How *Autonomic Fault-Management* Can Address Current Challenges in *Fault-Management* Faced in IT and Telecommunication Networks

Ranganai Chaparadza[1], Nikolay Tcholtchev[1], and Vassilios Kaldanis[2]

[1] Fraunhofer FOKUS Institute for Open Communication Systems, Berlin, Germany
{ranganai.chaparadza,nikolay.tcholtchev}@fokus.fraunhofer.de
[2] VELTI S.A. - Mobile Marketing & Advertising, Athens, Greece
vkaldanis@velti.com

Abstract. In this paper we discuss the perspectives that should be taken into account by the research community while trying to evolve Fault-Management towards Autonomic Fault-Management. The well known and established FCAPS Management Framework for Fault-management, Configuration-management, Accounting-management, Performance-management and Security-management, assumes the involvement of human technicians in the management of systems and networks as is the practice today. Due to the growing complexity of networks, services and the management of both, it is now widely believed within the academia and the industry that the concept of Self-Managing Networks will address some of the current challenges in the management of networks and services. Emerging Self-Management technologies are promising to reduce OPEX for the network operator. There is still a lot of work to be done before we can see advanced, production level self-manageability aspects of systems and networks, beyond what has been achieved through scripting based automation techniques that have been successfully applied to management and network operation processes. The concept of autonomicity—realized through control-loop structures and feed-back mechanisms and processes, as well as the information/knowledge flow used to drive the control-loops), becomes an enabler for advanced self-manageability of networks and services, beyond what has been achieved through scripting based automation techniques. A control-loop can be introduced to bind the processes involved in each of the FCAPS areas, and the "autonomic manager components" that drive the control loops and are specific for different FCAPS should interwork with each other in order to close the gaps characterized by dependencies among FCAPS functional areas as the FCAPS functional areas go autonomic and realize self-management. The dependencies among FCAPS functional areas need to be studied such that the functions/operations and processes that belong to the different areas can be well interconnected to achieve global system goals, such as integrity, resilience and high degree guarantee of system and service availability.

Keywords: Autonomic Fault-Management; GANA architectural Reference Model for Autonomic Networking and Self-Management; Resilience; Self-Healing/Self-Repair; dependencies among FCAPS functional areas; Interactions between the Operator and the Autonomic Network.

R. Szabó et al. (Eds.): AccessNets 2010, LNICST 63, pp. 253–268, 2011.

1 Introduction

The main benefits of introducing self-management technology in systems and networks, from the operator's perspective are: to minimize operator involvement and OPEX in the deployment, provisioning and maintenance of the network, and increasing network reliability (self-adaptation and reconfiguration on the fly in response to challenges e.g. faults, errors, failures, attacks, threats, etc). According to the research/survey study conducted and published by authors of [1], operators have recently provided a set of requirements on the evolution of network and services management, offering an insight into the need (requirement) for self-management in networks and processes of the next generation Operations Support Systems (OSS's). In [1], we learn about other requirements that operators have identified as requiring serious attention by researchers in the coming years. The requirements noted by operators in [1] include, apart from *self-management: alarm correlation, self-healing, Auto-Configuration/Provisioning, Model/Interface Integration, Service Quality, Service Modeling and Discovery.*

The concept of *autonomicity*—realized through control-loop structures [2, 3] and feed-back mechanisms and processes, as well as the information/knowledge flow used to drive the control-loops, becomes an enabler for advanced self-manageability of networks and services, beyond what has been achieved through scripting based automation techniques. A control-loop can be introduced to bind the processes involved in each of the FCAPS areas, and the "autonomic manager components" that drive the control loops of different FCAPS areas should interwork with each other in order to close the gaps characterized by dependencies among FCAPS functional areas as the FCAPS functional areas go autonomic and realize self-management. Autonomic Manager Components (referred to as "Decision Elements" (DEs) in the GANA Model [3, 4]) must serve the purpose of automating management processes by even executing some scripts while at the same time governing the autonomicity for a particular functionality for which the autonomic manager is responsible for. The autonomicity for a functionality e.g. routing, considers the following: (1) the auto-discovery of items required by the functionality to perform an auto-configuration/self-configuration process; (2) predictions/forecasting and listening for some events and reactions by the "autonomic manager" that controls and adapts the behaviour of the functionality towards some goal, based on the events.

The dependencies among FCAPS functional areas need to be studied such that the functions/operations and processes that belong to the different areas can be well connected to achieve global system/network goals, such as integrity, resilience and high degree guarantee of system and service availability. In this paper we focus on the Fault-Management functional area of the FCAPS framework and illustrate how to close the gap of its dependencies with other FCAPS functional areas in the context of an autonomic, self-managing network. The following is an illustration of the chain of processes that form a control-loop of an "autonomic manager component" specifically designed to perform Autonomic Fault-Management (we refer the reader to [5, 6] for more details on the subject of Autonomic Fault-Management (AFM), as well as to the subsequent sections of this paper). Autonomic Fault-Management [5, 6, 7] is understood as a control

loop structure (Figure 1) that facilitates the interplay between the processes of Fault-Management as defined by TMN (Telecommunications Management Network) [8]: Fault-Detection – *"detect the presence of a fault"*, Fault-Isolation – *"find the fault (root cause) for the observed erroneous state"*, and Fault-Removal – *"remove or reduce impact of the root cause"*. The autonomic fault-management control loop is characterized by the behaviour of components (including the respective "autonomic manager components") and mechanisms that collaboratively work towards implementing the following chain of operations/functions:

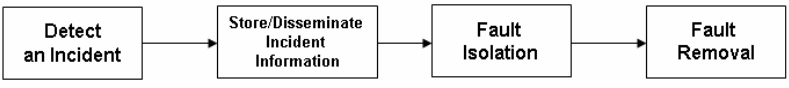

Fig. 1. Autonomic Fault Management Control Loop

The rest of the paper is organized as follows: Section 2 presents the implications of FCAPS going autonomic; Section 3 discusses Autonomic Fault-Management in relation to network and service management; sections 4 and 5 discuss further implications of Autonomic Fault-Management; Section 6 and 7 introduce the GANA Model in brief, which forms the basis for a unified architecture for realizing Autonomic Fault-Management, Resilience and Survivability. Finally we give concluding remarks.

2 Closing the Gap of Dependencies among FCAPS Functional Areas as the Areas Go Autonomic and Realize Self-management

In [9], we learn of the relationships between dependability of system(s) and security. These relationships i.e. dependencies should inspire efforts towards closing the gap of dependencies among FCAPS functional areas as the areas go autonomic and realize self-management. Indeed, efforts must be stepped up towards creating frameworks that show the dependencies and how to harmonize and integrate the corresponding operations of the diverse FCAPS functional areas. This would avoid having disjoint non-interworking solutions that are currently inherent in OSS (Operations Support Systems), consequences of which are well known and studied as described very well in [1]. In this section we present a consolidated picture on how to link (close the gaps) among the FCAPS functional areas for an autonomic/self-managing network: The links can be established primarily through (1) the interaction and synchronization of "autonomic manager components" that automate management processes involved in a particular FCAPS functional area with those in the other areas, in order to assure high degree of system/network integrity, reliability and availability; (2) the sharing of information/knowledge and data among the functional areas that need to use them in their respective functions; (3) the use of common models (Information Models, Data Models, Ontologies, Resource Description Models, Service Models, Network Description Models, Service Topology Models, Dependability and Causality Models,etc).

It is in such a consolidated picture that we seek to show the way Autonomic Fault-Management can help address some of the challenges that operators are facing with respect to network and services management.

The figure below (Figure 2) illustrates how the gaps can be interpreted and closed. In terms of the dependencies and desired interactions of the FCAPS functional areas, we see that Information/Knowledge sharing, use of Models, Ontologies, and collaborations of the functions belonging to the diverse functional areas are key to closing the gaps to achieve automation and self-management in an autonomic manner. This picture can be applied when reasoning about designing a system e.g. a router or the network architecture as a whole, required to be autonomic and self-managing it its operation. As can be observed on the figure, Autonomic Fault-Management requires, apart from information about detected faults/errors/failures and alarms, some access to information/knowledge about detected threats and attacks in order to use the information during Fault-Diagnosis/Localization/Isolation process, and even during fault-forecasting, and fault-removal operations i.e. in the associated algorithms. Autonomic Fault-Management may trigger reconfiguration procedures while attempting to remove certain types of detected faults, and so it needs to interact with Autonomic Configuration Management components and mechanisms. On the figure, we show some shared repositories that store information/knowledge such as models, required as input to the different FCAPS functional areas. Some of the Information/Knowledge maybe created and possibly used only by one or a few of the functional areas, as shown for the case of Fault-Management and Security Management. What is worthy to note is that the process of creating and populating the "Information/Knowledge Bases" depicted on the figure, with the types of information/knowledge depicted must be "automated" according to what the operators indicated in [1]. For example the auto-discovery of resources and capabilities of functional entities such as network elements and service components, and the building up of such information/knowledge in the corresponding repository must be something considered automated. Autonomic Fault-Management in particular, uses for example, information about Resource and Capability Descriptions to find appropriate mechanisms that can be applied to remove a fault when it has been narrowed down to a particular faulty resource. Capabilities information may then be used by Autonomic Fault Management in collaboration with the Configuration Management functions, to re-configure a system(s) based on knowledge about its Capabilities. Also, from resource descriptions obtained automatically by the self-description and advertisement of Resource and Capability Descriptions from the network elements, it is possible to create a picture of a candidate topology the operator could deploy as well as the advantages of a candidate topology. The advantages could be expressed in terms of employable resilience and recovery strategies (see [10] for more information on the subject of resilience and recovery strategies the operator needs to consider for different types of networks, resources and topologies, and even strategies that can be implemented through a Network Management System (NMS)). Later, in this paper, we describe how Autonomic Fault-Management uses some information/knowledge highlighted in the Figure 2.

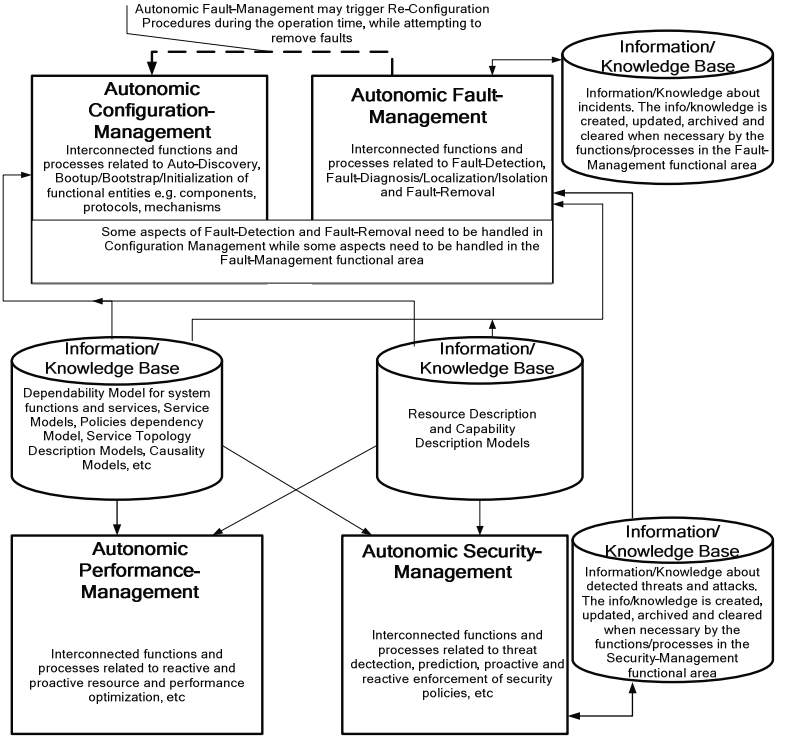

Fig. 2. Closing the gaps among dependencies of FCAPS functional areas

3 Implications of Autonomic Fault-Management (AFM) on a Network and Services Management

The implications of Autonomic Fault-Management on Network Management is that all the so-called traditional network management functions, defined by the FCAPS management framework (Fault-management, Configuration-management, Accounting-management, Performance-management and Security-management), as well as the fundamental network functions such as routing, forwarding, monitoring, discovery, fault-detection, fault-removal and resilience, are made to automatically feed each other with information (knowledge) such as goals and events, in order to effect feedback processes among the diverse functions. Since some degree of self-management and self-adaptation needs to be introduced into network device architectures e.g. for routers and switches, the implication is that the FCAPS functions become diffused within node/device architectures, apart from being part of an overall network architecture—whereby traditionally, a distinct Management Plane is

engineered separately from theother functional planes of the network [3, 4]. The recently emerged architectural Reference Model for Autonomic Network Engineering and Self-Management, dubbed GANA (Generic Autonomic Network Architecture), presents a framework on how the FCAPS becomes diffused into device architectures, while still maintaining an outer Management Plane that would be required to evolve to perform more sophisticated decisions for the operation and optimization of the whole network in an autonomic manner. In this paper, we present the GANA Model in brief, as we seek to illustrate how we have derived a GANA-orientated architecture that unifies Autonomic Fault-Management and Resilience and Survivability within device architecture and the overall network architecture. It is this unified architecture and the autonomic interworking of its components, placement of functions and algorithms that we consider as the machinery for addressing challenges currently faced by operators in Fault-Management.

4 Implications of Autonomic Fault-Management on a Network Element Architecture, Protocols and Services

Apart from the architectural implications that can be derived easily from the GANA Model as discussed above and also in [11, 5, 6, 12], the need for information sharing is very crucial. Looking at the current practices in the design of network elements, protocols, services, fault-tolerance implemented in such entities, in most of the cases, implies that the entities execute their intrinsic fault-tolerance mechanisms and do not share with other functional entities (through say a shared information/knowledge base), the information as to *what happened, what was detected and was the problem handled successfully?*. This information may be useful elsewhere, at a higher level where algorithms for fault diagnosis and resolutions would benefit from knowing such information. This requirement for information sharing in autonomic fault-management is described in more detail in [5], where the need for information repositories for registering and sharing such information is discussed in detail. Later, in this paper we briefly touch this subject when we discuss the unified framework for Autonomic Fault-Management, Resilience and Survivability.

5 How Different Types of Faults Can Be Addressed Autonomically

In this section (in the Table below), we summarize different types of faults and discuss the different types of Faults that are handled by Resilience and Recovery mechanisms found in fault-tolerant systems, services and protocols (let's call this *Domain-A*), while contrasting to the ones that must be handled by Autonomic Fault-Management(let's call this *Domain-B*). The role of Autonomic Fault-Management in relation to any type of fault is also discussed. We also discuss the inter-working required between functions in *Domain-A* and functions in *Domain-B.*

Detected Faulty conditions, faulty input to network systems, and fault symptoms	*Domain-A:* **Resilience, Recovery and Fault-tolerance Mechanisms intrinsically built into Protocols, Services and Applications**	*Domain-B:* **Autonomic Fault-Management** We also discuss the inter-working required between functions in *Domain-A* and functions in *Domain-B*		
		Fault-Detection	***Fault-Diagnosis/Localization/Isolation***	***Fault-Removal***
Faults, Errors, Failures detected at the Network layer, Link and Physical Layers. Examples include, faulty components, faulty modules, component failure, Link Failure, Node Failure and other related types of errors and failures (e.g see [13]). According to today's systems engineering approaches, mechanisms exist for detecting all these kind of "problems" to enable reactive resilience and recovery behaviours in some protocols and layers [10].	Some protocols are designed with intrinsic mechanisms to detect faults/errors/failures and react by executing resilience and recovery mechanisms i.e. fault-tolerance mechanisms and strategies, e.g. Protection and Restoration Schemes in Telecommunication networks [14, 15, 10]. For example Telecommunication Protocols such as ATM, MPLS/IP-FR, SONET/SDH exhibit such resilience and recovery characteristics [15, 10]. From the perspective of Autonomic Fault-Management: To augment Alarms that are often generated for some faults, it is necessary to enrich the sets and *quality of the*	Fault-Detection needs to not only rely on Alarms generated as symptoms, but also access to detected errors and failures or at least knowledge about even some low level incidents that are normally not communicated to the Network Management System (NMS) through Alarms (simply because "humans" may have thought that some low level incidents need not be communicated or even archived for offline analysis).	Normally this process is preformed using Alarm information coming from the network elements. For Autonomic Fault-Management some information about even low level detected incidents often not considered necessary to be communicated to the Network Management System (NMS), as well as detected threats and attacks may be required in performing extensive Fault-Localization, provided that it is practically feasible to include all such information in a Causality Model of a device or a network of some scope. Diverse Fault Diagnosis/Localization and Isolation techniques exist today (see [16]) and may be applied in Autonomic Fault-Management as discussed in [5, 6].	Fault Removal mechanisms must infer whether resilience and recovery i.e. fault –tolerance mechanisms employed by some protocols upon the detection of an incident have executed successfully or not, in order to execute a strategy on the global level, to remove or reduce the impact of a fault, in order to ensure system/network integrity, reliability and availability. Fault-Removal relies on Dependability and Causality models and Service Topology (if not embedded as part of a Dependability Model) as discussed in [7] and later in this paper.. Decisions regarding Fault-Removal are taken locally by a system while at the same time the system performs actions requested by a higher Decision –level e.g. the NMS level. Some distributed and collaborative Fault-Removal can be performed by network systems [17].

	communicated incidents information and the outcome of reactions by resilience mechanisms also need to be communicated to Autonomic Fault-Management components that keep track of global state and perform long term fault resolutions.			
Faults, Errors, Failures detected at the Services and Application Layers.	There is a great deal of work achieved and research continues in systems engineering practices for the design of fault-tolerant service components and applications. Multi-Layer resilience strategies (see [14, 10]) also include how the service/application layer reactions should be taken into account.	As OSS are evolving to cover Service Management aspects, Fault Detection in the Services/Applications Layer follow similar requirements as discussed in the case of Fault-Detection at the network layer and below, that of information sharing with the Fault-Management functions of the system and network. Since Service Management is the "playground" whose responsibilities fall solely on the network operator, it really means that faults emanating from Service creation, deployment and maintenance must be solved by the tools available to the operator, i.e. Fault-Management processes. As the operator desire to see automated	It still remains a challenge: how to build management systems that unify the management of services and network resources. As noted in [1], MIBs and alarms specific to services and applications are not widely developed to aid in the overall Services Management processes including Fault-Management. The concept of Service Topology, as discussed in [1], must be instrumental to Fault-Diagnosis for Services. As illustrated in Figure 2, the Model describing Service Topology must be captured, stored and shared automatically by all the FCAPS functional areas.	Fault-Removal relies on Dependability and Causality models and *Service Topology* (if not embedded as part of a Dependability Model). It also remains a challenge as to how to automate the processing of trouble tickets in order to execute some steps towards removing problems which may be narrowed down to network level, though some OSSs' can perform some limited steps in that direction..

		service monitoring (refer to [1], it means Autonomic Fault-Management must perform Service Monitoring e.g. Service Probing to detect problems.		
Alarms	At this level ("*Domain-A"*), some resilience and recovery functions process alarms and react upon them, while at the same time generating other types of alarms to the Management Systems or OSS's (see [18] for alarm reporting function).	Alarms are used primarily as symptoms used for triggering Fault-Diagnosis/Localization.	We can eliminate the problem of *redundant Alarms* discussed in [1] through models that when used by network elements, the elements can collaborate to suppress some alarms to avoid multiple alarms being sent to the NMS when the alarms are triggered by the same fault. Advanced *Alarm correlation* techniques are required as integrated solutions with the Autonomic Fault-Management system [1].	Alarm "severity" indications are vital input to Fault-Removal functions and decisions made autonomically.
Human Errors during network and service management phase	Such Human Errors are not handled at this level ("*Domain-A"*), but rather in Fault-Management ("*Domain-B").*	As illustrated on Figure 2 there are some aspects of Fault-Detection and Fault-Removal that should be handled in Configuration Management area while some aspects that should be handled in the Fault-Management functional area. In [19], an insight is provided as to what sort of faults may be created during the configuration of the network by humans e.g. erroneous configuration data provided as input to systems. In 18[19], automation techniques are discussed that relieve the operator from introducing severe faults into the input supplied to configure network elements. We believe that as further automation is required beyond what is described in [19] via autonomics, the autonomic Fault-Manager components of the network (presented later in this paper) should incorporate Fault-Tolerance techniques described in [9], or generally speaking fault-removal techniques) such as Rollback, Rollforward, compensation, Reconfiguration and Re-initialization, etc.		
Unknown Faults resulting from insufficient Testing of software	We can not expect resilience and recovery mechanisms to handle such faults.	Such kind of faults are hard to detect, and often classified as "unknown" during Fault-Diagnosis/Localization/Isolation. Often they are handled by upgrading the software and hardware components by applying patches and resorting to new releases [19]. From the perspective of Autonomic Fault-Management, the detection of situations whereby faults can be classified as "unknown" due to the fact that the techniques used in Fault Localization may have reached an inconclusive verdict, means the Autonomic Manager Components (i.e. DEs in the GANA Model [3]) responsible for autonomically managing a device (s) should automatically be designed to be able to synchronize with each other to fetch patches and new releases from vendor servers, and schedule updates events for the network, after which they can perform self-validation to ensure that the systems and the network are working properly.		

6 Generic Autonomic Network Architecture in Brief

A central concept of GANA is that of an autonomic Decision-Making-Element ("DME" or simply "DE" in short—for Decision Element) i.e. an "Autonomic Manager Component". A Decision Element (DE) implements the logic that drives a control-loop over the "management interfaces" of its assigned Managed Entities (MEs). Therefore, in GANA, self-* functionalities such as self-configuration, self-healing, self-optimization, etc, are functionalities implemented by a Decision Element(s). The *Generic Autonomic Network Architecture (GANA)* is based on a set of requirements derived in [3, 4]. Since control loops on different levels of functionality are possible, e.g. on node or network level, GANA defines the Hierarchical Control Loops (HCLs) framework. The HCLs Framework fixes and establishes four levels of abstraction for which DEs, MEs and associated control-loops can be designed: **Level-1:** *Protocol-Level*, i.e. control loops embedded within protocol modules (e.g. within some routing protocol). **Level-2:** *Abstracted Functions-Level*, i.e. DEs managing some abstracted networking functions inside a device e.g. routing, forwarding, mobility management, **Level-3:** *Node Level* - the node level consist of a Node_Main_DE that takes care of the management of aspects related to the state/fitness of the overall node, e.g. Fault-Management and Auto-Configuration. **Level-4:** *Network Level* -DEs on that level manage different aspects that require to be handled at the network-level, e.g. routing or monitoring, of a group of nodes according to a network scope. Thereby, control loops (i.e. DEs) on a higher level manage DEs on a lower level down to the lowest-level "pure" MEs. Detailed information about all the presented concepts, examples, as well as discussions on the application of GANA to diverse aspects of Autonomic Networking can be found in [3].

7 Decision Notification in GANA for the "Human in the Loop"

Certain types of decisions made by DEs should be communicated to the administrator. Potentially, decision made at the "node-level" in GANA or at the "network-level" by Network-Level-DEs are candidate for "decision notification" to the human. The majority, though, should be handled by the autonomic entities. This requires that:

(a) When the administrator informs the network that for certain types of Decisions (the human will specify them using some means e.g. a Rule or Policy Specification Language) she wants a Decision Notification, the DEs should notify the administrator so that the human closes the loop by providing a response to the Decision Notification. This may happen during the early days/weeks/months of operating the autonomic network till the time when the administrator has build trust. **(b)** When the administrator deactivates Decision Notification, DEs shall proceed with executing decisions without Decision Notification to the human. **(c)** When Decision Notification is deactivated as in **(b),** DEs shall however inform (possibly by simply logging the decision(s) that were taken in response to "a triggering event"), meaning that both: decision(s) and associated event should be logged. Example cases for decision notification are given in [20].

8 GANA-Oriented Unified Architecture for Autonomic Fault-Management, Resilience, and Survivability in Self-managing Networks

According to [2], self-healing is defined as follows: *"To detect incidents such as adverse conditions, faults, errors, failures; diagnose, and act to prevent or solve disruptions"*. That is, on one hand a network equipped with autonomic self-healing mechanisms should aim at automatically preventing future fault activations, and on the other hand it should resolve occurred/activated faulty conditions. This corresponds to a number of concepts and approaches that have been investigated recently, such as Autonomic Fault-Management as well as reactive and proactive Resilience in autonomic networks.

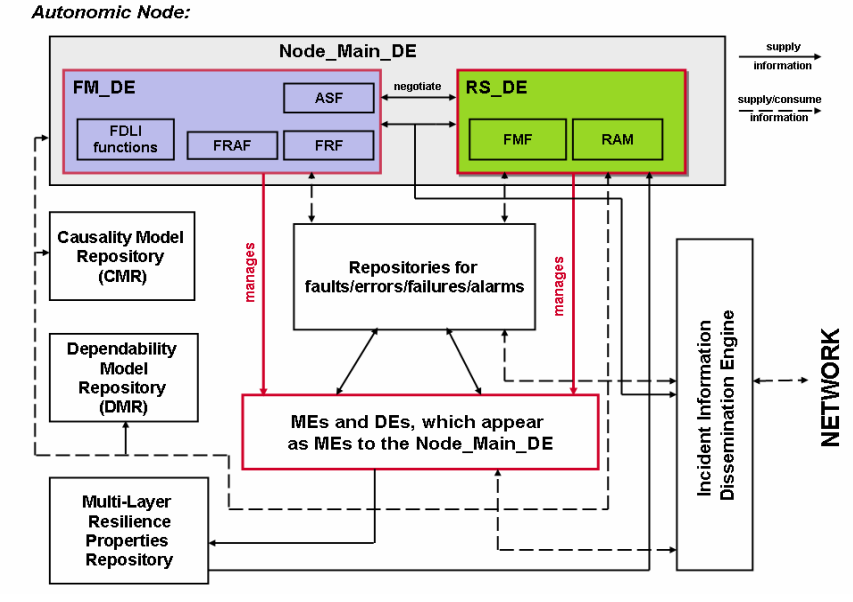

Fig. 3. The UAFAReS [11] architecture inside an Autonomic Node

Within the EFIPSANS [21] project, we introduced a *Unified Architecture for Autonomic Fault-Management, Resilience and Survivability in Self-Managing Networks* (UAFAReS). UAFAReS is based on the observation that the evolution of traditional Fault-Management towards Autonomic Fault-Management enables network devices to exercise self-healing and recover from faulty conditions. That is, the nodes of the network are then able to automatically self-heal (to some degree) without the need for human intervention. Hence, Autonomic Fault-Management has to interplay with concepts and mechanisms related to Fault-Tolerance, Fault-Masking, and Multilayer Resilience [14]. This implies harmonization (i.e. ordered time-scaling of reactions) to incidents, at different levels of autonomicity and self-management defined by GANA.

UAFAReS is based on the GANA reference model and specifies a number of components which aim at realizing the interplay of the aforementioned aspects. The node components of the architectural framework are illustrated in Figure 3. The main UAFAReS entities in a device are the Fault-Management Decision Element (FM_DE) and the Resilience and Survivability Decision Element (RS_DE). The RS_DE is responsible for an immediate reaction to the symptoms of an erroneous state while in parallel the FM_DE performs Fault-Isolation and Fault-Removal in order to eliminate the corresponding root cause(s). Both DEs are part of the Node_Main_DE, i.e. they are introduced on node level inside a GANA conformant device, in order to have exclusive access to all node functional entities (i.e. DEs and MEs) such that the overall autonomic behaviors of a node with respect to coping with incidents and alarms are synchronized to ensure node integrity. The UAFAReS DEs operate based on distributed control loops. The distributed nature of the UAFAReS control loops is enabled by a number of components that facilitate the incident information exchange across the network nodes. A *set of repositories for storing incident information* and an *Incident Information Dissemination Engine (IDE)* enable the synchronization of the faults/errors/failures/alarms knowledge known by UAFAReS DEs residing in different devices, and allow the DEs to perform Fault-Masking, Fault-Isolation and Fault-Removal in a node specific manner, based on the same information.

The FM_DE consists of four modules: 1) a component responsible for Fault-Isolation *(Fault-Diagnosis/Localization/Isolation functions* abr. *FDLI*), 2) *Fault-Removal Functions (FRF)*, 3) *Action Synchronization Functions (ASF)* – responsible for synchronizing (allowing and/or disallowing) tentative actions issued as by the RS_DE and the FM_DE control loops running in parallel, 4) *Fault-Removal Assessment Functions (FRAF)* – a component responsible for assessing and verifying the success of the fault removal actions issued as output of the FM_DE. The interactions of these modules towards the realization of an Autonomic Fault-Management control loop are illustrated in Figure 4. Specially instrumented monitoring entities, which have the capability to share incident information over the UAFAReS incident repositories, push descriptions of symptoms to the UAFAReS fault/error/failure/alarm registries such that the info gets conveyed (i.e. stored in the UAFAReS node registries) by the Incident Dissemination Engine (IDE) to the UAFAReS instances across the network scope, e.g. subnet/LAN. Once an incident description has been reported to the FM_DE over the UAFAReS incident repositories, it gets received and processed first by the FDLI functions as depicted in Figure 4. That is, the FDLI functions collect such events and correlate them in order to find the root cause for the observed faulty conditions. Algorithms that can be used for event correlation are presented in [6, 16, 22]. The correlation of incident events is realized by the FDLI functions based on a Causality Model that is kept in the *Causality Model Repository (CMR)* inside a node. The identified root cause(s) (faults) are then further submitted to the Fault-Removal Functions which implement an *"if-then-action"* logic that issues a reaction required to eliminate the faults, e.g. reconfiguration of an entity by using, e.g. the command line interface (CLI). Since it is possible that the tentative reaction would interfere with other actions that are intended to be performed by either the RS_DE control loop (next paragraph), or would interfere with a parallel Autonomic Fault-Management control loop process (i.e. a thread in multi-threading environment), the ASF should be invoked in order to allow or disallow the tentative action in question.

The ASF is based on techniques from the area of optimal control, and selects the optimal subset of tentative actions in order to better optimize the network performance reflected by the values of selected key performance indicators while ensuring integrity. An applicable algorithm can be found in our previous work [12]. Given that the ASF has allowed a tentative action, the FRF issues it on the MEs in question inside the device. Thereby the FRF can make use of information regarding the dependencies among protocol entities and services, kept in the *Dependability Model Repository (DMR)*. Finally, the success of the executed action is assessed by the FRAF functions, which may choose to notify the network operator in case when the UAFAReS mechanisms can't cope with the pending challenges.

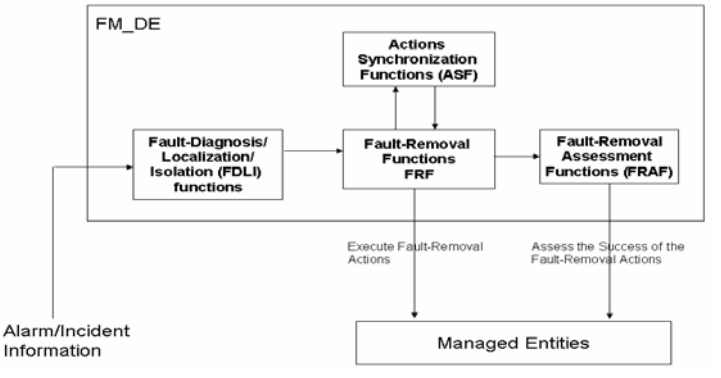

Fig. 4. The Autonomic Fault-Management control loop inside a node

The Resilience and Survivability DE contains the *Fault-Masking Functions (FMF)* component and a *Risk Assessment Module (RAM)*. The Fault-Masking Functions realize a reaction immediately after the symptoms of a faulty condition have been registered into the UAFAReS alarm/incident repositories. Thereby, the goal of the FMF is to implement a fault-tolerant behavior such that some fundamental level of service can be sustained in the face of a pending challenging condition. The FMF follow a similar logic as the Fault-Removal Functions of the FM_DE, and consult the Actions Synchronization Functions of the FM_DE to react first in order to ensure that the best possible set of actions is executed. The FMF, as the instance of first reaction, should also consider the aspect of Multilayer Resilience while orchestrating a fault-tolerant/masking behavior. Multilayer Resilience is a model that deals with the capabilities of functional entities at different layers in the protocol stack to execute their own embedded resilience behaviors. For instance, in IP networks generated ICMP messages enable systems (especially end systems) to overcome issues occurring in the network, e.g. sudden changes of PMTU (Path Maximum Transmission Unit) during the lifetime of a connection. Thus, the FMF is expected to allow the protocol modules to recover based on their own intrinsic capabilities and should intervene only in the case when these mechanisms fail. [14] proposes the usage of "hold-off" timers specifying the time that should be given to a protocol to recover on its own. Information on

how to handle the resilience properties of a protocol module (e.g. protocol module ID and corresponding "hold off" timer) is kept in the *Multi-Layer Resilience Properties Repository*. In addition, the operation of the Risk Assessment Module (RAM) is based on monitoring information about diverse key performance indicators (e.g. CPU temperature) that are used to calculate the probability for failures in the future. This results in notifications to the FMF which consequently have to trigger mechanisms that help proactively avoid significant degradation in the QoS in the future.

9 Concluding Remarks

In this paper, we presented some perspectives on how Autonomic Fault Management can address some of the challenges in Fault-Management faced in IT and Telecommunication Networks. The work presented is framework oriented, and is inspired by the need to move the well known and established FCAPS management framework towards automated management through concepts and principles of autonomic networking and self-management. Looking at the fact that there exist dependencies between dependability of systems and security, we see the need to close gaps characterized by dependencies among FCAPS functional areas as the FCAPS functional areas go autonomic and realize self-management. The dependencies among FCAPS functional areas need to be studied such that the functions/operations and processes that belong to the different areas can be interconnected well to achieve global system goals, such as integrity, resilience and high degree guarantee of system and service availability. We have illustrated how this can be achieved by providing a framework that can be further refined while doing actual implementations. We believe that such a framework should be the basis for reasoning about how Autonomic Fault-Management can address some challenges faced by Operators and vendors in the design and operation of Self-Managing Future Networks. We categorized some types of faults and discussed how certain faults are handled by resilience and recovery mechanisms intrinsic to some protocols and service components, and how certain faults can only be handled within the realm of Configuration Management phase, and some within the real of Fault-Management during the operation time of the network. The role of enriched information/knowledge sharing has also been discussed as part of the glue required to interconnect functions and operations belonging to the different FCAPS functional areas. We do not claim that there are no obstacles to implementing the framework, since what we offer are guidelines and so, issues such as scalability and complexity need to be addressed during the derivation of implementation architectures from the framework we provided. Our further work will be based on the evaluation of the frameworks, such as the UAFARes Framework we proposed, whose architectural components are built on the concepts and principles prescribed by the emerging, standardizable GANA architectural Reference Model for Autonomic Networking and Self-Management.

Acknowledgement. This work has been partially supported by EC FP7 EFIPSANS project (INFSO-ICT-215549).

References

[1] Wallin, S., Leijon, V.: Telecom network and service management: An operator survey. In: Pfeifer, T., Bellavista, P. (eds.) MMNS 2009. LNCS, vol. 5842, pp. 15–26. Springer, Heidelberg (2009)

[2] Autonomic Computing: An Architectural Blueprint for Autonomic Computing. IBM White Paper (2006), http://www-01.ibm.com/software/tivoli/autonomic/

[3] Chaparadza, R.: Requirements for a Generic Autonomic Network Architecture (GANA), suitable for Standardizable Autonomic Behavior Specifications for Diverse Networking Environments. International Engineering Consortium (IEC), Annual Review of Communications 61 (2008)

[4] Chaparadza, R., Papavassiliou, S., Kastrinogiannis, T., Vigoureux, M., Dotaro, E., Davy, A., Quinn, K., Wodczak, M., Toth, A.: Towards the future internet - A European research perspective. In: Creating a viable Evolution Path towards Self-Managing Future Internet via a Standardizable Reference Model for Autonomic Network Engineering, pp. 136–147. IOS Press, Amsterdam (2009); published by the Future Internet Assembly (FIA) in Europe

[5] Chaparadza, R.: Unifaff: a unified framework for implementing autonomic fault management and failure detection for self-managing networks. Int. J. Netw. Manag. 19(4), 271–290 (2009)

[6] Tcholtchev, N.: Scalabale Markov Chain based Algorithm for Fault-Isolation in Autonomic Networks. Accepted to appear in the Proceedings of the NGN Symposium of Globecom (2010)

[7] Li, N., Chen, G., Zhao, M.: Autonomic fault management for wireless mesh networks. Electronic Journal for E-Commence Tools and Applications (eJETA) 2(4) (January 2009), http://www.cs.uml.edu/~glchen/papers/fault-ejeta09.pdf

[8] The FCAPS Management Framework. ITU-T Rec. M.3400 (February 2000)

[9] Avizienis, A., Laprie, J.C., Randell, B., Landwehr, C.: Basic concepts and taxonomy of dependable and secure computing. IEEE Trans. Dependable Secur. Comput. 1(1), 11–33 (2004)

[10] Autenrieth, A.: Differentiated Resilience in IP-Based Multilayer Transport Networks. Ph.D. thesis, Technische Universität München (2003); Presented in 2003 at "Lehrstuhl für Kommunikationsnetze"

[11] Tcholtchev, N., Grajzer, M., Vidalenc, B.: Towards a Unified Architecture for Resilience, Survivability and Autonomic Fault-Management for Self-Managing Networks. In: MONA 2009: Proc. of 2nd Workshop on Monitoring, Adaptation and Beyond, MONA+ (2009)

[12] Tcholtchev, N., Chaparadza, R., Prakash, A.: Addressing stability of control-loops in the context of the GANA architecture: Synchronization of actions and policies. In: Spyropoulos, T., Hummel, K.A. (eds.) IWSOS 2009. LNCS, vol. 5918, pp. 262–268. Springer, Heidelberg (2009)

[13] Markopoulou, A., Iannaccone, G., Bhattacharyya, S., Chuah, C.N., Ganjali, Y., Diot, C.: Characterization of Failures in an Operational IP Backbone Network. IEEE/ACM Trans. Netw. 16(4), 749–762 (2008)

[14] Touvet, F., Harle, D.: Network Resilience in Multilayer Networks: A Critical Review and Open Issues. In: Lorenz, P. (ed.) ICN 2001. LNCS, vol. 2093, pp. 829–838. Springer, Heidelberg (2001)

[15] Types and Characteristics of SDH Network Protection Architectures. ITU-T Rec. G.841 (December 1997)
[16] Steinder, M., Sethi, A.S.: A survey of fault localization techniques in computer networks. Science of Computer Programming 53(2), 165–194 (2004), http://dx.doi.org/10.1016/j.scico.2004.01.010
[17] Tcholtchev, N., Chaparadza, R.: On Self-Healing based on collaborating End-Systems, Access, Edge and Core Network Components. In: SELFMAGICNETS 2010: Proc. of the International Workshop on Autonomic Networking and Self-Management. ICST ACCESSNETS 2010 (November 2010)
[18] Information Technology - Open Systems Interconnection - Systems Management: Alarm Reporting Function, ITU-T Rec. X.733 (February 1994)
[19] Juniper Networks Inc.: Juniper Network Whitepaper: What's Behind Network Downtime? Proactive Steps to Reduce Human Error and Improve Availability of Networks (2008)
[20] Tcholtchev, N., Chaparadza, R.: Autonomic Fault-Management and Resilience from the Perspective of the Network Operation Personnel. In: IEEE MENS 2010: IEEE International Workshop on Management of Emerging Networks and Services (MENS), Miami (December 2010); in conjunction with IEEE Globecom 2010
[21] EC FP7-IP EFIPSANS Project (2008-2010), INFSO-ICT-215549 http://www.efipsans.org
[22] Hasan, M., Sugla, B., Viswanathan, R.: A conceptual framework for network management event correlation and filtering systems. In: Sloman, et al. (eds.), pp. 233–246 (1999)