J Netw Syst Manage (2009) 17:457–481
DOI 10.1007/s10922-009-9127-3

Chasing a Definition of “Alarm”

Stefan Wallin

Published online: 27 May 2009

Abstract Alarm management has been around for decades in telecom solutions. We have seen various efforts to define standardised alarm interfaces. The research community has focused on various alarms correlation strategies. Still, after years of effort in industry and research alike, network administrators are flooded with alarms; alarms are suffering from poor information quality; and the costs of alarm integration have not decreased. In this paper, we explore the concept of ‘alarm’. We define ‘alarm’ and alarm-type concepts by investigating the different definitions currently in use in standards and research efforts. Based on statistical alarm data from a mobile operator we argue that operational and capital expenditures would decrease if alarm sources would apply to our alarm model.

Keywords Communication system operations and management · Alarm management · Network management standards

1 Introduction

Network administrators are flooded with alarms, which require their action, either to resolve the problem or confirm the alarm as irrelevant. Therefore, the quality of the alarm information is vital. Despite this, we are still in a situation where we see fundamental problems in alarm systems with alarm floods, standing alarms, lack of priorities, and incomprehensible messages [1]. The alarm text in Fig. 1 is an actual example from a Network Operations Centre. Imagine facing the alarm system and asking yourself the following questions:—*What does it mean?—Do I dare to ignore*

S. Wallin (✉)
Luleå University of Technology(LTU) Skellefteå and Data Ductus Sweden,
Torget 6, 931 31 Skellefteå, Sweden
e-mail: stefan.wallin@dataductus.se

```
*A0628/546
/08-07-01/10 H 38
/N=0407/TYP=ICT/CAT=SI/EVENT=DAL/NCEN=AMS1/AM=SMTA7/AGEO
=S1-TR03-B06-A085-R000
/TEXAL=IND/RECEPTION/COMPL.INF: /AF=URMA7/ICTQ7
AGCA=S1-TR03-B06-A085-R117/DAT=08-07-01/HRS=10-38-14
/AMET=07-020-01/AFLR=175-011/PLS/CRC=NACT
/NSAE=186/NSGE=186/NIND=14/INDI=956/NSDT=0
```

Fig. 1 Example alarm text

the alarm?—Does it affect services and customers?— How can the problem be resolved?—Does the alarm report an alarm raise or alarm clear?

Although the example presents an extreme case, many alarms suffer from poor information quality, and accompanying alarm-interface definitions are often vague. This implies operational (OpEx) and capital expenditures (CapEx) for the operators [2]. Operational costs are incurred since network administrators must spend time analysing alarm information. At the same time, the integration of the alarm interface to the network management solution creates a capital expenditure in the form of integration, filtering and correlation projects.

Network management history is the pursuit of a silver-bullet protocol [3] without much attention to defining the *meaning* of management information. This is especially true of alarm management where we have seen alarm attributes, as defined by X.733 [4], being transferred by means of different protocols like Q3, SNMP, Corba, and now SOAP. There is a need to focus on the definition of the alarm concept and on alarm information itself. In this paper we will argue that the costs of alarm management for large wireless operators would decrease if equipment vendors improved their alarm and alarm-interface quality.

Section 2 analyses the alarm concept in light of relevant telecom alarm standards and shows the alarm chain from the source to the human operator. Alarms can be modelled using state machines, where the life cycles are different for the alarm source and the management system. We show the respective state machines in Sect. 3. One of the main themes of this paper is that alarm-producing sources are currently passing on alarm-interpretation problems to the alarm consumers. We discuss the roles of producers and consumers in Sect. 4.

Our contributions are:

- An analysis and critique of existing alarm standards are presented in Sect. 5.
- We give a simple and generic definition of alarm and alarm types in Sect. 6 that, if used, would decrease integration costs and enhance alarm quality.
- We propose an alarm taxonomy in Sect. 7 that gives a structured framework to analyse alarms.

Throughout the paper we will illustrate the alarm concepts with qualitative and quantitative analyses of real alarm data from a mobile operator. Some of the empirical data have been published in a previous article [5].

2 What is an Alarm?

2.1 Background

In its widest sense an alarm is an important ‘observable’ network ‘phenomenon’, where a phenomenon is defined as *any state or process known through the senses rather than by intuition or reasoning* [6]. Network administrators observe undesirable states such as link failures, power problems and high dropped-call rates, and thereby make an hypothesis of the underlying problem. They judge the effect of the problem, making predictions of the impact on services and customers.

In order to arrive at a less abstract definition we can use the alarm definition framework by Stanton [7], see Fig. 2. In the *stimuli-based* model, an ‘alarm’ refers to the state in the resource. This is a positivist (and engineering) view in the sense that alarms can be defined technically. The *response-based* model refers to an alarm as the interpretation and response by management processes. A response-based model is phenomenological in that what constitutes an alarm depends on interpretation by human operators. The telecom industry focuses on alarm interfaces where the term ‘alarm’ is often used to refer to the alarm *notifications* exchanged by systems. This is illustrated by the message-based model in Fig. 2.

With regard to alarm standards, various definitions of the term *alarm* exist. We will look at the definitions used in the most important telecom alarm standards; ITU-T X.733 [4], 3GPP [8, 9], IETF Alarm MIB [10], DMTF Event Model [11], and ITIL [12]. We will look for definitions by asking questions relating to the chain of observations: What *general* state-changes can be observed?—What is broken?—What is the negative *effect*?—What information about negative effects is published?

What general changes in state can be observed?

X.733 No concept

3GPP *Event:* this is a generic term for any type of occurrence within a network entity

IETF *Event:* something that happens which may be of interest—a fault, a change in status, crossing a threshold, or an external input to the system, for example

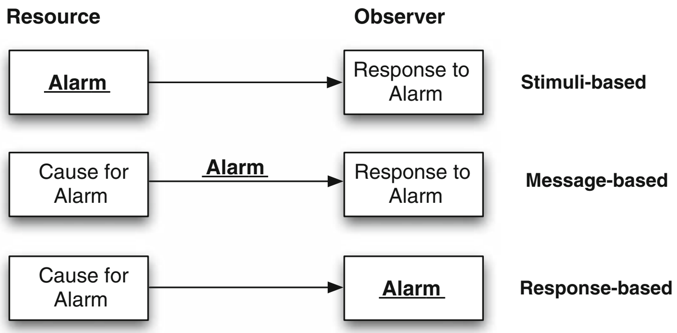

Fig. 2 Alarm definitions, extended from Stanton [7]

DMTF *Event:* a change in state such as stop/start, or as "An occurrence of a phenomenon of interest"
ITIL *Event:* a change of state, which has significance for the management of a Configuration Item or IT Service.

What is broken?

X.733 *Fault:* the physical or algorithmic cause of a malfunction. Faults manifest themselves as errors
3GPP No concept
IETF No concept
DMTF No concept
ITIL No concept.

What changes in state imply negative behaviour?

X.733 *Error:* a deviation of a system from normal operation
3GPP *Fault:* a deviation of a system from normal operation, which may result in the loss of operational capabilities of the element or the loss of redundancy in case of a redundant configuration
IETF *Error:* a deviation of a system from normal operation. *Fault:* Lasting error […] condition
DMTF No concept
ITIL *Incident:* an event which is not part of the standard operation of a service and which causes or may cause disruption to or a reduction in the quality of services and customer productivity.

What Information about negative behaviour is published?

X.733 *Alarm:* a notification, […], of a specific event. An alarm may or may not represent an error. […] Alarms are specific types of notifications concerning detected faults or abnormal conditions
3GPP *Alarm:* abnormal network- entity condition, which categorises an event as a fault
IETF *Alarm:* persistent indication of a fault
DMTF *Indication:* the representation of the event. *Alert* is a special case of an indication which contains information about the severity, cause, and recommended actions
ITIL *Alert:* a warning that a threshold has been reached, something has changed, or a failure has occurred.

First, we can agree that we have events in general which constitute any state changes in network resources. Alarms are a special kind of event in that they imply negative effects and require action. Another general observation is that alarms refer to the *symptoms* rather than the actual fault. We can use the X.733 path to see the chain: we have an underlying *fault*; associated resources will deviate from the expected behaviour as *errors*, until finally this is represented as an *alarm*.

The Distributed Management Task Force CIM Event Model [11] has a similar definition of these three steps. It differentiates between the different levels of ‘events’, ‘indications’, ‘alerts’. The ‘indication’ is the *representation* of the event. The relationship between ‘events’ and ‘indications’ is not one-to-one; several ‘indications’ can be reported for the same ‘event’, the ‘indication’ may not report the true event, etc. Finally, an *alert* is a special type of ‘indication’ which contains information about the severity, cause and recommended actions, thereby turning the ‘indication’ into an ‘alarm’.

We see some different approaches to the definition of ‘alarm’. While X.733 defines ‘alarm’ as the message-carrying alarm-information, 3GPP and IETF refer to ‘alarm’ as the undesired state. DMTF avoids using the term ‘alarm’ and uses ‘alert’ in the same sense as X.733. Alarms are as seen above mostly associated with an underlying fault. This is in some cases not true, since high resource utilisation is an undesired state without a corresponding underlying fault.

So far we have looked at the alarm concept as such. However, there are vital *characteristics* regarding alarms that need to be looked at:

Alarm states does an alarm have states?

Alarm type how do we know if two alarms are referring to the same kind of undesirable state?

Alarm equivalence how do we determine whether different alarms are the same? For example, how do we match an alarm-raise with a corresponding alarm-clear?

We will elaborate on these issues in the rest of this paper.

In sum, it is important to understand the overall alarm situation from an operator’s perspective. Figure 3 illustrates 3 months of alarm data. There are about 4 million alarm notifications per month (90 notifications per minute). The line ‘managed alarm’ shows the number of alarms represented in the alarm management system where duplicates and updates of the same alarm are grouped. ‘Alarms with

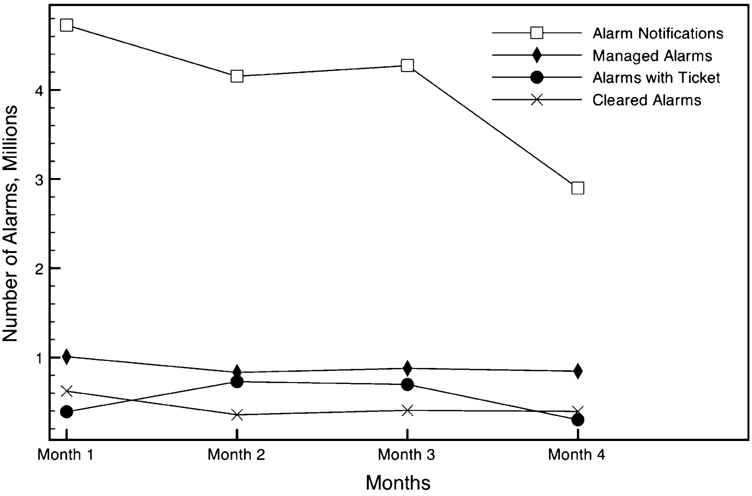

Fig. 3 Number of alarms

Ticket' shows how many of the alarms that are handled by operators using a trouble-ticket system to control the diagnosis and resolution process. We also show the number of alarms that are cleared by the resource. Figure 3 clearly illustrates the large number of alarm notifications that need to be managed, and the over-representation of alarm notifications versus the alarms that actually need manual attention.

The current alarm situation was expressed in the following way in an operator survey we made [13]:

> [*around*] *40% percent of the alarms are considered to be redundant as many alarms appear at the same time for one 'fault'. Many alarms are also repeated* [...]. *One alarm had for example appeared 65,000 times in today's browser. Correlation is hardly used even if supported by the systems,* [*the current correlation level is*] *1–2% at most.*

2.2 The Alarm Chain

Alarms are transformed and managed in a chain of abstractions (Fig. 4). First, we have states that manifest themselves as alarms. Every managed resource has a *set of* alarm states. This comes naturally to tangible resources like hardware and software. But from a management perspective, logical objects representing functions and their alarm states are also of vital importance. We denote the list of resources with associated alarm states $[R]$.

The alarm states themselves are in turn represented as entities of their own, often called an 'alarm list', which we denote $[A]$. The alarm interface published by the network element is the third step in the alarm transformation. It reports all changes to the alarms, ΔA, as alarm notifications. A management system will try to estimate the resource states, $[\overline{R}]$, and alarms, $[\overline{A}]$. Finally, alarm-management processes add characteristics to the alarm, $[M]$.

In Fig. 4 we see that we have two views of alarms: one resource-oriented view for A and one management view for M. These different alarm life-cycles are discussed in Sect. 3. We also see a chain of transformations that occurs:

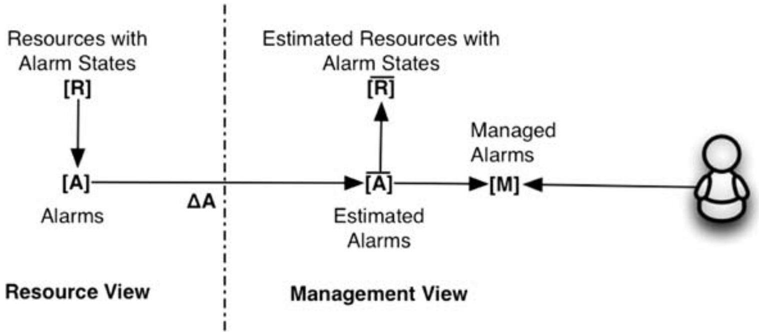

Fig. 4 The alarm chain

$$[R] \rightarrow [A] \rightarrow \Delta A \rightarrow [\overline{A}] \rightarrow ([\overline{R}],\ [M]) \tag{1}$$

in order to explain the state of resources and the overall alarm status.

3 Alarm State Machines

3.1 Resource Alarm States

The life-cycle of the resource alarm, A, is fairly simple and straightforward (Fig. 5). The main issue is to separate alarms that have a clear notification, Fig. 5a, or not, Fig. 5b. Note that we do *not* include administrative states of the alarm in this layer. We consider this to be part of the management view. Furthermore, in simplified terms, this represents the true state of the alarm, and not an approximation, as in the management system.

An alarm is created when the alarm first occurs. It may later change in severity, alarm text and possibly other non-identifying attributes. It then toggles between the active and clear states. A vital characteristic is hidden in this simple state-machine: an alarm going back and forward between active and clear is still the same alarm. Much equipment will generate new alarms for every toggle. This is common for flapping links, for example, and it might generate thousands of alarms. Rather, throttling and hysteresis functions should be used to send a few summary alarm notifications of the state changes. Administrative procedures like ageing may end the alarm object.

So far we have described state-full alarms. It is important to separate clearly state-full and stateless alarms. Administrators need to know if the alarm has a potential clear. Figure 5b illustrates the state-machine for stateless alarms. These kinds of alarms are common in many types of equipment where there is no notion of an alarm state. There is no instrumentation to detect the alarm clear state and therefore the alarm state is left to be judged by management applications or operators. The trap-directed polling architecture recommended by SNMP is a paradigm that promotes these kinds of solutions.

The alarm data we studied showed some interesting ‘clear’ phenomena connected to the simple state-machine. Our analysis shows that only some 50% of the alarms had an associated alarm clear at the end of the alarm life-cycle. Furthermore, a kind of anomaly occurs where we have an alarm clear reported for an alarm that is never reported—this accounted for almost 30% of the alarm notifications in the 3G

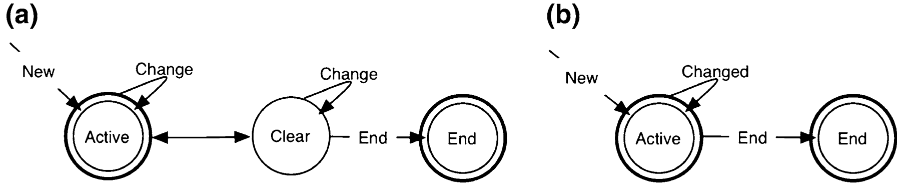

Fig. 5 Finite state machine for resource alarms: **a** state-full alarms, **b** stateless alarms

network, while it was less than 1% in the 2G network. This is probably a maturity concern and also an indication that the 3G network alarms are still suffering from low alarm quality.

3.2 Managed Alarm States

We have defined a Finite State Machine for managed alarms, *M*, as shown in Fig. 6. It is a simplification and abstraction of major standards and typical telecom management systems. In the end it can only be an example, since different processes and systems apply various alarm characteristics. It is also worth noting that the alarm state in the management system is always an *approximation* of the true alarm state.

From the resource point of view, the main notifications over the alarm interface, ΔA, are new and clear, which moves the alarm into the active and cleared state. Note, however, that the cleared state does not imply that the network administrator considers the problem solved, the trouble ticket process manages this. In order to manage the underlying fault, the user acknowledges and associates a trouble ticket with the alarm. We will refer to this as 'handling' the alarm. A trouble ticket contains information such as priority, affected services, and responsible work group. When the problem is solved, the administrator closes the trouble ticket. The life cycle of an alarm ends when a user decides that the alarm needs no further attention. This is indicated with the end state. In many cases, this is automatically performed

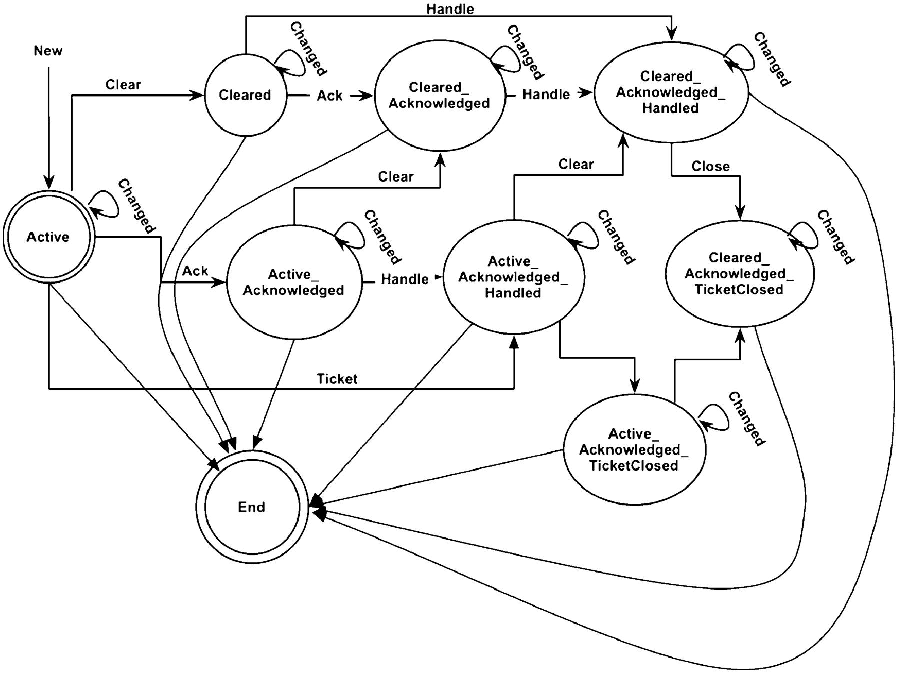

Fig. 6 Finite state machine for managed alarms

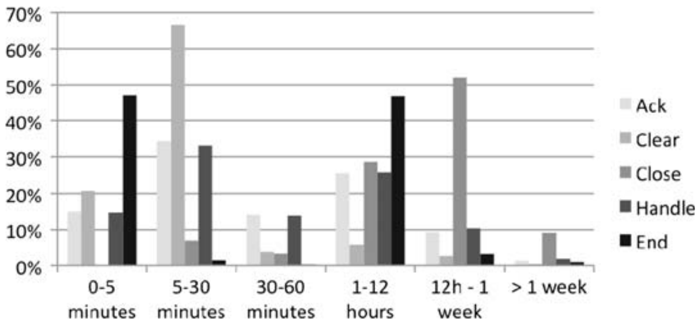

Fig. 7 Time distribution of state changes

when the associated trouble ticket is closed. There is a short cut, which bypasses the trouble ticket process in order to end alarms that do not represent real problems. Note that there is nothing that enforces alarms and managed alarms to end synchronously; different administrative processes may end the same alarm at different times.

The life cycle of an alarm can be studied by analysing the time distribution for the different state changes. In Fig. 7 we show the distribution of managed alarm state changes into time intervals. The figure shows some behaviour worth noting:

1. 20% of the alarms are cleared in less than 5 min, it is questionable if these should qualify as alarms at all and filtering these out would reduce the number of alarms.
2. 47% of the alarms are ‘ended’ within 5 min and this is again an indication of alarms that could be filtered out. Most of the alarm types in this case were automatically ‘ended’ by automatic rules and should never have been sent from the alarm producer in the first place.

We also studied the average times to *handle* and *end* alarms. Alarms with severity critical were on average handled more than 500 times quicker than alarms with severity warning. However, the alarm severity as emitted from the network element has low correlation with the priority as set by network administrators [5]. Therefore, the current quality of alarm severities as sent by the equipment is misleading and has a negative effect on network service.

4 Alarm Consumers and Producers

Currently most of the alarm improvement techniques are introduced on the consumer side. The actual *users* of the equipment are investing in filtering, correlation and enrichment. The input for these efforts is an avalanche of low-quality alarms. Looking at the current situation we recommend that efforts should rather be put into the alarm-producing sources to define simple alarm interfaces with high quality alarm information.

4.1 Consumers

The classical usage of alarms is the *alarm management system* with human users, the *network administrators*. Informal easy-to-understand texts are preferred in favour of rigorous formal parameters where syntactical structures are sometimes in conflict with readability. Our studies [5] show that the operators mostly use the two X.733 fields of additional text and managed object while omitting all the others. In many cases they introduce an 'informal' additional managed object field in preference to the syntactical focused original field.

It is a well-established fact that the number of alarms is too high for operators to be able to handle the situation [1]. Hollywell and Marshall [14] showed that administrators could read and comprehend familiar, structured text messages at a rate of 15–30 messages per minute. But if the administrators are required to assess the information and take action, then the alarm rates must be much lower than this. The rates we have observed are above these limits, especially in larger failures in the network.

Another important characteristic is the number of alarm *types* (see Sect. 6.1). Every alarm type implies different hypotheses and consequent actions. So, for efficient network administration, the different alarm types need to be known [15].

There are several process definitions for the alarm handling process [7, 16, 17]. All of these more or less contain the following simplified steps: detection (detecting the fault), diagnosis (identifying the cause of the fault) and correction (dealing with the fault). A fundamental problem is that the current level of alarm quality only supports the first step [7]. Klein, Pliske, Crandall and Woods [18] stress the fact that '*we do not directly perceive faults. We notice the disturbances they produce—their symptoms, which we experience as the cues that alert us to the existence of a fault*'. In their research, problems are detected as part of an inference and hypothesis process. It is triggered by a discrepancy of expectancies. They further argue that the capability to identify problems is affected by stance, expertise and attention management.

While ITIL [12] and eTOM [19] describe structured processes to handle alarms, they give little insight into the actual reasoning process behind the diagnosis performed by human operators. Woods [20] describes the diagnosis process as *abductive reasoning*. Abductive reasoning can be described as 'inference to the best explanation'. It is a method of reasoning in which one chooses the hypothesis that would, if true, best explain the relevant evidence. Abductive reasoning starts from a set of accepted facts and infers their most likely, or best, explanations:

- $D =$ collection of observations, givens, facts
- H explains D, that is, H, if true, would account for D
- No other available hypothesis explains D as well as H does
- Therefore, H is probably true.

We see that the network administrators work with probabilistic hypotheses with Bayesian rather then objective probabilities. The probabilities are constantly tuned when new evidence, alarms, arrive. Therefore, they need to revise their hypotheses given new alarms. Furthermore, many hypotheses are dependent on checking previous and future alarms, giving a complex temporal dimension to the reasoning.

At the end they are challenged with time-pressure to act. Should they wait for more evidence or take the best option and try to resolve the problem with the given facts?

4.2 Producers

The *resource instrumentation* is the main alarm producer and is closest to the true alarm state. On the other hand, alarms generated at this level have limited knowledge of contexts and desired states and therefore too many alarms are typically generated with non-relevant severities.

Management systems can themselves generate or change alarms according to the received alarm state changes and local knowledge. This is a level that is currently underdeveloped and the management systems are more or less a one-to-one mapping to the network element alarm state and information.

Correlation systems and *alarm impact analysis tools* analyse basic alarms using contextual information to filter and/or generate higher levels of alarms.

Active and passive probes act externally to the resource and test its black-box behaviour. On the basis of the tests they can generate alarms referring to a non-functioning resource. A basic problem here is that of resource naming. The native resource alarms and the probes do not necessarily name the resource in the same way. The same state may be detected by the probe as well as the original alarm, which results in duplicate alarms.

5 Existing Alarm Interfaces

Alarm interfaces are at the heart of network providers' service assurance solutions. Numerous alarm interfaces need to be integrated into the overall network management solution. This is currently a costly and complex task despite available standards and technologies. We will in this section outline the most common strategies and give some qualifying measures.

Alarm interfaces normally define the following items:

1. Notifications to report alarm state changes, ΔA.
2. Definition of the alarm object attributes and states, A.
3. List of active alarms, . This is used by the management system to be able to synchronise the list of active alarms.
4. Alarm equivalence, $a_1 \sim a_2$: how can we compare whether two alarms a_1 and a_2 refer to the same alarm state?
5. Alarm type: what is the mechanism to identify different kinds of alarms?
6. Naming rules for referring to the alarming resource R (often called managed object).

5.1 X.733

X.733 [4], defined by ITU-T, is the *de facto* standard for alarm interfaces. Almost all later standard efforts are based on the alarm information defined in X.733 to some degree. It is targeting the interface between a network element and the management system. Therefore, it does not contain any management aspects of the alarms. The model is notification-focused, and it defines the actual alarm state-change messages. The notification focus has created some confusion as to whether alarms are isolated messages or a set of messages related to the same alarm. This was recently resolved primarily by the state-focused 3GPP standard.

The value of the standard lies in the definitions of the various alarm attributes. It defines a rich set of attributes to describe the meaning of an alarm, but many of them, such as 'proposed repair action', are seldom used since capabilities of the agents do not support the corresponding information.

X.733 provides two alarm equivalence mechanisms, one using managed object and alarm type (eventType, probableCause, specificProblem), the other based on a sequential notification identifier. The X.733 idea of having global standards for statically defined alarm types still creates concern in the management community. Many systems support probableCause but the definitions are in many cases not unique enough to be useful in alarm identification scenarios.

ITU-T has defined a corresponding 'Generic Network Model', M3100 [21] to model the resources in the network, R. Three attributes are used to represent the alarm status of an object: operationalState, alarmStatus (the highest alarm severity), and currentProblemList (list of alarms for this object).

5.2 3GPP

3GPP [9] defines alarms using a state-based definition rather than a notification-based definition as in X.733. It targets the interface between a mid-level manager and an upper-layer management system and defines the management systems alarm list, $[M]$, and the corresponding management alarm changes ΔM. This means that management state changes like 'acknowledgment' are reported as well. The alarm attributes cover X.733 with the addition of different time-stamps to cover the alarm object life cycle and management states.

3GPP introduces an alarm equivalence attribute *alarmId*. The exact semantics of alarmId are unclear. It is used as a key for referring to alarms over the interface; systems need to share the knowledge of alarmId. The relationship with the inherited X.733 equivalence mechanisms is not well described and the standard shows examples where the two mechanisms actually create conflicts. It is also questionable if an identifier for the actual alarms is a sound approach since resource alarms and managed alarms have different life-cycles—they can be ended at different occasions, see Sect. 3.

The 3GPP Fault Management *requirements* [8] defines two kinds of alarms:

ADAC Automatic Detect Automatic Clear, Fig. 5a
ADMC Automatic Detect Manual Clear, an alarm where the resource does not report clearance, Fig. 5b.

The corresponding 3GPP *Alarm IRP* [9] uses the term 'event' to denote a stateless alarm while the 3GPP fault management requirements [8] define events more generally as network events. These two (conflicting and overlapping) 3GPP definitions of events have created confusion around stateless alarms versus events and they are not related to the 'ADAC' and 'ADMC' definitions above. Certain alarms are stateless, such as authentication-failure. Is this then an event? On the other hand, there are events that are not alarms like 'board inserted'.

In order to handle alarms without a clear notification, 3GPP introduced the concept of 'manual clear', which means that an operator can clear an alarm. In our view a resource-initiated clear is very different from an operator considering the alarm as cleared. This should be represented as two different states in order for the model to be correct, since the first is a resource view while the latter is a management view.

Another issue of the 3GPP alarm definition is that it does not separate the clear state from the perceived severity of an alarm. This means that we cannot represent the severity of a cleared alarm.

3GPP selected Q3 and Corba as protocols. According to our operator surveys [13] this has added drastically to the costs of alarm integration. The associated 3GPP generic object model, "Network Resource Model" [22], uses distinguished names for naming objects. It defines general classes such as SubNetwork and ManagedElement. The model does not define any alarm state attributes for the classes.

5.3 IETF Alarm MIB

The Alarm MIB represents the resource view, A. It is different from the other alarm interface standards in that it introduces an 'alarm model'. The alarm model is capable of mapping native SNMP notifications and object states to alarm states using the following underlying assumptions:

- Not all notifications report alarm state changes.
- Not all state changes are reported with notifications. In this case variables in the agent can be read to deduce the state.

This is an interesting approach and addresses our alarm type definition in Definition 4 (see Sect. 6.1). The alarm MIB assumes that alarm types can be differentiated by separate SNMP notifications and one integer-based notification parameter (var-bind). The alarm types are explicitly expressed as instantiated rows in an alarm model table. This is a unique feature whereby a management system can query the managed system for a list of alarm types and the corresponding notifications. It certainly adds complexity and management systems need to implement a discovery mechanism for this table before notifications can be semantically interpreted. The background for this is understandable, as the telecom community has dedicated alarm notifications, but in the SNMP world we have notifications in the various MIBs, which do not tell whether these are alarms or not. Another approach would have been to produce an SNMP-based X.733 clone. But the problem then would be to manage the installed base of existing notifications.

Every alarm *type* is given a unique identifier called "alarm id", whereas in 3GPP the alarm id refers to an alarm *instance*. Alarm equivalence is not explicitly stated but from the examples given on alarm and alarm clearance we see that the resource identity together with alarm type are used.

All in all, the Alarm MIB has an interesting approach to actually modelling the alarms. On the other hand, it implies complexity and the industry's acceptance is still to be obtained.

A somewhat associated MIB is the ENTITY-MIB [23] and its extension ENTITY-STATE-MIB [24]. It is a MIB that in an abstract way models the logical and physical resources of a managed system, *R*. The resources are modelled as entities with attributes, relationships and states, including an alarm status.

5.4 TM-Forum Standards

TM-Forum defines a broad set of interfaces including different attempts like MTOSI [25] and OSS/J [26].

MTOSI added an alarm notification attribute to indicate if the alarm state is clearable: "*isClearable : True if the alarm/event represents a condition that will not be cleared at a later time.*" This is a solution to explicitly telling the upper management system and network administrators about the stateless/state-full nature of the alarm type. Furthermore, MTOSI has explicit definitions of alarm types.

OSS/J has made some notable adaptations of 3GPP. First, it introduces a separate clearance status of the alarm object. A separate clear status is important in order to support unclear and also occurs naturally since we can have a major alarm that is cleared.

In 3GPP, alarm objects are identified with an 'alarmId', which is a unique key in one alarm list. Since, however, OSS systems manage a collection of alarm lists, we need a key that is unique across different lists. This is the purpose of the OSS/J 'alarmKey' attribute. But again, we have an identifier without semantics, which needs to be shared between systems.

Finally, OSS/J also opens up the static probableCause problem in making it into a string defined at run-time.

5.5 Syslog

Syslog [27] was once created as a logging solution for sendmail. Syslog has since become the standard logging solution on UNIX systems and IP devices. As the name indicates it is targeting a general logging service and is therefore focused on distributing event messages in general. This includes debug messages as well as alarm notifications. Syslog as such is designed to be flexible and therefore puts few requirements on its usage. Interpretation of messages is therefore very much left to the consumer and is based on string pattern matching. There is no concept of alarm states and no explicit way to express clear severities. Still, the actual information quality is in many cases high compared to other interfaces.

The MSGID field identifies different event types, but further string matching may be required to guarantee uniqueness. The emitting resource is identified by (HOSTNAME, APP-NAME, PROCID). An internet-draft [28] has just been published that maps alarm information into syslog. It resolves how to map alarm severities (including clear), probable cause and resource identification into syslog messages.

It is interesting to note that syslog is the only standard of those we studied that gives time-stamps serious treatment and therefore avoids problems in interpreting time-stamps from different devices.

5.6 Summary of Alarm Standards

Table 1 gives a summary of relevant alarm standards as elaborated above. The following acronyms and terms are used:

- ET: event type, PC: probable cause, SP: specific problem from X.733.
- DN: Distinguished Name, a text-based hierarchical name of the resource such as paths in file systems, DNS or LDAP names.
- OID: SNMP Object Identifier.

6 Definition of Alarm

We will use an adapted stimuli-based definition from Hollifield [29] in order to provide a *functional* definition of the term alarm. This is in contrast to the alarm standards definitions which focus on descriptive definitions. The definitions are summarised in Fig. 8.

Our definition also builds on the definition of ‘event’ from the CIM Event Model [11] in the sense that we refer to the state of the resource.

Definition 1 An alarm signifies 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.

An alarm is defined by

$$A = (R, T), [(t, S, I)]$$

where R is Resource, T is Alarm Type, t is time, S is Severity, I is Information

Table 1 Overview of alarm standards

	X.733	3GPP	IETF alarm MIB	OSS-J	Syslog
Models	ΔA	$\overline{\Delta A}$ and $\overline{A}$	$\overline{\Delta A}$	$\overline{\Delta M}$	Event
Alarm type	ET, PC, SP	ET, PC, SP	Trap and var-bind	ET, PC, SP SP is string	–
Equivalence	Not if ID and MO or ET, PC, SP, MO	Alarm id	Alarm type and resource OID	Alarm key	–

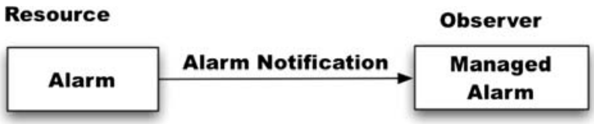

Fig. 8 Alarm definitions

The resource R and alarm type T identifies an alarm; this key is constant during an alarm life cycle. Note that this is logically based on alarm types and resource identifiers in contrast to for example the 3GPP alarmId. Alarm equivalence in our definition is based on matching the tuple (R, T). The other fields (t, S, I), are fields representing the history of the alarm, they form a tuple that is repeated for every alarm state change. Whether the meaning of an alarm is interpreted using the latest or highest severity depends on the observer. The state might be discrete, meaning it will only be reported once, with a single alarm raise notification as in Fig. 5b.

The focus of this definition is that every alarm needs attention and manual investigation including possible actions. This interpretation is close to the origins of the term 'alarm' which stems from the Old French, *à l'arme*; 'to the weapon', telling armed men to pick up their weapons and get ready for action (because an enemy may have suddenly appeared). It is also in line with the definition of alarms used in the process industry [30]: '*Alarm—a warning to the operator that immediate action is required to correct a condition in the plant*'. Note that an alarm based on this definition may not be associated with a fault. A traffic management application may generate an alarm when the utilisation of a link is getting too high.

Many conclusions can be drawn from this simple definition:

- Only important states representing abnormal situations that require action should be alarms.
- The alarms must be informative enough for the network administrator to assimilate his knowledge.
- The alarm must provide enough information to support reasoning, hypothesis and prediction.

Our definition also enforces network elements to have a model of alarms rather then emitting events whenever an implementer believes it is a good idea. We argue that the equipment provider *should* provide an alarm model for its equipment, rather then pass the problem to the service providers' management application. It is costly to understand events, create filters and map events to alarm states. These mechanisms should be performed by the emitting equipment.

Whenever the alarm changes state it is vital to inform subscribing and dependent systems of the status change. We call this an 'alarm notification'.

Definition 2 An *alarm notification* is a message about an alarm state change such as raise or clear. The alarm notification contains the change of an alarm

$$\Delta A = (R,\ T, t, S, I).$$

Finally, we have the representation of the alarm in management systems. This representation is calculated mainly from the received alarm notifications.

Definition 3 A *managed alarm* is the *management representation* of the alarm derived from alarm notifications. The managed alarm has added states and attributes that relate to the alarm management process.

If we denote a managed alarm by M, we can define it as

$$M = \overline{A} + m \tag{2}$$

where m is the pure management aspects of the alarm and $\overline{A}$ represents the management approximation of the resource alarm A. Managed alarms give the alarms further attributes and states related to the alarm handling process, for example alarm acknowledgement. It is worth noting that managed alarms may reside in the same system as the resource alarms in the case of an embedded system management function. If we go back to the previous discussion on the 3GPP manual clear, our standpoint is that resource clear is part of A whereas operator clearance is part of m.

6.1 Alarm Types and Instances

Alarm types, T, are of fundamental importance for modelling alarms; craft alarm interfaces, and structuring user interfaces and knowledge management databases. Dynamic definition of alarm types is the basis of any serious attempt to manage alarms more efficiently [31].

An alarm type is a classification of alarms that refers to the same underlying undesirable state. Examples of alarm types are LossOfSignal, BatteryFailure and AbisToBTSInterfaceFailure. Alarm types partly answer the question, if an alarm is the same as another alarm. The general rule is that two alarms refer to the same original alarm state if they refer to the same resource and alarm type, alarm equivalence. Formally we will use the following definition:

Definition 4 An alarm type, T, defines an equivalence class of alarms referring to the same undesirable state. It is defined as predicate P with associated severity S over the resource states. Every alarm type should have an associated operator instruction.

$$\mathrm{T} = ([(\mathrm{Predicate},\ \mathrm{Severity})],\ \mathrm{OperatorInstruction}).$$

The basis for an alarm type definition is to define the *undesirable states*, we represents this as a list of resource state evaluations and corresponding severity. In order to clarify our definition we show an example of the (Predicate, Severity) list of a linkOutage alarm type:

$$((\mathit{ifOperStatus} = \mathit{down} \wedge \mathit{ifAdminStatus} = \mathit{unlocked}), \mathit{Major}),$$
$$((\mathit{ifOperStatus} = \mathit{down} \wedge \mathit{ifAdminStatus} = \mathit{locked}), \mathit{Clear}),$$
$$((\mathit{ifOperStatus} = \mathit{up} \wedge \mathit{ifAdminStatus} = \mathit{unlocked}, \mathit{Clear}).$$

Note that we require that the operational state is not only ‘down’ but also differs from the administrative state of the interface. This will guarantee that the alarm represents a deviation from normal operation. However, if we compare this with the current practice of interface monitoring, most management applications will generate an alarm whenever the interface status is down irrespective of the administrative state. Furthermore, the alarm is cleared whenever the operational state equals the administrative state, including the case when operational state is down. The existence of a clear predicate makes the alarm type state-full according to Fig. 5a. Default severities are attached to the type but individual alarms can override the severity such as increasing the severity for important ports. The *operator instruction* gives instructions how to analyse and remedy the problem. If it is not possible to state an algorithm or heuristic to resolve the problem, it does not qualify as an alarm type.

One resource can only have one alarm type active at the same time. Alarm types must be fine-grained enough to guarantee that two underlying undesirable states do not map to one single alarm type. This restriction will allow for operators to work on isolated problems rather then events. Alarm types should not be confused with different notifications or states for the same alarm type. LinkUp and LinkDown are examples of the first case; these SNMP notifications refer to two states of the same alarm type, *linkOutage*. Different severity thresholds for the same problem are examples of the latter.

Some alarm types are specific to the managed object class while others are not. A battery problem can be related to a base station site as well as a toaster. It makes sense to further qualify the alarm types with a managed object class. We call this a *qualified alarm type*.

Definition 5 A qualified alarm type binds an alarm type to a specific resource class.

Maintaining and defining alarm types is cumbersome. There are several sources: network element types, standards, enterprises and network management systems. An integrated management solution needs a strategy for managing the type definition. Typical challenges are:

1. Addition of types at run-time
2. Different vendors using different names for the same alarm type
3. Too low resolution to enable unique alarm instance identification
4. Unclear predicates for alarm types
5. No specified operator instructions.

Alarm types are defined within a scope such as standard organisations or enterprises. A problem we see in current alarm practices is that a flat name space is used for alarm types like the global enumerated list of probable cause. We need a hierarchical name space making vendor extensions possible while maintaining a way of matching alarm types. Dot-separated names like DNS or X.500 would resolve this issue. Therefore, we assume T to be named like X733powerProblem.acme.upsfailure that specialises the X733 powerProblem into an ACME Inc. UPS failure.

When designing alarm types we find it useful to ask the following questions from BP Oil Alarm Management experiences [30]:

- *What is the purpose of the alarm? What potential hazard is the alarm intended to prevent?*

- *What are the consequences of the operator failing to respond to the alarm?*
- *What action is required of the operator?*
- *How likely is it that the operator will be able to prevent the hazard from occurring?*
- *How quickly is the operator required to respond?*

Every alarm type definition should have clear answers to these questions in order to qualify as an alarm. According to our previous studies [5] a minor subset (20 of 3,500) of all identified alarm types account for almost 90% of the trouble tickets. The same observation was done by Levy et al. [32]. This is an indication that it is important to study the alarm types and that automation of a very limited subset of all alarm types can to a large degree decrease the operational costs.

7 Alarm Taxonomy

We are analysing alarms using a taxonomy of four levels, partly borrowed from linguistics and semiotics.

- *Level* 0, *Phenomenon:* the resource state change or event that is interpreted as an alarm.
- *Level* 1, *Syntax and grammar:* the protocol and information modelling language to define the alarm interface.
- *Level* 2, *Semantics:* what does the alarm say?
- *Level* 3, *Pragmatics:* what is the meaning of the alarm when using contextual information?

The different levels in the taxonomy apply to different steps in the alarm chain (Fig. 9). The taxonomy can be illustrated with an example from a 3G network. At *Level* 0 a base station detects a problem considered to be an alarm, so the 3GPP Corba solution [9] set is used to transfer the alarm to the management system over *Level* 1. At *Level* 2 we interpret the various alarm fields such as managed object, severity, probable cause, and additional text to understand the Virtual Circuit alarm sent from a Radio Network Controller, RNC, node. It tells us that a virtual connection has been cut in the transport network between the RNC and Radio Base Station. Finally, at *Level* 3 we need to understand which cells and channels are affected by the alarm. At the last pragmatic level we investigate SLA and demographic data to understand the real impact of the problem. In the following sections we will use the alarm taxonomy to analyse the alarm concept.

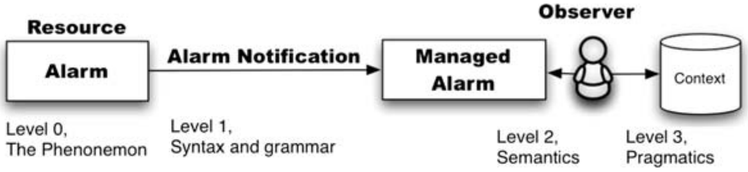

Fig. 9 Alarm taxonomy

7.1 Level 0: The Phenomenon

The first level is the most overlooked, and operators are flooded with messages that are not real alarms. Events that are for information only and not requiring actions should not be defined as alarms. Other important questions to ask before triggering the alarm are [29]: *Is this the best indicator of the situation's root cause? Is this alarm truly resulting from an abnormal situation?* Our investigations [5] show that the alarms at telecom network management centres do not comply with our basic definition of alarms (Definition 1). We studied an alarm database from a large mobile operator. It contained a total of more than 15 million alarm notifications for about 3.5 million alarms. A total of 90,000 trouble tickets were associated with these alarms. This tells us that only 2.5% of the alarms required manual actions. Bransby and Jenkinson [1] report the usefulness of alarms in the plant industry. According to their studies, less than 50% of the alarms qualify as true alarms.

7.2 Level 1: The Syntax and Grammar

Level 1 in our alarm taxonomy is the basic level that defines the alarm interface. From a technical point of view alarm interfaces have not been very successful since integrations are still complex and expensive [13] despite the existence of various alarm interface standards. The modelling languages and corresponding models have been too informal, leaving room for a lot of variants and not enabling the semantic level. It is also worth noting that most of the efforts in the telecommunication industry focus on the actual alarm protocol with little attention given to defining an alarm model.

7.3 Level 2: The Semantics

The semantic level deals with understanding the alarm information. An alarm carries a defined set of parameters according to the grammar, but the contents need to be understood and provide meaning to the network administrator. Some challenges that appear at this level are: understanding the resource reference, too coarse-grained resource reference, incorrect alarm severities and semantics embedded in free text fields.

We illustrate the semantic problems by some examples of how to interpret the alarm severity. The severity alarm field provides an indication of how the capability of the resource has been affected. X.733 defines and enumerates the following values: Indeterminate, Warning, Minor, Major, Critical, Clear. Our studies [5] show that the pragmatic severity level assigned manually by network administrators has very little correlation with the severity passed in the alarm.

In Fig. 10 we show the distribution of the severities from our alarm database. We have also included a normalised plot of the distribution of manually assigned severities (priorities) in the associated trouble ticket system. The graph also shows the recommended severity distribution from Hollifield [29].

If we compare the total alarms with the recommended distribution we see that the distribution is over-represented for high severities. It is even more interesting to

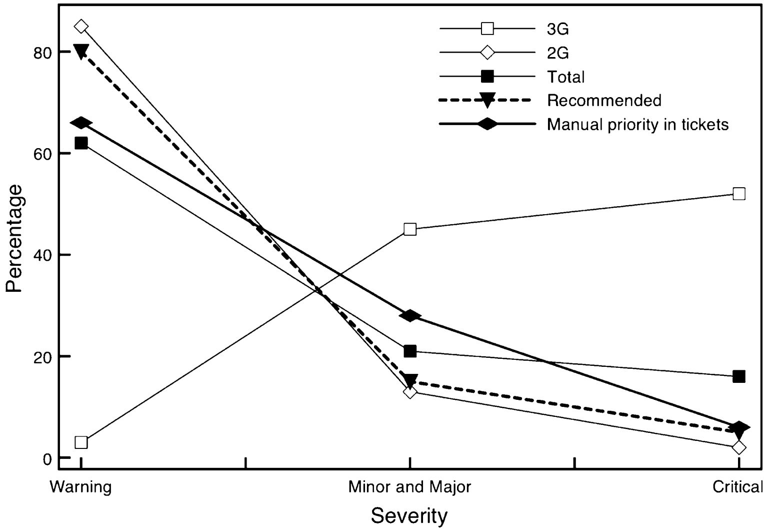

Fig. 10 Severity distribution

study the distribution of 2G and 3G alarms. The mature 2G technology is almost aligned with the recommendation whereas 3G alarms are orthogonal to the recommendation.

Several semantic challenges are also connected with the alarm time stamps. Time synchronisation is still an issue in network management environments. Some nodes do not have an internal clock, which means that time-stamps are introduced later in the chain, networks cross time-zone borders, and not all nodes use the network time protocol. The introduction of post-processing tools and probes also introduces challenges on how to time-stamp alarms retrospectively.

Finally, we make some observations on semantics embedded in the additional text field of alarms. In the worst case, alarm identity information like managed object instance or alarm type is embedded as part of the additional text. A common pattern is that the managed object parameter only points to the box, but the additional text contains detailed information such as rack/slot/port. Our experience is that semantic alarm information is embedded in different ways in additional text and it has vital importance compared to the other, at least on a syntactical level more formal, attributes. It is interesting to quote the X.733 statement about additional text: '*Understanding the semantics of this field is not required for interpretation of the notification*'.

7.4 Level 3: The Pragmatics

Finally, at the pragmatic level we need to understand the impact and context of the alarm:—*What is the effect on services and customers?—What needs to be done?* Network administrators use contextual information like topology and SLAs to understand the alarm pragmatics. Most important of all is their informal expert knowledge. In current network management solutions contexts are mostly added manually in associated trouble tickets.

Although contexts are fundamental in the interpretation of the alarms, the term context is a slippery one. As Dourish points out, contextual information is '*defined dynamically*, *is an occasioned property and arises from the activity*' [33]. Apart from the above examples of context which indicate business and network information, we have contextual information related to the resource itself. Alarms mean a different thing if the alarming system is, for example, in a start-up state or part of a protection configuration. These sorts of system contexts should be presented as part of every alarm. We also believe that there is still much potential for improving knowledge management solutions supporting alarm management.

8 Related Work

The process control and power industry areas seem to have a high degree of research focusing on the alarm concept as such [1, 34, 35]. This is probably because of the human safety risks involved. Much of our work on the underlying concepts are based on Stanton's [35] inspiring work on alarms. Stanton emphasises the meaning of an alarm at the same time as a concern for the contextual interpretation of alarms.

Alarms are often modelled as finite state machines as in the work by Rouvellou [36]. In this approach, the specific state machines for specific alarm types need to be discovered in an initial step. At run-time the various alarm notifications are applied to the correct state machines. Rouvellou shares our basic assumptions that all alarms carry alarm identification information and a time-stamp. The model supports basic alarm operations: add, delete, change, re-order (sort). Deletion of alarms is relevant in several cases such as filtering and the underlying resource being removed from operations. This is a solid approach and maps to the typical telecommunication needs of mapping events to alarms. However, the complexity of actually discovering and understanding the different state machines has forced us to relax this approach and rather retain the history of state changes. If, at the end, a state machine is known it can be derived from our model.

Zheng et al. [37] define an alarm model in order to support their data-mining approach to finding correlation rules. They define an alarm type in the following way:

> *An alarm type is defined as* e_i = *<object_class, object_instance, alarm_num, desc>,* […]*, alarm number is the alarm type* […]

This is an example of the problem of defining alarm types since it easily becomes recursive. Bellec and Kechadi [38] provide a model of alarms as the last step in the chain: fault, error, symptom, alarm. They separate the static and time-dependent attributes of an alarm and argue that alarms are identical if they share the same static attributes. This maps to our definition of alarm type and is also an interesting approach to defining alarm equivalence.

While we have assumed a strict equivalence operator, Julisch [39] uses taxonomies of alarm attributes to find alarm clusters. By using taxonomies, alarms and alarm types can be matched on a semantic level not really available in the alarm attributes as such.

Event algebras and active databases [40–42] represent another relevant research area with connections to the alarm model. Event algebras define temporal patterns of discrete events to form composite events. Composite events can be compared to alarms based on individual alarm notifications (events), see for example Zimmer's work [43]. Zimmer defines event types and event instances in the following manner:

> *An event type* […] *describes on an abstract level the essential factors that unambiguously identify the occurrence of an event of that type* […] *Each concrete occurrence of an event type is represented within the system by its specific event instance* […], *whose main task is to save all relevant information about the event in parameters.*

Network event recognition engines like NERL [44] represent a connected category of related work. This is an approach where domain-specific languages are used to express models of network events. These are possible architectures to implement an alarm notification engine where our alarm model is used as input.

Research efforts around alarms are mostly focused on alarm correlation. The focus of alarm correlation is to interpret alarm notifications and give (new) meaning to them [45]. Alarm correlation research efforts address several underlying problems: correlation of notifications relating to the same underlying fault, repeated alarm notifications, the same fault as identified by several components, and fault propagation [46]. Our definition of alarms would ease the alarm correlation effort since the input to the correlation engine would from the beginning be qualified as an undesirable state.

9 Conclusions and Future Work

We have analysed and proposed definitions for the alarm and alarm type concepts. Our definitions point to alarms as deviations from normal operation that require operator intervention. On the other hand, we have analysed alarm data from a large mobile network and found that the majority of the alarm notifications from network elements do not represent true alarm states according to our definition. Service providers therefore spend operational and capital expenditures in alarm filtering and correlation efforts in order to understand the network's status. We argue that the situation for network service providers could improve drastically if alarm sources were compelled to comply with our alarm model.

Our future work includes an alarm model that expresses our definitions in a formal framework. This will enable vendors and standards to express the alarm interfaces in a more stringent way and therefore reduce integration costs and improve alarm quality.

Acknowledgements Thanks to Jörgen Öfjell at Data Ductus for his excellent work with the alarm databases.

References

1. Bransby, M.L., Jenkinson, J.: The management of alarm systems. HSE Contract Res. Rep. **166**, (1998)
2. Verbrugge, S., Pasqualini, S., Westphal, F.J., Jager, M., Iselt, A., Kirstadter, A., Chahine, R., Colle, D., Pickavet, M., Demeester, P.: Modeling operational expenditures for telecom operators. In Optical Network Design and Modeling, 2005, Conference, pp. 455–466 (2005)
3. Schonwalder, J., Pras, A., Martin-Flatin, J.P.: On the future of Internet management technologies. Commun. Mag. IEEE **41**(10), 90–97 (2003). doi:10.1109/MCOM.2003.1235600
4. ITU-T. X.733: Information technology—Open Systems Interconnection—Systems Management: Alarm Reporting Function (1992)
5. Wallin, S., Leijon, V., Landén, L.: Statistical analysis and prioritisation of alarms in mobile networks. Int. J. Bus. Intell. Data Min. (2009) (To Appear)
6. Websters online dictionary, July 2008
7. Stanton, N.: Modelling human alarm initiated activities: implications for alarm system design. In: Man-Machine Interfaces for Instrumentation, IEE Colloquium, p. 8 (1995)
8. 3GPP. 3GPP TS 32.111-1: 3G fault management requirements (2007)
9. 3GPP. 3GPP TS 32.111-2: Alarm Integration Reference Point (IRP) (2007)
10. Romascanu, D., Chisholm, S.: Alarm Management Information Base (MIB), RFC3877 (2004)
11. DMTF: CIM Event Model White Paper, June 2003
12. itSMF. Service Management—ITIL Version 3 Publications, Service Operations. Office of Government Commerce, ITIL Version 3 Publications (2007) http://www.best-management-practice.com
13. Wallin, S., Leijon, V.: Telecom network and service management: an operator survey. In: 12th IFIP/IEEE International Conference on Management of Multimedia and Mobile Networks and Services (2009) (Submitted)
14. Stanton, N.: Human Factors in Alarm Design, chap. 3. An experiment to support the design of VDU-based alarm lists for power plant operators. Taylor & Francis, London (1994)
15. Edworthy, J., Hellier, E.: Alarms and human behaviour: implications for medical alarms. Br. J. Anaesth. **97**(1), 12–17 (2006). doi:10.1093/bja/ael114
16. Edwards, E., Lees, F.P.: The Human Operator in Process Control. Taylor & Francis Group, London (1974)
17. Marshall, E., Baker, S.: Alarms in nuclear power plant control rooms: current approaches and future design. In: Human Factors in Alarm Design, pp. 183–191. Taylor and Francis (1994). ISBN:0-74840-0109-1
18. Klein, G., Pliske, R., Crandall, B., Woods, D.D.: Problem detection. Cogn. Technol. Work **7**(1), 14–28 (2005). doi:10.1007/s10111-004-0166-y
19. TeleManagement Forum. Business process framework (eTOM). GB921, version 7.3, July 2008
20. Woods, D.D.: The alarm problem and directed attention in dynamic fault management. Ergonomics **38**(11), pp. 2371–2393 (1995). doi:10.1080/00140139508925274
21. ITU-T. M.3100: Information technology—Open Systems Interconnection—Systems Management: Generic Network Model. (2005)
22. 3GPP. 3GPP TS 32.622: Network Resource Model (2003)
23. Bierman, A., McCloghrie, K.: Entity MIB (Version 3). (4133), August 2005
24. Chisholm, S., Perkins, D.: Entity State MIB. RFC 4268 (Proposed Standard), November 2005
25. Caruso, F., Milham, D., Orobec, S., Technologies, T.: Emerging industry standard for managing next generation transport networks: TMF MTOSI. In: Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP, pp. 1–15 (2006)
26. TeleManagement Forum: OSS/J. http://www.tmforum.org/ossj/. Accessed 14 Aug 2008
27. Gerhards, R.: The Syslog Protocol. RFC 5424 (Proposed Standard), March 2009
28. Chisholm, S., Gerhards, R.: Alarms in syslog, 2009. http://www.ietf.org/internet-drafts/draft-ietf-opsawg-syslog-alarm-01.txt. Accessed 18 May 2009.
29. Hollifield, B., Habibi, E.: The Alarm Management Handbook, p. 21. PAS (2006). ISBN:0-9778969-0-0
30. Brown, D.C.C.: Alarm management experience in BP oil. In: Best Practices in Alarm Management (digest no. 1998/279), IEE Colloquium, pp. 1/1–1/10, March 1998
31. Grossglauser, M., Koudas, N., Park, Y., Variot, A.: Falcon: fault management via alarm warehousing and mining. In: Proceedings of Workshop on Network-Related Data Management (2001)

32. Levy, D., Chillarege, R.: Early warning of failures through alarm analysis—a case study in Telecom Voice Mail Systems. In: Proceedings of the 14th International Symposium on Software Reliability Engineering, p. 271. IEEE Computer Society Washington, DC, USA (2003)
33. Dourish, P.: What we talk about when we talk about context. Pers. Ubiquitous Comput. **8**(1), 19–30 (2004). doi:10.1007/s00779-003-0253-8
34. Hollifield, B., Habibi, E.: Alarm Management Handbook. PAS (2006). ISBN:9778969-0-0
35. Stanton, N.: Human Factors in Alarm Design. Taylor & Francis, London (1994)
36. Rouvellou, I., Hart, G.W.: Automatic alarm correlation for fault identification. In: IEEE INFOCOM'95. Fourteenth Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 2, pp. 553–561, April 1995
37. Zheng, Q., Xu, K., Lv, W., Ma, S.: Intelligent search of correlated alarms from database containing noise data. In: 2002 IEEE/IFIP Network Operations and Management Symposium, 2002. NOMS 2002, pp. 405–419 (2002)
38. Bellec, J.H., Kechadi, M.T.: Towards a formal model for the network alarm correlation problem. In: 6th WSEAS International Conference on Power Systems (PE'06). World Scientific and Engineering Academy and Society, Ag. Ioannou Theologou 17–23, Athens, 15773, Zographou (2006)
39. Julisch, K.: Mining alarm clusters to improve alarm handling efficiency. In: Computer Security Applications Conference, Annual, 0:0012 (2001)
40. Carlson, J., Lisper, B.: An event detection algebra for reactive systems. In: Proceedings of the 4th ACM International Conference on Embedded Software, pp. 147–154. ACM New York, NY, USA (2004)
41. Hinze, A., Voisard, A.: A parameterized algebra for event notification services. In: Proceedings of the 9th International Symposium on Temporal Representation and Reasoning (TIME 2002). Manchester, UK (2002)
42. Mishr, D., Chakravarthy, S.: Snoop: an expressive event specification language for active databases. Knowl. Data Eng. J. **14**, 1–26 (2004)
43. Zimmer, D., Unland, R.: On the semantics of complex events in active database management systems. In: Data Engineering, 1999. Proceedings, 15th International Conference, pp. 392–399 (1999)
44. Bhargavan, K., Gunter, C.A.: Network event recognition. Form. Methods Syst. Des. **27**(3), pp. 213–251 (2005). doi:10.1007/s10703-005-3398-4
45. Jakobson, G., Weissman, M.: Real-time telecommunication network management: extending event correlation with temporal constraints. In: Proceedings of the Fourth International Symposium on Integrated Network Management, table of contents, pp. 290–301. Chapman & Hall, Ltd., London (1995)
46. Houck, K., Calo, S., Finkel, A.: Towards a practical alarm correlation system. In: Proceedings of the Fourth International Symposium on Integrated Network Management, table of contents, pp. 226–237. Chapman & Hall, Ltd. London (1995)

Author Biography

Stefan Wallin received his M.Sc. at Linköping University in Sweden in 1989. Since then, he has worked with network management solutions and standards; first for Ericsson from 1989 until 1995 and thereafter as a Senior Partner at Data Ductus AB, Sweden. Most of his work addresses integrated management solutions for mobile operators. Stefan is an appreciated trainer and speaker in the area of network management. He is also a part-time Ph.D. student at Luleå University. His research interests are network and service monitoring for mobile networks.