Autonomic networks: engineering the self-healing property

Autonomic Networks:

Engineering the Self-Healing Property

Roy Sterritt

School of Computing and Mathematics,

Faculty of Engineering, University of Ulster,

Northern Ireland.

E-mail: R.Sterritt@ulster.ac.uk

Abstract

As the size and complexity of networks and communications continue to grow, there is a heightened need to develop new techniques capable of achieving a level of service with successful operations upon which users can place even more reliance. Key emerging strategies for meeting this demand is 'autonomic networks' and 'autonomic communications', concepts similar to autonomic computing while specific to the communications field. This paper considers the 'self-healing' aspect of autonomic networks, examining, in particular, techniques for event correlation to aid fault identification. A three-tier rule-discovery framework and associated support and analysis tools are described. These assist with the development, management and maintenance of correlation rules and beliefs.

1. Introduction

Autonomic computing is rapidly becoming established as a significant strategic approach to the design of more reliable, easier-to-manage computer based systems. When launching

the autonomic computing initiative, IBM highlighted the growing complexity crisis in the IT industry, comparing it with telephony in the 1920s. There, the rapid increase in use of the telephone led to estimates that by the 1980s half of the population of the USA would have to be employed as telephone operators to meet the demand (Horn 2001). The implementation of automated switching and other technological developments avoided this crisis. By analogy, IBM is expecting autonomic system implementations to achieve similar productivity gains. It is anticipated, however, that significant research and development will be required to achieve that goal.

The envisaged goal of autonomic computing is the production of systems that are self-managing in four main respects: *self-configuring*, *self-healing*, *self-protecting* and *self-optimizing*. Some of the prerequisites for autonomic computing include complete visibility of the managed platform, complete control of that platform without undesirable side effects, and complete knowledge of how to relate visible situations to concrete actions. Most importantly is the ability to capture and represent both enterprise and personal policy (rules). Because of the need for differing levels of human involvement, autonomic computing maturity and sophistication has been categorized into five "stages of adoption" (Bantz and Frank, 2003, IBM, 2003): *Basic, Managed, Predictive, Adaptive,* and *Autonomic.* These prerequisites are priorities in the work reported in this paper while evolving along the autonomic computing maturity stages.

There are two strategies for introducing autonomic behavior. The first is to engineer it into systems and the second is to achieve it through adaptive learning. The first approach can be progressed immediately, with human experts generating or overseeing the generation

of rules for autonomic functions. Over time, this could be increasingly supplemented with self-learning processes (Sterritt, 2002).

During 2003 a research study took place with a large telecommunications company on Autonomic Computing and Telecommunications (Sterritt, 2003), the findings confirm and motivate the research in this paper, which is concerned specifically with the self-healing and problem determination aspects of autonomic communications, with particular focus on the analysis of alarm events in the telecommunication distributed systems. After a discussion of the autonomic concepts, the paper presents an overview of problem determination through alarm event correlation in fault management and the three-tier process for correlation rule discovery which may be used to assist in engineering autonomic capability into the systems. This extends earlier work in this area (Sterritt and Bustard, 2002b) with two support tools: (i) HACKER to support the interactive identification and documentation of correlation rules; and (ii) acCAT to test potential rules.

The Autonomic initiative is about much more than faults and self-healing, yet it is a critical area to address considering that it has been estimated that companies spend 33% to 50% of their total cost of ownership recovering from or preparing against failures (Patterson et al, 2002).

2. Towards Autonomic Networks

Since the 1920's, automation in telephony has evolved substantially. The Internet, with its vast infrastructure supporting millions of interconnected computers is perhaps the most significant development. The complexity of networks has grown in various ways (Oates, 1995). As user demands and expectations become more varied and complex so too do the

networks themselves. Data, voice, image, and other information now travels under the control of different protocols through numerous physical devices manufactured and operated by different vendors. It is expected that the trend towards increasing complexity will continue, due to several factors such as the increasing complexity of individual network elements, the need for sophisticated network and communication services and the heterogeneity of connected equipment (Cheikhrouhou et al, 1999). The promise of Autonomic Networks, networks that manage themselves, will substantially abate this complexity crisis.

2.1. Telecommunication Networks

Telecommunication networks are designed to be robust as it is simply not acceptable for millions of calls/connections to be cut-off due to a faulty network element or a software upgrade. This leads to design approaches that incorporate back-up mechanisms that allow for recovery from certain classes of fault. One technique, for example, is the use of a ring topology for node connection as in the Survivable Network Architectures (SNA) illustrated in Figure 1. In SDH/Sonet systems, traffic travels in both directions. Any fault occurring that prevents progress in one direction will cause an automatic switch in traffic direction to avoid the failure area, thus sustaining traffic throughput. This fits with the autonomic goal that there should be no failure at the system level. Components of the system will fail but self-configuration is used to ensure minimal disruption (Ganek, 2003).

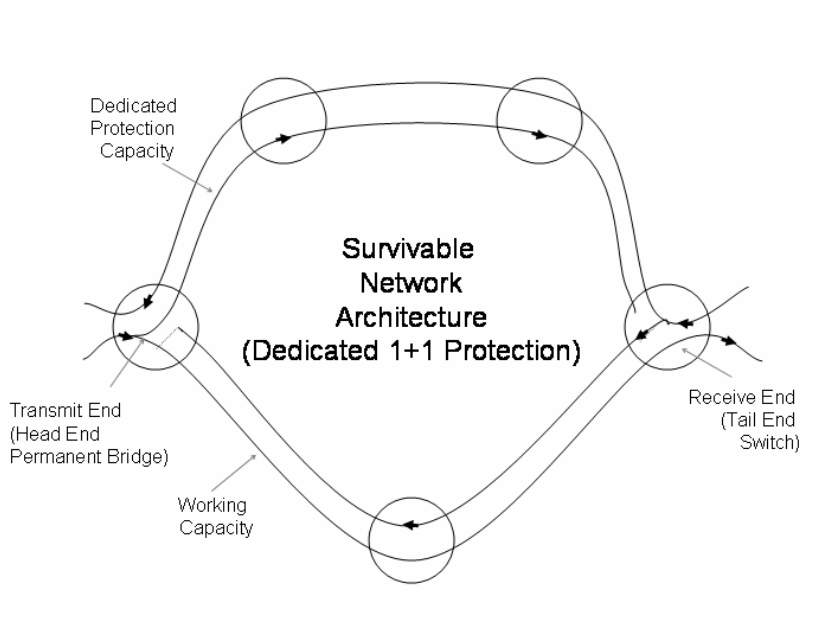


Figure 1 Survivable Network Architectures (SNA)

For major hub traffic applications, survivability tends to be implemented through an additional dedicated protection ring (Figure 1). In metropolitan, junction and trunk network applications this robustness may be achieved through the less expensive option of a shared protection ring, which reserves protection capacity in the existing ring in case of failure.

Robustness, in general, is achieved through redundancy in the hardware and software components of the network. Unfortunately this can increase complexity even further, made worse by facilitating (legacy) non-synchronous traffic to co-exist with synchronous traffic.

2.2. Autonomic Networks Scope

The preceding presentation of the SNA demonstrated some autonomic behavior in the physical layer of the telecommunications networks, yet this is just the beginning of the autonomic vision; zero touch, self-sensing, context-aware, dynamic, self-programming and evolvable networks. To create Autonomic Networks will require the co-operation of the

industry to develop open standards to evolve from the current network elements (NEs) to autonomic network elements (ANEs). From a Telco's perspective the physical layer tends to be outside their immediate design control as the NEs are supplied by third party vendors.

Telco's offer communications and services across a large variety of technologies. Each technology within the network; SDH (SONET in USA), PDH, ATM, IP and so on, all have their own specific domain technology fault managers. SDH frames may be carrying ATM frames which may be carrying IP and so on. As such at the physical layer Autonomic Networks may resolve their own management issues, but these may have affected the traffic/service they are carrying. This can only be determined at a higher layer.

Essentially due to the complexity the situation has arisen that a large number of uncorrelated alarm event messages may reside on a network at any one time. One estimate concerning BT's UK network was that 95% of all alarm events raised remain uncorrelated, amounting to tens of thousands alarm events being active at any one time. Over time this amounts to a substantial load of data. Another concern that is these problems with root cause analysis are preventing the development of further autonomics particularly in self-healing and with increasing mean-time to human intervention.

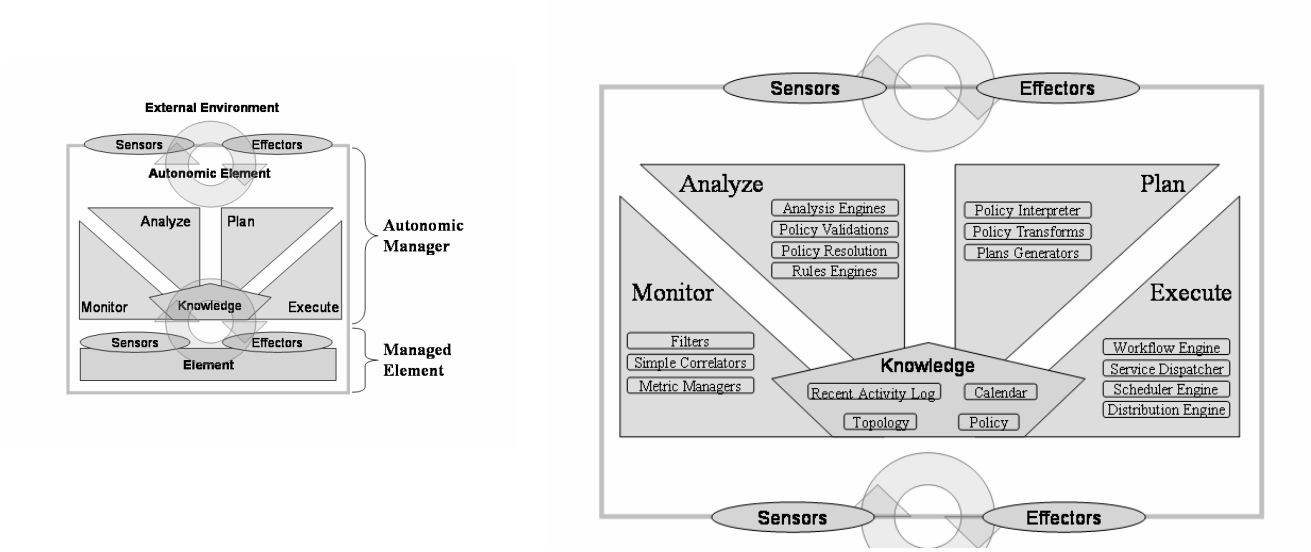
Autonomic Networks in themselves will not be an easy goal to achieve, yet the longer term goal of Autonomic Communications is much more than this, having commonality with Ubiquitous and Pervasive Computing, a vision of communications services anytime, anyplace from any device adapting to the users current needs and situation. Effective problem determination in the networks will assist in enabling other autonomics to advance.

2.3. Autonomic Architecture

The basic building blocks of any autonomic system architecture include *sensors* and *effectors* (Ganek and Corbi, 2003). By monitoring behavior through sensors, comparing this with expectations (historical and current data, rules and beliefs), planning what action is necessary (if any) and then executing that action through effectors, creates a control loop (IBM, 2001). The control loop, a success of manufacturing science for many years, provides the basic backbone structure for each system component (Ganek, 2003).

Figure 2 is IBM's view of the necessary components within an autonomic manager. (For an alternative artifacts view, see Sterritt and Bustard, 2003). It is assumed that an autonomic manager is responsible for a managed element within a self-contained autonomic element. Interaction will occur with remote autonomic managers through virtual, peer-to-peer, client-server (Bantz et al, 2003) or grid (Dean et al 2003) configurations.

The monitor and analyze parts of the structure process information from the sensors to provide both self-awareness and an awareness of the external environment. The plan and execute parts decide on the necessary self-management behavior that will be executed through the effectors. The *simple correlator* in the monitor parts and the *rules engine* in the analyze part use correlations, rules, beliefs, expectations, histories and other information known to the autonomic element, or available to it.



(a) General concept of an Autonomic Element

(b) Necessary Components within the Autonomic Manager

Figure 2 IBM's view of the Architecture of an Autonomic Element

3. Problem Determination

3.1 Event Correlation

The introduction of autonomic principles requires the monitoring of individual system components through sensors and the ability of those components to respond to requests through effectors. Monitoring will typically involve the *correlation* of several related pieces of information (Figure 2). Correlation is important in both self-assessment (self-awareness) and in the assessment of a component's operating environment (environment awareness). This helps in deciding when action is required and what should be done.

By analogy with the human autonomic nervous system event messages are similar to the electric pulses that travel along nerves. When a fault occurs in an SDH network a series of triggered events are usually reported to the element controller (manager). The behavior of the alarms is often so complex it appears non-deterministic (Bouloutas et al, 1994), making

it very difficult to isolate the true cause of the fault (Klemettinen, 1999). Yet at this level this is one of the primary goals of Autonomic Networks.

Problems and failures in the network are unavoidable but quick detection and identification of their source is essential to ensure robustness. The correlation of alarm event messages is an important part of this analysis (Jackobson and Weissman, 1993). The major telecommunication equipment manufacturers deal with event correlation through alarm monitoring, filtering and masking as specified by ITU-T (ITU-T, 2000) and other international standard bodies. Resulting rule type diagnostic systems provide assistance to the operator whose expertise is then used to determine the underlying fault (or faults) from the filtered set of alarms reported.

Currently, the skill of the operator is central to identifying faults. So although automation prevents the immediate loss of traffic and preserves the general function of the system (as in the SNA), intervention is necessary to determine and resolve problems that arise. The promise of autonomic networks would bring about a significant reduction in the role of the operator.

Event correlation is a conceptual interpretation of multiple events, giving them a collective meaning. This produces a new higher-order compound event that helps determine what action is required. Jakobson and Weissman (1993) describe correlation as a generic process involving six operations: *Compression, Suppression, Count, Boolean Pattern, Generalization,* and *Specialization.*

The principle aim of event correlation is the interpretation of the events involved. The event signals or messages represent *symptoms*. Rules and beliefs identify which events to correlate and how they should be transformed. These tend to vary over time creating a

significant maintenance burden (Bratko and Muggleton, 1995). Machine learning, data mining and other AI techniques can assist in the discovery of correlation rules and beliefs (Sterritt, 2002a, Sterritt and Bustard, 2002a). However, a human-centred discovery process is more effective than either a human or computer operating independently (Uthurusamy, 1996). For example, it is useful to provide various visualizations of data throughout the knowledge discovery process to build user trust in the process and hence instill more confidence in the mined patterns. The transformation from data to knowledge requires interpretation and evaluation, which can also benefit from visualization of the processes involved. Visualization techniques can make use of the highly tuned perceptual abilities that humans possess, such as a capacity to recognize images quickly and to detect the subtlest changes in size, color, shape, movement or texture. Any patterns that emerge may indicate the presence of potential for new rules. Human interpretation is then required to transform them into 'knowledge'. Human input typically produces more meaningful insights into the discovered correlations, enabling them to be coded as useful rules for fault identification and management.

The next section describes a framework and support tools to assist such rule discovery.

3.2 Event Correlation Framework and Tools

A three-tier architecture model for rule discovery is shown in Figure 3. This extends earlier work described in (Sterritt, 2001, Sterritt and Bustard, 2002b). The main difference is enhanced management control and co-ordination across the three tiers, achieved through the newly created HACKER and acCAT tools. It also makes explicit a recommendation for extending tier 2 activities from the development phase into the operational phase by using

knowledge management techniques to capture operators' manual live correlations of alarms, bring this knowledge into the development lifecycle and test to see if the rules are of general use.

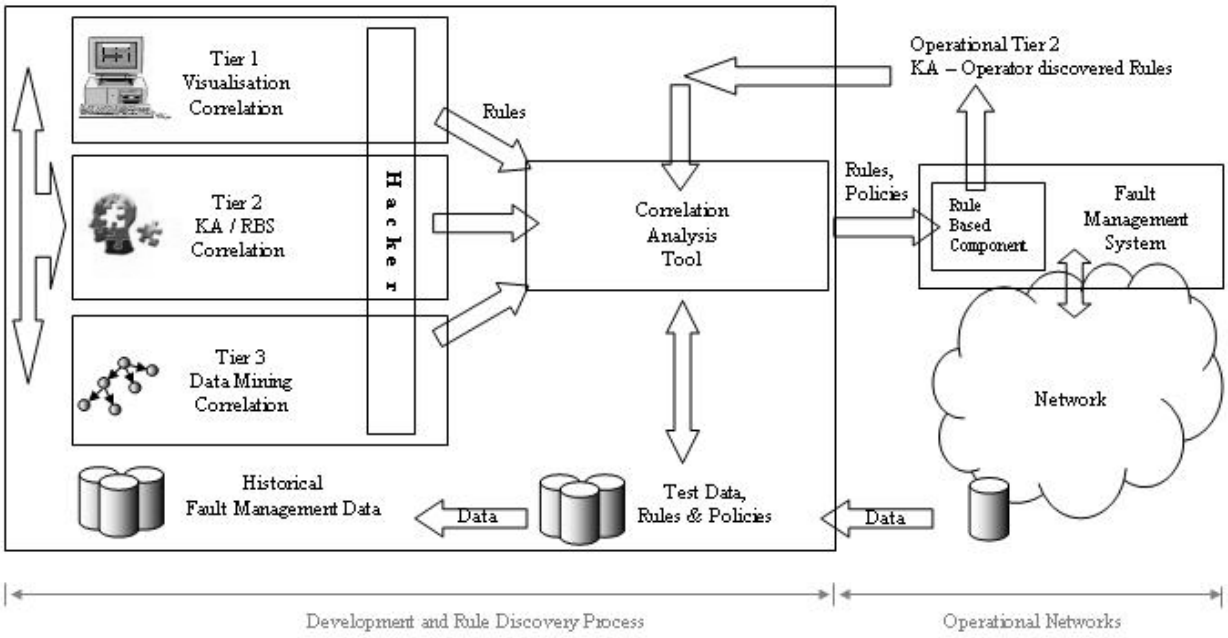


Figure 3 Three-tier alarm event correlation rule discovery process

The right-hand side of the diagram represents the managed operational network and the left-hand side the discovery or learning process. Data flows from the network to the discovery process; while rules flow from the discovery process to the network manager. The representation suggests a cycle of activity, reflecting the necessary review that must take place after changes have been made to the network. Computer-assisted human discovery and human-assisted computer discovery techniques can be integrated in the three-tier framework for the discovery of alarm correlations to support the deduction of fault management rules. The responsibility of the tiers is as follows:

Tier 1. *Visualization Correlation* (Computer-aided, human discovery). New alarm correlations are discovered from visualizing the fault management data.

Tier 2. *Knowledge Acquisition or Rule Based Correlation*. New alarm correlations are discovered through consultation with experts and analysis of documentation. Correlations from tiers 1 or 3 may also be validated in this tier.

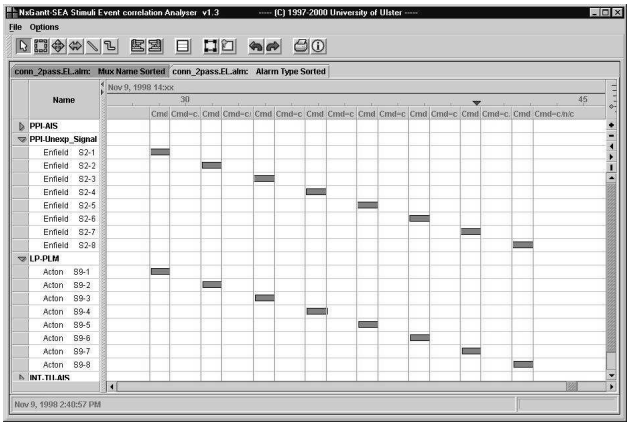
Tier 3. *Data Mining Correlation* (Human-aided, computer discovery). New alarm correlations are revealed by mining the fault alarm data.

New rules may emerge from any of these tiers. The first tier, visualization correlation, supports the visualization of data in several forms. Visualization has a significant role throughout the knowledge discovery process, from data cleaning to mining. In particular, it facilitates the analysis of data to help identify alarm correlations (knowledge capture). The second tier aims to identify correlations and rules using more traditional knowledge acquisition techniques, with experts and documentation. At the same time it has a supporting role to confirm that discoveries from tiers 1 and 3 are indeed new and useful information. The third tier mines the telecommunications management network data to produce more complex correlation candidates.

The rest of this section describes two new prototype tools, HACKER and acCAT, which assist knowledge discovery, integrating analysis activity across all three tiers of the discovery framework. Previously (Sterritt and Bustard, 2002b), there was no automated linkage across the tiers. This meant that it was difficult to appreciate if the same rules were being discovered in different tiers or indeed if contradictions were present. Such issues were left to the human expert to resolve. Another advantage of the cross-tier support is that it enables potential rules emerging from any tier to be tested against other data sets before adding them to the rule base.

3.2.1. HACKER Tool

The HACKER tool supports (i) the human visual discovery of rules; (ii) knowledge capture from the experts explaining the discovered rules; and (iii) visual presentation of data mined rules. The starting point for tool development was an existing visualization tool (Sterritt et al, 2000), *NxGantt-SEA* (SEA: Stimuli-Event correlation Analyzer). Its role was within tier 1 to present alarm and other event data (found in element controller event logs) in a visualized tabular form (Figure 4). It uses a Gantt chart to represent the life span of each alarm. The tool can reveal patterns of occurrence, which suggest possible correlations (as illustrated). The tool was found to be a great improvement on scanning through tens of thousands of events in ASCII text logs in an attempt to gain an indication of a problem.



(a) Tier 1 visualization tool

```
Rule portDisabled {
  when {
    ?x PPI-Unexplained_Signal, …,
    ?y LP-PLM, …,
  } then {
    retract ?x
    retract ?y
    assert portDisabled, …,
  }
}
```

(b) potential rule from discovered correlation

Figure 4 NxGantt-SEA *Stimuli-Event correlation Analyser*

The illustration, Figure 4 (a), highlights a potential (human discovered) alarm correlation from scanning through the event logs using the tool (Sterritt 2001). The two alarms are raised on two different NEs within the same time window (display shows on eight

occasions). From expert knowledge this could be used to manually develop a rule or belief to replace the alarms with the root cause (simplified example in Figure 4 (b)).

NxGantt-HACKER is an extension of this tool into tiers 2 and 3. It may also collect information from the domain expert (tier 2) to explain why a certain correlation should or should not be used to develop a rule. This facility is similar to the 'trouble ticket' approach used in many commercial fault management systems. It semi-automates the rule and case coding, for instance the rule in Figure 4 (b) produced automatically in the correct syntax (e.g. ILog rules), given information found and additional details supplied by the operator. One of the goals is to capture additional information and develop a 'case' as well as a 'rule'. This will facilitate the future automation, for instance the introduction of case-based reasoning, firstly as part of a fault management system using the correlations and expert case knowledge; and secondly in automating more fully the rule discovery process by taking into account what lessons and beliefs can be learnt from the historical information on correlations chosen or rejected by the expert.

Figure 5 shows the two main threads of activity supported by HACKER: (i) automated alarm correlation discovery (data mining); and (ii) manual alarm correlation discovery (visualization). The basic process is one of rule discovery, case description, and rule specification.

It is important in any automated process to avoid placing too much trust in the analysis software. The design of the application thus allows manual adjustment of the produced rules and cases as well as visual feedback on how the candidate correlations have been discovered.

In the example in Figure 4 the visual correlation between alarms PPI-Unexp_Signal (unexplained signal on the PDH physical interface) and LP-PLM (lower order path - path label mismatch) was discovered, Figure 4 (a), and manually coded as a rule, Figure 4 (b). With HACKER, in Figure 9, screenshots (a)-(c) show the manual/human discovery approach that matches the right hand side of the flow in Figure 5. Screenshots (d)-(f) show automated/computer discovery that matches the left hand flow in Figure 5. Screenshot (a) depicts the visualization of data in the standard form used in the original tool (Figure 4), with two alarms highlighted and the '*rule writer'* option selected. Screenshot (b) shows the system requesting (i) a rule name; (ii) a higher order rule name that will be used in the system to represent the set of correlated alarm events; (iii) some explanatory reasoning from the expert on why this should be considered a correlation rule; and (iv) a diagnostic explanation that may be supplied to a user encountering this set of alarms. Screenshot (c) then shows the resulting rule that emerges, in the required format, with the facility to edit or save.

Screenshot (d) shows the (semi) automatic '*find correlations'* option being chosen, with the results from the mining analysis given in (e). Choosing to develop a rule from a discovered correlation will lead to the same position, shown in (b); (f) illustrates the case where a user has decided to ignore the correlation and provides an explanation for this decision. The rule discovery is semi-automatic since not all information is available to construct a complete rule—hence the need for a case dialog.

The mining algorithm searches for instances of alarms occurring at the same time. This is problematic in that often the local clock on a network element will not be synchronized with clocks in other network elements. Relying instead on the network manager's time

stamp does not solve this problem since there is no guarantee that the alarms take the same time to arrive with the manager and so match their order of occurrence. This is handled by using 'time windows' that in effect, events are considered potentially related, if they both fall within the defined time range of the window. This can, however, result in some spurious suggestions for relationships. Rule development will favor combinations that occur frequently, though, ultimately, the decision lies with the domain expert.

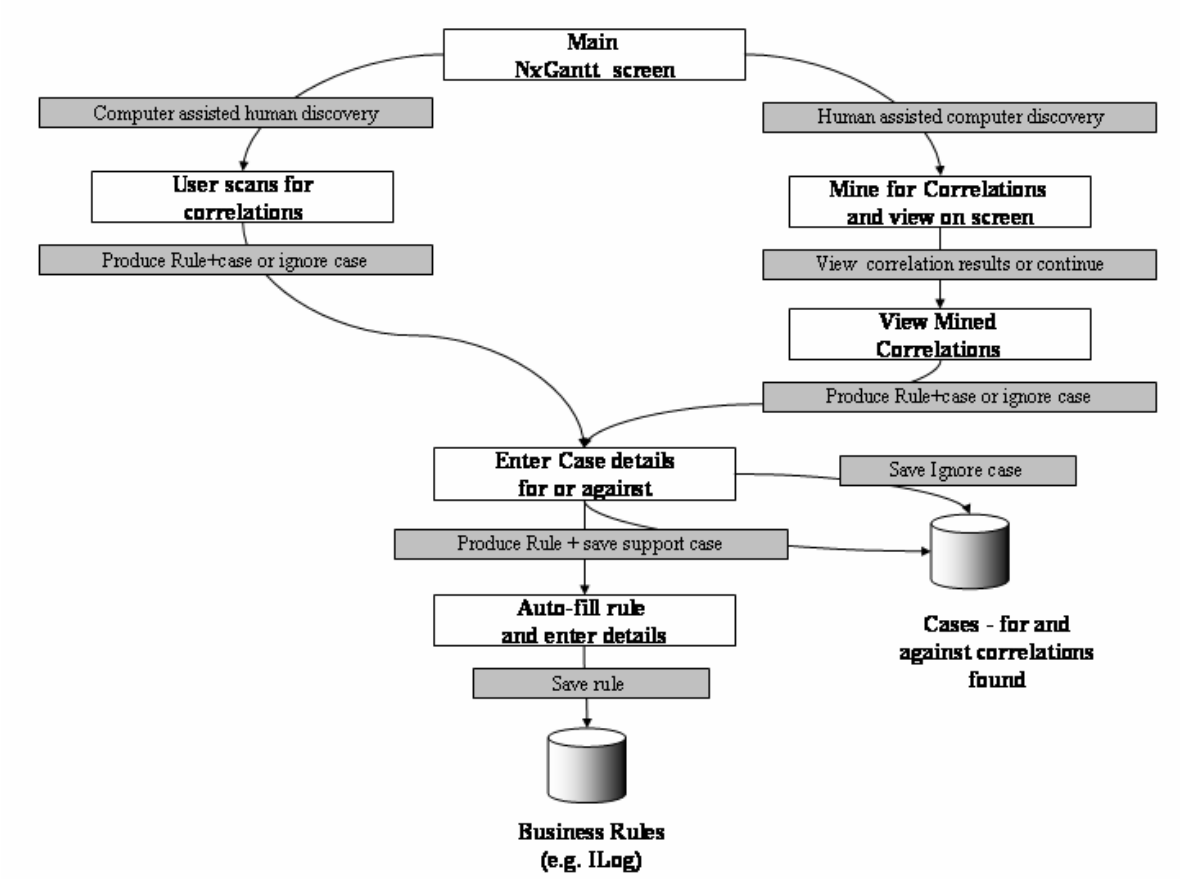


Figure 5 HACKER Computer and Human discovery options

The rule visually discovered in Figure 4, may also be discovered by a mining algorithm in tier 3. HACKER's incorporation of all three tiers allows instances of the mined pattern to be visualised in the tool, while also recording expert's details (tier 2) concerning why it is

valid or invalid; the rule may also be automatically generated from the tool in the correct format (e.g. ILog JRules) from this fuller knowledge.

Possible extensions to HACKER include the use of a more sophisticated mining algorithm to better handle the uncertainty in the data and a case-based reasoning capability, taking advantage of the additional rule correlation information now recorded.

3.2.2. Autonomic Computing Correlator Analysis Tool (acCAT)

The acCAT prototype is an interactive tool to test and execute discovered correlation rules, discovered in any tier, using the six transformation rules identified in the previous section: *compression*, *suppression*, *count*, *Boolean patterns*, *generalization*, and *specialization*. As such the rule discovered in Figure 4 may be tested using acCAT against large sets of historical event logs to confirm if the rule holds while also identifying other possible problem scenarios that may conflict.

Figure 6 shows the high-level structure of the tool. The inference engine comprises:

1. The *user interface*—through which the user is able to influence the analysis strategy.
2. The *control process*—which controls the sequencing of the strategy and the components which perform the control.
3. The *correlation engine*—which contains the lower level components for performing the correlation.

Correlation rules are maintained in the *rule base*. To facilitate the addition of new rules discovered through other tools in the three-tier architecture, XML has been adopted as a standard rule format.

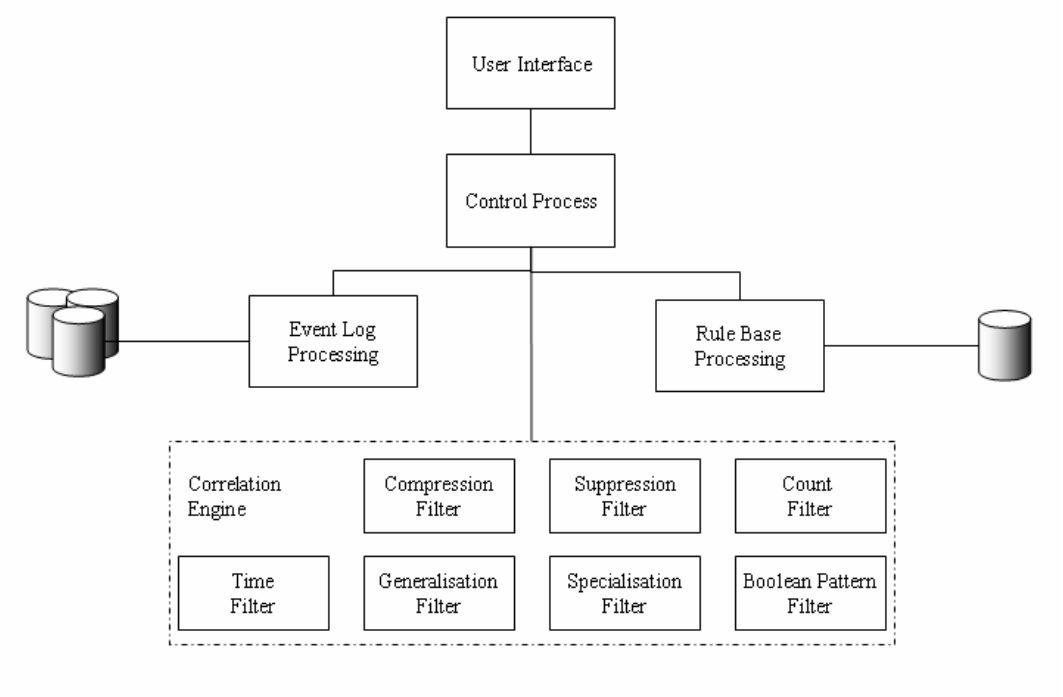


Figure 6 High-Level design architecture for acCAT

The user interface is responsible for managing all interactions with the user. It uses the API provided by the control process to perform all operations and is not directly aware of any of the underlying classes. Screenshot 1 to Screenshot 3 demonstrate some of the functions of the tool. The control process provides methods to access alarms and objects. File processing is conducted from here, and it contains *EventList*, *RuleList* and *CorrelationEngine* objects that control the flow of data among these processes and between the objects and the user interface. These objects contain the 'knowledge' of the system. The *Log Processing* object is responsible for taking in data from the *Event Logs* and creating *Event Objects*.

Rule Base processing, like Log File Processing, is responsible for reading from a file and creating objects—in this case, Rule Objects. As rules can be created, edited and deleted this

component requires full privileges to the Rule Base database. Rule Base processing is also responsible for allowing access to the Rule Objects.

The correlation engine contains all the filters required to perform the correlation. The EventList and RuleList (where required) are passed between these filters resulting in the return of a correlated EventList. The engine contains seven filters, each partially configurable. These filters include a Time Filter and filters for the six generic correlation transformations described above.

Essentially acCAT can take discovered rules from tier 1 (visualization) or tier 3 (mined rules) in XML format and allow a user to experiment by applying the rules to event logs to see the effects of the new correlations (note there are colored visual indicators on the right-hand side of the screenshots to indicate a rule has fired).

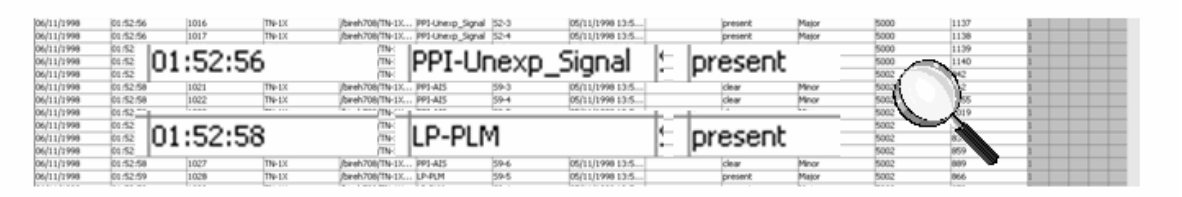


Figure 7 acCAT - testing the previously discovered rule (magnified to illustrate)

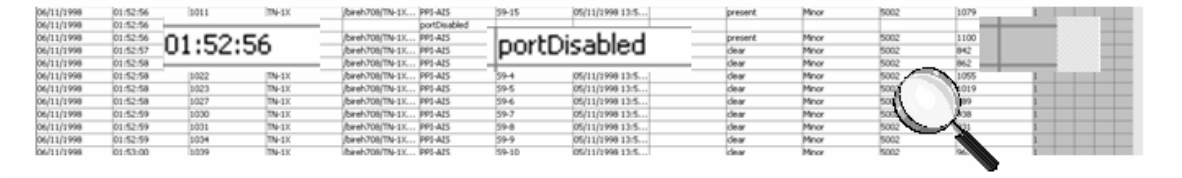


Figure 8 acCAT - rule test and resulting correlation as portDisabled (magnified to illustrate)

Once again taking the discovered correlation coded as a rule; Figure 7 illustrates the two alarms located among the alarm event data in acCAT becoming active (present) within a 2 second time window and Figure 8 illustrates once the rule has been executed.

The tool may assist in debugging as well as managing discovered rules and testing these and existing rules against new network equipment and situations. acCAT can also be used directly by an expert with implicit or tacit network knowledge to develop and experiment with rules.

4. Related Work

This is an inspiring time, whereby autonomic and self managing initiatives are providing a wealth of related and relevant work in this area. These initiatives are also reaching out to existing research areas providing a focal point for cross dissemination and pollination. The key relevant aspects and a comparative analysis with the work presented in this paper are considered.

IBM concurs with this assessment that root cause analysis in complex systems is key to achieving autonomics. In their white paper 'Autonomic problem determination: A first step towards self-healing computing systems' (Oct. 2003) they state that in effect complexity in problem determination is diluting the effectiveness of computing in the corporate environment (IBM, 2003). The same can be said for communications and networks. It has been estimated that companies now spend from one third to one half of their total cost of ownership recovering from or preparing against failures (Patterson et al, 2002). While many of these outages, with some estimates at 40%, are caused by operators themselves (Patterson, 2002).

The IBM white paper highlights the multitude of ways that different parts of a system report events, conditions, errors and alerts as a major factor contributing to the complexity in problem determination. They propose a common format for log/trace information, called

the Common Base Event (CBE) format, to create consistency across systems and ease cross-product problem determination and heterogeneity. The paper also proposes a finite set of canonical situations to categorise the events and reduce the diversity between logs.

The CBE format is accepted as an industry standard and is a significant evolutionary step forward. In terms of the 3-tier framework it would substantially reduce the amount of data cleaning and data pre-processing (including the necessary development of propriety yacc, ack and perl scripts).

Also in October 2003, IBM announced they had entered an agreement with Cisco Systems and released a joint white paper "Adaptive Services Framework". The collaboration is recognition of the synergies between Cisco's next-generation Adaptive Network Care (ANC) solution and IBM's Autonomic Computing initiative. The joint paper (IBM and Cisco Systems, 2003) proposed a set of common interfaces for remote service and support systems. It sets out the stages of adaptive networking in Autonomic Computing as active (connected), reactive, predictive and adaptive. The Adaptive Services Framework has two main goals from the specifications and standardization work; interoperability among different implementations and extensibility.

The CBE format has been included in this initiative. This collaboration should substantially benefit the developments in Autonomic Networks. The greater the problem determination situated in the physical layer, the fewer problems requiring attention in higher layers and as such the lesser the burden on cross-domain problem determination.

In terms of autonomizing legacy systems, agents are being utilised to add capabilities without requiring direct alterations to the legacy code (Haas et al, 2003, Kaiser et al, 2003).

This is occurring in areas such as instant messaging, spam detection, load balancing and middleware (Kaiser et al, 2003).

Several tools with similar general aims to the three-tier rule discovery architecture and acCAT, have recently been released by IBM through their AlphaWorks autonomic zone website (IBM, 2003).

- There is a generic Log and Trace tool (similar in purpose to acCAT) that correlates event logs from legacy systems to identify patterns.
- The Tivoli Autonomic Monitoring Engine essentially provides server level correlation of multiple IT systems to assist with root cause analysis and automated corrective action.
- The ABLE rules engine can be used for more complex analysis. In effect it is an agent building learning environment that includes time series analysis and Bayes classification among others. It correlates events and invokes the necessary action policy.

The three-tier framework is not prescriptive but an approach encouraging the prerequisites previously described; complete visibility, complete control, complete knowledge, capture and represent policy (rules and beliefs). As such the state of the art described in this section are complementary to the three-tier framework approach with potential for integration.

An EU brainstorming workshop in July 2003 discussed novel communication paradigms for 2020 and identified '*Autonomic Communications*' as an important area for future research and development (EU IST FET, 2003, Smirnov and Popescu-Zeletin, 2003). This can be interpreted as further work on non-conventional networking (self-organizing

networks, ad-hoc, sensor, peer-to-peer, group communications, active networks, and so on) but is undoubtedly a reflection of the growing influence of IBM's Autonomic Computing initiative launched in 2001 (Horn, 2001). In effect, autonomic communications (Clark et al, 2003) has the same motivators as the autonomic computing concept with particular focus on the communications research and development community. Hence Autonomic Communications is dependant on a successful Autonomic Networking infrastructure (Agosta and Crosby, 2003).

5. Conclusion

Although commercial networks achieve high reliability (99.999%) (Gray, 2001), their ever-growing size and stringent user demands mean that new techniques are needed to help manage their operation and communications. The autonomic communications paradigm promises to be a useful strategy for meeting these requirements and moving beyond. This will involve moving more decision-making down into the systems to enable them to self-manage their activity, including self-healing.

This paper has considered the general area of problem determination and rule discovery for fault identification in telecommunications systems based on the analysis of alarm events. To help understand the faults underlying the alarms, the alarm events can be reduced through a correlation process involving six standard operations and a developed rule base. A three-tier framework has been outlined to support the identification and recording of rules. This is a collaborative process between domain experts and support software that has data mining capabilities that may be used in autonomic networks

development. Progressively it is hoped to make more and more of the rule identification and verification process automatic, moving everything closer to the autonomic ideal.

Two prototype tools were described: acCAT, which supports rule validation; and HACKER, which aids rule identification and documentation. The information recorded is in a form suitable for case-based reasoning and future work will progress in that direction.

Since the telecommunication domain consists of *systems within systems* many of these systems will be at different maturity levels. The ability to automatically determine the root cause of any event is clearly an enabler to opening new autonomic options that will assist in attaining higher levels of autonomic maturity within the systems. As currently stands the ability to correlate event messages in complex telecommunications lies between managed and predictive. The work described in this paper is an attempt to move to the predictive and adaptive space.

Acknowledgements

The development of this paper was supported by the Centre for Software Process Technologies (CSPT), funded by Invest NI through the Centres of Excellence Programme, under the EU Peace II initiative. The basic research on event correlation was undertaken in the Jigsaw project (ITS Start 187), funded by Nortel Networks' Belfast Labs and the Industrial Research and Technology Unit (IRTU). The extension to autonomic communications was explored through a BT Exact Short Term Research Fellowship (2003). This journal paper expands on concepts first expressed in conference papers (Sterritt et al, 2002, Sterritt et al 2003).

References

Agosta, J.M., Crosby, S., 2003, Network integrity by inference in distributed systems, NIPS 2003 Workshop on Robust Communication Dynamics in Complex Networks, Whistler, Canada, December 12-13.

Bantz, D.F., Bisdikian, C., Challener, D., Karidis, J.P., Mastrianni, S., Mohindra, A., Shea, D.G., Vanover, M., 2003, Autonomic personal computing, IBM Systems Journal, 42 (1), pp 165-176.

Bantz, D.F., Frank, D., 2003, Challenges in Autonomic Personal Computing, with Some New Results in Automatic Configuration Management, Proceedings of IEEE Workshop on Autonomic Computing Principles and Architectures (AUCOPA 2003) at INDIN 2003, Banff, Alberta, Canada, 22-23 August, pp451-456.

Bouloutas, A. T., Calo, S., Finkel, A., 1994, Alarm correlation and fault identification in communication networks, IEEE Trans. on Comms., 42(2), pp 523-533.

Bratko, I., Muggleton, S., 1995, Applications of Inductive Logic Programming, Communications of the ACM, 38(11), pp 65-70.

Cheikhrouhou M., Conti P., Labetoulle J., Marcus K., 1999, Intelligent Agents for Network Management: Fault Detection Experiment. In Sixth IFIP/IEEE International Symposium on Integrated Network Management, Boston, USA, May.

Clark, D., Partridge, C., Ramming, J.C., Wroclawski, J.T., 2003, A Knowledge Plane for the Internet, Proc. Applications, technologies, architectures, and protocols for computer communication, ACM SIGCOMM 2003, Karlsruhe.

Deen, G., Lehman, T., Kaufman, J., 2003, The Almaden OptimalGrid Project, Proceedings of the Autonomic Computing Workshop, 5th Int. Workshop on Active Middleware Services (AMS 2003), Seattle, WA, June, pp 14-21.

EU IST FET, 2003, New Communication Paradigms for 2020, brain storming meeting July 2003, Brussels, Belgium, (report published Sept 2003).

Ganek, A., 2003, Autonomic Computing: Implementing the Vision, Keynote presentation at the Autonomic Computing Workshop, 5th Int. Workshop on Active Middleware Services (AMS 2003), Seattle, WA, 25th June. pp1-1.

Ganek, A.G., Corbi, T.A, 2003, The dawning of the autonomic computing era, IBM Systems Journal, 42(1), pp 5-18

Gray, J., 2001, talk on Dependability in the Internet Era at High Dependability Computing Consortium. May.

Haas, R., Droz, P., Stiller, B., 2003, Autonomic service deployment in networks, IBM Systems Journal, 42 (1), pp. 150-165.

Horn P, 2001, Autonomic computing: IBM perspective on the state of information technology, IBM T.J. Watson Labs, NY, 15th October Presented at AGENDA 2001, Scotsdale, AR. (available http://www.research.ibm.com/autonomic/).

IBM, 2001, Autonomic Computing Concepts, IBM White paper.

IBM, 2003 An architectural blueprint for autonomic computing, April, revised October.

IBM, 2003, alphaworks Autonomic Computing site, http://www.alphaworks.ibm.com/autonomic

IBM, 2003, Autonomic problem determination: A first step towards self-healing computing systems, White Paper, October.

IBM, CISCO Systems, 2003, Adaptive Services Framework, White Paper, version 1.0, October 14.

ITU-T, 2000, M.3010 principles for a telecommunications management network, ITU-T Recommendations, Feb.

Jackobson G., Weissman, M.D., 1993, Alarm correlation, IEEE Network, 7(6), Nov., pp. 52-59.

Kaiser, G. , Parekh, J., Gross, P., Valetto, G. , 2003, Kinesthetics extreme: An External Infrastructure for Monitoring Distributed Legacy Systems, Proceedings of the Autonomic Computing Workshop, 5th Int. Workshop on Active Middleware Services (AMS 2003), Seattle, WA, 25 June, pp 22-30.

Klemettinen, M, 1999, A knowledge discovery methodology for telecommunication network alarm databases, Ph.D. Thesis, University of Helsinki, Finland.

Oates, T., 1995, Fault identification in computer networks: A review and a new approach. Technical Report 95-113, University of Massachusetts at Amherst, Computer Science Department.

Patterson, D., 2002, Availability and Maintainability Performance: New Focus for a New Century, USENIX Conference on File and Storage Technologies (FAST ’02), Keynote Address, Monterey, CA January 29.

Patterson, D.A., Brown, A., Broadwell, P., Candea, G. , Chen, M., Cutler, J., Enriquez, P., Fox, A., Kiciman, E., Merzbacher, M., Oppenhiemer, D., Sastry, N., Tetzlaff, W., Traupman, J., Treuhaft, N., 2002, Recovery-Oriented Computing (ROC): Motivation, Definition, Techniques, and Case Studies, U.C. Berkeley Computer Science Technical Report, UCB//CSD-02-1175, University of California, Berkeley, March 15.

Smirnov, M., Popescu-Zeletin, R., 2003, Autonomic Communication, presentation at the EU IST FET brainstorming meeting on Communication Paradigms for 2020, Brussels, Belgium, July.

Sterritt, R., 2001, Discovering Rules for Fault Management. Proceedings of Eighth Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS '01), April 17-20, pp190-196.

Sterritt, R., 2002a, Facing fault management as it is, aiming for what you would like it to be, Soft-Ware: 1st International Conference on Computing in an Imperfect World, Belfast 8-10 Apr., LNCS2311, Eds. D.W. Bustard, W. Liu, R. Sterritt, Springer-Verlag, pp 31-45

Sterritt, R., 2002b, Towards Autonomic Computing: Effective Event Management, Proceedings of 27th Annual IEEE/NASA Software Engineering Workshop (SEW), Maryland, USA, December 3-5, pp 40-47

Sterritt, R., 2003, xACT: Autonomic Computing and Telecommunications, BT Exact Research Fellowship.

Sterritt, R., Bustard D.W., 2002a, Fusing Hard and Soft Computing for Fault Management in Telecommunications Systems, IEEE Trans. Systems Man and Cybernetics part C, 32(2)

Sterritt, R., Bustard, D.W., 2002b, Practical Intelligent Support for Rule Discovery in Fault Management Systems, Cybernetics & Systems: An International Journal, 33(6), Taylor & Francis, pp 579-601.

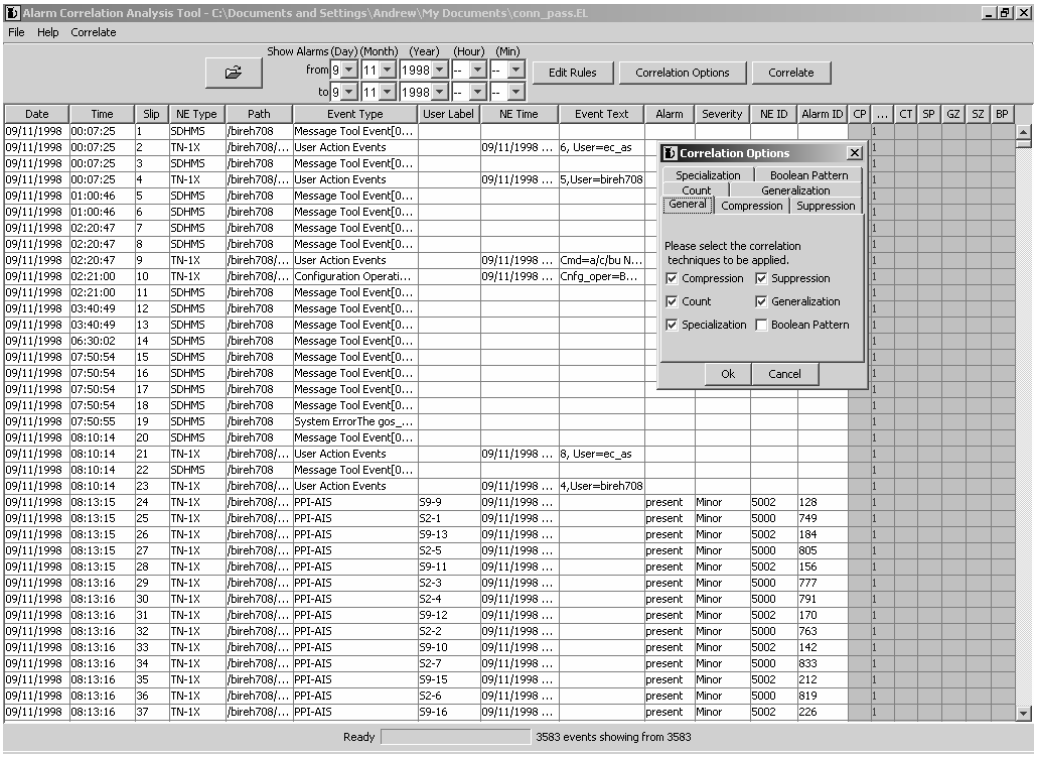
Sterritt, R., Bustard, D.W., 2003, Towards an Autonomic Computing Environment, 1st International Workshop on Autonomic Computing Systems at 14th International Conference on Database and Expert Systems Applications (DEXA'2003). Prague, Czech Republic Sept. 1-5, pp694-698.

Sterritt, R., Bustard, D.W., McCrea, A., 2003, Autonomic Computing Correlation for Fault Management System Evolution, Proceedings of IEEE International Conference Industrial Informatics (INDIN 2003), Banff, Alberta, Canada, 21-24 August, pp 240-247.

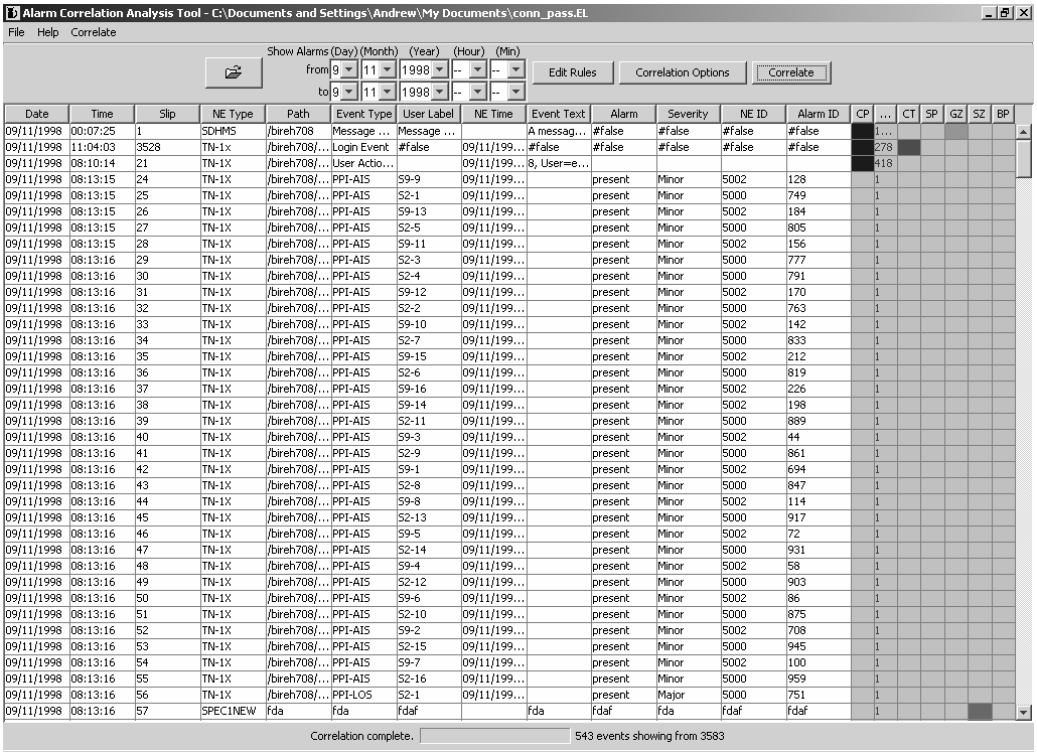
Sterritt, R., Curran E.P., Adamson K., Shapcott C.M, 2000, Visualization for Data Mining telecommunications network data, Data Mining II, (eds.). Ebecken F.F., Brebbia C.A., Weigend A., WIT Press, Southampton UK, pp 445-454.

Sterritt, R., Curran, E.P., Song, H., 2002, HACKER: Human And Computer Knowledge discovered Event Rules for Telecommnuications Fault Management, Proceedings of 2002 IEEE International Conference on Systems, Man and Cybernetics, Hammamet, Tunisia, October 6-9.

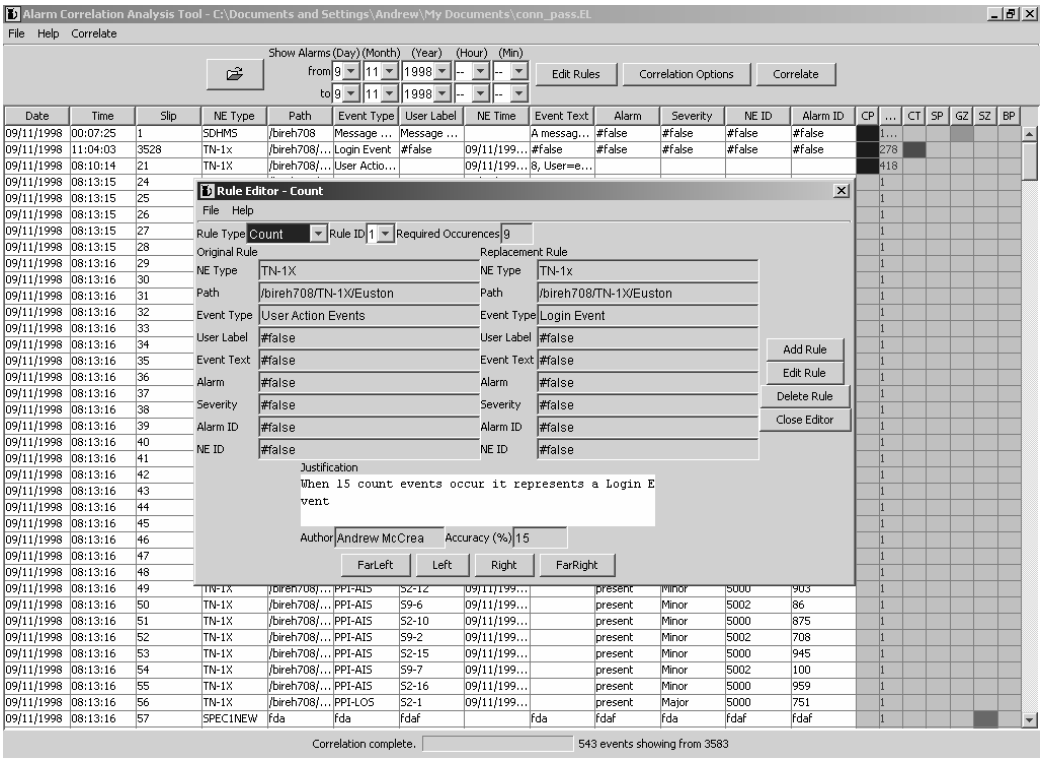
Uthurusamy, R., 1996, From Data Mining to Knowledge Discovery: Current Challenges and Future Directions, Advances in Knowledge Discovery & Data Mining, AAAI Press & The MIT Press: California, pp 561-569.



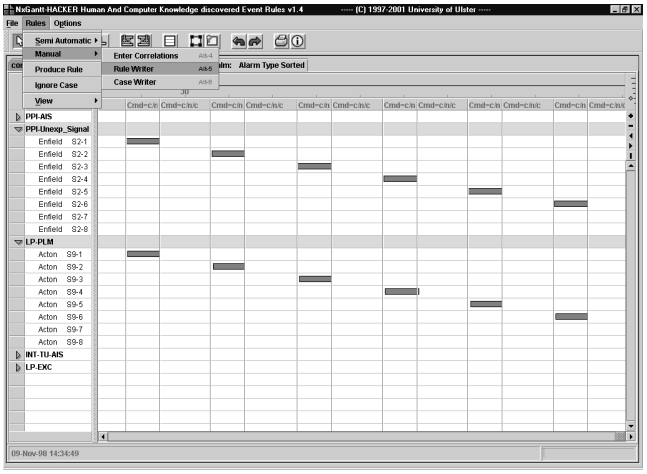
Screenshot 1 Viewing an Event Log and the Correlation Options



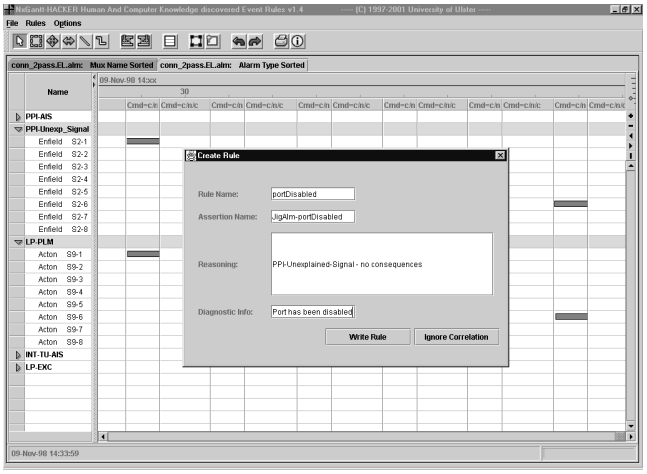
Screenshot 2 Correlation Results showing events that have been correlated



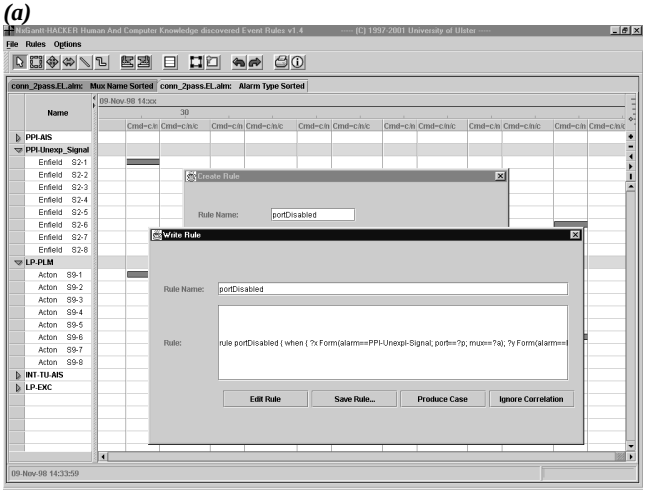
Screenshot 3 Correlation Rule Editor



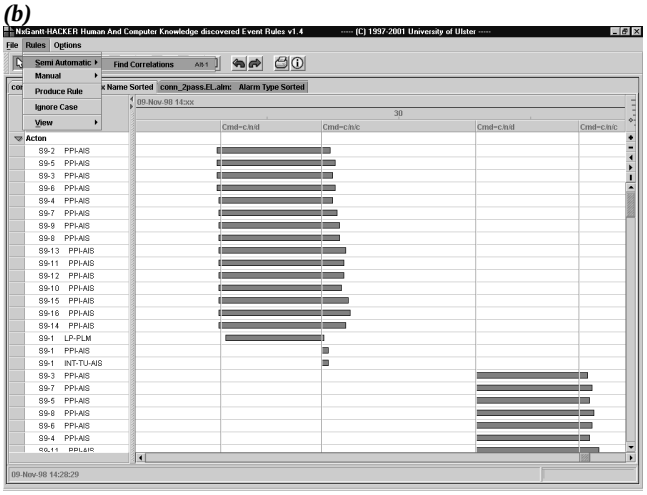
(a)



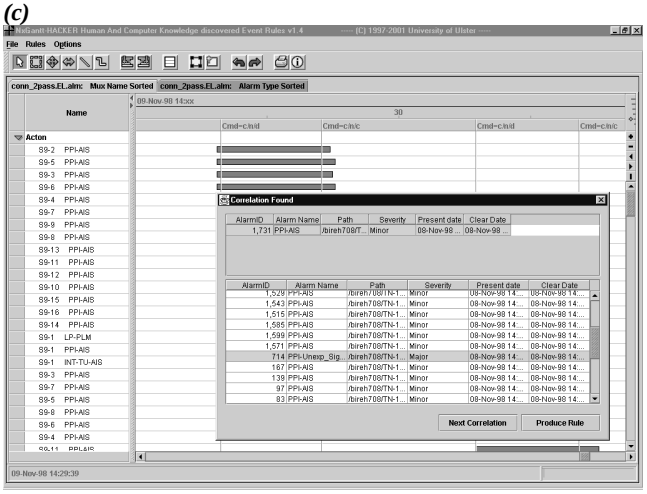
(b)



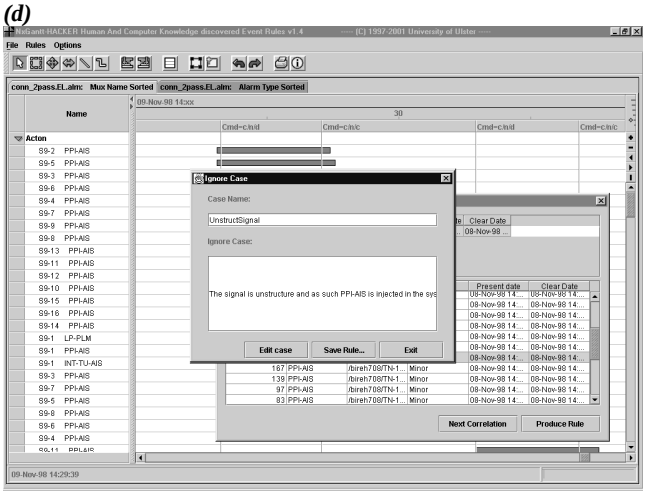
(c)



(d)



(e)



(f)

Figure 9 Screenshots of the Prototype "NxGantt - HACKER" - Human And Computer Knowledge discovered Event Rules