, the description of the components behaviour endorsed in the policy artefact and using a reputation based trust model to improve the reliability, termed ARMAN. Our work has been illustrated in the frame of a critical infrastructure in the field of electrical power distribution as considered in CockpitCI.

Enterprise architecture models are frameworks that allow representing the information system (IS) of companies in (or on a set of) schemas called views. Those models have undergone major improvements during the first decade of the 21st century and some significant frameworks have been developed since, such as ArchiMate® [25], the Zachman framework [26], or TOGAF [27]. These models are traditionally structured in layers that correspond to different levels of the organizations' IS. The business layer, for instance, models the concept that exists at the business layer such as the processes, the actors, their business roles, and so forth and which are supported or represented by IT application layers. At this application layer the concepts of the IS that are modelled are the applications, the databases, or for instance, the application data. The advantages of these enterprise architecture models are that they allow improving the connections between the concepts from each layer and, thereby, allow a better integration and an enhanced support for the decision making processes. Up to date, the components are represented at the business layers [28][29][30] have been considered human actors playing business roles. However, rising security requirements for the management of heterogeneous and distributed architecture calls for a rethinking of distribution of the security procedures in both: human and software autonomous entities. Although having been handled by human employees for years, the management of complex systems, nowadays, needs to be shared with intelligent software items, often perceived being more adapted to act in critical situations. Those intelligent software items must behave according to automatic reaction strategy. This statement is enforced by the characteristic ability of the agent to act autonomously in open, distributed and heterogeneous environments, in connection or not with an upper authority. Acknowledging this situation, we are forced to admit that software agents are no longer to be considered only as basic software components deployed to support business activities, but

that they are part of the business actors as well, that they plays some kind of “business role”, and they perform “business tasks” accordingly. Since then, acquiring an innovative enterprise architecture framework to represent the behaviours of such component appears fully justified and required by the practitioners, especially the ones engaged in the management of those critical infrastructures.

In this Section, we propose to explore ArchiMate® and to redraw its structure in order to fit with components’ specificities and domain constraints. The main focus concerns the design and the consideration of the policies that are centric concepts related to the activation of components’ behaviours. The section is structured as follows, after having sighted the related works concerning (meta)models which allow representing the component’s architecture in Section 4.2.2; we review the reputation base trust that is exploited in the modelling of the components smartness and which we consider as a core element for the component behaviour in Section 4.2.3. We model the concept of policy that represents the engine for applying the behaviour rules of the CI component in Section 4.2.4. This framework includes a motivation extension related to the reaction strategy at the 3 abstraction layers (management, application and technical). This extension motivation is explained in the ArchiMate 2.0 specifications and is modelled using the model motivation extension (MME). In Section 4.2.5, we focus on the reputation which play a major role in the decision making process and we explain layer by layer the entire Reputation based System Metamodel and illustrates its different components. In Section 4.2.6, we present a case study that illustrates the exploitation of the enhanced ArchiMate® and we perform real-time simulations in Section 4.2.7. Finally, Section 4.2.8 highlights how the metamodel for CI component may be exploited to support the automatic reaction strategy.

4.2.2 State Of The Art and Motivations

For the modelling of the CI system architecture, we firstly focus on the components which are organized and which collaborate under the form of agents. However, the CI system architecture aims to model not only agent components but all type of components which compose the CI system. By reducing the analysis to the agents system, we aims at facilitating the understanding of the approach by addressing the management of CI. Secondly, the Literatures explain methodologies to model Multi-Agent System (MAS) and their environment as a one layer model and give complete solutions or frameworks. Gaia [28] is a framework for the development of agent architectures based on a lifecycle approach (requirements, analysis, conceptualization and implementation). AUML [31] and MAS-ML [29] are extensions of the UML language for the modeling of MAS but do no longer exist following the release by the OMG of UML 2.0 [32] supporting MAS. Prometheus [30] defines a metamodel of the application layer and allows generating organizational diagrams, roles diagrams, classes’ diagrams, sequences diagrams and so forth. It permits to generate codes but does not provide links between diagrams and therefore makes it difficult to use for alignment purposes or with other languages (e.g. MOF [33], Dsml4mas [34]). CARBA [35] provides a dynamic architecture for MAS similar to the middleware CORBA based on the role played by the agent. Globally, we observe that these solutions aim at modelling the application layer of MAS. CARBA goes one step further introduces the concept of Interface and Service. This approach is closed to the solution based on *ArchiMate*® that we design in our proposal but offers less modelling features.

As we have notice that agent systems are organized in a way close to the enterprises system, our proposal analyses how an enterprise architecture model may be slightly

reworked and adapted for MAS. Therefore, we exploit *ArchiMate*® which has the following advantages to be supported by the Open Group[1]. It has a large community and proposes a uniform structure to model enterprise architecture. Another advantage of *ArchiMate*® is that it uses referenced existing modelling languages like UML. With this aspect we think that it is relevant to provide a lean and simple structure compliant with the new version of UML to model any MAS. As a conclusion of our state of the art, we acknowledge the many other models or frameworks that provide solutions to MAS models and which are compliant or not with other modelling languages. As far as we know no existing approach provides a multiple layer view or an integrated view of these layers.

4.2.3 Metamodel for reputation based trust

The proposed reputation-based trust management scheme is used to predict the future behaviour of a component in order to establish trust among agents and hence to improve security in the system. The goal of using an architecture using a trust Policy within a metamodel core is to improve the agent assignment according to his policy. The trust Policy component depicted in the Figure 15 signifies the lower value that is necessary for agent to be assigned to a role. Moreover according to his role fulfilment, a reputation score is used to assess this level of trust. Indeed we consider reputation as *a measure that is derived from direct and/or indirect knowledge of earlier interactions if any, and is used to access the level of trust an agent puts into another* as in [36]. This trust policy is linked with the behavioural policy. Indeed these Behaviour and Trust Policy are combined into Policy. The rest of the metamodel component is explained in the next section.

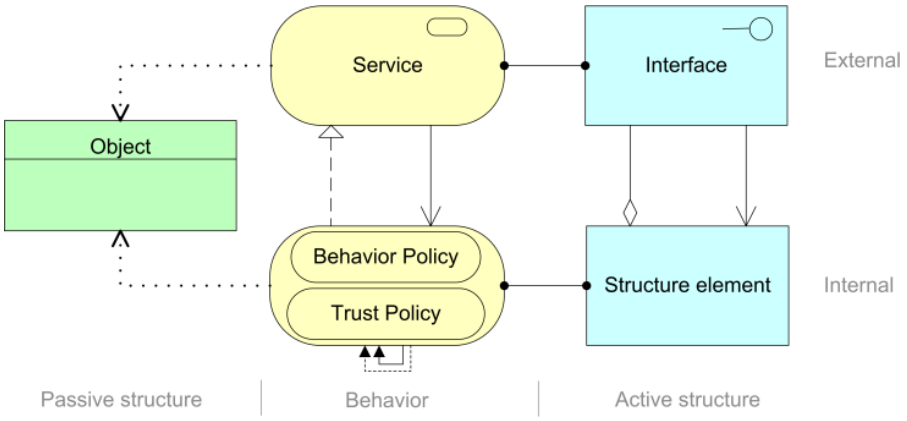

Figure 15: Metamodel Core with Trust Policy

4.2.4 Policy Concept And Metamodel Core

Our goal in modelling the multi-agent system into architecture metamodel is to provide system architects and developers tools to create their own multi-agent system including the notions of *Agents Policy*. As explained earlier, we have selected the *ArchiMate*® language to provide a multiple layered view of multi-agent system using policies.

[1] http://www.opengroup.org/subjectareas/enterprise/archimate. The Open Group is a global consortium that enables the achievement of business objectives through IT standards.

To create this metamodel, we have realized a specialization of the original *ArchiMate®* metamodel for agent architecture. Firstly we have redefined the *Core* of the metamodel (Figure 15) to figure out the concept of *Policy* that hosts the behaviour and the trust decision mechanism. The *Core* represents the handling of *Passive Structure* by *Active Structures* during the realization of *Behaviours*. For the *Active Structures* and the *Behaviour,* the *Core* differentiates external concepts that represent how the architecture is being seen by the external concepts (as a *Service provider* attainable by an *Interface*) and the internal concept which is composed of *Structure Elements* (*Roles*, *Components*) and linked to a *Policy Execution* concept. *Passive Structures* contain *Object* (*Data Object, Organizational Object, Artefacts,…)* that represents information of the architecture.

Secondly, the concept of *Policy* has been defined in accordance with our specialization of the *ArchiMate®* metamodel. The proposed representation is composed of three concepts defining the *Policy* (Figure 16):

Figure 16: Policy concept

1. ***Event***: Events are something done by a *Structure Element* that generates an execution of a *Policy*.
2. ***Context:*** *Context symbolizes a configuration of Passive Structure that allows the Policy to be executed. (e.g. a security level, value for an object, event trigger, …)*
3. ***Rules***: Rules are a set of behaviors to be performed by *Structure Elements.* The behaviors can use *Object* from *Passive Structure* or modify their values.

With these three concepts, we consider *Policy* as an execution of a set of *Rules* in a specific *Context* and in response to an *Event*. Compared to the original *ArchiMate®* language model, we have modified the headline of the *Business Layer* into *Organizational Layer*. Organizational view is more suitable to our vision of an agent and allows the consideration of multi-agent systems as an organization that is relevant to policy rules.

Concepts and colours are from the original *ArchiMate®* metamodel except for *Organizational Function* and the *Application Function* that have been replaced by the *Organizational Policy* concept and the *Application Policy* concept. Behind the *Policy concept,* we try to show that each operation done by the agent can be transposed into a *Policy execution*. Although in *ArchiMate®* there is a semantic difference between the application and the user that exploits the application, in our model and in the agent world, both concepts correspond to a unique one. This is resulting in the fact that the concept of agent is not managed at the organizational layer, thus by human operators. The latter tends to consider the role as a set of entities managed by an existing application. In our proposal, the metamodel, such as for *ArchiMate®* is structured in three layers:

1. The ***Organizational Layer*** that offers products and services to external customers, which are realized in the organization by organizational processes performed by *Organizational Roles* according to *Organizational Policies*.
2. The ***Application Layer*** that supports the organizational layer with *Application Services* which are realized by *Applications* according to *Application Policies*.

3. The ***Technology Layer*** that offers *Infrastructure Services* needed to run applications, realized by computer and communication hardware and system software.

Figure 17 represents the different domains covered by the metamodel in relation with those layers and strategies that influencing the *behaviour*, giving a guideline for the definition of the *Organizational Policies* and *Application Policy*.

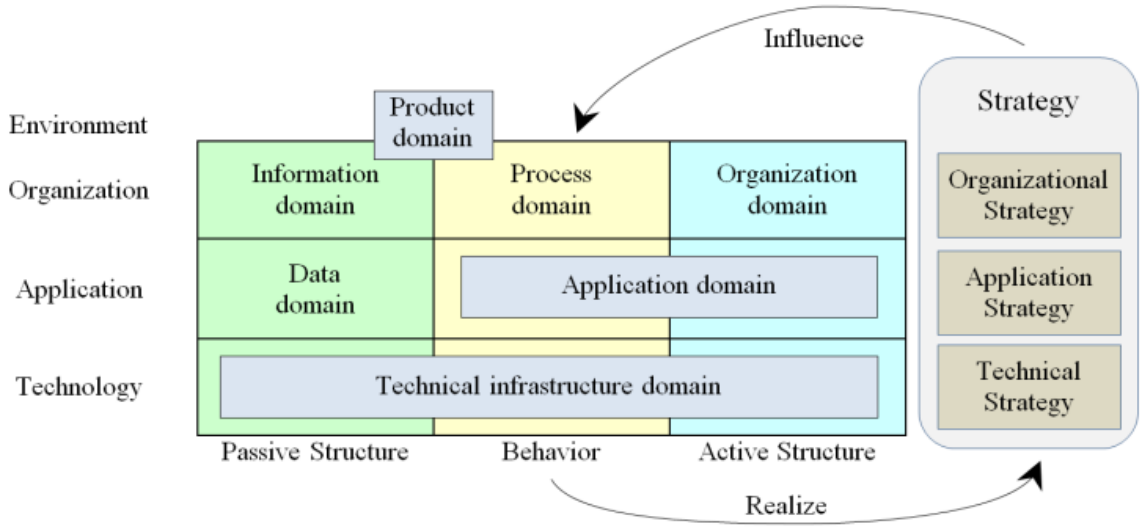

Figure 17: Metamodel structure

Based on this analysis we defined the *Organizational Policy* as:

> *The set of rules that achieves the organizational strategy and that governs the execution, by the Organization domain, of behaviours that serve the Product domain in response to a Process domain occurred in a specific context, symbolized by a configuration of the Information domain*

And we defined the *Application Policy* as:

> *The realisation of behaviour by the Application domain in a configuration of the Data domain necessary to achieve the application strategy.*

4.2.5 Agent System Metamodel

As explained in the previous Section the metamodel is a specialization of the original *ArchiMate*® metamodel. Next paragraphs present the concept from *ArchiMate*® illustrating each layer of the metamodel considering in parallel the concept of *Policy*.

4.2.5.1 Organizational Layer

The *Organizational layer* highlights the organizational processes and the links they have with the *Application layer*. At first the *Organizational layer* is defined by an *Organizational Role (e.g. Alert Detection Agent)*. This role, accessible from the outside an *Organizational Interface*, performs behaviour on the basis of the organization's Policy (*Organizational Policy*) associated with the role. Depending on the role played by the agent, it is possible to interact with other roles in order to perform behaviour; symbolized by the concept of *Role Collaboration*.

Organizational Policies are behavioural components of the organization whose goals are to achieve an *Organizational Service* to a role depending on *Events*. *Organizational Policies* are also influenced by the *Organizational Strategy* in their achievement. *Organizational Services* are contained into *Products* accompanied by *Contracts. Contracts* are formal or informal specifications of the rights and obligations associated with a *Product*. *Values* are defined as an appreciation of a *Service* or a *Product* that the *Organization* attempts to provide or acquire. The *Organizational Objects* define units of information that relate to an aspect of the organization.

4.2.5.2 Application Layer

The *Application layer* is used to represent the *Application Components* and their interactions with the *Application Service* derived from the *Organizational Policy* of the *Organizational layer*. The concept of the components in the metamodel is very similar to the components concept of *UML* and allows representing any part of the program. Components use *Data Object* which is a modelling concept of object and object types of *UML*.

Interconnection between components is modelled by the *Application Interface* for representing the availability of a component to the outside (implementing a part or all of the services defined in the *Application Service*). Concept of *Collaboration* from the *Organizational layer* is present in the *Application layer* as the *Application Collaboration* and can be used to symbolize the cooperation (temporary) between components for the realization of behaviour. *Application Policy* represents the behaviour that is carried out by the components to realize the *Application Strategy*.

4.2.5.3 Technical Layer

Technical layer is used to represent the structural aspect of the system and highlight the links between the *Technical layer* and the *Application layer* and how physical pieces of information called *Artefacts* are produced or used to realize the *Technical Strategy*. The main concept of the *Technical layer* is the *Node* which represents a computational resource which on *Artefacts* can be deployed and executed. The *Node* can be accessed by other *Node* or by components of the *Application layer*.

A *Node* is composed of a *Device* and *System Software*. *Devices* are physical computational resources where *Artefacts* are deployed when the *System Software* represents a software environment for types of components and objects. Communication between the *Nodes* of the *Technology layer* is defined logically by the *Communication Path* and physically by the *Network*.

4.2.5.4 Inter-Layer Link

The complete metamodel (Figure 20) is the union of the three layers. As it is shown new connections between the layers have appeared. For the *Passive structure* (Figure 18) we see that *Artefact* of the *Technical Layer* realizes *Data Object* of the *Application Layer* that realizes *Organizational Object* of the *Organizational layer*.

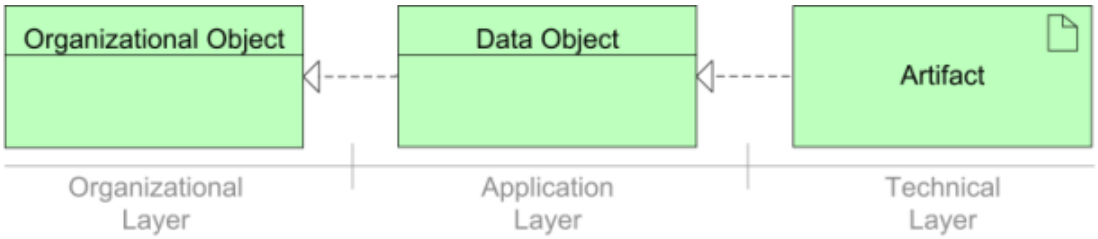

Figure 18: Passive structure connections

Behaviour element (Figure 19) layers show that the *Application Service* uses the *Organizational Policy* to determine the services he proposes. In the same way the *Technical layer* bases his *Infrastructure Service* on the *Application Policy* of the *Application layer*. These concepts collaborate to reach the *Strategy*.

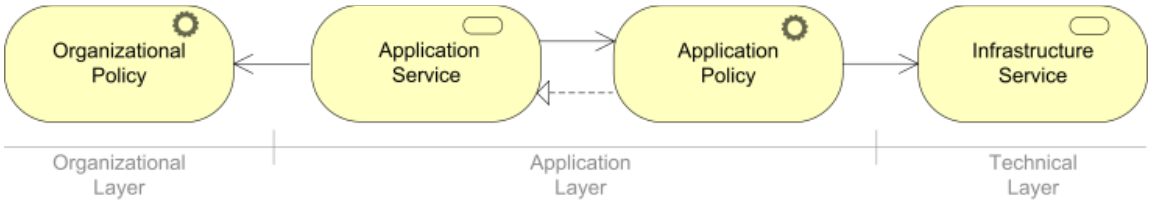

Figure 19: Behaviour element connections

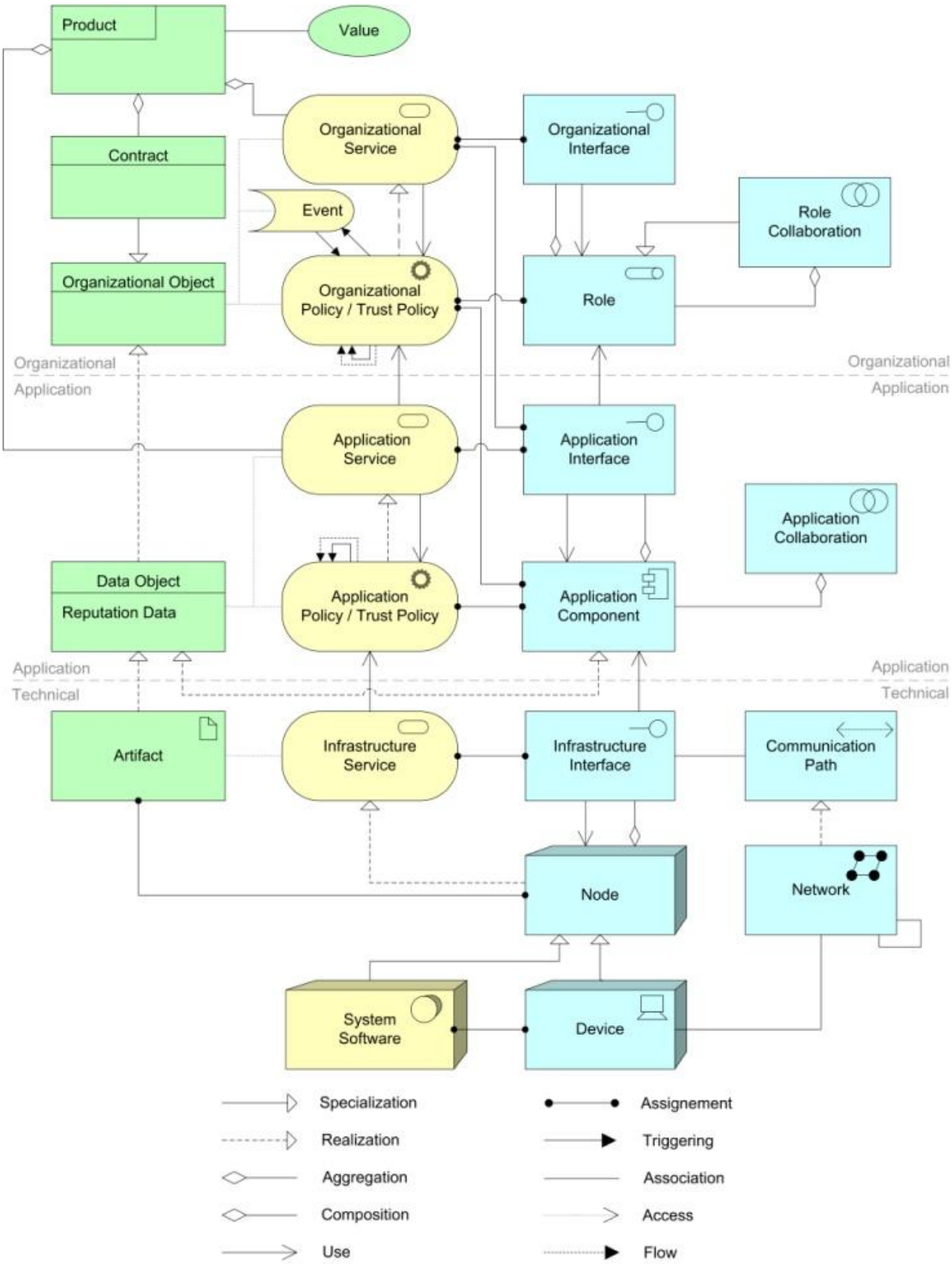

Figure 20: ArchiMate® metamodel for MAS

Concerning the *Active Structure* connection (Figure 21) the *Role* concept determines with the *Application Component* the *Interface* provided in the *Application layer*. Also the *Interface* of the *Technical layer* is based on the components from the *Application layer*.

4.2.5.5 Policy modelling

The organizational and the application policies may, afterwards, be modelled as follows:

4.2.5.5.1 Organizational Policy**.**

In the *Organizational Layer*, *Organizational Policy* can be represented as an *UML Use Case* [32] where concepts of *Roles* represent the *Actors* of the *Use Case* and the *Collaboration* concepts show the connections between them. Concepts of *Products*, *Value* and *Organizational Service* provide the *Goal* of the *Use Case*. *Pre* and *Post conditions* are modelling the context of the *Use Case* and are *symbolized* in the metamodel as the *Event* concept (Precondition) and the *Organizational Object* (Pre/Post condition).

4.2.5.5.2 Application Policy.

Application *Policy* from the *Application Layer* is defined in Section III as the realisation of behaviour by the *Application domain* in a configuration of the *Data domain*. UML provides support to model the behaviour performed by the *Application domain* as *Sequence Diagram*. Configuration of the *Data domain* can be expressed as *Preconditions* of the *Sequence Diagram* and symbolized by the execution of a test-method on the lifeline of the diagram.

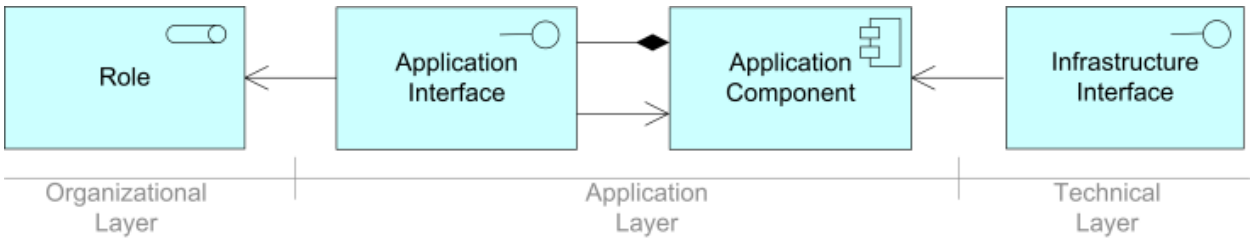

Figure 21: Active structure connections

4.2.5.5.3 Reputation based Trust Policy

The trust and reputation model (TRM) evaluates the policy(ies) that apply to each component involved in the architecture. A large review of computational trust models can be found in [37]. We consider that *reputation is a measure that is derived from direct and/or indirect knowledge of earlier interactions if any, and is used to access the level of trust an agent puts into another*. Each agent uses reputation to derive the trustworthiness that it puts in another based on information provided by probes. Implementation of TRM mechanisms are translated into agent behaviours through the concept of *Policies* called *Trust Policies*. As it will be later illustrated through the broadcasting mechanism (Figure 22), the trust value of each component at an upper level, for instance MSP agents, is derived from sublevels agents. That signifies that, for two given agents A and B, the trust value of agent B computed by agents A is calculated using equation 1, adapted from [36] as such:

$$T_{AB}=OR_{AB}= \gamma DR_{AB}+ (1-\gamma)(\mu_1 IR_{i1B}+ \mu_2 IR_{i2B}+\mu_1 IR_{i3B}) \quad (1)$$

$$DR_{AB}=E(Beta(\alpha,\beta))= \frac{\alpha}{\alpha+\beta} \quad (2)$$

with $\mu_1+\mu_2+\mu_2=1$ and $0<\gamma<1$

DR_{AB} represents the direct reputation of agent B view by agent A and is obtained through direct interactions using the mean of the beta distribution calculated from equation 2 extracted from [36]. IR_{i1B} represents reputation coming from other agent i1 (as well as i2 and i3) and μ_1, μ_2 and μ_3 represent the trustworthiness of the associations between each agent. Applying 1 to the broadcasting mechanism of Figure 22, it gives:

$$T_{MBP\_ACE}=\gamma DR_{MBP\_ACE}+(1-\gamma)(\mu_1 IR_{i1\_ACE}+\mu_2 IR_{i2\_ACE}+\mu_1 IR_{i3\_ACE}) \quad (3)$$

with μ_1, μ_2 and μ3 values calculated based on strategic broadcasting decision e.g. *prioritisation of regional broadcasting* or *technology threat mitigation*.

4.2.6 Case study In Electric Power Distribution Infrastructure

To represent the modelling of MAS with *ArchiMate* for MAS, we complete, in this section, the case study presented in [38]. It is important to know that: *electricity is a good which is difficult to store. Its production has to precisely fit with its consumption. To maintain and guarantee that balance, electric companies supervise the transport of the electricity and manage the electric network. They keep watching in real time both production (wind turbine) and consumption (electric warmer) values to maintain the safety of the system. In case of productivity problem, solutions are deployed like the importation of electricity from adjoining countries or user request, made via TV and newspapers, to adapt the usage of electric machines (e.g. stop washing machine or dryer).*

The broadcasting mechanism (Figure 22) aims at sending alerts to the population using media such as the SMS or tweets whenever a weather alert occurs. This section presents the core components of the broadcasting mechanism. The solution relies on a MAS technology on top of the JADE framework [31]. Agents are disseminated on three layers of the infrastructure corresponding to geographical region (city, region or country) and they retrieve information from probes located in weather station and on the electric networks and representing with different values: pressure, temperature and electric voltage.

The agents that compose the critical architecture are the following: the Alert Correlation Engine (ACE) collects, aggregates and analyses weather information coming from probes deployed over the network and weather stations. Confirmed alerts are sent to the Policy Instantiation Engine (PIE). The PIE receives confirmed alert from the ACE and sets the severity level and the extent of the geographical response. The PIE instantiates high level alert messages to be deployed. Finally the high level alert messages are transferred to the Message Supervising Point (MSP).

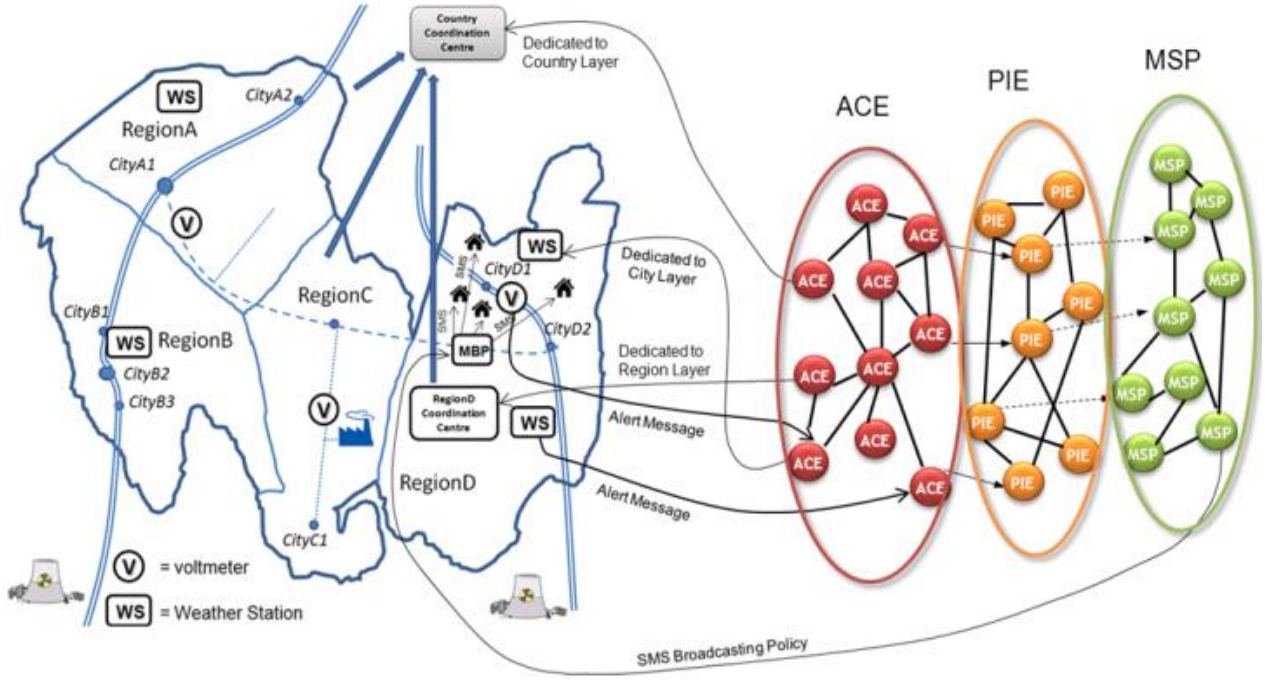

Figure 22: Broadcasting mechanism inside for electric power distribution monitoring

The MSP, as explained in detail in [38] is composed of two modules. The Policy Analysis (PA) is in charge of analysing the policies previously instantiated by the PIE. For that, the Policy Status database stores all communication policies and their current status (in progress, not applicable, by-passed, enforced, removed…) so that the PA module can check the consistency of the newly received message to be deployed. The second module is the Component Configuration Mapper that selects the appropriate communication channel.

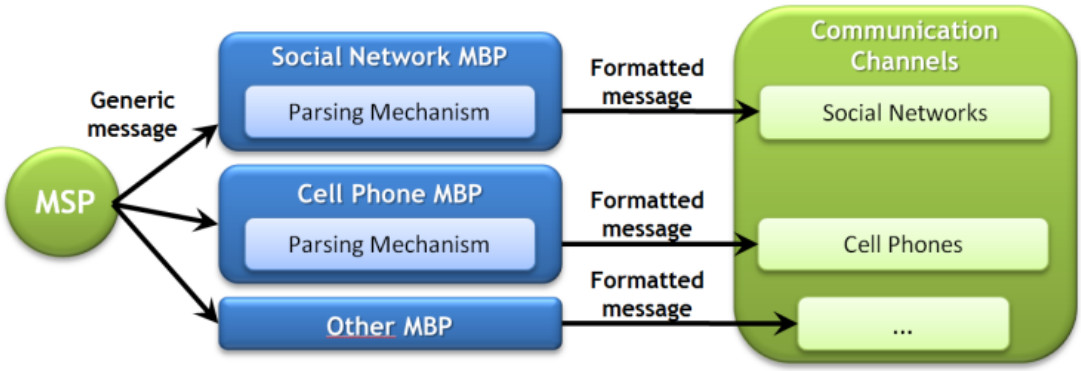

Figure 23: MBP architecture

Figure 23 presents two different kinds of Message Broadcasting Point (MBP). Indeed, another advantage of MAS is that it is very easy to implement from a given model. Concretely it enables us to use different channel of communication (e.g. SMS, e-mail, micro-blogging) to send alerts to citizens, hospitals, etc. By this way our electric blackout prevention system is easily extensible for future communications facilities. MBPs receive generic alert messages from the MSP. Then a specific parser converts the incoming alert message to the appropriate format according to the channel.

To consider the mutual trust between agents, each agent maintains within it a database of levels of trust towards its pairs. This means e.g. that the MBP has a dedicated level of trust for the ACE and the MSP.

The broadcasting alert architecture presented in this section is based on the ReD project [30]. The ReD (Reaction after Detection) project defines and designs a solution to enhance the detection/reaction process and improves the overall resilience of critical infrastructures. Figure 23 introduces the developed architecture illustrated with our weather broadcast alert system. The flow is supposed to begin with an alert detected by a probe.

This alert is send to the ACE agent (City layer) that does or does not confirm the alert to the PIE. Afterwards, the PIE decides to apply new policies or to forward the alert to an ACE from a higher layer (Region Layer). The PIE agent sends the policies to the MSP agent, which decides which MBP is able to transform the high level alert message into an understandable format for the selected communication channel.

In order to manage access rights, we have incorporated to ReD a Context Rights Management module (CRM). Block on the right on Figure 24. The CRM is in charge of providing access rights to agents (E.g. MBP to the probes and Logs File database, MSP to the Policy Rules Status database). The CRM uses the agent links and the crisis context database. The first database includes the link between two agents (type of contextual access right). The second database includes a set of crisis contexts. Thanks to these databases the CRM agent is able to detect the agent right to access each other's at the operational layer depending on the context.

4.2.6.1 ACE Organizational layer

In the *Organizational layer* of the *ACE Agent* (Figure 25) we have represented separately the monitoring aspect from the transaction aspect. We call a transaction a communication of information from one agent to another (e.g. the ACE sends an alert to a PIE) and then we consider the monitoring as the representation of information from an external device. Firstly the *Organizational Role* of the *ACE* is represented as a *Collaboration* of the *PIE Role* and the *Device Role*.

Each *Role* of the *Collaboration* communicates with the *ACE* through a proper *Organizational Interface* one for the monitoring and another one for the transaction. *ACE Role* is providing two *Organizational Services* depending on only one *Organizational Policy* which is dealing with two *Events* respectively for the monitoring and the transaction. Secondly the two *Organizational Services* provided by the *ACE* agent are regrouped into a correlation service symbolized by the *Product* concept. This *Product* has the objective *Value* to reduce a crisis by giving a guaranty of short reaction time represented by the *Contract concept*. Finally the *Contract* is applied on *Organizational Object* as monitoring information and transaction information.

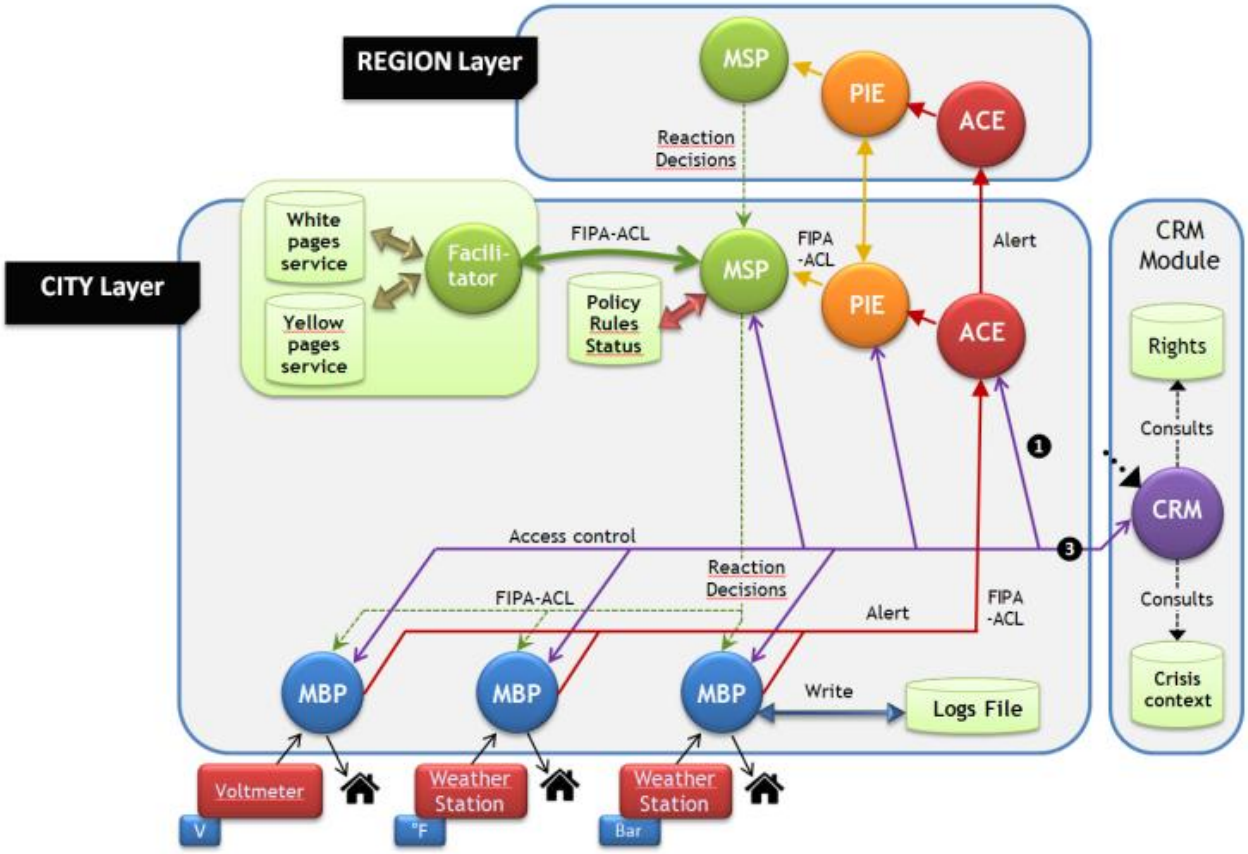

Figure 24: Detailed reaction architecture for electricity distribution adaptation based on weather parameters

4.2.6.2 ACE Application layer

For the *Application layer* of the *ACE Agent* (Figure 25) we consider a separation between the transaction and the monitoring. *Application Services* for transactions and monitoring are, as in the *Organizational Policy*, linked to only one *Application Policy*. To highlight the collaboration between the *ACE* and the *Monitored Device*, we created a *Collaboration* concept named *Monitoring Administration* and shows that this collaboration is constituted of the *Components of the ACE* and the *Components of the Device*. Device's components use the *Application Monitoring Interface* to communicate with the ACE's components and the ACE's components are composed of the *Application Monitoring Interface*.

We use the same approach for the transaction part and rapidly show that the ACE's components are composed of two interfaces deserving the two *Application Services*. Again the *Application layer* contains *Data Object* as *Transaction Messages* and *Monitoring Messages* used by the different *Application Components* of the layer.

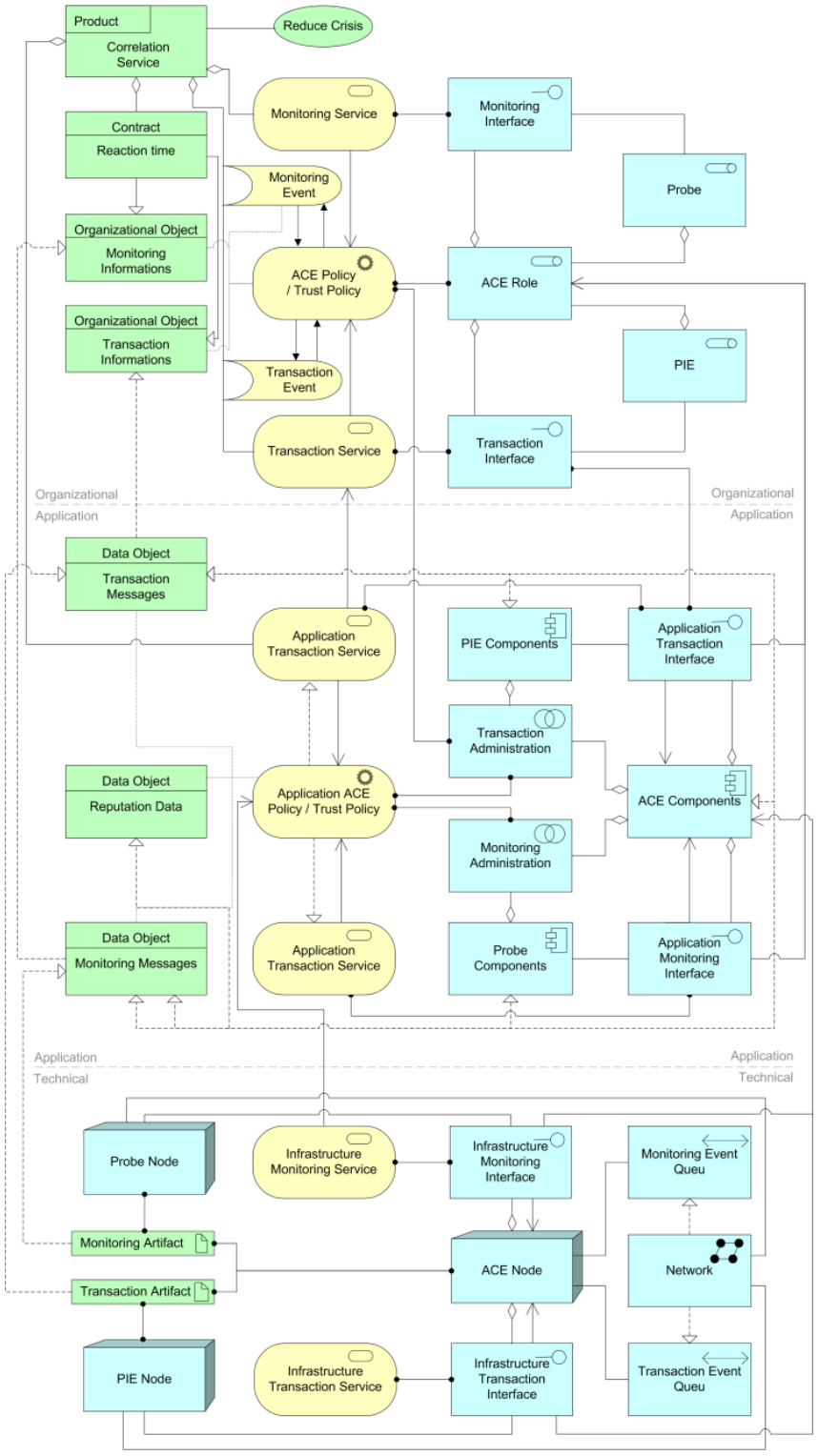

Figure 25: ACE agent model

4.2.6.3 ACE Technical layer

We found in the *Technical layer* of the *ACE Agent* (Figure 25) another representation of the two collaborators of the ACE agent. *Transaction* and *Monitoring Infrastructure* are separated from each other. Both of them have *Infrastructure Service* connected to the ACE agent's *Node* and an *Infrastructure Interface* where the collaborators can interact with it. Each *Node* is respectively connected to a *Communication Path* (represented by a logical *Event Queuing*) and uses different *Artefacts* to communicate. We have intentionally not instantiated *Nodes* for readability but the reader can easily imagine that an ACE agent can be deployed on a computer who's running an operating system. Also the *Network* concept is not defined in our instantiation for the same reason. For example *Monitoring Event Queue* between the ACE agent and the *Device* can be represented as a *Network* concept, as an USB cable and for the *Transaction Event Queue* by an RJ45 cable.

4.2.6.4 ACE Organizational Policy

To illustrate the *Organizational Policies* of the *ACE* we choose to represent the monitoring part of the *ACE Role* as an *UML Use Case* (Figure 26). *Monitoring Events* are illustrated in the *Use Case* as *Extension Points* and show their impacts on the behaviours realized in the *Perform Monitoring Policy*. *Roles* are presented as *Actors* and *Collaborations* are highlighted by the different link between the behaviours.

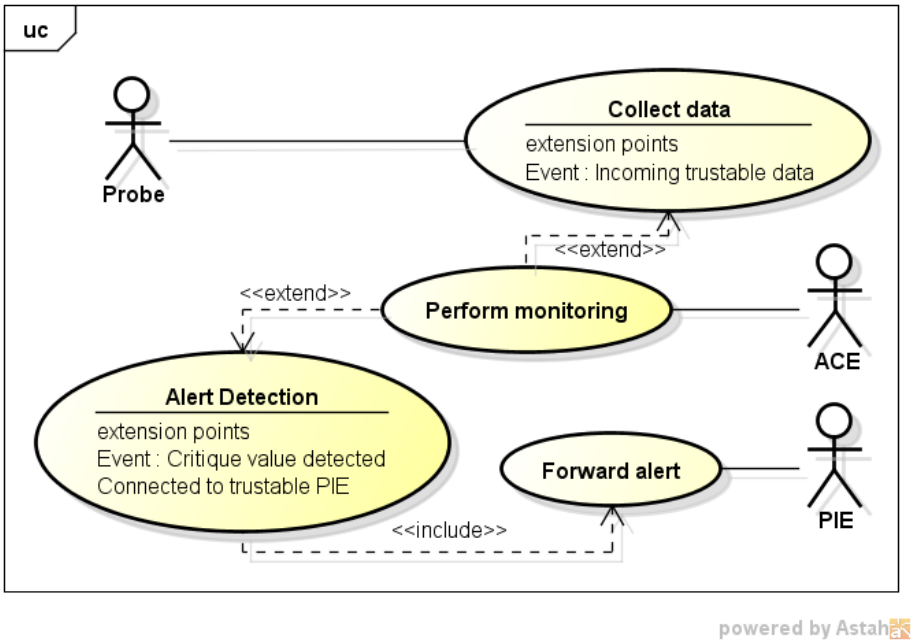

Figure 26: ACE Monitoring Organizational Policies Use Case

4.2.6.5 ACE Application Policy

Sequences Diagrams have been used to represent the behaviours performed by the *Application Domain* of the *ACE Agent* for the *Application Policy*: *Perform Detection* (Figure 27).

In the Sequence Diagram, behaviour of each component is fit to his lifeline and in/out Events presented as inter-component methods call. Context analyse is performed by the component during the execution of his behaviour.

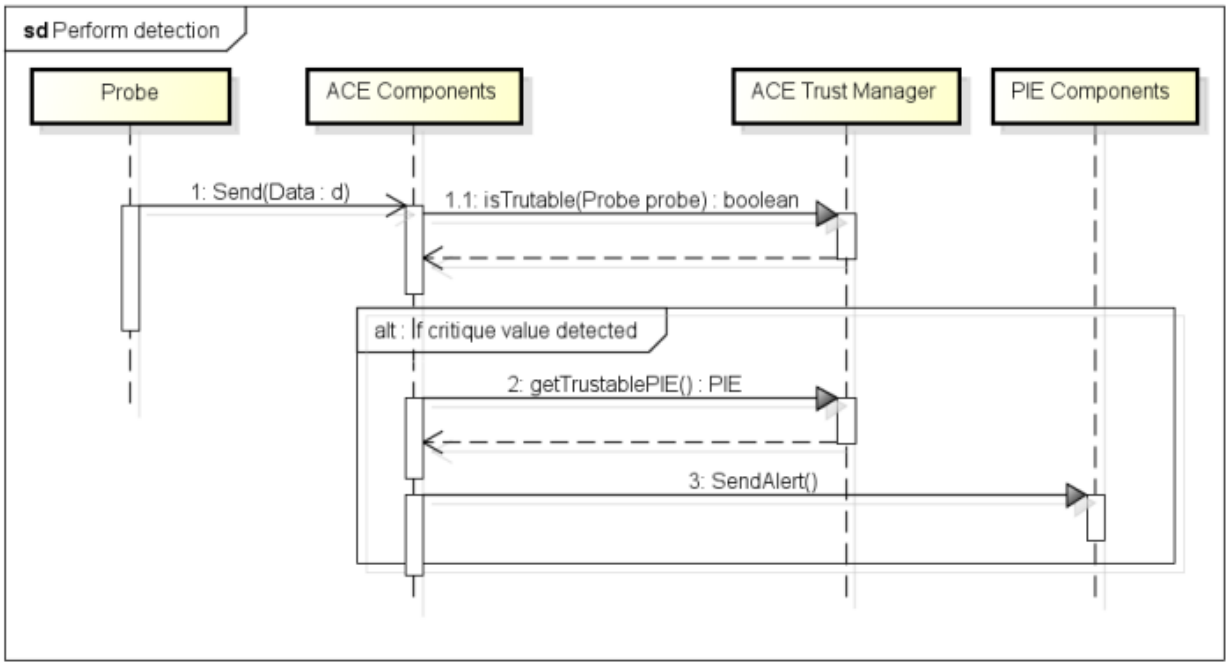

Figure 27: Perform Detection – High level Sequence Diagram

4.2.7 Simulations

In this paragraph we have realized a simulation to a heterogeneous network of ACE and PIE (Figure 28) agents running the reputation model in [38]. Given the complexity and scope of the proposed model, this simulation has voluntary been kept reduced, not to take up too much room for the deployment of the basic components involved in the simulation. The framework used for the test environment has been developed in JAVA and simulate MAS network in a graphical environment. Each created agent is deployed on thread and is only connected to a central supervisor (Composed of an *Agent Manager* and a *Graph Supervisor*) that give him the list of his neighbors depending of his location on the network with a maximum edge size between agents. The protocol used asks ACE agents to send a message containing the collected data from the probe to the nearest PIE every five seconds. Test environment represents a city of 50x50km with a maximum of 5 km connection distance between agents. Also simulations have been running several times during 120 seconds with different load of malicious agents, respectively 10%, 50% and 90%.

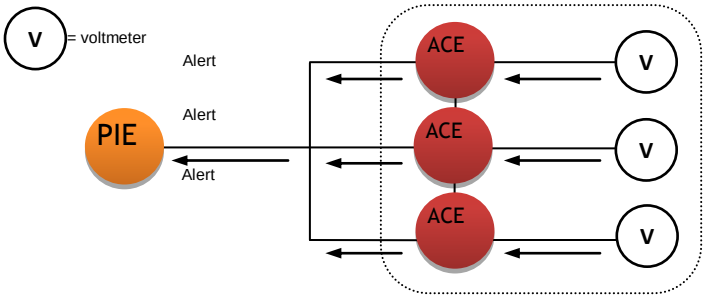

Figure 28: Simulation network

For each load of malicious agents in the network we have collected the trust table - equation 3 - of the same PIE agent, representing his perception of his neighbors ACE (Table 3)

Table 3: PIE perception evolution

Malicious percentage					
10%		**50%**		**90%**	
ACE	*Rep*	*ACE*	*Rep*	*ACE*	*Rep*
A73	0.8	**A73**	0.75	**A73**	0.62
A71	0.86	**A71**	0.87	**A71**	0.81
A80	0.69	**A80**	0.55	**A80**	0.15
A45	0.72	**A45**	0.98	**A45**	0.76
A55	0.91	**A55**	0.93	**A55**	0.9
A56	0.93	**A56**	0.0	**A56**	0.36
A66	0.82	**A66**	0.85	**A66**	0.72
A32	0.8	**A32**	0.81	**A32**	0.44
A35	0.84	**A35**	0.92	**A35**	0.99
A0	0.73	**A0**	0.71	**A0**	0.66

As the percentage of malicious grows, the threshold evolves according to the reputation. For instance, the reputation of PIE “A35” growth from 0.84 to 0.99 as the percentage of malicious PIE grows from 10% to 90%.

4.2.8 Automatic policies paths for reaction strategy

Based on the metamodel of Figure 20, an entire SCADA architecture may be designed. For instance, from the SCADA building blocks defined in the project (Figure 29), each building block may be modelled using the metamodel presented in Section 4.2.3. The model engineered from the building blocks is represented on Figure 30.

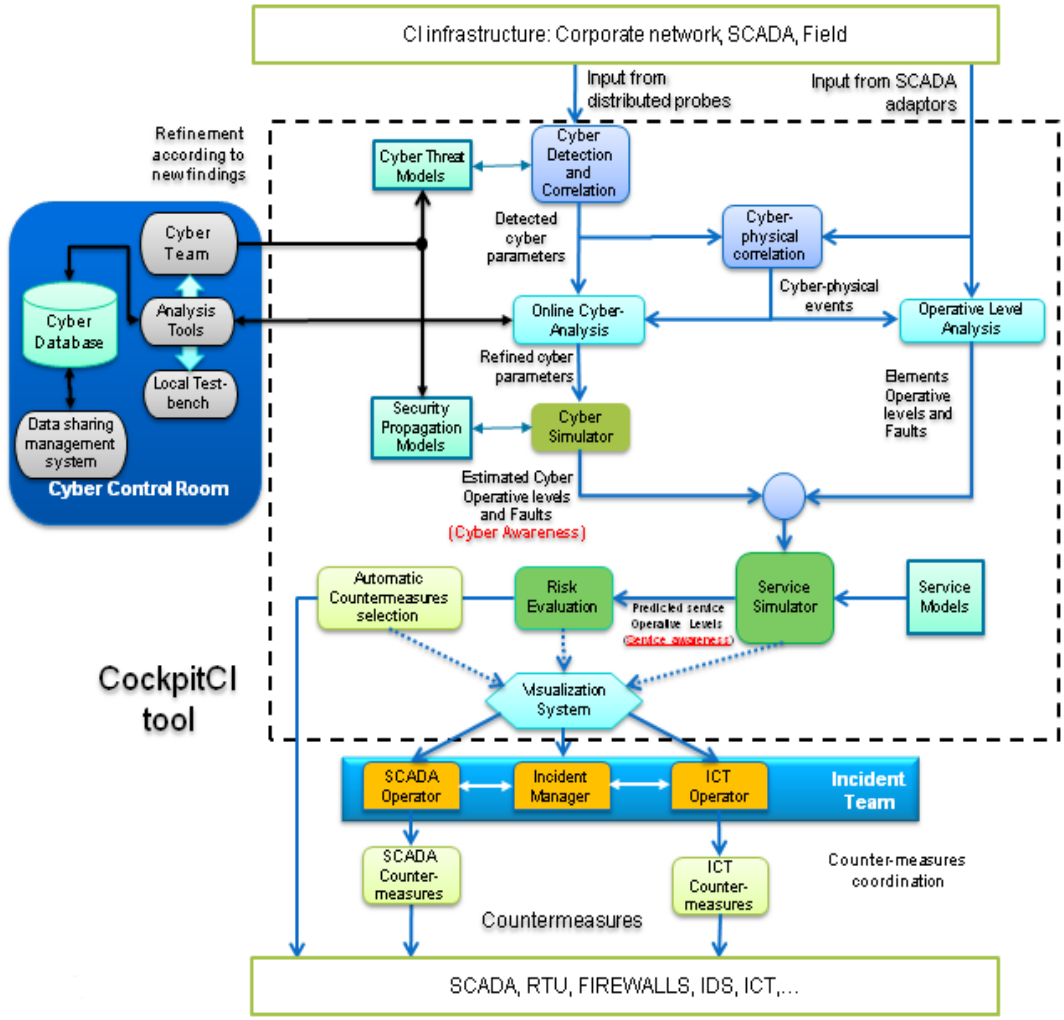

Figure 29: SCADA building blocks

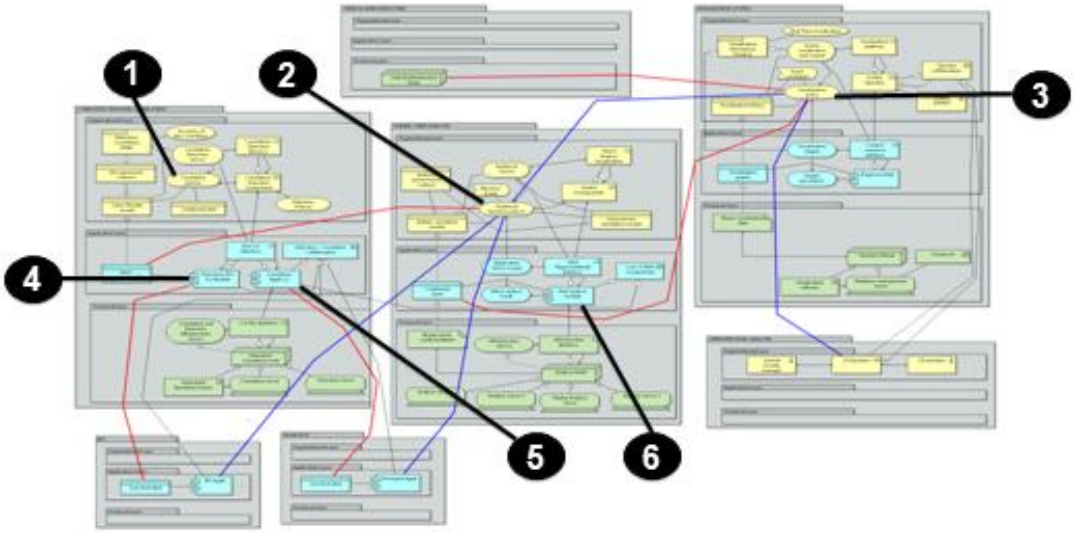

Figure 30: SCADA building blocks model from the meta model

Figure 30 represents each unitary SCADA blocks (unitary SCADA component) of the architecture in grey. These components are modelled with organizational artefact in yellow, application artefact in light blue and technical artefact in green.

4.2.8.1 Policies paths between artefacts

The unitary SCADA component models are used in the second step to picture the global structure of the SCADA architecture and of the connections, in terms of policies, amongst the components of the architecture. This permits to define two types of path which correspond (1) to the rules related to the behaviour of one artefact of the component architecture and enforced by another component (these rules are defined by cognitive paths and correspond to behaviour policy that we name *Cognitive policy*) and (2) to rules related to the acquisition of knowledge from the *Master* to the *Slave* artefact (there rules are defined by permissive paths (knowledge which is allow to know) and correspond to behaviour policy that we name *Permissive policy.*

The following steps are needed to identify the paths:

- *1. Identification of the structure of the CI architecture in terms of unitary modules (components) including their abstraction 3 layers build upon the SCADA metamodel (i.e., organization, application, and technical)*
- *2. Identification of the external parameters of the CI such as potential threat probes and indicators that may impact the CI normal functioning (flood, hijacking,…), the physical environment, the contractual SLA (service level agreement)*
- *3. A. Identification of the Cognitive Policies – artifact of a CI component which needs information from succeeding artifact.*
- *4. B. Identification of the Permissive Policies – artifact of a CI component which needs permission upon the succeeding lower layer artifact.*

4.2.8.2 Automatic Reaction Strategy

We define the Automatic Reaction Strategy (ARS) as "*the rules ($r_{1 \to n}$) uses by the Main CI Investigator which helps to choose between the available reaction policy ($RP_{1 \to m}$) option in accordance with the critical infrastructure Expected Automation Levels (EAL) and considering the RP at the Organization (o) and/or at the Application (a) level.*

This definition means that, in order to support the SCADA for CI manager (the Main CI Investigator), an automatic reaction strategy must be defined and must provide the tool to choose between the possible RP existing amongst the artefact of the CI components. Therefore the RP may be assimilated to the *Cognitive* or *Permissive policy* explained in Section 4.2.8.1.

Acknowledging the semantic of the Automatic Reaction Strategy, we suppose that it may be represented as an entire SCADA component as well. This SCADA component acts afterwards as the main reaction component. Such as all SCADA blocks, it is modeled using the artifact from the three abstraction layers. Figure 31 highlights the 2 upper layers of the ARS.

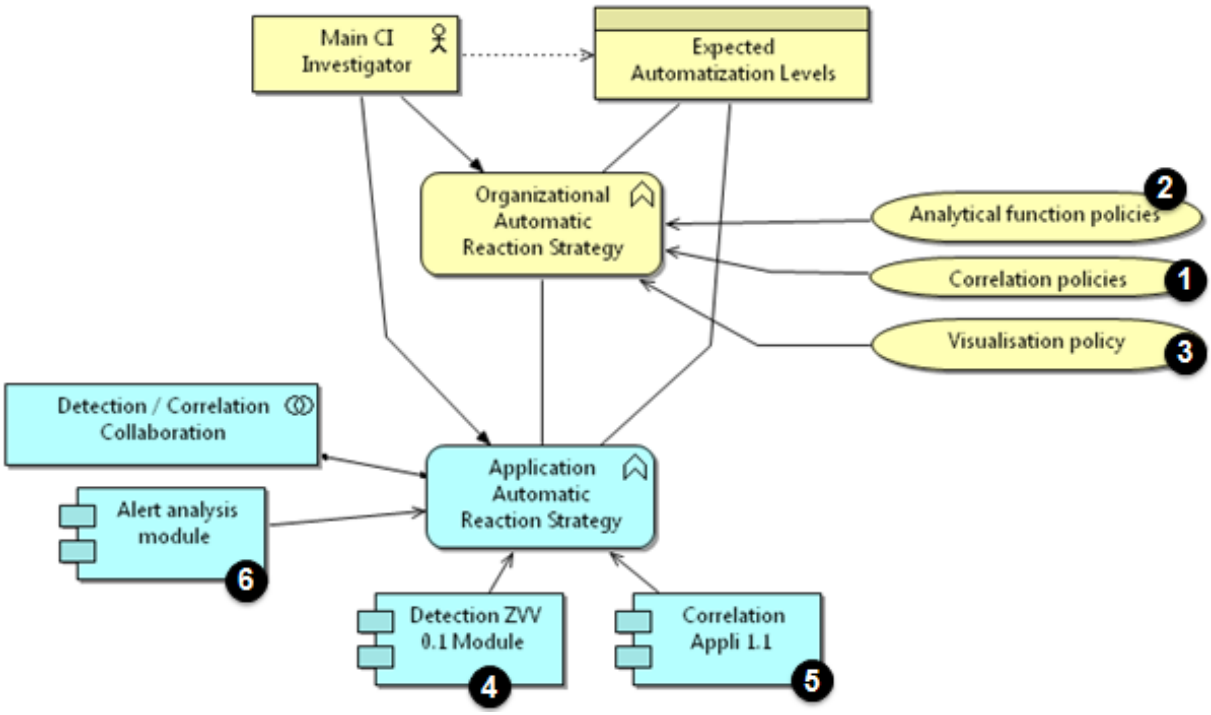

Figure 31: ARS model and connection with the other SCADA artefacts

On this Figure 31, the artifacts from the Cyber Detection and Correlation, the Online Cyber Analyze, the Visualization system, IDS, Honeypots (the two last components being associated to the SCADA architecture through the two first – See Figure 30) are associated to the ARS model following two types of relation as explained in Table 4.

Table 4:Types of relation between artefact from the SCADA building blocks artefact and the ARS.

CONCEPT	Organizational ARS	CONCEPT	Application ARS
SPECIALIZATION	Analytical functional policies	REALIZATION	Alert analysis module
	Correlation policy		Detection ZW 0.1 Module
	Visualisation policy		Correlation Appli 1.1
	…		…

Four of the main ARS model artifacts are represented in Figure 31:

- The Main CI Investigator which is a type of Business actor with access the Expected Automation Levels and which is associated to the Organizational Reaction Strategy
- The Expected Automation levels which encompasses the automation expected according to external constraints, SCADA topology, CI topology, Regulatory framework, Security level (provided through CERN paper for instances), and so forth.

This expected automation levels may thus be associated to different type of application or organization object, but this is not represented on the figure.

- The organizational automatic reaction strategy is the engine that defines, maintain and monitor all the network's *cognitive* and *permissive* policies according to the expected automation levels. To that end, the engine is represented as a function which realizes according to [12] the "*rules* ($r_{1 \to n}$)" from the ARS definition.
- In an equivalent way, the Application Automatic Reaction Strategy is associated to three application artifacts from the application layers of two SCADA components, to know, Cyber Detection and Correlation, the Online Cyber Analyze. This association is dedicated to the enforcement of *permissive* policies to these SCADA applications by means of a dedicated ARS application.

As summarized, we have elaborated an innovative version of ArchiMate® for CI components purpose (that we have firstly considered as MAS of the SCADA architecture) to enrich the component society collaborations and, more particularly, the description of the component behaviour endorsed in the policy component, using a reputation based trust model termed ARMAN and motivated by a strategic reaction extension. To illustrate our work, a case study has been performed in the frame of a critical infrastructure related to electrical power distribution. This case study has allowed illustrating and validating the definition of policies according to automatic reaction strategy on the first hand, and depending on evolving trust parameters amongst agents on the other hand. Finally, we have provided a strongly reduced simulation of a heterogeneous network of ACE and PIE component running the reputation model and where different load of malicious component have been integrated.

We have additionally underlined the possibility to model the strategic automatic reaction by means of an independent SCADA component which acts as an intermediary between the detection and correlation layer , the set of cognitive and permissive policies deployed amongst the SCADA components, and the ARS engine that manage the automatic reaction.

Additional validations are expected in the larger scale infrastructures and in considering the all range of SCADA component. In parallel, a supporting tool is being developed. The upper validation has been allowed by the primary functionalities of it. Additional features of that latter will allow modulating the environment parameters in which the agents' network is running and thereby, it will allow refining and validating the trust based policies evolution along more complex situations. The policies' semantic through a unified component modelling approach with the aim of providing a homogeneous and coherent framework will also be analysed and adapted for the governance of the system by all SCADA and non-SCADA operators.

PART II – Smart RTU policies

5 State of the art on RTU reaction/response strategies

Our aim is to identify policies and strategies that allow a local decision making capability to field components, with particular reference to the RTUs.

5.1 Existing technologies

Before advancing with the study of possible strategies, we want to proceed with the analysis of the contributions made by project partners on technologies already implemented in their infrastructure or still under development, and the material available in the literature.

5.1.1 Implemented technologies

5.1.1.1 Lyse

Lyse has during 2012 implemented a ICS from ABB Ventyx called Network Manager. The following RTU-types are being used:

- RTU560, RTU232 and RTU211 from ABB
- SICAM AK-1703 from Siemens.

Lyse use only private (self-operated) communication lines to their RTUs, mainly on optical fiber and microwave links. Communication lines between the core ICS network and external networks (corporate LAN and remote RTUs) are protected by Cisco ASA 5550 firewalls and HP TippingPoint intrusion prevention system (IPS).

It is likely that Lyse in the near future will install RTUs at small hydro-, solar- and wind-power plants and charging stations for electrical vehicles. These RTUs must be “smart” in such a way that they can be programmed to perform local switching and regulating actions automatically in the grid and communicate with other RTUs to avoid “counter regulating” actions. The introduction of local “intelligence” in the RTUs will, of cause, require increased security and reliability regarding communication lines and RTU operating environment.

5.1.2 Current literature

We did not find anything in the literature that dealt with SMART RTU from the point of view of information security. The intelligence of the RTU is usually related to a more efficient energy consumption and to optimize the control operations. In this sense, they are often referred to as SMART-METERS.

6 Designing a Smart RTU

From the point of view of computer security, since the SMART RTUs have not yet been treated in the literature and none of the project partners has implemented or studied anything about it, it was considered appropriate to start from scratch with a very structured approach.

First, we define the concept of Smart Control and how this applies to the Smart Industrial Control Systems, hereinafter we will analyse in detail the basic idea of the Smart RTU and the advantages in terms of safety in the event of its real implementation.

6.1 Smart Control

We define the **Smart Control** as:

A multi-level approach that adapts behaviour strategies defined at the global level (e.g. for a SCADA system) to the local "smart" components of the control system (e.g. RTU / PLC, RTU clusters), allowing each component to make decisions independently.

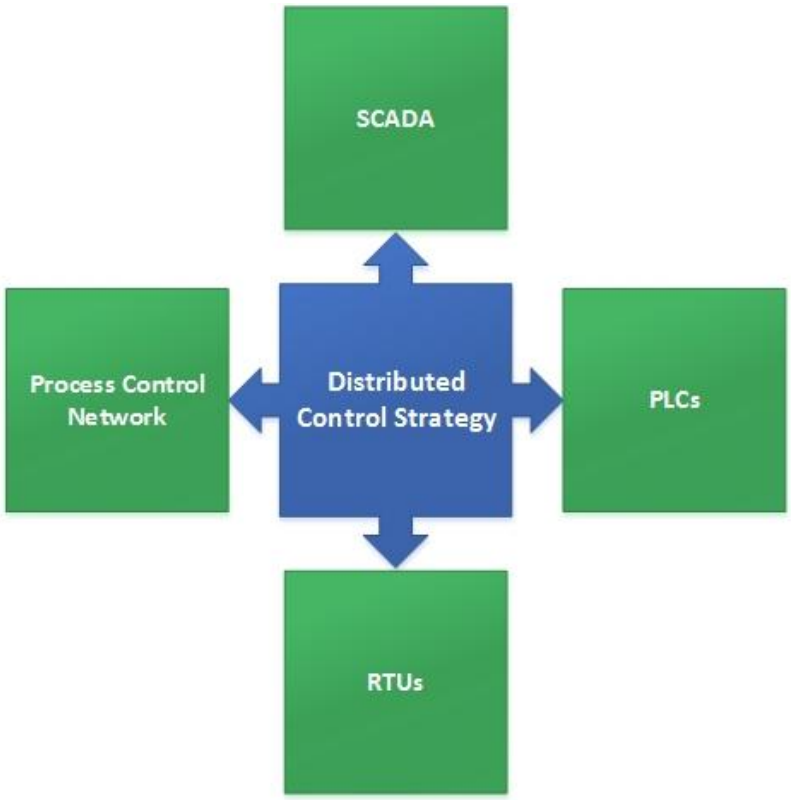

Figure 32: Smart Control

In accordance to this definition the logic behind the Smart Control is shown in Figure 32. In the design phase it will be decided which parts of the control system can be considered

"smart" and configured in such a way as to reflect local control policies defined at the global level. Every single "smart" component will thus contribute to strengthening overall security.

6.2 Smart Industrial Control Systems (ICS)

The industrial control systems are used by most of the critical infrastructures of a country and, despite the latter are 90% owned by private individuals, the general architecture does not vary so much.

As shown in Figure 33 a conventional Industrial Control System (ICS) consists of the following subsystems [39]:

- Control subsystems:
    - Control Server
    - SCADA unit
    - Human-Machine Interface (HMI)
    - Intelligent Electronic Devices (IED)
    - Data Historian
    - RTUs
    - PLCs
    - Input/Output (IO) Server.
- Network subsystems:
    - Fieldbus Network
    - Control Network
    - Communications Routers
    - Firewall
    - Modems
    - Remote Access Points

The key components of a standard ICS include the following:

- Control Loop
- Human-Machine Interface (HMI).
- Remote Diagnostics and Maintenance Utilities

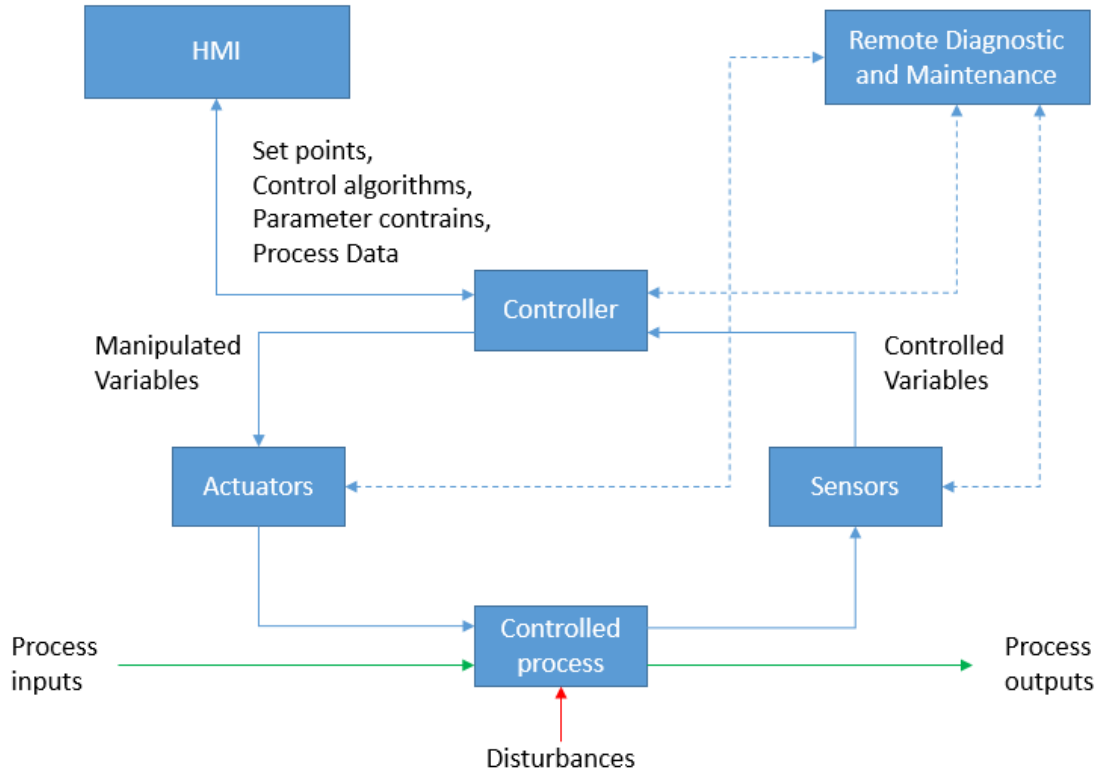

Figure 33: Standard ICS

A Smart ICS should be capable of:

1. Coordinate Control Strategy determined at design both globally and locally
2. Allow an automatic response of the system without going through the operator
3. Allow local decision making to the field components
4. Allow local information sharing between field components

The goals of a Smart ICS are:

1. Enhance the resilience of the system
2. Increase the global awareness
3. Increase the local awareness
4. Create a Distributed awareness

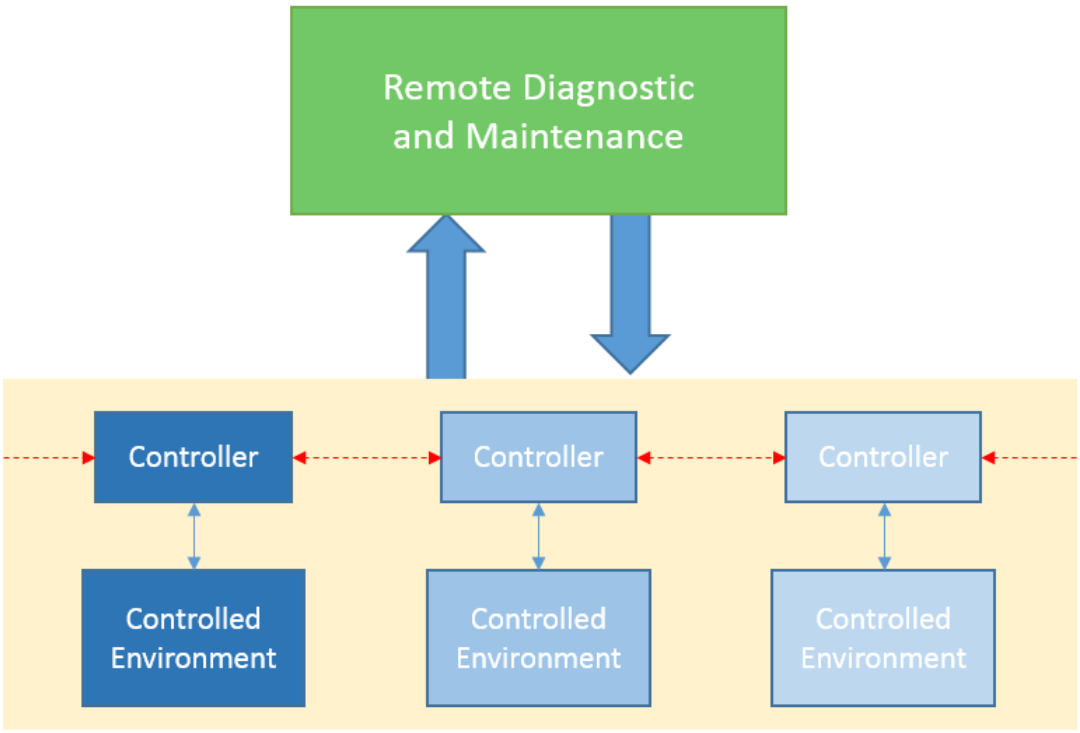

Figure 34: Smart ICS

In Figure 34 the Remote Diagnostic and Maintenance block is connected to a set of control cycles that simultaneously exchange information between them. Each control cycle then receives information from both components with a high processing capacity and a more global view, and local components, equipped with a limited capacity for information processing. This configuration increases local awareness also allowing the local components to take decisions independently, if necessary.

6.3 Smart SCADA

We define a **Smart SCADA** as:

a SCADA system (Supervisory Control and Data Acquisition), which has advanced features compared to conventional SCADA systems, on all levels.

These additional features include:

- Process optimization
- Monitor and manage information on all levels
- Identify the optimal response strategies in case of attack or contingency
- Perform (or suggest to the operator) automatic reactions at global level
- Coordinate automatic reactions at local level

It is therefore necessary to have local components that can manage, interpret and process information by organizing themselves in accordance with the pre-set response policies and / or if they cannot communicate with the control room.

6.4 Smart RTU

First we present the standard definitions for RTU and PLC [39]:

Remote Terminal Unit (RTU). The RTU, also called a remote telemetry unit, is a special purpose data acquisition and control unit designed to support SCADA remote stations.

Programmable Logic Controller (PLC). The PLC is a small industrial computer originally designed to perform the logic functions executed by electrical hardware (relays, switches, and mechanical timer/counters

In this study we will refer to the PLC as an RTU, as they often cover the same role.

6.4.1 Logical errors

In addition to recording and communicating the normal operating parameters, currently a generic RTU can detect physical errors (fire, flood, tampering, etc…). We want to add a new class of logic errors (contradictory instructions, dangerous or out of the normal operating cycle, etc…) processed directly by the RTU, as shown in Figure 35.

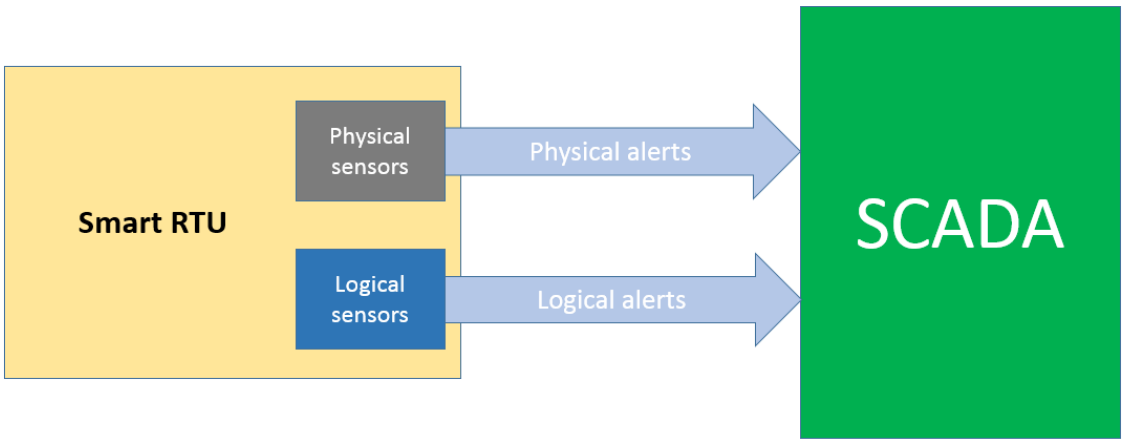

Figure 35: Smart RTU alerts flow

The logical errors will be processed by a pre-determined set of rules and if necessary the Smart RTU will alert the SCADA system about a potential threat.

We also want the Smart RTU to be able to recognize and manage potentially dangerous situations independently from the control room.

Depending on the type of attack will also change response policies. Smart RTUs will then be able to decide and implement the best strategy in each case.

A Smart RTU will be able to:

1. Recognize anomalous or dangerous situations
2. Change its settings in accordance with a pre-set strategy
3. Alert the SCADA system and / or the other closest Smart RTUs
4. Cooperate with other Smart RTUs (Distributed awareness)

6.4.2 Hybrid rule based approach for Smart RTU

A RTU, during its normal operating cycle, continuously receives and sends informations to and from the control room. This informations can be both routine communications and important control instructions. When an RTU receives an instructions from the control room it executes them, without knowing if those instructions are lawful or not and what might be the consequences.

To give the RTU the opportunity to recognize and manage anomalous or dangerous instructions we use a rule based approach along with a RYG (Red, Yellow, Green) alert scale.

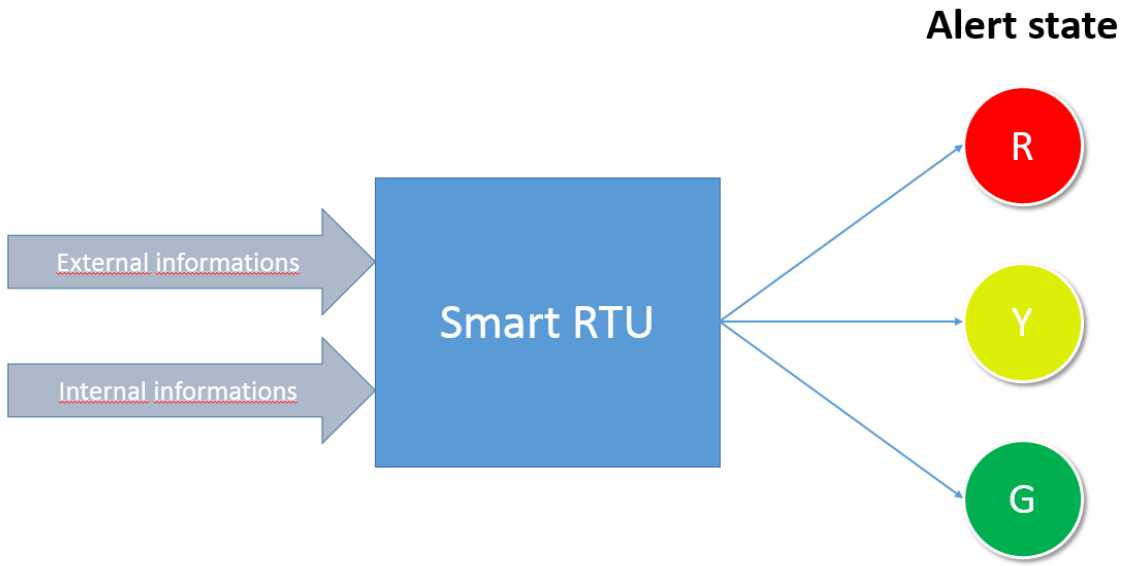

Figure 36: Alert states

The RTU level of alert will be set to GREEN during its normal operating cycle. In the case faults are recognized, the alert level is raised to YELLOW and if the danger is established or anomalies continue to repeat themselves, the alert level becomes RED.

In Figure 36 the RTU processes the information from the control room and other RTUs through a set of rules and decide the alert level appropriate to the situation. In exceptional cases may be the control room to want to force a state of alert on the RTU.

At each level of alert will be associated with a different configuration. The choice of a configuration rather than another will depend on various factors, specific to each RTU:

1. The risk associated with the malfunction or improper use of the RTU
2. The importance of the RTU within the process control
3. The risk of propagation of the anomaly from the RTU to the adjacent ones.

For example, an RTU whose task is to collect and send information from field sensors hardly possess the same configuration in a RED state of alert of an RTU that controls the opening and closing of a relief valve in a nuclear power plant.

Examples of configurations for different alert levels could be:

GREEN:

- Parameters in Standard Mode - Accept all incoming instructions - Communications enabled

YELLOW:

- Parameters in Safe Mode - Accept only certain instructions - Communications enabled

RED:

- Parameters in Lock Mode - Do not accept instructions - Communications filtered

It is obvious that a configuration that is blocking communication and puts the RTU in a Lock Mode can create problems during the control operations, especially in critical situations. The Lock Mode of a RTU should last the minimum necessary and be decided at the design stage of the control system.

In Figure 37 we show an example of the rule based approach in which the Smart RTU is designed to check the number of times it receives the same instruction in a short amount of time. The instruction may be the closure of an electric switch or the opening of a watertight bulkhead. The RTU starts its control loop in the Green alert state. As soon as the counter variable `count` reaches the threshold of 3 times in a minute, the alert state is raised to Yellow. The configuration of the RTU changes in accordance with the Safe Mode. If the RTU continues to receive the same instruction in the next few minutes the alert state will be set to Red. Communication is filtered and the RTU won't accept further instructions for a predetermined period of time.

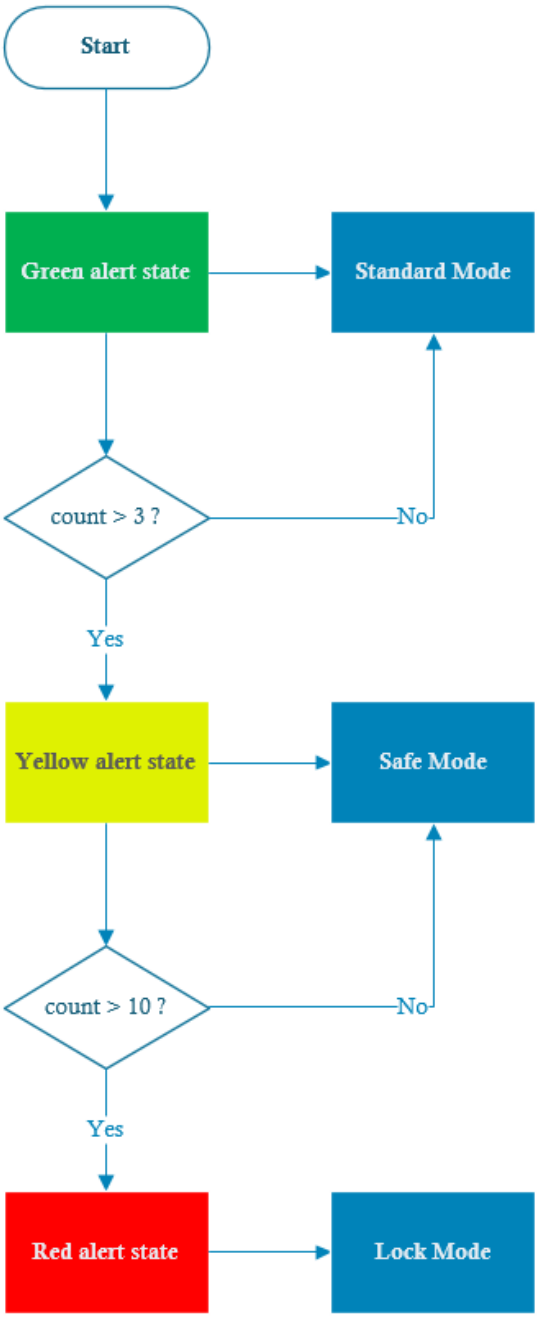

Figure 37: Rule based approach for Smart RTU

Same as the configurations, the rules are decided during the design process and must take into account the specifications of the RTU in which they're implemented.

Each SMART RTU will have a different set of rules that will be active at the same time and will respond to different activation parameters. Some rules may be sensitive to variables

such as the number of TCP/IP packets received within a predetermined period of time, others will react to dangerous instructions, others will send an error message when being asked to perform a given operation outside the usual time window .

6.4.3 Basic SMART RTU structure and definition

A standard RTU/PLC has a modular design, and usually consists of a Power Supply, a CPU, and an Input/output card.

To facilitate the initial implementation of the Smart RTU we decided to bypass the incoming and outgoing communications by inserting a Smart Extension.

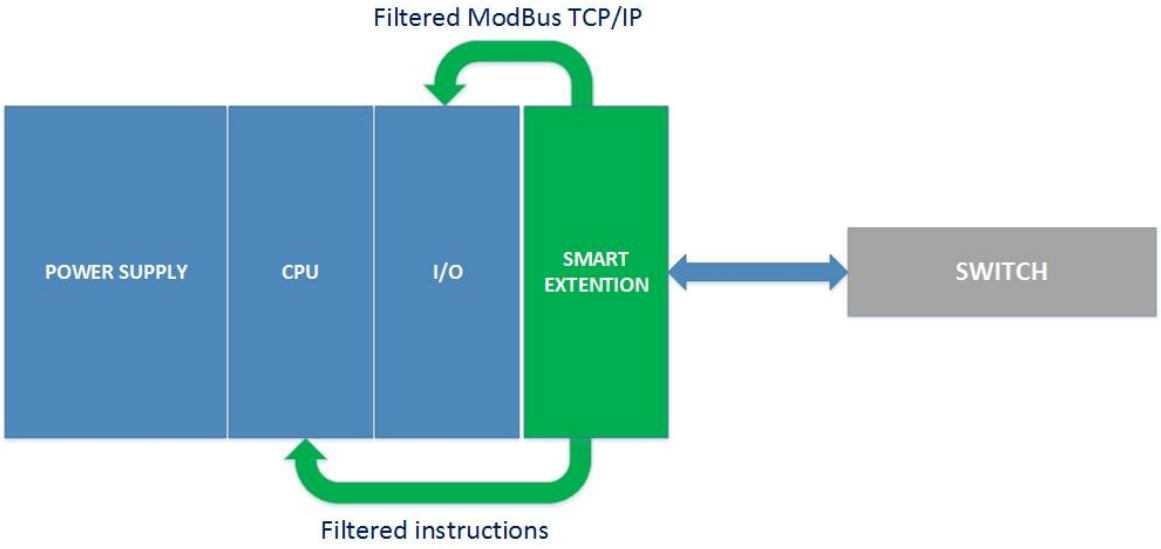

Figure 38: Smart RTU structure

The Figure 38 shows how this extension filters both the raw ModBus TCP / IP packets (acting as a standard firewall) and the instructions received, making it invisible to the RTU itself and the rest of the control network.

The Smart Extension is a network card with additional processing capabilities. Open source IDS software such as SNORT can be installed and modified, to allow it to work at the application level.

Returning to what was said in section 6.2, with this modification a RTU may be considered "smart" and in this way communicate with the others, going to form one SmartCluster, as shown in Figure 39.

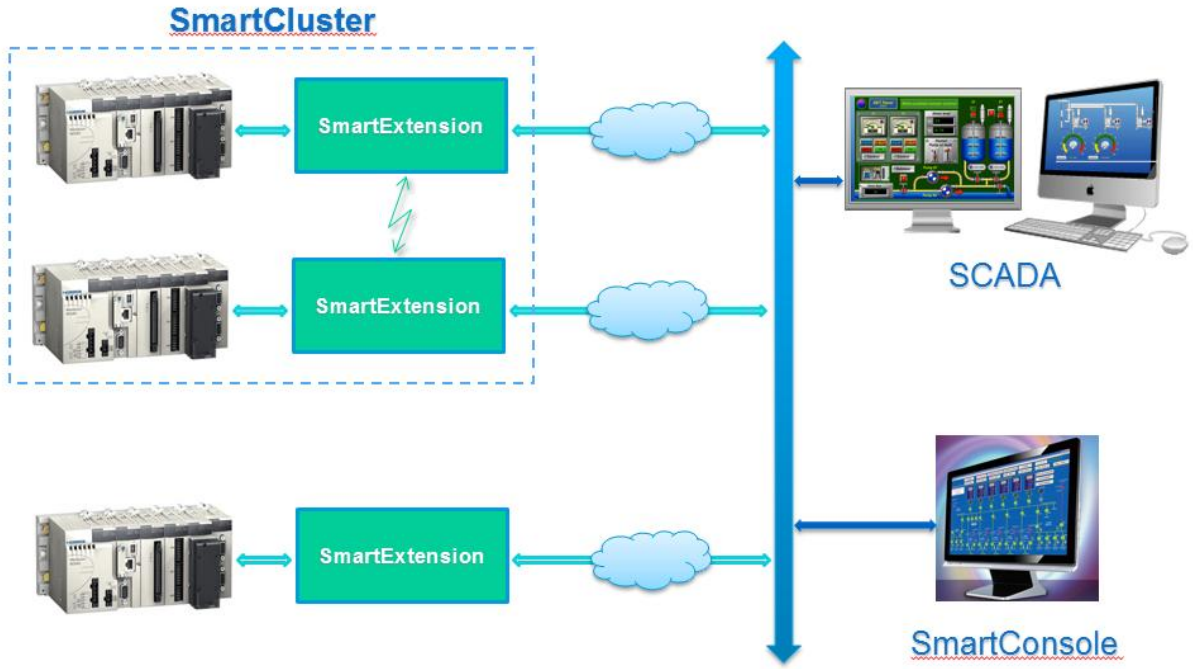

Figure 39: Smart Ecosystem

We can finally give a definition of Smart RTU according to what has been said so far:

A Smart RTU is a modified RTU able to recognize logical errors and/or anomalous instructions, automatically respond to attacks, communicate with the others Smart RTU and take decision locally, without support of the control system.

7 Conclusion

In the event that susceptible behaviour is detected, it is essential to take necessary actions to prevent attacks and ensure safety of the targeted resources. Such actions are known as intrusion response. In designing intrusion prevention systems the intrusion response module receives much less attention due to the inherent complexity in designing and deploying response in an automated manner. Part I of this deliverable addresses solutions towards this problem. The report first provides fundamental concepts on different intrusion reaction approaches (notification systems, manual and active response systems, and passive and active systems) and discusses advantages and disadvantages of those methodologies. Then state-of-the-art automatic intrusion reaction strategies are discussed. Here a discussion on several machine learning/modelling and signal (information) processing based reaction strategies are presented. Finally, detailed discussions on the proposed intrusion response strategies, (i) a combined rule and fusion based reaction strategy and (ii) a reputation based system for automatic reaction is presented. Part II of the document discusses the Smart RTU policies. Since it is a topic that has not been treated nor in literature, nor by our partners, it was decided to proceed with a structured approach. First the concept of Smart is defined, and how it's closely related to information security. Then this concept is applied to the control systems. In a Smart control system there are components able to take decisions independently on the basis of information from both the central control system, either from other local components. One of these components is the Smart RTU, capable of managing faults and attacks independently, at a local level. In the design phase a hybrid rules-based approach has been chosen together with a RYG alert scale. A set of rules is inserted into a Smart Extension within the RTU allowing it to process the information received and to change its configuration according to the situation. Finally, a definition of Smart RTU is presented. Combining work carried out during this task and other related tasks of the project, CockpitCI will be able to contribute to a safer living environment for people especially by providing smart detection tools, early alerting systems and strategic security system. The distributed framework of the system will ensure an operational deployment of the security all over Europe and will improve the European Critical Information Infrastructure Protection (CIIP) strategy.

8 References

[1] Tipping Point intrusion prevention systems. Available from ”http://www.tippingpoint.com”. Accessed 20 February 2006

[2] Natalia Stakhanova, Samik Basu, Johnny Wong, “A taxonomy of intrusion response systems”, International Journal of Information and Computer Security, Vol-1, issue-1 pp 169-184, 2007.

[3] Jahnke, M., Thul, C., Martini, P: “Graph based metrics for intrusion response measures in computer networks”, In: Proceedings of the 32nd IEEE Conference on Local Computer Networks, LCN 2007, pp. 1035–1042. IEEE Computer Society, Washington, DC (2007)

[4] The Snort Inline Project Team. Snort Inline Project Homepage. Online accessible at http://snort-inline.sourceforge.net/, 2007

[5] T. Toth and C. Kruegel. Evaluating the impact of automated intrusion response mechanisms. In ACSAC ’02: Proceedings of the 18th Annual Computer Security Applications Conference, 2002.

[6] C. Carver, J. M. Hill, and J. R. Surdu. A methodology for using intelligent agents to provide automated intrusion response. In Proceedings of the IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop, West Point, NY, June 6-7, 2000, pages 110–116, 2000.

[7] D. Ragsdale, C. Carver, J. Humphries, and U. Pooch. Adaptation techniques for intrusion detection and intrusion response system. In Proceedings of the IEEE International conference on Systems, Man, and Cybernetics at Nashville, Tennessee, pages 2344–2349, 2000.

[8] S. Axelsson. Intrusion detection systems: A survey and taxonomy. Technical Report 99-15, Chalmers Univ., March 2000.

[9] Carver, A. C. 2002. Intrusion Response Systems: A Survey. Department of Computer Science, Texas A&M University. Collage Station, USA

[10] E. A. Fisch, “Intrusion Damage Control and Assessment: A Taxonomy and Implementation of Automated Responses to Intrusive Behavior,” Ph.D. Dissertation, Texas A&M University, College Station, TX, 1996.

[11] A. W. Krings and A. Azadmanesh, “A graph based model for survivability applications”, European Journal of Operational Research, vol. 164, issue 3, pages 680-689. 2005.

[12] T.J. Dasey and J.J. Braun, Information Fusion and Response Guidance, Lincoln Laboratory Journal, Vol.17, No.1, 2007.

[13] Sadoddin R., Ghorbani A. (2006) Alert correlation survey: framework and techniques, In proceedings of the 2006 International Conference on Privacy, Security and Trust, ACM New York.

[14] Porras P. A., Fong M.W, Valdes A. (2002): A Mission-Impact-Based Approach to INFOSEC Alarm Correlation. RAID 2002: 95-114.

[15] Alsubhi K., Al-Shaer E., Boutaba R. (2008) Alert Prioritization in Intrusion Detection Systems, In proceeding of NOMS 2008, pp 33-40

[16] Beth T., Borcherding M., Klein B. (1994): Valuation of Trust in Open Networks, In proceedings of ESORICS 1994, pp.3-18.

[17] Ouedraogo M., Khadraoui D., Mouratidis H., Dubois E. (2012): Appraisal and reporting of security assurance at operational systems level, Journal of Systems and Software 85(1), pp.193-208, Elsevier.

[18] N. B. Anuar, M. Papadaki, S. Furnell, N. Clarke, A Response Strategy Model for Intrusion Response Systems, Information Security and Privacy Research, IFIP Advances in Information and Communication Technology Vol. 376, 2012, pp 573-578.

[19] B. Schneier. Secrets & Lies: Digital Security in a Networked World. John Wiley & Sons, 2000.

[20] S. A. Zonouz, H. Khurana, W. Sanders, and T. Yardley. RRE: A game-theoretic intrusion response and recovery engine. In DSN, pages 439 –448, 2009.

[21] S. Tanachaiwiwat, K. Hwang and Y. Chen, Adaptive Intrusion Response to Minimize Risk over Multiple Network Attacks (ACM Trans on Information and System Security, 2002).

[22] Schnackenberg D., Djahandari K. and Sterne D. 'Infrastructure for intrusion detection and response', DARPA Information Survivability Conference and Exposition, 2000. DISCEX '00. Proceedings (Volume:2)

[23] Nor Badrul Anuar, Maria Papadaki et al, 'A response strategy model for intrusion response systems', SEC 2012, IFIP AICT376, pp573-578, 2012;

[24] Natalia Stakhanova, Samik Basu, Johnny Wong, 'A taxonomy of intrusion response systems', International Journal of Information and Computer Security, Vol 1, pp169-184, 2007

[25] G. Neumann and M. Strembeck. A scenario-driven role engineering process for functional rbac roles. In SACMAT '02, New York, NY, USA, 2002. ACM.

[26] M. Lankhorst. ArchiMate language primer, 2004

[27] J. A. Zachman. 2003. The Zachman Framework For Enterprise Architecture : Primer for Enterprise Engineering and Manufacturing By. Engineering, no. July: 1-11.

[28] F. Zambonelli, N. R. Jennings, and M. Wooldridge. 2003. Developing multiagent systems: The Gaia methodology. ACM Trans. Softw. Eng. Methodol. 12, 3 (July 2003), 317-370

[29] V.Torres da Silva, R. Choren, and C. J. P. de Lucena. 2004. A UML Based Approach for Modeling and Implementing Multi-Agent Systems. In Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - (AAMAS '04), Vol. 2. IEEE Computer Society, Washington, DC, USA, 914-921.

[30] Prometheus Methodology. http://www.cs.rmit.edu.au/agents/SAC2/methodology.html

[31] AUML (Agent UML), http://www.auml.org/

[32] UML 2 (http://www.uml.org/)

[33] J. J. Gomez-Sanz, J. Pavon, and F. Garijo. 2002. Metamodels for building multi-agent systems. Proceedings of ACM symposium on Applied computing (SAC '02). ACM, New York, NY, USA, 37-41.

[34] C. Hahn. 2008. A domain specific modeling language for multiagent systems. In Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems - (AAMAS '08), Vol. 1. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 233-240

[35] W. Jiao; Z. Shi; A dynamic architecture for multi-agent systems, Technology of Object-Oriented Languages and Systems, 1999. TOOLS 31. pp.253-260.

[36] G. Guemkam, D. Khadraoui, B. Gâteau, Z. Guessoum, ARMAN: Agent-based Reputation for Mobile Ad hoc Networks. To appear in 11th Conference on Practical Applications of Agents and Multi-Agent Systems Salamanca, Spain 22nd-24th May, 2013.

[37] J. Sabater and C. Sierra, Review on computational trust and reputation models, Artificial Intelligence Review, vol. 24, no. 1, pp. 33–60, September 2005.

[38] G. Guemkam, C. Feltus, C. Bonhomme, P. Schmitt, D. Khadraoui, Z. Guessoum, Reputation based Dynamic Responsibility to Agent for Critical Infrastructure, IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2011, Lyon, France.

[39] Keith Stouffer, Joe Falco, Karen Scarfone. 2011. Guide to Industrial Control Systems (ICS) Security.