System monitoring and fault management methods and techniques

Técnicas y métodos para gestión de fallas y monitoreo de sistemas

John Willian Branch, PhD, Sergio Armando Gutiérrez, I.S.
Escuela de Sistemas, Facultad de Minas, Universidad Nacional de Colombia, Sede Medellín.
(jwbranch|saguti)@unal.edu.co

Recibido para revisión 16 de abril de 2010, aceptado 4 de junio de 2010, versión final 28 de junio de 2010

Abstract—**This paper presents the most notable trends in the field of Fault Management, as an important issue in the evolution of service platforoms, basically in Telco Industry, because of the criticity of services running on them.**

Keywords— **Fault Management, Monitoring, Active Monitoring, Active Probing, Passive Monitoring, Finite State Machines, Autonomic Computing.**

Resumen— **Este trabajo presenta las más notables tendencias en el campo de la gestión de fallas, como un aspecto importante en la evolución de las plataformas de servicios, básicamente en la industria de las telecomunicaciones, dada la criticidad de los servicios que corren sobre éstas.**

Palabras Clave—**Gestión De Fallas, Monitoreo, Monitoreo Activo, Sondeo Activo, Monitoreo Pasivo, Máquinas De Estado Finito, Computación Autónoma.**

I. INTRODUCTION

Telco industry has showed a very accelerated evolution in the recent times, producing technologies highly sofisticated, and every time more involved in the whole fields of human activities. Education, Financial, Health sectors among other, have become highly dependent on systems and platforms, and every time is more common the term "critical mission" to identify the technological platforms which support the activities and processes of this kind of institutions and companies which offer first need services to people [1].

Critical Mission means that systems should be available 100% of time, and must not be interrupted by any circumstance, as this detention means a critical process can not be performed. This condition requires that any event associated to failures at any component of the system be managed in such a way that degradation or unavailability of system be avoided or minimized as much as possible.

Because of the high heterogenity and distribution of platforms and infrastructure commonly found nowadays in telco services, and the highly complex relationships which can be found among theses components, Fault Management has become a big issue in the field of operation and engineering, as in many cases, the ability of human operators and supervisors is not quick an efficient enough to respond in the best way to the events of failures.

This article presents a review of some outstanding works which present approaches to Fault Management and System Monitoring in several fields of computing. It is important to mention that Fault Management is quite specific to field where it is applied, so that is the reason of many approaches which are found when reviewing the literature.

II. System Monitoring Approaches

Current fault management solutions have two typical features:

- They are generally added a-posteriori to existing applications. This means that the applications are not designed for being managed. They often lack a suitable architecture andefficacious means for the diagnosis and repair of faults.

- They typically use external management functionality State or behavior information is extracted from the application and analyzed by an external manager. This means that problems are torn out of their live context, which (1) makes them much more difficult to diagnose and correct, and (2) breaks encapsulation. This defeats good software engineering principles, hinders reusability, and does not favor automatic management solutions.

System monitoring is a very important issue in Fault Management, as it becomes the main product for any analysis which drives to any intervention to manage faulty behaviors in the system The following sections of this paper present and analysis of different approaches to system monitoring, and relevant works reported on literature regarding this element in the Fault Management process.

A. Self Maintenance

Maintenance is a task which is performed on systems, and its main purpose is keep or restore the condition of the system to a predefined state. Maintenance can be preventive, when it is performed on a timely basis, to deal with degradation or aging or system parts, or corrective, when it is performed after a failure occurs. There is another novel approach named proactive maintenance, when current and past information and data of system state is processed to detect possible failures yet to occur, even forecasting the time before a failure occurs.

With the goal of offering increased availability in systems, industry has defined the concept of Self-Maintenance (SM). SM is taken from nature, emulating the capacity of self-repairing which is exhibited by some living organisms.

Mei-hui et al[2] introduce an approach to SM. According to authors, SM is part of a wider eature, Self-Recovery (SR).

SR can be understood as the combination of Self-Maintenance and Self-Repairing.

Self-Maintenance is composed by technologies as Recomposition, Dynamic Reconfiguration, Cold Backups, and Reconstruction, that is to say, technologies which allow to modify or even restore the state of the system to fulfill with some service levels.

Self-Repairing is the feature which allows to the system respond to any change presented on its environment. Self-Repairing is composed of responsing capacities, monitoring and adjusting internal system structure, and compensate malfunctions during operation, where applicable [3].

Self-maintenance is not a feature considered indispensable for the systems; it is more an evolution and improvement of conventional maintenance systems. Self-maintenance component is conformed by Detection System, which are the sensors on charge to detect events and issues; the information of these sensors is then transmitted to the Analysis System[4], which evaluates the situation according to fault type, and then, transmit to Maintenance System the decision about how to perform the maintenance itself.

Self-Maintenance as presented in this work is based on several techonologies, mainly from the world of Artificial Intelligence. Here, the main goal is that detection of events occur in very real-time, and analysis bases on smart models and algorithms, which make possible prognose, diagnose monitor and manage the issues. Self-maintentance highly contributes to increase the overall reliability, by allowing on-time FM for every component subsystem. Figure 1 presents a flow diagram which illustrates the concept of Self-Maintenance.

Labib [5] presents the notion of Next Generation Maintenance Systems, as an approach for the design of Self-Maintenance machines. In the first part of this work, the generations of maintenance are presented, with a first generation, lasting approximately until World War II, when maintenance was merely corrective, having fault as something unavoidable, so maintenance consisted in fixing equipment, a second generation lasting until 70's decade, when maintenance was focused in offering a high availability on equipments, while costs tried to be reduced. In this second generation, some statistical techniques started to be used in fault modelling, and computer systems, although the ones of these times were quite big and slow. The third generation, which is considered to extend to present time, when maintenance is focused in offering high availability and reliability, but with greater emphasis in environment and security.

So, the fourth generation will be an evolution of third, aiming to zero downtime in systems, by means of hot redundancies. It might be that the vision of this generation aims to avoid failure consequences, rather than failures themselves. To gain this, maintenance techniques should combine self-maintaning, self-repair and self-healing features.

This work also presents some unmet needs in responsive maintenance, this is, features which might be considered as desirable in Self-maintenance systems: a) Intelligent monitoring, prediction, prevention and compensation for sustainability, b) Prioritization, Optimization and responsive scheduling for reconfiguration needs and c) Autonomous information flows from market demands to factory asset utilization.

In particular, the last point is very important, as mentions that most of the maintenance method do not provide feedback to bussiness process within the organization, being these isolated of corporate policies and merely focused on infrastructure rather than services and processes.

An important difference which is presented between traditional maintenance (preventive and corrective) and Self-Maintenance is the fact that traditional maintenance consists in interventions to avoid more catasthrophic failures, while Self-

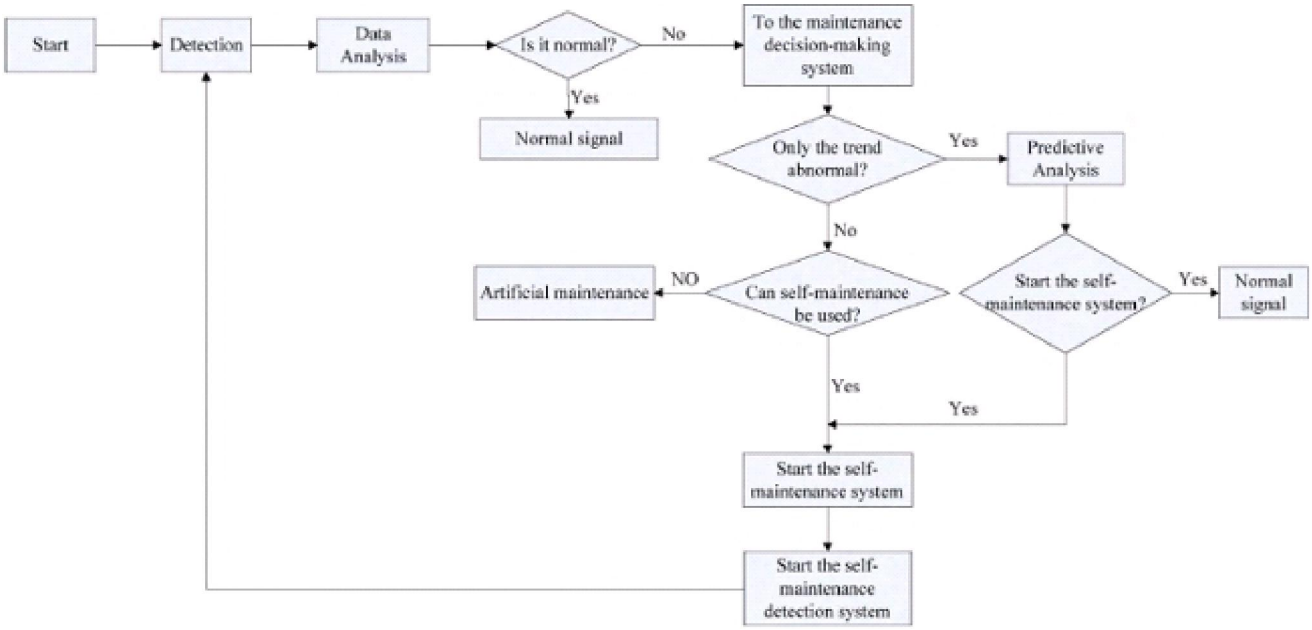

Figure 1. The development of Self-Maintenance Process [2]

Maintenance is based in intelligent monitoring, with adaptive and responsing behavior in the system. The features required in the system to be able to implement Self-Maintenance are perception, fault classification and diagnosis, failure prediction, repair planning and repair execution[6].

Author refers a classification for maintenance strategies[7]: Attributive and Functional maintenance. The attributive maintenance consists in replacing failing parts (which implies stopping the system) while functional maintenance tries to repair the functions rather than components. Functional maintenance focuses in the whole system, and would allow that Self-Maintenance be implemented.

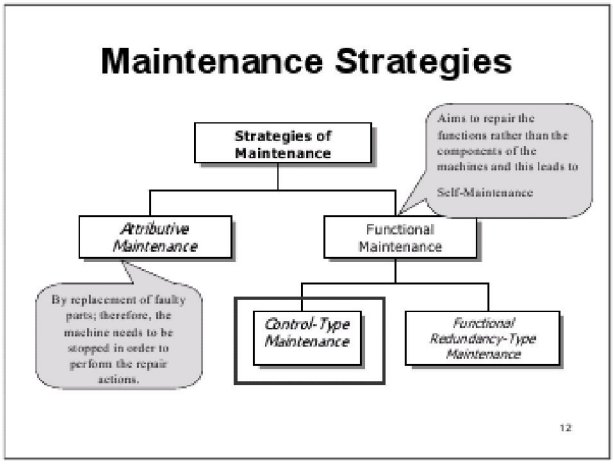

Figure 2. Maintenance Strategies [5]

Some features wich are required for Self-Maintenance are presented, as follows:

- Monitoring: Self-Maintenance systems must have sensor capabilities; sensors would collect raw data from system components, and this data would be sent to a processing entity.
- Fault judging: The processing entity would take the information coming from sensors and would judge whether system component is on normal or abnormal state. It would be desirable also that from this raw data, the system would try to predict time left for a failure.
- Diagnosing: After analysing the data, whether an abnormal state is found, the system should classify the failure, and identify if possible its root causes, so that a reparing plan can be carried out.
- Repair planning: From the information collected, the system should be capable to propose one or several maintenance plans. This plans would include also information previously stored, taken from experts. As many repair plans might be proposed, the system should decide to apply an optimized one.
- Repair execution: The action of maintenance which is executed by the system itself. This is performed by control and actuators on the system components.
- Self-Learning and improvement: Although an unexpected problem arises, the system is expected to include the applied solution as knowledge for a possible next time where the same failure arises. This feature will allow the system to repair itself in a shorter time frame, being more efficient and effective.

B. Active Monitoring

Monitoring is the periodic verification which is performed on service components and on service itself. According to Mohammed et al[8], monitoring can be classified in two main approaches: Active Monitoring, which is basedn in sending probes to system component, and Passive monitoring, which is based on evaluating the output of the system or the one of a particular component, and see whether it belongs to a right behavior.

The majority of works reported on literature are focused in active monitoring, as most of modern networl entities (both hardware and software) are highly instrumented, and exist many solutions of network managers, which are designed to handle probing in these network entities and process and analyze the information collected and sent by those entities t detect malfunctions and failures.

Because of the distributed nature of service components, usually, for fault detection and identification, a lot of probes are required. This has a high impact, related to information processing, storing and the delay to initiate any action to correct or repair the problem.

The work presented by the authors tries to solve a Constraint Satisfaction Problem, whose goal is find an optimal and adequate set of probes which allow fault identification. This set of probes (which is really a subset of the original set or probes required, still large) should be as accurate and should have the same diagnosis power as the original set of probes.

This technique tries to offer a very accurate and cost effective diagnosis; as it is an Active FM approach, it avoids problems and issues related to event correlation, which might be inherent to Passive FM. The approach presented in this work has the advantage to be highly adaptive, having the capacity to adapt itself to particular network conditions; it has the disadvantage that in some cases it could drive to solutions with a high cardinality, with consequent high mathematical complexity.

Figure 3 illustrates the architecture of the probing system proposed in [8]

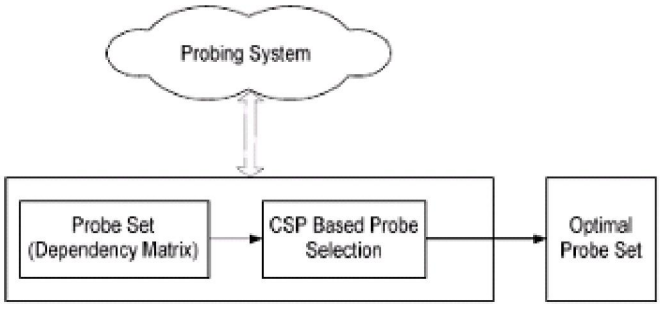

Figure 3. Architecture of Probing systems [8].

Li et al [9] study the active monitoring of Internet Services as Fault Detection mechanism. This approach tries to solve the problematic related to balance the number of probes which are sent, keeping the balance between enough probes for diagnose, but not increasing it on a measure which could saturate the network. This approach proposes the integration of *a-priori* fault distribution learning, and fault diagnosis in a Hidden Markov Model. The approach as presented, is very accurate to work, even in noisy and uncertain environments, reducing the complexity related to priory knowledge acquisition. Figure 4 illustrates the diagnosis probing selection method.

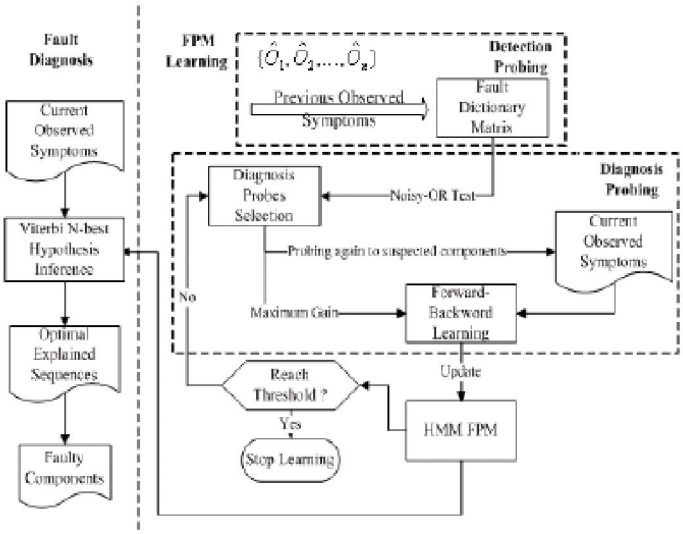

Figure 4. Diagnosis Probe selection [9]

Chu et al [10] present an approach based in two algorithms, one for probe selection, to monitor the whole components, and other for the fault diagnosis, by sending more probes to obtain a more detailed view of system state. This work focuses in Service Concept, conceiving the service as a composition of interdependent components, in such a way as whether a component fails, it for sure will cause a degradation or failure in the other components.

Chu proposes a layer model to understand the service, having the following layers:

- Demand Layer: It comprises the quality of service requirements defined for a particular service. By measuring the service performance, and evaluation to see whether the service is fullfiling these requirements or not. In case that requirements do not be fulfilled, corresponding alarms will be issued and sent to monitoring component.
- Service Layer: In this layer appear service themselves as defined in the system. There exists a one-to-many mapping among service and quality of service requirements.
- Dependency Layer: Contains the dependency relationships among system components. This layer applies for the cases where dependency model is adopted to define the system.

- Component Layer: This layer contains the components which integrate a service. In general, each service depends on several components, and different service may share dependent components.
- Mode Layer: This layer defines the modes or states where a service can be located at. One of these modes is defined as working mode, and the other will represent different degrees of degradation.

This work show and approach that reduces the necessity of probes, but produces a high diagnosis rate, and a low number of false positives. Figure 5 illustrates the representation of dependency model at service layer.

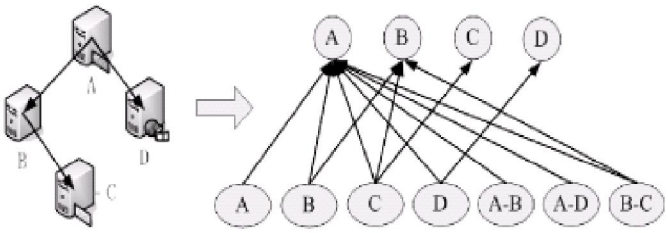

Figure 5. Dependencies model at service layer [10]

Chen [11][12] proposes an scheme named Proactive Probing and Probing on demand which can be used to monitor the whole components of a service. He defines an entity called Probing Service (PS), which is implemented as a Web Service, capable of performing a specific transacion, record its result, and send it to a management station. These services are deployed by means of Probing Service Outlets (PSO), where PS can invoked on a regular basis, or can be invoked at anytime.

The scheme proposed by the author is composed by three stages. A first stage, when PSO and PS are selected to monitor service components; the goal is select a set of PS as small as possible. At this point, the information is collected with the only purpose of monitoring the health of the system, and will not be useful for diagnosing the particular problem. The advantage of decoupling monitoring and diagnosing is reducing overhead and traffic, as system will be most of the time monitoring, with, as mentioned, the smallest set of PS.

The second stage is offline rule development, that is to say, provide and strategy in the case of one or several PS reports problems. This strategy should consider the whole possible pools of PS, for the failure detection. The last stage of this approach is enabling On Demand and Incremental probing, for the failure diagnosing and location. It is performed by using one or several PSs to get additional information, and it is guided by the strategy which was defined on second stage. Figure 6 illustrates the generic architecture of probing systems.

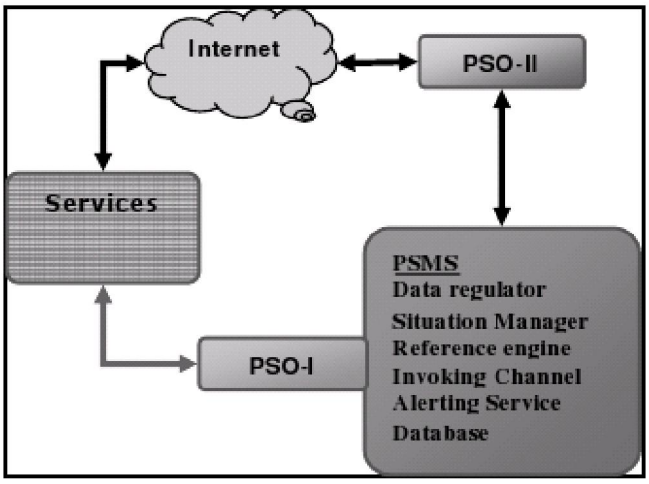

Figure 6. Generic architecture or probing systems [11], [12].

Kirmani and Hood [13] propose a mechanism to diagnose network states by means of a technique named Intelligent Probing, presented by Brodie et al [14]. Intelligent probing consists in finding an optimal set of probes, which tends to be small but still enough to get an accurate diagnosis, which be efficient in terms of precision and time.

C. Passive Monitoring

Passive monitoring is based on the correlation of events which are sent by network entities (hardware and software), where the information of the events is processed and analyzed.

Because of the increasing complexity of current service platforms, which are formed by many components as hardware, software and communication links, a lot of events usualy in the order of tenths of event logs might be produced. These logs might contian information regarding problems occuring, or yet to occur, but usually they do not offer information sbout the root cause of the problem. Another problem is that in most of the cases, failure do not happen one at time, but many failures occur simultaneously, being this a problem for the human operator who has the responsability of analyzing the alarms from the network elements to detect the cause or causes of the problem and correct them.

Al-Fuqaha et al [15] present a system called CHARS: Call Home Analysis and Response System. CHARS receives a stream of events, which summarize the state of Software Based Services and Network Elements, and also performs tasks which traditionally were performed by System Operators. CHARS has the ability of correlate the messages produced by services and elements, to detect the root cause of events and issues, and associate them with a solution procedure. This system has the ability of defining relationships and dependencies among service components, and allows to human operators bring knowledge into te system by defined scripts which can react to

problems, modifying the configuration of system elements. CHARS has as its core a Rule-Based Expert System.

The main design goals of this system are:

- Intelligence: By using Expert Systems, CHARS has the possibility of store and forward chaining the documention and solutions which were previously inserted by human experts, or deduced when solving a particular problem.
- Reactiveness: By using scripts which can interact with network entities, CHARS has the capacity of isolating outages which are detected by forward chaining.
- Scalability: As the knowledge of the system is stored in the database of the Expert system, bring new external knowledge into the system is not a hard issue.
- Extensibility: By using technologies as XML, extending the components of the system, for example the rules engine, is quite easy to perform.

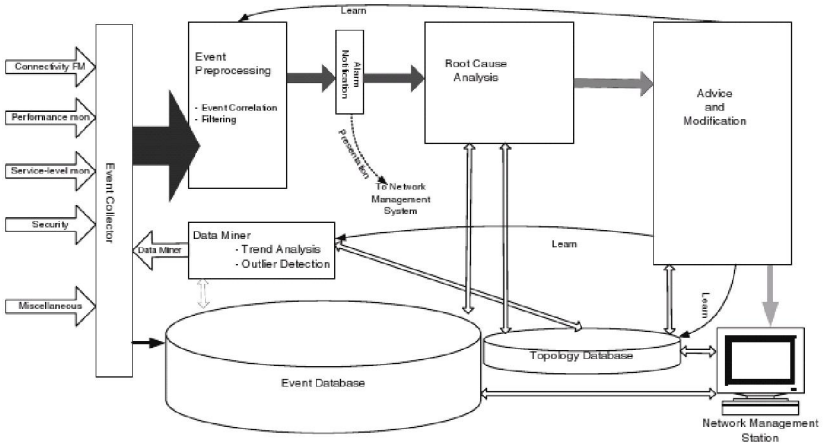

Figure 7. Event Processing Module [17]

Alam and Deters [16] present and architecture for FM in Web Services (WS). FM in web services is a very important topic, as WS have become in the recent years a de-facto standard for building distributed systems, and for integration among heterogenous platforms and environments. Nowadays, WS can be found in very active environments as Banking, E-commerce, Database Management; that causes a high dependence on this kind of entities, for many internet based services and applications. Because of the high distribution and dispersion which is typical in WS environments, traditional monitoring tools are not suitable; for this case, active monitoring might not be meaningful, as components which conform a service might be seen healthy when responsing to probes, but they might exhibit degradation and poor performance. Also, because WS mask and hide complex interactions between software and hardware which implement services, monitoring is not just checking availability of individual components but the right functioning and performance of these related entities. The approach presented, which is based on event storage and analysis and the distributed nature of these two components showed a good behavior in the monitoring of distributed components.. It is composed by following components:

- Enterprise Service Bus (ESB)
- Event Databa se (ED)
- Event File Generator (EFG)
- Event Object Generator (EOG)
- Event Router (ER)
- Event Processor (EP)

Varga and Moldovan [17] propose an Integrated Service Level Monitoring and Fault Management (ISF) framework. This framework offers a solution for service-level monitoring, with the goal of offering end-to-end Quality of Service. This framework is applied in managing several Local Area Networks, thus showing its applicability in multiprovider environments.

A system implemented following this framework would be integrated by the following elements:

- Event collector: It is an entity where the messages sent by all the nodes and elements in the system send their messages. At this element, they are normalized, stored in the event database, and sent to preprocessing module.
- Data Miner: It is an entity on charge of analyze the event database. It applies analysis algorithms to detect non matching patterns, and in case they are found, alarms are sent to event collector, because they might indicate possible problems in system.
- Event Processing: It is integrated by two submodules, correlator, which tries to find correlations among event messages, and filter, which eliminates unnecesary events.
- Alarm Presentation: It is the entity which displays the alarms, and triggers a new Root Cause Analysis for each one.

- Root Cause Analysis: It is an entity which using the descriptive information on the alarm, tries to find a Root Cause for it.
- Advice and Notification: It produces fault descriptions after the Root Cause Analysis. It uses a database to compare the descriptive information and provide possible corrective actions to be taken. This module could just notify, or take directly the actions on the network elements.
- Event Database: It stores the information of the collector, and is used by the data miner for the analysis.
- Topology Database: It stores information of the real network structure. It contains node addresses, node functionalities and connections among nodes.

In particular, for the Root Cause Analysis, this approach uses Petri-Nets as analysis technique as mentioned in [18]. Figure 7 shows the event processing module, which is the core and most important component of the proposed framework.

Bhattacharyya et al [19] present a modelling scheme for fault management by means of Discrete Event Systems, using Finite State Machines and their state transations to identify faulty states. They define three classes of faults: Input Faults, related to invalid or unexpected inputs coming into the system, Transition Faults, when system passes to an invalid state or to an state not matching the received input, and Output Faults, which is the case when transition is performed as expected, but for any cause, system produced an invalid output. In this work, and according to modelling used, the failures related to output could be detected and diagnosed, but input and transition could not. This work also introduces a method which allow to build a global diagnoser to be used across the whole system, by extending the idea of local diagnosers.

Sterrit et al [20] introduce the concept of Autonomy in event correlation. Starting from the ideas formulated by Horn [21], authors present an approach where the goal is avoiding failures at system level, by performing self-configuration to ensure minimal disruption. This approach is composed by three main areas:

- Correlation: Consists in processin information which comes from several sources, and trying to find relationships among the information pieces, which be useful to decide when and what to do.
- Rule discovery: The information sent by network entities is just symptoms. A further processing needs to be performed according to rules guiding what events correlate, and how to process the information within them. Techniques as machine learning or data mining might be useful at this point [22], although in most of cases the most useful knowledge will be the one which could be provided by a human operator [23].
- Autonomic Computing Correlator Analysis Tools: It is the mechanism which perform the analysis to correlate the events reported, and try to find the root cause of the problem or failure reported. It implements the self-diagnosis, and triggers the self-healing on system components.

Figure 8 illustrates the high level design of this last component, which is the core of the approach presented in this work.

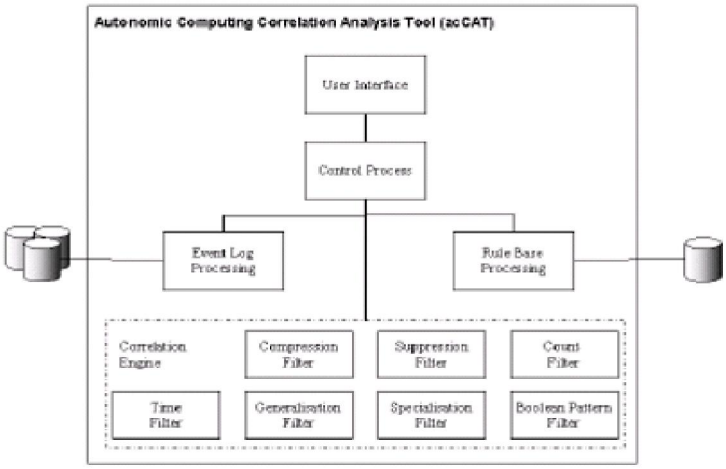

Figure 8. High level design of Autonomic Computing Correlator Tool [20]

Sterrit [24] describes in detail the process of rule discovery by means of acquisition of knowledge from events captured in network and from human expert knowledge, and later applying data mining to this data to obtain new rules for event and alarm correlation. This work opens a novel research topic, as the knowledge acquisition is vital to build monitor and fault management systems which really adapt to changing network conditions and be able to work in real time.

Sterrit et al [25] present HACKER, which is a tool which integrates visualization and data mining, incorporating principles of Computer and Human Discovery. The tool is designed to assist in discovering unknown rules from events in such a way that these new rules can be used to feed systems of event correlation based in rules. This process of knowledge discovery is performed in three tiers: Visualization Correlation, Knowledge

acquisition (Rule based correlation) and knowledge discovery (Data Mining Correlation). Figure 9 shows a diagram illustrating the three tier knowledge discovery process.

Miller and Arisha [26] present a model based on Communicating Finite State Machines (FSM), which is a solution for monitoring and fault management with passive monitoring. In this model, authors represent every node in network by means of a FSM, with communicating channels among them. In an alternative way to common communication protocols, based on send/receive, the communication model employed by these FSM uses input/output labeling on transitions. Also for faults, model specifies three kind of failures: related to output, related to input and related to communication channels themselves, and uses two assumptions related failures: They occur one at time, and they are persistent for nodes and non-persistent for communication channels. This scheme considers that faulty component is the one who exhibits a different behavior on its FSM which differ from specification FSM. Authors finally apply this model to a mobile IPv6 network as experiment, and show a very good behavior in fault detection.

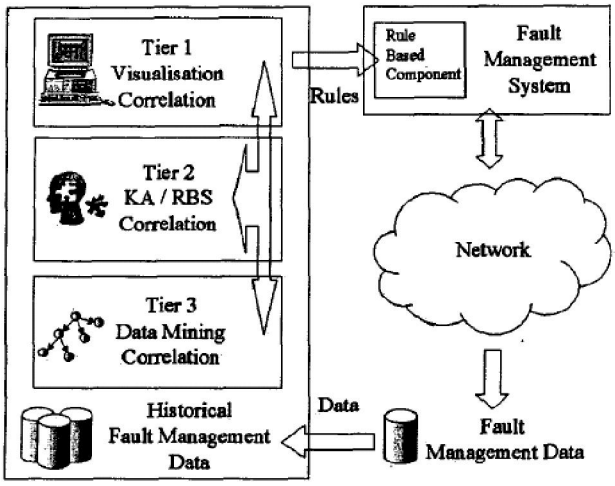

Figure 9. The three tier knowledge discovery process [25]

Sabin et al [27] analyze an approach to Fault Management in Groupware Services by means of a Constraint Satisfaction Problem. They define the information gathering for these services, by means of evaluation of current system state, to see whether an inconsistence is presented against a given reference state. The system is capable to perform both predictive and reactive diagnosis.

Gürer et al [28] present a review of several Artificial Intelligence Techniques for alarm correlation. In first hand, authors present the general limitations of AI based techniques in particular, those based on Expert Systems:

- They are not designed to handle dynamic knowledge; because of this, they might not be accurate when facing unforeseen situations.
- They have limitations to learn from experience. Rules defined at develop time might not adapt well to network evolution.
- They do not scale well. Add new rules requires knowledges in AI to value the meaning of a new rule, and the impact it might cause in rule base.
- They require high maintenance. When there are changes within the application domain, new rules have to be added, and old rules should be eliminated or adapted.
- They are not suitable to handle probability or uncertainty. The rule base requires high previous knowledge of the domain, as previously mentioned, they have high limitations to adapt to changes.
- They are not suitable to handle high volumes of data. Expert Systems are not designed to process high rates of data, either real time processing.

Other AI technique explored by authors is Neural Networks (NN). According to authors, NN are quite adequated for Event Correlation, because of their processing speed, their ability to handle incomplete information and their learning capacity.

Bayesian Belief Networks (BBN) is another technique which is explored by authors. This exhibit good results in event correlation, because of its ability to model dependencies and thus, cause and effect relationships between events; they can also help in diagnosis, can handle noisy environments, and have a compact and well-defined problem space.

The authors finally propose a hybrid system, which combine several techniques, and show to be very successful in event correlation.

This work has been analyzed although is quite old, but the survey it presents of AI techniques to be used in event correlation, passive system monitoring.

D. Agent Based

Traditionally, the most used computer paradigm for many kind of applications has been the Client/Server (CS) model. In this model, there is a main processing entity named server, which offers services to usually many clients. Network administration has mainly used this approach in its architectures.

There is a Management Station or Management Server which access information in Managed Nodes, which are the resources around the network.

In general, in computer applications, and in particular in Network Management, CS model is suitable for applications

which have not high needs of distributed control [29], and because of this centralized nature, scalability and reliability issues arise, and limit the capacity to be used in modern environments, which are by nature highly distributed and sparsed. Taken from Artificial Intelligence domain, the concept of Mobile Agents [30] appears as a solution which could be used to address the issues presented by CS model, by allowing

a high degree of distribution in the processing task.

Al-Kasassbeh and Adda [31] present a detailed analysis where they compare the performance of two approaches to Network Management: Client/Server and Multi Agent based They analyze the performace of both approaches using as metric the traffic generated by the management process itself, and the time to execute the management tasks.

In the experiments, authors show that MultiAgent approach exhibits a noticeable reduction in network traffic, and a reduction of execution time, compared to Client/Server. Multi Agent approach also shows an almost constant behaviour regarding resources consumption, although network nodes increase. Client/Server presents a resource consumption which is directly proportional to number of managed nodes.

III. CONCLUSIONS

This work presents a preliminary revision of techniques which are used in system monitoring as main component in Fault Management. Because of the highly particular nature of Fault Management analysis of other techniques and technologies stay open, according to profundization within a particular domain of application.

It also can be seen that autonomic computing is becoming a desirable approach for monitoring and fault management, because of the agile utilization which can be performed of the human knowledge when it is combined an integrated to computer knowledge discovery to gain faster and more accurate diagnosis and interventions on system components.

REFERENCES

[1] T. Fujisaki, M. Hamada, and K. Kageyama, 1997. A scalable fault-tolerant network management system built using distributed object technology, in Proc. First International Enterprise Distributed Object Computing Workshop.

[2] W. Mei-hui, L. Chuan, M. Zhao, and Z. Dong, 2010. A discussion of using self-maintenance technology to achieve high reliability of equipment, in Prognostics and Health Management Conference, 2010. PHM '10.

[3] J. Gao, B. Ma, and Z. J., 2006. Research on fault self recovery engineering, Dalian University of Technology, vol. 46.

[4] M. N. Yuniarto and A. W. Labib, 2006. Fuzzy adaptive preventive maintenance in a manufacturing control system: a step towards self-maintenance, International Journal of Production Research, vol. 44.

[5] A. W. Labib, 2006. Next generation maintenance systems: Towards the design of a self-maintenance machine, in IEEE International Conference on Industrial Informatics.

[6] S. Sirvunnabood and A. Labib, 2005. Towards the development of self-maintenance machines, in Proc. of the 2005 Third European Conference on Intelligent Management Systems in Operations. 28th and 29th June. Greater Manchester U.K.

[7] Y. Umeda, Y. Tomiyama, and H. Yoshikawa, 1995. A design methodology for self-maintenance machines. ASME Journal of Mechanical Design, vol. 177.

[8] A. Mohamed and O. Basir, 2009. A new probing scheme for fault detection and identification, in Proceedings of IEEE International Conference on Electro/Information Technology, 2009. EIT '09.

[9] C. Li, S. Zou, and C. Lingwei, 2009. Online learning based internet service fault diagnosis using active probing, in Proc. IEEE International Conference on Networking, Sensing and Control, Okayama, Japan.

[10] L. W. Chu, S. H. Zou, S. D. Cheng, W. D. Wang, and C. Q. Tian, 2009. Internet service fault management using active probing in uncertain and noisy environment," in Proc. Fourth International Conference on Communications and Networking in China, ChinaCOM 2009.

[11] Z. Chen, 2005. Proactive probing and probing on demand in service fault localization. The international journal of Intelligence control and systems., vol. 2, pp. 107–113.

[12] ——, 2006. Service fault localization using probing technology, in Proceedings of the 2006 IEEE International Conference on Networking, Sensing and Control, 2006. ICNSC '06.

[13] E. Kirmani and C. Hood, 2004. Diagnosing network states through intelligent probing, in Network Operations and Management Symposium, 2004. NOMS 2004. IEEE/IFIP.

[14] M. Brodie, I. Rish, and S.Ma, 2002. Intelligent probing: a cost-effective approach to fault diagnosis in computer networks, IBM Systems Journal, vol. 41, pp. 372–385.

[15] A. Al-Fuqaha, A. Rayes, M. Guizani, M. Khanvilkar, and M. Ahmed., 2009. Intelligent service monitoring and support, in Proc. IEEE International Conference on Communications, 2009. ICC '09.

[16] S. Alam and R. Deters, 2009. Distributed event processing for fault management of web services, in IEEE Asia-Pacific Services Computing Conference, 2009. APSCC 2009.

[17] P. Varga and I. Moldovan, 2007. Integration of service-level monitoring with fault management for end-to-end multi-provider ethernet services, IEEE Transacions on Network and Service Management, vol. 4.

[18] A. Aghasaryan, E. Fabre, A. Benveniste, R. Boubour, and C. Jard, 1997. A petri net approach to fault detection and diagnosis in distributed systems, in Proc. 36th IEEE Conference on Decision and Control, IEEE (CDC 97).

[19] S. Bhattacharyya, Z. Huang, V. Chandra, and R. Kumar, 2004. A discrete event systems approach to network fault management: detection & diagnosis of faults, in Proceedings of the 2004 American Control Conference.

[20] R. Sterritt, D. Bustard, and A. McCrea, 2003. Autonomic computing correlation for fault management system evolution, in IEEE international conference industrial informatics.

[21] P. Horn, 2001. Autonomic computing: Ibm's perspective on the state of information technology, IBM Corp, Tech. Rep.

[22] R. Sterritt and D. Bustard, 2002. Fusing hard and soft computing for fault management in telecommunications systems," in IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews.

[23] R. Uthurusamy, 1996. From data mining to knowledge discovery: Current challenges and future directions, Advances in knowledge discovery and Data Mining, pp. 561–569.

[24] R. Sterritt, 2001. Discovering rules for fault management, in Proc.

Eighth Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems, 2001. ECBS 2001.

[25] R. Sterritt, E. Curran, and H. Song, 2002. Hacker: human and computer knowledge discovered event rules for telecommunications fault management, in IEEE International Conference on Systems, Man and Cybernetics.

[26] R. Miller and K. Arisha, 2001. On fault management using passive testing for mobile IPv6 networks, in IEEE Global Telecommunications Conference, 2001. GLOBECOM '01.

[27] M. Sabin, A. Bakman, E. Freuder, and R. Russell, 1999. A constraint-based approach to fault management for groupware services, in Proc of the Sixth IFIP/IEEE International Symposium on Integrated Network Management, 1999. Distributed Management for the Networked Millennium.

[28] D. Gurer, I. Khan, and R. Ogier, 1996. An artificial intelligence approach to network fault management. SRI International, Tech. Rep.

[29] S. Pleisch and A. Schiper, 2004. Approaches to fault-tolerant and transactional mobile agent execution: An algorithmic view." ACM Computer Surveys, vol. 36, pp. 219–262.

[30] N. Nikaein, 1999. Reactive autonomous mobile agent, Master's thesis, Sophia Antipolis.

[31] M. Al-Kasassbeh and M. Adda, 2008. Analysis of mobile agents in network fault management, Journal of Network and Computer Applications, vol. 31, pp. 699–711.