UNIVERSITY OF GOTHENBURG

Validating a Unifying Alarm Model

JESPER LINDBERG
DANIEL NILSSON

Bachelor of Software Engineering and Management Thesis

Report No. 2010:005
ISSN: 1651-4769

University of Gothenburg
Department of Applied Information Technology
Gothenburg, Sweden, May 2010

VALIDATING A UNIFYING ALARM MODEL

Jesper Lindberg
lindberg00@gmail.com

Daniel Nilsson
deinils@gmail.com

***Abstract*—**
The constant growth and complexity of Telecommunication networks has resulted in a situation where alarm operators are flooded with alarms. A medium-sized telecommunication network center can receive several hundred thousands alarm per day. The amount of alarms have created a need for an automatic event correlation strategy. Thus far, researchers and industry have looked to address the problem through techniques such as model-based correlations, neural networks and sequential data mining. Despite these efforts the problem still remains. (Wallin 2009) argues that strong alarm models is the key to an effective alarm management and have created the foundation for a unifying alarm model. In this study an implementation of the unifying alarm model was developed and tested. The result show that the unifying alarm model provide abilities to perform automated correlations hence solving issues in alarm mangement.

***Index Terms*—**
Network Management, Unifying Alarm Model, Alarm Correlation

1. INTRODUCTION

Network management is becoming more and more complex as a result of the growth of the telecommunication networks (Jacques-H. Bellec 2006). Due to the complexity of network management, a single fault can produce a cascade of network alarms and alarm operators working at management centers can receive up to 90 alarms per minute (Jacques-H. Bellec 2006, Wallin 2009). This makes it extremely important to ensure high quality alarms. However, as the quality of the alarms received is currently poor, both humans and computers struggle to interpret them (Wallin. et al. 2008).

Thus far, researchers and industry have looked to address the problem through standardizing alarm interfaces. However, the problem still remains and network management systems are still flooded with poor quality alarms. Today's network management systems use event correlations to filter out important events from the constant flood of alarms. Different approaches have been taken to identify the events for correlation such as *model-based correlation, neural networks* and *sequential data mining* (Jacques-H. Bellec 2006). Wallin (2009) argues that the focus should be put on alarm quality and definitions of alarm interfaces at a semantic level. Wallin, Leijon and Landen have subsequently looked to address the current situation in several research projects (Wallin 2009, Wallin et al. 2009, Wallin & Leijon 2009). Based on these studies, they have created the foundation of a semantic alarm model which they refer to as the unifying alarm model (UAM). The proposed benefits with a semantic model is that it allows computers to filter out the actual cause of an alarm. One single fault/error in a telecommunication base station can start a chain reaction, resulting in several alarms sent to the network management system, even do several alarms are caused by each other. By using a semantic alarm model the system can identify the root cause of the alarm and filter out all other alarms. If this is the case a semantic model could solve the alarm problem.

This paper shares the results of a research project that looks to validate if the theoretical unified alarm model can be translated into a technical implementation, and subsequently if the UAM stands up to actually improving the quality of alarms as is argued by Wallin et al. (2008). The contribution made thus lies in illustrating how the theoretical UAM may be implemented in practice, and as a first feasibility test of extant research on UAM. The alarm database that we are using in this study comes from a leading telecom organization, and thus represents exactly what operators are currently facing. We have, for this paper, limited the alarms to 3G alarms from one specific hardware vendor as the goal at this stage of the research is to produce a first validation of the UAM and not (at this point) to provide a full fledged product to telecom organizations.

The paper is organized as follows: Chapter 2 will present related research, chapter 3 describes the research method, chapter 4 contains the data collection and results, chapter 5 is the analysis section where we analyze our findings and chapter 6 contains the conclusion of this study.

2. RELATED RESEARCH

This section presents the topics related to our study, providing it with a richer contextual background by elaborating on the reality of network management at the moment, and

details on event correlation as well as the unifying alarm model needed to implement and validate our findings.

2.1. Network Management

The network management system is responsible for recording the alarms generated by the nodes in the network and presenting them to the operator (Jacques-H. Bellec 2006). Network management is becoming more and more complex as a result of the growth of the Telecommunication networks. A medium-sized telecommunication network center receives several hundred thousands alarm per day (Wallin et al. 2009). Due to the complexity of network management a single fault can produce a cascade of network alarms (Jacques-H. Bellec 2006, Devitt et al. 2005). The operational activities at the network management center is to manage the constant flood of alarms and the primary goal is resolve the most important faults as soon as possible (Wallin et al. 2009). Due to the amount of alarms received it is impossible to quickly prioritize the alarms. As of today alarms are often prioritized manually, the network administrators need to use their experience and several support systems to prioritize the alarms (Wallin forthcoming). This approach makes the organization heavily dependent on a few individuals and their knowledge (Wallin & Leijon 2006). These individuals are invaluable for the organizations, but their knowledge is hard to reuse as the process of prioritizing alarms are carried out manually and relies much more on experienced interpretation then formal methods. The size and complexity of telecommunication networks, makes it almost impossible to continue with the manual prioritizing. The amount of network operators needed to resolve the flood of alarms are prohibitively high (Devitt et al. 2005). Wallin & Leijon (2006) argues that the next generation of network management must apply knowledge management. Knowledge management will capture the knowledge of the operators which in turn will enable the alarm prioritizing to become self-learning and automatic. High amounts of alarms in telecommunication networks are unavoidable, according to Jacques-H. Bellec (2006) quick detection, identification causes and resolution of failures can make the system more robust and reliable. Therefore, many systems use event correlation engines to manage the large amount of alarms.

2.2. Event Correlation

Event correlation is a technique for locating interesting patterns of events in flood of information (Jayaram & Eugster 2009). Different approaches have been explored to identify the events for correlation.

Model-based correlation uses a system description, often formalized as a set of formulas expressed in logic (Wietgrefe et al. 1997). The model is used to predict expected behaviors, the model is a representation of the correct system which makes it useful for error detection. The model-based system needs to be maintained, which is costly and in some cases complex (Wallin et al. 2009). Wallin et al. (2009) argues that the reason for this is that the change rate of network topology and service structures can be hard to handle. Whenever a new piece of hardware is introduced someone has to "implement" the alarms sent from the hardware into the model.

Knowledge-based correlations approaches gathers correlations rules from interviews with experts/operators (Burns et al. 2001). The benefit is that this approach captures the informal knowledge in the telecom organization, however interviewing experts is are expensive in both time and money (Burns et al. 2001). When ever new hardware is introduced new interviews has to be conducted, the problem is that there is no one t interview because no one in the organization have encountered alarms form the new hardware.

Neural networks is an artificial intelligence technique which can be used for event correlations (Wietgrefe et al. 1997). Neural networks systems may require no experts as input for alarm correlation (Devitt et al. 2005), instead it can use a learning algorithm (Wietgrefe et al. 1997). A database containing the most important fields from the alarms is sufficient for learning the neural network (Wallin et al. 2009). When the network management system receives an alarm, it pose a series of questions to the fully trained neural network which generates a suggested privatization (Wallin et al. 2009). This privatization indicates whether the network operator should handle the alarm or whether it will clear itself. An advantage with the neural network approach is that one does not have to maintain it in the sense that it constantly learns how to correlate new alarms. However, as of today its privatization only gives the network operator a hint of what to do with the alarm, its is not fully automatic.

Sequential data mining seeks to identify the specific problem of relationships or correlations between events in a data set (Devitt et al. 2005). These data sets are often inherently sequenced by nature, due to the fact that all events are timestamped. The output of this mining approach can be used as input for rule-, code- or model based approaches (Devitt et al. 2005). The main objective with a sequential data mining approach is to locate *noteworthy* event sequences or patterns which in turn points out relationships between event (Devitt et al. 2005). In reality this means that a noteworthy sequence is an frequently occurring sequence in a data set. The downside with sequential data mining as a correlation tools is that it does not indicate redundancy of sequences.

All of these correlation strategies are currently used in network management, but they are facing two issues. The *first issue* all strategies, is that when new equipment is introduced the telecom company have to invest time and money to be able to correlate new alarms. Even systems using neural networks have to spend time teaching the system how to correlate. *Second*, these four event correlation strategies are all dependent of high quality

information to make correct correlations the alarms. If the alarm specifications provided by the hardware vendor is in poor quality it will effect the correlation process (Wallin. et al. 2008). Wallin. et al. (2008) favors the model-based approach and argues that the hardware manufacturers should use such approach to express their alarms. By doing this, one transfers the problem from the telecom company to the hardware manufacturer. Today the telecom companies have to gather information from alarm documentation to be able to include it into their correlation strategy. It would be much easier if the hardware manufacturers implemented their alarms into a model. As mentioned above a model allows for automatic correlations.

Correlations are used to determine the root cause of a fault and to filter out redundant alarms (Jacques-H. Bellec 2006). A lot of effort have been made researching alarm correlations, resulting in that all alarm systems support advanced filtering mechanisms, Wallin et al. (2009) argues that the problem lies in defining the rules used to filter the alarms. By filtering out all redundant alarms the network operators would only have to handle relevant alarms which would make the network management center more efficient (Wallin et al. 2009). In a survey from 2009 one representative for a leading telecom operator estimated the use of alarm correlation to 1-2% of all the alarms and the overall attitude of the survey was that the technique is expensive and complex (Wallin & Leijon 2009).

2.3. Unifying Alarm Model

To implement and maintain alarm interfaces is expensive due to the increasing number of hardware and services in the network. Most research and industry efforts in event management have focused on late stages in the alarm chain such as alarm correlation (Wallin. et al. 2008). Despite standards in alarm interfaces there is a confusion around alarm notifications, alarm states and how to match notifications to alarms. Wallin. et al. (2008) argues that the problem lies in the lack of strong alarm models.

The unifying alarm model adds quality to the static information about alarms. In an alarm chain there are two contexts to consider, the managed system and the management system. In the first context, the alarm affects the actual system, in the second the management system tries to estimate the alarm and resource states. The estimation is based on the alarm notification using reasoning algorithms with topology knowledge (Wallin. et al. 2008). The first provide static information and the latter dynamic information which is dependent of the quality of the static information. Alarm interfaces, alarm operator instructions and resource models are the most relevant static information. The unifying alarm model is a semantic model of alarm specification in equipment in the telecom industry (Wallin. et al. 2008). By adding formal semantics to the current alarm specifications provided by the hardware vendor the model provide a solution to the current problem in network management.

Alarms are described in basic alarm specification syntax (BASS). There is a tradition in hardware vendors providing alarm specifications in informal English, where the alarm is described according to the standards together with information about routines to handle it (Wallin. et al. 2008). An implementation of the UAM includes a Domain Specific Language (DSL) called BASS to describe the alarm according to a predefined compilable grammar. BASS is describing the relations of alarms similar to an ontology. In computer science ontology refers to expressing the relational and hierarchical nature of entities in a knowledge domain. Gruber calls this *"specification of a conceptualization, an explicit formal specification of the terms in the domain and relations among them"* (Gruber et al. 1993). To make an ontological commitment is to use a vocabulary in a way that is consistent within the knowledge domain, hence enabling a common understanding of the *structure* of information (Gruber et al. 1993, Noy & McGuinness 2001). A domain consists of entities, entities that can be grouped and related in a hierarchy, and subdivided according to similarities and differences.

The UAM defines an alarm as an *"abnormal state in a resource for which an operator action is required. The operator is alerted in order to prevent or mitigate network and service outage and degradation"* (Wallin. et al. 2008). This definition is important to limit the amount of alarms an operator receives to only include alarms for which he or she actually can take an action against. The actions an operator can perform are different if the alarm is an error or fault. A fault represent a root cause and actions can be taken to resolve the underlying problem, though it is important to understand that not alarms have an underlying fault. An error is a symptom and action can be taken to minimize negative effects (Wallin. et al. 2008).

The unifying alarm model would give several advantages. The most useful is that an alarm management system now understand the alarms due to their formal description which includes relationships and attributes. This would remove the overhead for manual management of alarms. Another important aspect is the alarm quality. Currently an alarm is described in a text document and they differ in quality. Ultimately it is a matter for a human operator to interpret the meaning of the alarm. To replace the specification with a formal semantic model described in a domain specific language will give a new layer of quality assurance because each alarm description needs to pass a compiler testing the rules of the model. It will also capture the knowledge of the experts hence decrease the risk of dependency of key staff.

Fig. 1 show the correlation process as it is today and fig. 2 show the process when using the UAM. The overhead of alarms would be reduced and expert knowledge capture

3. Method

As our study at this point does not aim at a specific telecom organization, the research setting is somewhat generic to us

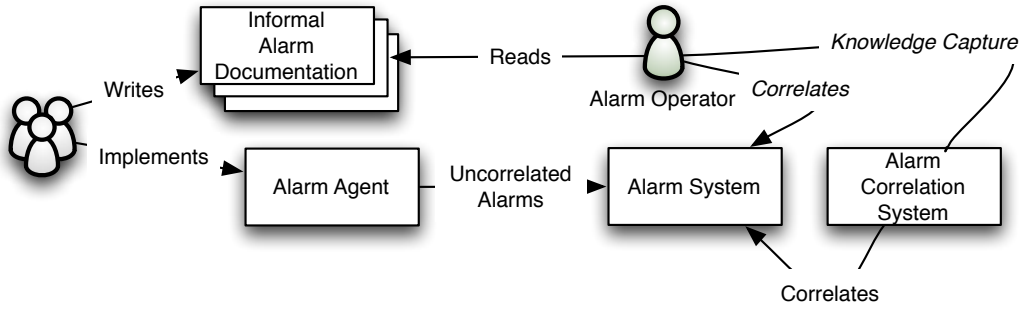

Fig. 1. *Show the current alarm correlation process with the problem of dependency of experts in the correlation chain.*

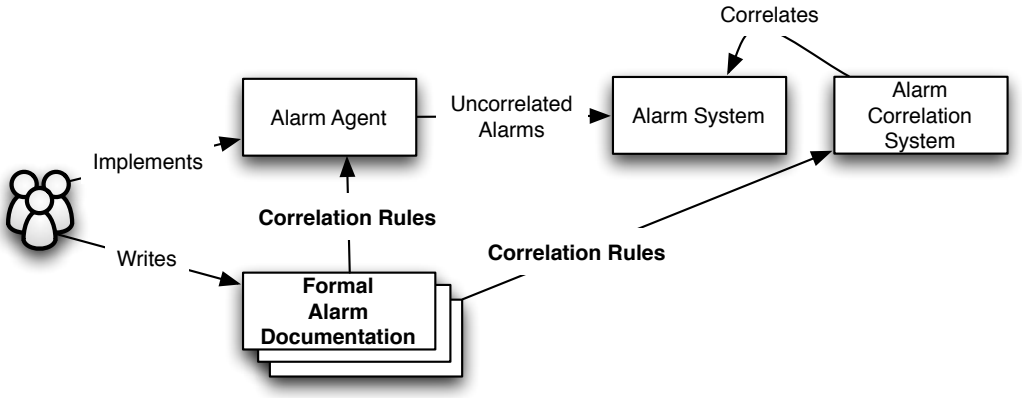

Fig. 2. *Show how the dependency of the operator can be avoided and an automated correlation process would be achieved using the UAM.*

and also why we covered it as background information in section 2.1. Thus, the method section below focuses on the limitations we have placed on this paper (3.1), the detailed elements that have gone into our implementation of the UAM (3.2), and the process we have followed through this research (3.3), including the role of test cases to validate our findings.

3.1. Limitations

The mobile network today consists of a large variety of hardware vendors, different types of communication platforms and services. For this study equipment in the umts (3GPP) network was studied. This study aims to validate to what extent the unifying alarm model is sufficient in correlating alarms in an isolated part of a real situation. A correlation of an alarm refers to pin point a root cause of an alarm with the attribute ”Error”. The isolated reality consist of alarms sent from equipment from one hardware vendor. The alarm database is provided by a large telecom company. These limitations implies that alarms affected by other equipment could not be fully investigated and was ignored in this study.

3.2. Research process

Our research was divided into four steps, the translation step where the alarm documentation was translated into DSL, the alarm database step where the unsuitable data was removed, the correlation artifact which was developed and the validation step.

3.2.1. Translation

The alarm documentation was provided by the hardware vendor and as previously mentioned covers their equipment in the 3G network. The translation of the documents to the model language was conducted in an iterative process. The results from the probing and re-building of the alarm database created a focus lists of prioritized alarms to translate. The translated alarms gave statistics to further trace the root cause of alarms. The final product was used to gather the data which is presented in this study. The translations were conducted with collaboration with an external expert with knowledge in alarm documentation.

3.2.2. Alarm Database

To effectively probe for statistics in the database the above mentioned limitations and a simple rule where used when

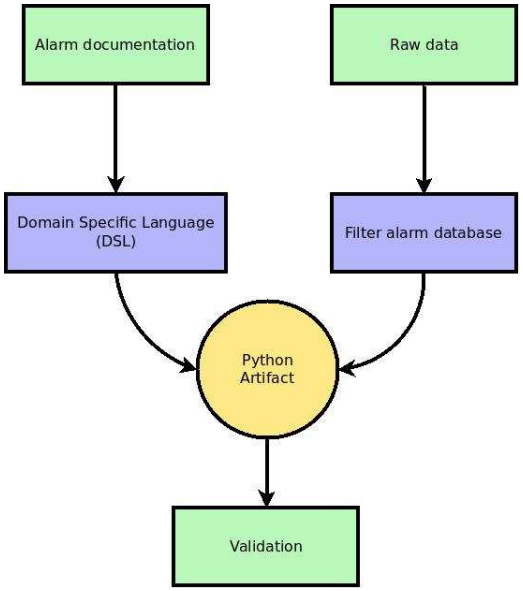

Fig. 3. *show the work process of this study*

filtering and rebuilding the database. AlarmType is a naming convention and a variable described in the alarm standard document X.733 (a standard in telecommunications). The alarm type variable is used in the uam for alarm name and mapping to the original alarm specifications. If the alarm type could not be found the alarm was deleted from the database

3.2.3. Correlation artifact

A tool was developed to handle the massive amount of data, this tool is described in greater detail in section 4.2. The tool is an implementation of the UAM and creates internal representations of alarms as objects. The internal representation adds UAM abilities to correlate alarms. The validation artifact creates the SQL queries based on the modeled representation attributes and gather statistics from the database. The statistics will be used to answer the question whether the UAM can be used to effectively correlate alarms.

3.2.4. Validation

The validation step was divided into two parts, the automatic- and the sample correlations. The automatic correlations is closely linked to the correlation artifact as the statistics is a product of the artifact. The sample correlations on the other hand where conducted manually. The entire validation strategy is described in section 3.3.

3.3. Validation strategy

As a part of this project we translated written alarm specification documents provided as PDF's into the domain specific language. We limited the alarms in this study to 3G alarms from one specific hardware vendor. By focusing on alarms from one hardware vendor we limit the validation of alarm specifications and aim at ensuring that alarms from the equipment vendor can be formalized into the model.

To be able to validate whether the theory of the unifying alarm model holds, we implemented the unifying alarm model according to its specifications in Wallin. et al. (2008) paper. To validate that the implementation of the unifying alarm model is providing proclaimed correlating abilities we examined it in conjunction with samples from an actual alarm database.

To test the unifying alarm model we created tests which covers all scenarios discovered during the process of validation and translation of the alarm specifications. These tests contained several thousand alarms all caused by one or more root causes. The outcome from all tests served as our quantitative data and we used statistical analysis to illustrate how many tests our implementation of the unifying alarm model passed.

There are two primary sources of data needed to validate the unifying alarm model, alarm documentation and alarm database. The model is based on a manual translation of existing alarm documentation to the DSL. The translated alarms are used to automatically back trace root causes in the alarm database. To ensure that the model is tested in a real environment the sample database was provided by a major telecom operator. The sample contains over 3,5 million alarms from a three months period of normal operations and cover the entire Swedish network. Quantitative data were gathered and analyzed according to the description below.

4. Data Collection and Results

The Data collection section is where the data used in this study is presented. It is divided in the following topics: *Alarm database* and *Correlation Results*

4.1. Alarm database

The exported alarm database raw file contained 3,592,868 rows. A filter mechanism was used to delete rows excessive for the study. 66,068 rows was deleted according to the first criteria.

Criteria 1: The alarm type must be known.

If an attribute "alarm type" was unknown it was impossible to use in the study. In the cases where the alarm type was unknown it was noticed that the string holding the values was interrupted before the desired value was fully stated. This interruption most likely occurred during the export phase of the database which was beyond the control of this study. The position of these alarm was registered and used to later exclude samples of data that might would have been affected by the deleted alarms. The reason the alarm type have to be known is due to the UAM's usage of name mapping using the dotted notation in alarm names. The

alarm type of an entry in the database need to be known for automatic mapping against the alarms notation in the model language. An alarm can be categorized using the following format where the alarm type is the last of the dot seperated names.

category.protocolType.alarmType

The above fictive example shows how an alarm can be categorized using the dotted notation. The alarmType was used when collecting and mapping alarms from the database.

Criteria 2: The alarm must have available documentation.

The number of alarms in the database that there was no available documentation for was 3,281,041 in number and after deletion the database consisted of 245,759 divided on 71 different alarm types. All statistics discussed in this section is collected in table I.

TABLE I
FILTERING STATISTICS

Raw input	+3,592,868
Criteria 1	-66,068
Criteria 2	-3,281,041
Remaining	245,759
alarmTypes	71

4.2. PYTHON IMPLEMENTATION

The alarm types translated to DSL is the foundation in the UAM. The syntax follows a specified grammar outlined in a alarm grammar file that can be used to implement the model in any language of choice. In this study Python was used to implement alarm type objects and to perform a statistical analysis in a correlation process.

The definition of a correlated alarm in this study is an alarm for which the underlying fault can be found in a specified time window. Alarms with the type ”error” is a symptom on an underlying fault and the fault should occur in a reasonable time window of the original error. Figure 4.2 show an abstraction of the process to correlate the alarm.

For each alarm type, an object is created to be an internal representation of that type. This process is handled by the parser which use the alarms in DSL to create an instance for each alarm type. Each instance holds the variables of an alarm as specified in the UAM and are hereafter called *typeObj*.

The correlation process needs to know in what *time window* root causes should be found. There were two variables which could be set before runtime. They limit how far in the future and how far in the history alarms should be investigated, with the time stamp of the current alarm under investigation as a starting-point. The reason to look for the fault after the creation of the error is because the symptom might appear before the root cause in the alarm records.

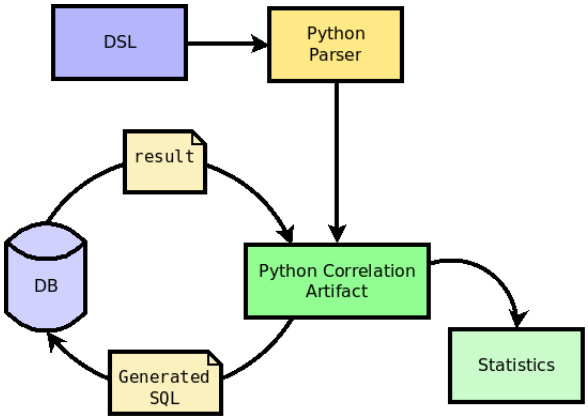

Fig. 4. *An abstraction of the correlation process.*

For each *typeObj* a SQL query is generated which ask for all alarms of that specific type. For each returned alarm a *alarmObj* is created. The alarmObj is an internal representation of an alarm entry in the database. It holds all values that are needed for the correlation process.

To summarize the correlation process:

- For each alarm type an SQL query is generated to return all alarms of that type.
- For each returned alarm a new SQL query is generated to return ALL alarms in a specified time window with no regard to alarm type.
- The correlation engine use the list held in typeObj.rootcause to match with alarmObj.alarmType in all alarms returned by the second SQL query.

The alarms in the time window fall in either one of these categories:

- **Perfect match:** The alarm type of the alarm exist in the typeObj list of root causes and the node of the alarm match the node specified in the alarmObj.
- **Root cause found:** The alarm type of the alarm exist in the typeObj list of root cause but does NOT match the node in the alarmObj.
- **Same node:** There is no match of any root causes but there is at least one other alarm sent from the same node in that time window.

The last category is used for a post analysis to see if one alarm could possibly have an affect on the alarm under investigation. The last category and all other cases in which a root cause was not found is counted as uncorrelated alarms. Statistics from all categories was collected in an automated report and are shown in table II. The report also contains all related alarms found in each time window, grouped and linked to its original alarm.

TABLE II
AUTOMATED CORRELATION STATISTICS

Alarm type	Occurrence	Num of related faults	Root cause/60min	Perfect match/60min	Root cause/6min	Perfect match/6min
Loss of link redundancy group	1	5	0%	0%	0%	0%
Loss of system clock	610	3	2,62%	1,97%	0,33%	1,8%
PDH alarm indication signal	2773	1	33.18%	1,48%	4,76%	1,37%
Switch internal link group fault	19	5	0%	5,26%	0%	5,26%
Switch plane a fault	16	5	12,50%	81,25%	0%	81,25%
System clock in holdover mode	11408	19	99,92%	65,27%	75,15%	62,12%
System clock quality degradation	7150	24	100%	10,36%	95,19%	6,59%

4.3. CORRELATION RESULTS

The 71 alarm types found in the database was examined with the automated process. 7 alarm types was chosen for further analysis which all were errors; they all have at least one underlying fault. The results are divided in two parts. The first part will show statistics about a specified alarm type taken from the automated correlation process. The second part will look at a selection of samples of alarms of that particular alarm type.

4.3.1. AUTOMATED CORRELATION RESULTS

Table II show the result from the automated correlation. The variables are:
Occurance: The total number of alarms of the specified type.
Num of related fault: The number of faults that possibly is the cause for the symptom alarm.
Root cause/60 min: The number of root causes found 60 minutes prior and 60 minutes after the original symptom alarm, this only is counted if no *Perfect match* is found.
Perfect match/60 min indicates when an alarms related fault has been found in the specified time window and on the same node. This number only have relevance when the fault and error is explicitly on the same node.
Root cause/6 min and *Perfect match/6 min* is the same as the variables described above except the time window was reduced to 6 minutes prior and 6 minutes after the original symptom alarm.

4.3.2. SAMPLE CORRELATION RESULTS

10 samples was chosen from one alarm type from the generated report in the previous correlation process. Each alarm was back traced manually. The objectives for the manual analysis was to cross check the automatic correlation report and to identify anomalies that could possibly affect the correlation procedure.

The following variables was under investigation:

- Alarms found in the time window of the original alarm happening on the same node.
- Multiple instances of found root causes for the original alarm
- Multiple instances of the same alarm type on the same node in the time window
- The creation time stamp of the alarms.

In none of the cases did the automated correlation report contradictions with the result from the manual test. Using the same variable for the time window showed that the correlations found was valid.

For the second objective to find anomalies that could affect the result a shorter time window of trace total 12 minutes of database entries. 6 minutes before and 6 minutes after the original symptom alarm. The result of the second objective revealed that there is some aspects which needs to be taken in consideration when interpreting the automated correlation results. These issues is further discussed in the following analysis section.

5. ANALYSIS

For this analysis there are two ground cases that needs to be considered. Is the error a symptom of a fault on the same node or is it a symptom of a fault on a node somewhere else in the network? The first case is fairly easy to back trace while the latter require an investigation of all nodes in the network. So how do we know that our symptom error is an effect of the fault? In the case of the symptom and fault explicitly occurring on the same node, a simple search for multiple instances of both fault and errors found in the time window can tell us if the alarm really could be said to be correlated. If the fault have a possibility of existing on any node in the network a relation needs to be established between the error and fault to confirm a correlation. In table II there are two variables that represent the two cases. *Root cause* is the correlation rate of one alarm type if a root cause was found in any node in the network. *Perfect match* represent the correlation rate if the fault was found on the same node.

As an example; alarm *System Clock Quality Degradation* had a correlation rate of 66,27% by only look at alarms sent from one node during a time window of 120 minutes in total. If to search for root causes on all nodes in the network, matches was found in all cases; 100%.

The alarms found in a time window could fall in to either of three categories as described in the Python implementation section. So if at least one alarm matched the rules of the

perfect match or root cause, it was counted as a correlation success. The complexity of the relationship between alarms is a problem here. If an error (symptom alarm) have many related faults (root cause alarms) it is hard to link the error to a specific fault. Sometimes there was multiple faults related to the error found in the time window. Either there was multiple problems causing the error or the error was related to only one of them. Which one could not be said for sure.

The sample correlation results show that in most cases, especially in the *Root cause* category, multiple matches was found, sometimes as many as over 500 matches for every alarm in the 120 minute time window. Even in the 12 minute window there are still sometimes over 50 matches. The *System Clock Quality Degradation* alarm have 24 possible underlying faults. That is 24 alarm types to search for in a database containing 71 alarm types in total. The variables which effect the amount of matches is off course how many alarm types exist as possible root causes and the length of the time window.

The question is, which of the matches was the actual root cause? Is the alarm really correlated? The answer is that in this study there is no way to know for sure. But the result is still important and answer our research question. Without the UAM there was no clear distinction between error and fault. There was no possibility to search for a root cause because the only place to find this information is in a written document and the task to manually search for faults is not trivial.

Another aspect to consider in the automated correlation result is that each error alarm was examined and counted with no attempt to group redundant alarms. Still, the result clearly show that the UAM is effective in finding related root causes for an error.

6. CONCLUSIONS

The unifying alarm model shows many benefits for alarm management in theory. This study have focused on the common task to correlate an alarm with the attribute ”error” by automatically trace related alarms with the attribute ”fault” in a database of alarms.

Three issues arises in this task.

- If there are multiple instances of related alarms with the attribute ”fault” which one is actually the root cause for the original symptom alarm?
- If there are multiple instances of the symptom alarm on the same node, are they issued by the same fault?
- If the fault is causing errors in other nodes, which error belongs to what fault?

This should be taken in consideration in the assessment of the statistics presented in this paper.

Translation and added quality. This study can conclude that a translation from informal alarm documents to formal basic alarm specification syntax is fully possible and easy to use when developing a correlation tool. The BASS adds a layer of quality assurance, if the original alarm documentation is incomplete it will reveal itself in the correlation results because loose ends will show.

Formality improves automated correlation. The study show that a formal translation is indeed very useful for tracing root causes for errors in a network. The alarms description in domain specific language can be used to implement a correlation tool in any programming language and that correlation can be done in an automated process.

For future research it is suggested to increase the number of translated alarms to cover one complete system. It is also recommended to perform a comparison study with the correlation from an actual alarm management center or perform a test of the UAM in a collaboration with staff from an alarm management center.

REFERENCES

Burns, L., Hellerstein, J., Ma, S., Perng, C., Rabenhorst, D. & Taylor, D. (2001), A systematic approach to discovering correlation rules for event management, in ‘Proceedings of the 7th IFIP/IEEE International Symposium on Integrated Network Management’, pp. 345–359.

Devitt, A., Duffin, J. & Moloney, R. (2005), Topographical proximity for mining network alarm data, in ‘Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data’, ACM New York, NY, USA, pp. 179–184.

Gruber, T. et al. (1993), ‘A translation approach to portable ontology specifications’, Knowledge acquisition **5**, 199–199.

Jacques-H. Bellec, M.-T. K. (2006), Towards a Formal Model for the Network Alarm Correlation Problem, in ‘Proceedings of the 6th WSEAS International Conference on Simulation, Modelling and Optimization’, ACM New York, NY, USA, pp. 458–463.

Jayaram, K. & Eugster, P. (2009), Context-oriented programming with EventJava, in ‘International Workshop on Context-Oriented Programming’, ACM, pp. 1–6.

Noy, N. & McGuinness, D. (2001), ‘Ontology development 101: A guide to creating your first ontology’.

Wallin, S. (2009), ‘Chasing a definition of älarm¨’, Journal of Network and Systems Management pp. 457–481.

Wallin, S. & Leijon, V. (2006), ‘Rethinking network management solutions’, IT Professional **8**, 19–23.

Wallin, S. & Leijon, V. (2009), Telecom network and service management: An operator survey, in ‘12th IFIP/IEEE International Conference on Management of Multimedia and Mobile Networks and Services (MMNS)’.

Wallin, S., Leijon, V. & Landen, L. (2009), ‘Statistical analysis and prioritization of alarms in mobile networks’, International Journal of Business Intelligence and Data Mining (IJBIDM) **4**, 4–21. Special Issue on Intelligent Techniques for Network Applications.

Wallin., S., Norlander., J. & Leijon, V. (2008), ‘A unifying alarm model’, Journal of Network and Systems Management .

Wietgrefe, H., Tuchs, K., Jobmann, K., Carls, G., Frohlich, P., Nejdl, W. & Steinfeld, S. (1997), Using neural networks for alarm correlation in cellular phone networks, in ‘Proceedings of the International Workshop on Applications of Neural Networks to Telecommunications 3’, Lawrence Erlbaum, p. 248.