LICENTIATE THESIS

Alarm and Service Monitoring of Large-Scale Multi-Service Mobile Networks

Stefan Wallin

Luleå University of Technology

Alarm and Service Monitoring of Large-Scale Multi-Service Mobile Networks

Stefan Wallin

ICT
Dept. of Computer Science and Electrical Engineering
Luleå University of Technology
Luleå, Sweden

Supervisor:

Associate Professor Christer Åhlund and Dr. Evgeny Osipov

Tryck: Universitetstryckeriet, Luleå

ISSN: 1402-1757
ISBN 978-91-86233-34-1

Luleå 2009

www.ltu.se

And here he remained in such terror as none but he can know, trembling in every limb, and the cold sweat starting from every pore, when suddenly there rose upon the night-wind the noise of distant shouting, and the roar of voices mingled in alarm and wonder. Any sound of men in that lonely place, even though it conveyed a real cause of alarm, was something to him. He regained his strength and energy at the prospect of personal danger; and springing to his feet, rushed into the open air.

- Charles Dickens, Oliver Twist

Abstract

Two of the most important challenges in network service assurance are

- an overwhelming flow of low-quality alarms
- understanding the structure and quality of the delivered services

This thesis proposes solutions for alarm and service monitoring that address monitoring of large scale multi-service mobile networks.

The work on alarms is based on statistical analysis of data collected from a real-world alarm flow and an associated trouble ticket database containing the network administrators' expert knowledge. Using data from the trouble ticketing system as a reference, we examine the relationship between the original alarm severity and the human perception of the alarm priority. Using this knowledge, we suggest a neural network-based approach for alarm prioritization. Tests using live data show that our prototype assigns the same severity as a human expert in 50% of all cases, compared to 17% for a naïve approach.

In order to model and monitor the services, this thesis proposes a novel domain-specific language called SALmon, which permits efficient representation of service models, along with a computational engine for evaluation of service models. We show that the proposed system is a good match for real-world scenarios with special requirements around service modeling.

Contents

Preface

This thesis is based on close to 20 years of industrial experience working with mobile operators in providing network and service management solutions. Many of these solutions have been unnecessarily complex and costly and have not supported service and customer-oriented functions to the desired degree. While frustration is not always the best starting point, I thought it was time to both summarise my experience and to consider some necessary changes in the field.

In order to focus on relevant subjects, I have worked with colleagues, partners and customers – service and equipment providers – to understand the challenges and changes that are needed. As such, this is the correct place to thank all the various friends who have helped me gather vital information on the subject. Also, I wish to thank the operator that provided me with a large database of alarms and trouble tickets.

Thanks to Christer Åhlund, my supervisor, who lets me work in a chaotic and flexible mode of operation. Also, I wish to thank the whole team at LTU in Skellefteå for a great working atmosphere. I do much of my work with Viktor Leijon, whose qualifications as a devil's advocate and at the same time supporting research colleague have pushed the results to a higher level.

I'd like to thank my partners at Data Ductus, Urban Lundmark, Stefan Lundström and Lennart Wiklund for letting me spend time on this project.

David Partain has provided both substantial insights and language reviews; I owe him several pints. Thanks to Gary Webster as well for help with language reviews.

My research is based on struggling with network management challenges in the industry. There would have been no challenges if Ingemar Häggström had not kept throwing assignments on my desk from his different positions at Ericsson, Hewlett-Packard and Ulticom.

Finally, and most important of all, my complete gratitude and love to my family: Inger, Axel and Ellen, my everything.

Part I

Chapter 1

Thesis Introduction

1.1 Introduction

Network service providers have rapidly transitioned from providing few services over fixed lines with limited competition to multi-faceted Internet-based services with intense competition. Not only do operators provide different *access* networks such as xDSL, optic fiber, and mobile broadband, they are also struggling with new *consumer services* like IP telephony and mobile TV. Traditional network operators are extending their business in several countries with local competition, at the same time as new players like the utility market are entering the service provider arena. In Paper A, we look at how the business model for operators has changed drastically while the corresponding business and network support systems have troubles dealing with the new needs.

Based on our industrial experience we asked major telecom service providers to tell us about their network management problems and expectations for the future. According to this survey, presented in Paper B, the following requirements will drive the solutions in the coming years: *Service management and quality*, *Customer satisfaction*, and *Cost reduction*. One operator summarized the current status of service management as followings:

> *Services are not currently managed well in any suite of applications and require a tremendous amount of work to maintain.*

One of the main themes of this thesis is service quality. This is obviously also a major contributor to achieving customer satisfaction.

We also asked the operators to identify the most important research areas. Alarm correlation, self-healing, and auto-configuration were identified as the most urgent research topics. Out of these we picked alarm correlation but with a focus on filtering and prioritization rather than traditional correlation. This means that we see other ways of reducing the number of alarms and at the same time providing better alarm quality.

In many cases, alarm messages go unaltered, unfiltered, and uncorrelated from the network elements to the network management system. This leads to a chaotic situation

which needs to be improved before we can move into service and customer management. The current alarm situation was expressed in the following way by one of the operators:

> *[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 65000 times in today's browser. Correlation is hardly used even if it supported by the systems, [current correlation level is] 1-2 % maybe.*

If we could find ways to deliver only the relevant alarms, this will result in cost reductions and better use of expert network administrators.

Aside from these prioritized items, the operators mentioned interface integration, service quality and service modeling as relevant research areas. Interface integration will be part of our future work, while service quality and modeling is our current focus.

1.2 Research Topics

Based on the background surveys, we focus on two contrasting themes: the problems of *alarm management* and a future solution for *service modeling and monitoring*. These two areas contrast with each other in many ways. While alarm management is an existing solution, this is not true for service models and corresponding service-focused monitoring functions. Alarms are a necessary pain connected with cost, and although the problems have been around for decades we have not seen any major improvements. Service modeling on the other hand is a journey about to start.

1.2.1 Alarm Management

Alarm management is an area with an extensive research community. Most of the current research focuses on alarm correlation [1]. This is an approach where the existing alarm rates and alarm quality are accepted and correlation techniques are applied at the end of the alarm flow. It is costly developing the correlation solutions and coping with the alarm rates. We believe that the situation would improve if we focused on getting the right alarms from the outset. As a starting point we are trying to find the characteristics of alarms using a database of alarms and associated trouble tickets from a large mobile operator.

Our findings so far are that filtering and prioritizing alarms will improve the situation (see Section 2.2). Paper D elaborates on our two hypotheses in this area: that statistical analysis techniques can be used to find alarm filtering strategies and that a learning neural network could suggest relevant priorities by capturing network administrators' knowledge.

1.2.2 Service Modeling and Monitoring

Operators want to manage services rather than the network resources which are used to deliver the services. This change of focus is driven by several factors, including increased competition and more complex service offerings. A result of this change is an increasing need to predict, monitor and manage the quality of the service that is delivered to the end users. However, the complexity of understanding and modeling services is a serious obstacle.

In section 2.3 we suggest a possible solution for this called SALmon. The main components of SALmon are a dedicated service modeling language and a run-time environment for calculating the service status. The language is an object-oriented functional language tailored to the domain-specific requirements. SALmon tries to address some of the drawbacks and features missing in available commercial service monitoring systems, features such as time-awareness, flexibility, and well-defined semantics. A description of the work up to this point can be found in Papers C and E. An example of how SALmon can be used as a component in a solution is given in [2].

1.3 Related Work

1.3.1 Alarm Management

Most research efforts related to alarm handling focus on alarm correlation. A separate thesis is required to present an overview of alarm correlation efforts. See, for example, Meira [3]. Alarm correlation refers to the process of grouping alarms that have a reciprocal relationship [4] and the process of determining the root cause. The majority of these solutions are rule-based. Rule-based systems represent knowledge as a conditional if-else clause. Jakobsson's [5] work is a representative example of these kind of solutions.

As Sterrit [6] points out: *"correlation approaches based on rule development struggles with problems like knowledge acquisition bottleneck, maintenance burden, hidden implicit and tacit knowledge"*. Our work aligns with Sterrit's approach in that we use data mining techniques to acquire the domain knowledge.

Data mining, or knowledge discovery in databases, is being used in different domains such as finance, marketing, fraud detection, manufacturing and network management [7]. Bose et al. [8] performed a study to analyze the usage of data mining techniques across domains and problem types and found that 7% of the usage was in the telecom sector. They also identified five major categories of machine learning: induction, neural networks, case-based reasoning, genetic algorithms, and inductive logic programming.

The following problem types are typically addressed with data-mining:

- *Classification*, where the training data is used to find classes of the data set. New data can then be analyzed and categorized. This is the main theme of our work, where we are looking for trouble ticket priorities based on the alarm data.
- *Prediction* focuses on finding possible current and future values such as finding and forecasting faults in a telecommunication network.

- *Association* attempts to find associations between items such as dependencies between network elements and services [9].

- *Detection* tries to find irregularities in the data and seeks to explain the cause. A representative telecom application is to detect churn and fraud.

Gardner's work [10] belongs to the classification category where a self-organizing map, a Kohonen network [11], is used to categorize alarms. The primary application of Kohonen maps is analysis and classification of input where unknown data clusters may exist. This is in contrast to our prioritization scenario where we *have* a complete record of the correct output in the trouble ticket database.

Sasisekharan [12] combines statistical methods and machine learning techniques to identify "patterns of chronic problems in telecom networks". Network behavior, diagnostic data, and topology are used as input in the solution. This solution covers the challenging aspect of problem prediction and focuses on the data-mining techniques. Levy et al. [13] also combine data mining and machine learning in an alarm system. They come to the same conclusion as we do regarding the Pareto distribution of the alarms: "there is a lot of value in the ability to get an early warning on the 10% of causes that create 90% of field failures" [13]. This further emphasizes the foundation for the work presented in this thesis where we are focusing on pinpointing the relevant alarms to be handled.

Klemettinen [14] presents a complementary solution to alarm correlation, using semi-automatic recognition of patterns in alarm databases. The output is a set of suggested rules that the user can navigate and understand. This approach shows that data-mining is a way forward to solve the alarm management problem. Wietgrefe [15] uses neural networks to perform alarm correlation in order to find the root cause alarm, the learning process is fed with alarms as inputs and the triggering alarm as output. The results presented by Wietgrefe show that telecom alarm correlation based on neural networks behaves well in comparison to traditional rule-based approaches.

1.3.2 Service Modeling

There are three main classes of service modeling approaches: techniques currently used by service providers, industry and standard initiatives and finally research initiatives. The different available approaches are illustrated in Figure 1.1 and elaborated below.

1.3.2.1 Current Approaches

Current deployed approaches can be separated into *static* and *dynamic* service modeling solutions. *Static* solutions focus on maintaining the service model as such. The model instances are updated by batch-oriented routines driven by the provisioning process and uploads/discovery from the various network elements. In its simplest form *static* solutions are databases or spreadsheets with inventory-based information. *Dynamic* solutions on the other hand try to calculate the service state by using sources like alarms, probes and

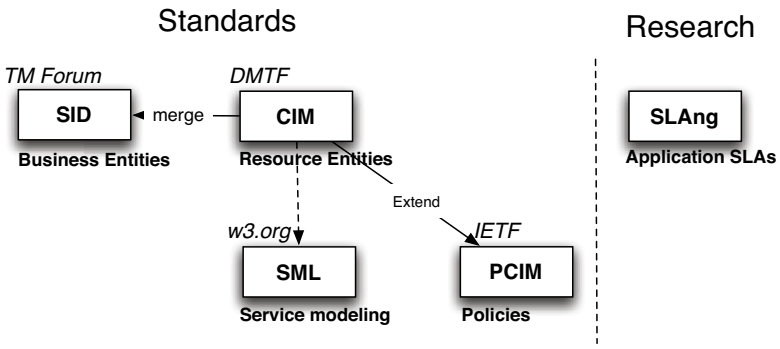

Figure 1.1: Service Modeling Languages

trouble ticket systems. They are in general less powerful with respect to service modeling and depend on an external service model database in order to create the service instances.

We focus on a small language, dedicated to service modeling and capable of handling both the static and dynamic aspects of service models.

1.3.2.2 Industry and Standard Initiatives

One of the most important sources for service and SLA modeling is the SLA handbook from TM Forum [16]. It provides valuable insights into the problem domain but not to the modeling itself. TM Forum has also defined an accompanying service model, SID, "System Information Model" [17]. SID is comparatively high level and models entities in telecom operators' processes. However, SID is being refined and moving closer to the resources by incorporating the Common Information Model, CIM [18]. The IETF has extended the CIM model to manage policies [19]. However, the policy work within IETF has not really taken off. CIM has an extensive and feature-rich model including a modeling language, *MOF* (Managed Object Format). MOF is based on the Object Management Group's Interface Definition Language, IDL. Key strengths in CIM are the modeling guidelines and patterns. However, CIM faces some major challenges since the UML/XML approach tends to create unwieldy models. It is also aimed more at instrumentation than end-to-end service modeling.

Some of the major players behind CIM are now working on the "Service Modeling Language", SML [20]. SML is used to model services and systems, including their structure, constraints, policies, and best practices. Each model in SML consists of two subsets of documents; model *definition* documents and model *instance* documents. SML is certainly a step forward and the work is based on experience gained with the CIM model.

It is also worth mentioning YANG [21], the data modeling language for NETCONF [22]. Although not targeting service modeling as such, it seems to be making a footprint in the resource-oriented network management domain. YANG is targeting configuration

data and not services and service parameters. However in the long run we expect to see YANG attributes as inputs to service models.

CIM, SML, and similar efforts focus on static class diagrams to express the service model. They are not intended for monitoring, but rather to help with the construction and composition of services, and tend to lack formally defined semantics. Our solution adds the capability to express the functional QoS calculations and time-series of QoS data. While there has been some effort put into graphical modeling tools, our experience is that it is easy to expect too much from graphical modeling tools for service modeling. This can to some degree be compared to the partial failure of graphical CASE tools compared to standard programming methods [23].

1.3.2.3 Research Initiatives

Garschhammer et al. [24] outline a generic service model which includes both some of the vital concepts for service modeling and the interaction between providers and users. This work has bearing on the design of generic model components, and forms a pattern for service model development in general. Garschhammer also identifies vital research items for service management:

1. The interaction and mapping between the abstract service model and the corresponding service implementation.
2. Mapping of the service-oriented QoS parameters and the service implementation.
3. Mapping of service agreements across service chains.
4. Interaction between customer and provider.

To some degree we are addressing the two first mapping problems by parameter calculations and explicit support for lower-level QoS inputs. We also support service chaining by references between service objects.

While Garschhammer establishes requirements for a generic service model, Gopal [25] identifies requirements for a service modeling language such as aggregation of components and calculations, and list-valued attributes, which we can express in SALmon. Marilly et al. [26] identify a set of main challenges in order for SLA management to appear. We have addressed three out of four challenges, namely: information model, scalability, and end-to-end view. Räisänen elaborates service management and mobile networks in [27]. Special attention is given to the resource allocation mechanisms in order to provide the required quality of service. The author also points to the need of an integrated aggregated customer-focused SLA across services.

SLAng [28] is a language focused on defining formal SLAs in the context of server applications such as web services. It uses an XML formalism for the SLAs. SLAng identifies fundamental requirements needed in order to capture SLAs but differs from our current effort in that it "focuses primarily on SLAs, not service models in general". SLAng is being further developed by UCL.

Chapter 2

Towards Better Network Management Solutions

2.1 Introduction

In this section we summarize our work on alarm and service management. At this point we want to emphasize the link between the two areas. A service model needs inputs about the status of the service from various sources. One of the sources is alarms. However, in order to use alarms to indicate the service state, we need to get the right alarms and the right severities for these alarms, rather than the current unreliable alarm quality. This is why the alarm improvements presented in Section 2.2 are a stepping stone towards using alarms as input to service models as outlined in Section 2.3.

The section on alarms, Section 2.2, is based on two articles [29, 30] of which the latter is included in this thesis as Paper D. Section 2.3 on service modeling is based on three articles[31, 2, 32] of which two are included in this thesis as Papers C and E

2.2 The Chaotic Alarms

Network administrators are flooded with alarms which require action, either to resolve the problem or to define the alarm as irrelevant. Therefore, the meaning and quality of alarms is vital. Despite this, we are still in a situation where we see fundamental problems in alarm systems with alarm floods, standing alarms, a lack of priorities, and alarm messages that are hard to understand [33]. The alarm text in Figure 2.1 is a real example from a Network Operations Center. Imagine being in front of the alarm system and ask yourself the following questions: *—What does it mean? —Do I dare ignore 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 is an extreme one, many alarms suffer from bad information quality which makes manual analysis hard. The poor alarm quality combined with the

```
*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
```

Figure 2.1: Example alarm text

sheer number of alarms to be managed results in a huge cost for network service providers. The problem has been around for decades without much improvement. Over the years we have seen standards and industry organizations chasing the silver bullet protocol [34] without much attention to defining the *meaning* of management information.

After considering various attempts to define what an alarm is, we have adopted the following:

Definition 1 *An* alarm *is 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.*

At the core of this definition is that every alarm needs attention and manual investigation and possible actions. This interpretation is close to the origins of the term "alarm" which stems from from Old French, *a l'arme*; "to weapons", telling armed men to pick up their weapons and get ready for action.

2.2.1 Alarm Taxonomy and related findings

We are studying 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 modeling language to define the alarm interface.

Level 2, Semantics: what does the alarm say?

Level 3, Pragmatics: what is the meaning and effect of the alarm when using contextual information?

In the rest of this section we will use the above taxonomy to describe our findings and recommendations.

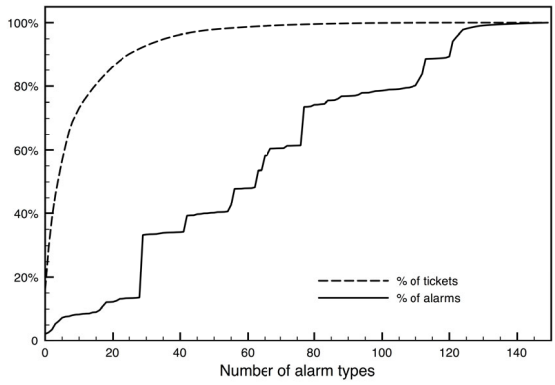

Figure 2.2: Cumulative distribution of tickets and alarms by alarm type.

2.2.1.1 The Alarm Phenomenon

We studied an alarm and trouble-ticket database from a mobile operator in order to understand the first level of the taxonomy. The alarm database contains close to 20 million alarms. Our investigation [30] shows that the alarms at telecom network management centers do not adhere to our basic definition. Only 2.5 % of the alarms required manual actions, and only these should really qualify as alarms [35].

Another observation we made is the "rule of vital few". In total, there are over 3500 different alarm types defined in the sample alarm database, but most of them are very rare. Given the distribution of tickets and alarms on different alarm types shown in Figure 2.2, we see that we get 90% of all our trouble-tickets from less than 30 of the most common alarm types. This indicates that there is a great deal to gain from improving automation for a few alarm types. On the other end of the scale, 10% of all alarms belong to alarm types that have never given rise to an actual ticket. While this requires a detailed study, it does suggest that a great reduction in alarm volume can be achieved by exploring filtering on these alarms.

2.2.1.2 The Alarm Syntax

Level 1 in the alarm taxonomy is the basic level which defines the alarm interface. Over the years, we have seen Q3/GDMO, TCP protocols with ASCII strings, CORBA/IDL, SNMP/SMI, and lately SOAP and Web Services. From a technical point of view, alarm interfaces have not been successful, because integration is still expensive. According to the survey presented in Paper B, service providers reported very few positive experiences with technical standards and the 3GPP alarm interface standard was actually seen to increase the integration costs.

As part of our future work, we are working on defining a semantic model of alarms in order to decrease the integration costs and improve the alarm quality.

2.2.1.3 Alarm 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 to provide meaning to the network administrator. Some challenges that appear at this level are

- understanding the managed object reference
- incorrect alarm severities
- and semantics embedded in free text fields

At this level we chose to study the meaning of *severity*. When network equipment sends an alarm, it usually contains a severity field as a guide to the network administrator for prioritizing the alarm. This severity is assigned by the equipment provider. After an initial manual analysis of the alarm, the network administrator assigns a priority in an associated trouble ticket. We studied the correlation of alarm severities as set by the equipment, versus associated trouble-ticket priorities. The results shows that there is very little correlation between these two. This is even more disturbing when realizing that the alarm severity is actually used for the initial sorting. On average, Critical alarms were acknowledged 500 times faster then Warning alarms.

2.2.1.4 The Alarm Pragmatics

Finally, at the *pragmatic level*, we need to understand the impact and context of the alarm. Network administrators use contextual information such as topology and (informal) service models to understand the alarm pragmatics, but most important of all is their informal expert knowledge. We see two primary ways to improve the situation at this level: knowledge management and service models. The latter is the subject of Section 2.3.

2.2.2 Alarm prioritization using neural networks

This part of our work is to set priorities of alarms at the time of reception by using neural networks. We let the neural network learn from the experienced network administrators rather than building complex correlation rules and service models. As stated by Pernido et al. [36]:

> *Experienced decision makers do not rely on formal models of decision making, but rather on their previous experience. They use their expertise to adapt solutions to problems to the current situation.*

We have integrated a neural network into an alarm and trouble ticket system. The neural network makes use of the manually assigned trouble ticket priorities and associated alarms as learning data. When the alarm system receives an alarm, it interrogates the trained neural network for a priority to be put into the alarm information.

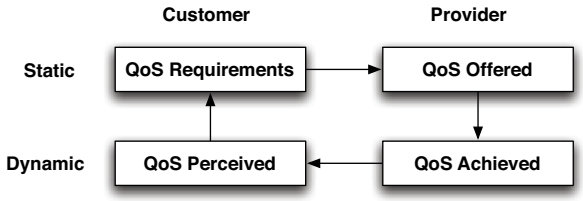

Figure 2.3: Different QoS scope

We compared the neural network to competing approaches and found that our prototype assigns the same severity as a human expert in 50% of all cases, compared to 17% for the severity supplied in the original alarm.

This solution has several benefits:

- Priorities are available immediately as the alarms arrive.
- Captures network administrators' knowledge without disturbing their regular duties.

2.3 Service Modeling

2.3.1 Quality of Service

There are many interpretations and solutions which aim to address Quality of Service. While most of them address a specific service or technology we want to find a solution which can express overall quality measurements or even Service Level Agreements, (SLAs).

ITU-T has defined an overall model for QoS [37], see Figure 2.3, which addresses different scopes like *consumer/producer* and *perceived/delivered*. Examining Figure 2.3 leads us to realize that QoS can mean different things depending on the role we play. As a customer, we *perceive* a certain Quality of Service, while as a provider, we think we *deliver* a certain Service Quality. Quality of Service also has static and dynamic aspects. Customers and providers negotiate a certain QoS (the upper quadrants in Figure 2.3) whereas a dynamic QoS is delivered or perceived while a customer is using the service. Services exist in various flavors such as access services, application services, customer services and the static services defined in service catalogs. All of these have service parameters that need to be described, modeled and calculated. So when modeling services and measuring service quality, we need to address all four quadrants in Figure 2.3.

Another way of looking at QoS is to view it as the *difference* between the right and left halves of Figure 2.3. This is the approach taken by Bloemer [38]:

> *"service quality is a function of the difference scores or gaps between expectations and perceptions (P - E)".*

All of these different perspectives have led us to use a general definition of Quality of Service:

Definition 2 *Quality of Service is defined as a function of the service parameters.*

This lets us cover all the aspects mentioned above. With that as a base we designed a domain-specific service modeling language and monitoring engine called SALmon.

2.3.2 SALmon Overview

Monitoring complex services requires a service modeling language that can express the service models in an efficient way. At the most basic level, a service model consists of relationships between attributes, with different performance indicators coming from the network elements and support systems. An additional complication is that some attributes, such as average quality or minimum bandwidth, depend on how the values of other attributes change over time.

In essence SALmon is a declarative data flow language without a specified flow of time and with the possibility to destructively update old input values. The basic building blocks are attributes expressed as pure functional computations using other attributes and inputs. Inputs are data from external sources such as alarms, trouble tickets, probes and customer care systems. A program consist of a sequence of requests for attributes and updates for input variables.

The evaluation of attributes is performed by time-indexing. Time indexes are restricted to constants or constant functions of the NOW parameter such as

```
system.status@(NOW-1h)
```

The expression can be used both as right-hand value and left-hand value, the latter to change a value retrospectively. Intervals of a time-variable can also be retrieved by specifying a time range as in the following example:

```
dailyStatus = worstOf system.status@(NOW, NOW-1day)
```

SALmon can be used to model the different aspects of QoS, as expressed in Figure 2.3. In the example in Listing 2.1, we present two classes that represent the actual *delivered* quality of service for a VOIP service and the associated network access. The inputs, line 2-3, are collected from voice probes and decoders. Note that these kinds of data are typically based on time-series of data and therefore the time-index of a variable like `rFactor` can be used in expressions. R-Factor [39] is derived from metrics such as latency, jitter, and packet loss and assesses the quality-of-experience for VoIP calls. Typical scores range from 50 (bad) to 90 (excellent). The anchor on line 4 is a reference from the VOIP service access point to the associated network access point. Line 5-8 aggregates inputs from the network instance. The network service access point in line 10 follows the same pattern. It also defines some properties which are static attributes

Listing 2.1: SALmon definition of Service Access Points

```
class VoipSAP
  input clarity, echo;
  input rFactor;
  anchor networkSAP;
  loss = networkSAP.loss;
  jitter = networkSAP.jitter;
  delay = networkSAP.delay;
  bandwidth  = networkSAP.bandwidth;

class NetworkSAP
  properties name, type, DS_Class;
  input in_loss, in_jitter, in_delay, in_bandwidth;
  loss = in_loss;
  jitter = in_jitter;
  delay = in_delay;
  bandwidth = in_bandwidth;
```

Listing 2.2: SALmon definition of a VoIP Service Level

```
class VoipSAPSL
   properties minRFactor, req_loss,req_jitter,
       req_delay, req_bandwidth;
   anchor voipSAP;

   rFactorStatus = linearThreshold voipSAP.rFactor 0 minRFactor;
```

assigned at instantiation. In this example they indicate the type of network and the DiffServ class.

The corresponding *required* quality of service is modeled in Listing 2.2. In this case, properties represent the service level requirements, for example the required R-Factor. From a modeling point of view, a service level is associated with the corresponding service instance as shown in line 4. The `rFactorStatus` calculation in line 6 shows an example of a QoS calculation in line with the Bloemer definition of quality of service in that it evaluates the difference between delivered and required QoS, (as a percentage).

We have implemented the SALmon language runtime and interpreter using the Java™ J2SE Framework [40], the ANTLR parser generator [41] and the Berkeley DB [42]. SALmon is different from many of the upcoming products targeting the problem domain. We believe that a small and efficient domain-specific language will be successful. With a dedicated language you can precisely express what you need and update your models with short turn-around. While graphical approaches may seem attractive at the outset, they often face obstacles when faced with the full complexity of the operator's reality. We have verified SALmon against scenarios from our industrial experience [32],

and we see that the basic requirements for a service modeling language are expressed in a straightforward way. Our approach also eases the transition from current monitoring solutions where most of the integration is performed by using modern, regular, languages like Python. A SALmon-based approach will attract skilled integrators and give them a tool where they can rapidly change and develop models.

2.4 Contributions

The contributions of this thesis include:

- A survey that covers 15 major telecom operators world-wide, including more than 100 million customers/subscribers. This survey forms a basis for deciding which research topics warrant the most attention.
- We examine the basic alarm characteristics using real alarms from a large mobile operator. The analysis identifies which alarm types are candidates for filtering and automation which would result in substantial savings for managing the network.
- Further, this thesis presents a neural network solution that automatically assigns priorities to alarms in real time based on the knowledge of expert network administrators. The training data is automatically gathered using data-mining techniques from the trouble-ticket database.
- Service management will be a strategic challenge for network operators in the coming years. We have designed and implemented a prototype of an efficient domain-specific language dedicated to service modeling and monitoring. We have shown that the capabilities of the language cover relevant functional requirements and large scale models.

2.5 Future Work

Part of this thesis is the building of a platform from which we can search for deeper understanding of alarm and service management. At this stage we see the following important further steps:

- A formal model of alarms to provide solid definitions of alarms and alarm interfaces.
- Further data-mining work to understand more aspects of real telecom network alarms, including more alarm sources.
- Automated solutions for filtering and prioritizing alarms.
- Further implementation and evaluation of SALmon.
- Work in service modeling using SALmon as an evaluation platform.

Chapter 3

Summary of Publications

3.1 Overview of Publications

Our publications and their relationships are described in Figure 3.1. We have chosen to include the most representative subset of these as papers A-E.

Paper A and Paper B describe the problem background and input from network and service providers. Papers C through E represent solutions in the alarm and service monitoring themes.

3.1.1 Paper A: Rethinking Network Management Solutions

Authors Stefan Wallin and Viktor Leijon

Journal IT Professional, November and December [43]

Year 2006

This paper is the starting point for our research and this thesis. It summarizes the problems we have observed, challenges and prioritized areas for improvement, and it is primarily based on our industrial experience. The basic ideas for service modeling and knowledge management technologies to prioritize alarms were initialized in this paper. I wrote most parts of the paper.

3.1.2 Paper B: Telecom Network and Service Management: an Operator Survey

Authors Stefan Wallin and Viktor Leijon

Conference Submitted to 12th IFIP/IEEE International Conference on Management of Multimedia and Mobile Networks and Services [44]

Year 2009

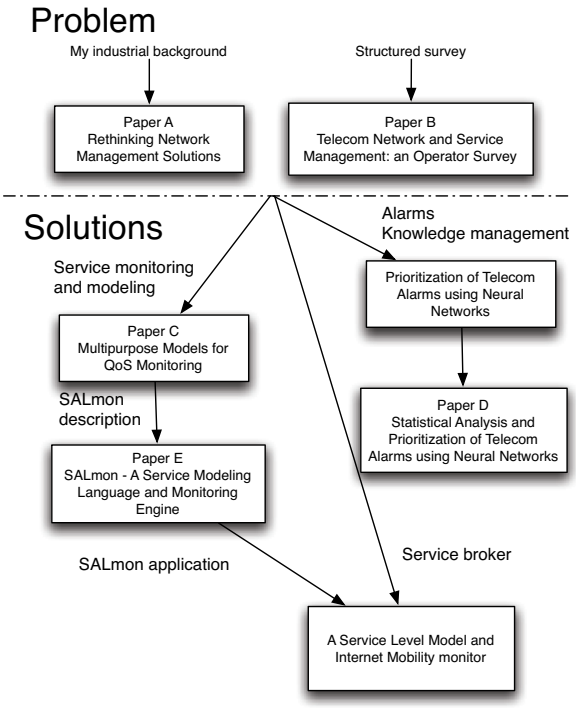

Figure 3.1: Publication Overview

If Paper A is subjective, Paper B is quite the opposite. We realized that we needed more formalized and objective input from challenging tier 1 operators around the world. We surveyed 15 operators regarding the challenges they faced in their operations and what improvements they saw the greatest need for. This is an extensive survey which gives important input to network management research. I did most of the survey work, analysis and writing.

3.1.3 Paper C: Multipurpose Models for QoS Monitoring

Authors Stefan Wallin and Viktor Leijon

Conference 21st International Conference on Advanced Information Networking and Applications Workshops, AINAW'07 [31]

Year 2007

This paper describes the background for a service modeling and monitoring language. The first sketches of the language are outlined at the end. In order to generalize the QoS

term we introduce a general definition of the term: "Quality of Service is defined as a function of service parameters". In this way we introduce the aggregation and modeling function of SALmon – it builds on existing technology specific QoS solutions.

I did most of the writing whereas the ideas are the joint effort of the authors.

3.1.4 Prioritization of Telecom Alarms using Neural Networks

Authors Stefan Wallin and Leif Landén

Conference 22nd International Conference on Advanced Information Networking and Applications Workshops [29]

Year 2008

Telecom Service Providers are faced with an overwhelming flow of alarms. Network administrators need to judge which alarms to resolve in order to maintain service quality. We have prototyped a solution that uses neural networks to assign alarm priority. The neural network learns from network administrators by using the manually assigned priorities in trouble-tickets. Real alarms and trouble-tickets from a large operator is used.

I created the idea and wrote most of the paper. Landén implemented the solution as his BSc thesis work.

3.1.5 Paper D: Statistical Analysis and Prioritization of Telecom Alarms using Neural Networks

Authors Stefan Wallin, Viktor Leijon, and Leif Landen

Journal To Appear in International Journal of Business Intelligence and Data-Mining [30].

Year 2009

This is an extended journal version of the above paper including an additional statistical evaluation. I worked on a more extensive description of the alarm and data-mining handling process. Leijon added the statistical evaluation.

3.1.6 Paper E: SALmon - A Service Modeling Language and Monitoring Engine

Authors Viktor Leijon, Stefan Wallin, and Johan Ehnmark

Conference 4th international Symposium on Service-Oriented System Engineering [32]

Year 2008

This paper describes our service modeling and monitoring language after Ehnmark implemented the first version as his MSc Thesis work.

3.1.7 Service Level Model and Internet Mobility monitor

Authors Christer Åhlund, Stefan Wallin, Karl Andersson and Robert Brännstrom

Journal Telecommunication Systems. Springer [2]

Year 2008

The research group "Mobile Systems" at Luleå University in Skellefteå led by Christer Åhlund works on methods for access network selection. This paper shows how SALmon could be used in that context as the basis of a Policy Based architecture including service and SLA monitoring. I wrote all sections related to SALmon and worked with Åhlund on the overview sections. The final sections on mobility management is the work of Åhlund, Andersson and Brännstrom.

From my research point of view, this paper picks up on an idea from Paper A in regards to a service broker. It also illustrates how SALmon can be used in an overall solution.

References

[1] M. Steinder and A. S. Sethi, "A survey of fault localization techniques in computer networks," *Science of Computer Programming*, vol. 53, no. 2, pp. 165–194, 2004.

[2] C. Åhlund, S. Wallin, K. Andersson, and R. Brännström, "A service level model and Internet mobility monitor," *Telecommunication Systems*, vol. 37, no. 1, pp. 49–70, 2008.

[3] D. M. Meira, *A Model For Alarm Correlation in Telecommunications Networks*. PhD thesis, Federal University of Minas Gerais, November 1997.

[4] R. D. Gardner and D. A. Harle, "Methods and systems for alarm correlation," in *Global Telecommunications Conference (GLOBECOM'96)*, vol. 1, 1996.

[5] G. Jakobson and M. D. Weissman, "Alarm correlation," *Network, IEEE*, vol. 7, no. 6, pp. 52–59, 1993.

[6] R. Sterritt, "Towards autonomic computing: effective event management," in *Proceedings of the 27th Annual Software Engineering Workshop*, pp. 40–47, NASA Goddard/IEEE, 2002.

[7] R. J. Brachman, T. Khabaza, W. Kloesgen, G. Piatetsky-Shapiro, and E. Simoudis, "Mining business databases," *Communications of ACM*, vol. 39, no. 11, pp. 42–48, 1996.

[8] I. Bose and R. K. Mahapatra, "Business data mining, a machine learning perspective," *Information & Management*, vol. 39, no. 3, pp. 211–225, 2001.

[9] C. Ensel, "Automated generation of dependency models for service management," in *Workshop of the OpenView University Association*, (Bologna, Italien), 1999.

[10] R. D. Gardner and D. A. Harle, "Alarm correlation and network fault resolution using the Kohonen self organising map," in *Global Telecommunications Conference (GLOBECOM'97)*, vol. 3, 1997.

[11] T. Kohonen, *Self-Organizing Maps*. Springer, 2001.

[12] R. Sasisekharan, V. Seshadri, and S. M. Weiss, "Data mining and forecasting in large-scale telecommunication networks," *IEEE Expert: Intelligent Systems and Their Applications*, vol. 11, no. 1, pp. 37–43, 1996.

[13] D. Levy and R. Chillarege, "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.

[14] M. Klemettinen, H. Mannila, and H. Toivonen, "Rule discovery in telecommunication alarm data," *Journal of Network and Systems Management*, vol. 7, no. 4, pp. 395–423, 1999.

[15] H. Wietgrefe, K.-D. Tuchs, K. Jobmann, G. Carls, P. Fröhlich, W. Nejdl, and S. Steinfeld, "Using neural networks for alarm correlation in cellular phone networks," in *International Workshop on Applications of Neural Networks to Telecommunications (IWANNT)*, 1997.

[16] TM Forum, "SLA management handbook," tech. rep., TM Forum, 2004.

[17] TeleManagement Forum, "Information Framework (SID) ." GB922 , Release 7.5, 2005.

[18] Distributed Management Task Force, "CIM Specification." Version 2.15.0, 2007.

[19] B. Moore, E. Ellesson, J. Strassner, and A. Westerinen, "Policy Core Information Model – Version 1 Specification." RFC 3060 (Proposed Standard), Feb. 2001. Updated by RFC 3460.

[20] W3C, "Service modeling language," May 2008. http://www.w3.org/TR/sml/.

[21] M. Björklund, "YANG - A data modeling language for NETCONF, draft-ietf-netmod-yang-03." Accessed 16th of March 2009. http://www.yang-central.org/.

[22] R. Enns, "NETCONF Configuration Protocol." RFC 4741 (Proposed Standard), Dec. 2006.

[23] B. Lundell and B. Lings, "Changing perceptions of case technology," *Journal of Systems and Software*, pp. 271–280, 2004.

[24] M. Garschhammer, R. Hauck, H. Hegering, B. Kempter, I. Radisic, H. Rolle, H. Schmidt, M. Langer, and M. Nerb, "Towards generic service management concepts a service model based approach," *Integrated Network Management Proceedings, 2001 IEEE/IFIP International Symposium on*, pp. 719–732, 2001.

[25] R. Gopal, "Unifying network configuration and service assurance with a service modeling language," *Network Operations and Management Symposium, 2002. NOMS 2002. 2002 IEEE/IFIP*, pp. 711–725, 2002.

[26] E. Marilly, O. Martinot, H. Papini, and D. Goderis, "Service level agreements: a main challenge for next generation networks," *Universal Multiservice Networks, 2002. ECUMN 2002. 2nd European Conference on*, pp. 297–304, 2002.

[27] V. Räisänen, "Service quality support—an overview," *Computer Communications*, vol. 27, no. 15, pp. 1539–1546, 2004.

[28] D. Lamanna, J. Skene, and W. Emmerich, "SLAng: A Language for Defining Service Level Agreements," *Proc. of the 9th IEEE Workshop on Future Trends in Distributed Computing Systems-FTDCS*, pp. 100–106, 2003.

[29] S. Wallin and L. Landén, "Telecom alarms prioritization using neural networks," in *International Conference on Advanced Information Networking and Applications Workshops (AINAW'08)*, 2008.

[30] S. Wallin, V. Leijon, and L. Landén, "Statistical analysis and prioritization of alarms in mobile networks," *International Journal of Business Intelligence and Data-Mining*, 2008. **To Appear**.

[31] S. Wallin and V. Leijon, "Multi-Purpose Models for QoS Monitoring," in *21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07)*, pp. 900–905, IEEE Computer Society, 2007.

[32] V. Leijon, S. Wallin, and J. Ehnmark, "SALmon, a Service Modeling Language and Monitoring Engine," in *The Fourth IEEE International Symposium on Service-Oriented System Engineering)*, 2009.

[33] M. Bransby and J. Jenkinson, "The Management of Alarm Systems," *HSE Contract Research Report*, vol. 166, 1998.

[34] J. Schönwälder, A. Pras, and J. Martin-Flatin, "On the future of Internet management technologies," *Communications Magazine, IEEE*, vol. 41, no. 10, pp. 90–97, 2003.

[35] N. Stanton, D. Harrison, K. Taylor-Burge, and L. Porter, "Sorting the Wheat from the Chaff: A Study of the Detection of Alarms," *Cognition, Technology & Work*, vol. 2, no. 3, pp. 134–141, 2000.

[36] M. C. Penido G, Nogueira J.M, "An automatic fault diagnosis and correction system for telecommunications management," in *Proceedings of the Sixth IFIP/IEEE International Symposium on Integrated Network Management*, pp. 777–791, 1999.

[37] ITU-T, "ITU-T Recommendation G.1000 Communications Quality Of Service: A Framework and Definitions," 2001.

[38] J. Bloemer, K. de Ruyter, and M. Wetzels, "Linking perceived service quality and service loyalty: a multi-dimensional perspective," *European Journal of Marketing*, vol. 33, no. 11, pp. 1082–1106, 1999.

[39] ITU-T, "Recommendation G.107," 2008.

[40] SUN Microsystems, "J2SE 5.0." Accessed 14th of July, 2008. http://java.sun.com/j2se/1.5.0/.

[41] T. Parr, "ANTLR parser generator." Accessed 14th of July, 2008. http://www.antlr.org/.

[42] M. Olson, K. Bostic, and M. Seltzer, "Berkeley DB," in *Proceedings of the FREENIX Track: 1999 USENIX Annual Technical Conference*, pp. 183–192, 1999.

[43] S. Wallin and V. Leijon, "Rethinking network management solutions," *IT Professional*, vol. 8, no. 6, pp. 19–23, 2006.

[44] S. Wallin and V. Leijon, "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**.

Part II

Paper A

Rethinking Network Management Solutions

Authors:
Stefan Wallin and Viktor Leijon

Reformatted version of paper originally published in:
in IT Professional, November and December, 2006.

Rethinking Network Management Solutions

Stefan Wallin and Viktor Leijon

Abstract

This paper looks at network management from an overall perspective. We try to explain what the current problems at big telecom operators are, and how things are changing in the operators environments. Finally we present the major steps needed to be taken as seen by the authors. The problem statement has been worked out with input from people in charge of large telecom network management centers. In order to cope with current problems and improve the quality and effectiveness the major steps forward are: (1) Service centric management, (2) Dynamic management, (3) Knowledge management, (4) Automation and correlation and finally (5)Managing network management interfaces.

The paper does not elaborate in detail in any of the items; rather it presents an outline of where to go.

1 Introduction

As with any technology, it's important to focus management solutions on the *users*, even when the users are those providing a service. In that broader context, network management has three types of users: *network operators*, which must earn money on their services, *network service users* (business and consumer), who pay for using services, and *network administrators*, who staff the network operations center. All three user types benefit from a well-thought-out management solution: operators increase their profits, service users get better service, and administrators streamline their workload.

In short, the right network management solutions empower network operators to provide new services, maintain service quality, and manage billing and usage [1].

By its nature, network management is a hierarchical, centralized function that puts the operator in control; therefore it makes sense to provide a centralized network management solution. Operators are under pressure to reduce network operating costs and provide new services at an increasing speed. These two requirements highlight the need for an effective, automated, network-management solution.

To explore such a solution, we interviewed people in charge of large telecom network management centers and identified six challenges facing big telecom operators:

Excessive alarms: A medium-sized operations center receives 100,000 to 1,000,000 alarms per day.

Constant changes: New or upgraded devices and new services launch frequently.

Complex services structure: Services are vital for business and customer interaction, but they are not really managed.

Customer interaction: Operators must handle customer complaints, customer care, selling services and service-level agreements (SLAs).

Cuts in operations costs: A small team must run a large, multifaceted network.

Difficult interface integration: Diverse equipment and support systems make managing interface integration a challenge.

We then considered strategies for tackling each of these challenges and determined several best practices.

2 Challenges

2.1 Excessive Alarms

The bulk of network administrators' daily work involves alarms. Unfortunately, the large number of alarms indicates that the systems produce many irrelevant and noncorrelated alarms, making it hard to understand the true state of problems in the network.

Today's alarms are more or less raw alarms from the different equipment and vendor-specific management systems. Operators must establish an efficient organization to handle the alarms, a process that typically follows three steps:

1. The first-line organization performs three tasks: check for alarms that indicate the same problem, group the alarms and attach them to a trouble ticket, and distribute problem information to affected parties, such as SLA customers and customer care.
2. If it's a simple problem, the first line resolves it and closes the ticket.
3. If it's a complex problem, the first line dispatches it to the second- and third-line organizations. This might involve equipment vendors or operator staff in the field who might perform onsite management, card replacement, and so on.

The ever-increasing number of systems and services increases the number of alarms. Still, operators can't afford to employ additional people to handle the alarm lists, and automatic solutions are limited.

Automatic trouble ticketing, for example, manages the workflow from problem identification to problem solution, but its usefulness doesn't extend to prioritizing the alarm's importance. Such knowledge is critical because an alarm's context determines if it affects services, customer SLAs, and the affected equipment's state. The resource emitting the alarm typically doesn't know the context, so the network-management system must supply it through alarm filtering and correlation.

Alarm-correlation projects are complex and not particularly successful. First, alarm quality is insufficient. The information carried in the alarm messages is not good enough

to feed automatic-correlation engines. Second, the network lacks an overall network topology. In many cases, alarms are symptoms of a failure somewhere in the network. When a network doesn't have a model or system in place that keeps track of all of its resources and their relationships, it's difficult to implement rules that can deduce a fault's root cause. Third, correlation knowledge is spread across the organization and over several domain experts. This makes the organization too dependent on individuals and hinders centralized efforts. Finally, operators fail to use important alarm contexts such as trouble ticketing, inventory, customer care, and SLA management systems. Trying to correlate alarms using only the alarm information will lead to only minor improvements.

The operators we interviewed also noted the high cost of integrating alarm interfaces from equipment into the overall management system. In typical Simple Network Management Protocol (SNMP) agents, every box has its own specific mechanisms for alarms. The Disman working group at IETF has tried to resolve it by defining a standard MIB for alarms, the Alarm Management Information Base [2]. However,we have not seen equipment vendors moving in this direction for their SNMP interfaces. Alarm integration at operators are still equipment specific and time-consuming.

Another organizational issue is managing the knowledge of how to resolve problems. Alarm texts from network equipment are usually cryptic, without any hints of how to fix the problem.Thus, operators incur steep training and productivity costs when hiring new people or introducing new equipment types.

2.2 Constant Change

Networks change. Network elements are upgraded, new services launch, and customers come and go.These daily changes are a challenge for operators and network management solutions.

Few operators have a fully controlled or automated process for handling these changes. Moreover, the network organizations are introducing critical equipment into the network without informing the network administrators. Surprises occur in the monitoring activities when unknown alarms and equipment suddenly appear. SLAs and business-critical services are sold to enterprise customers but without corresponding support in the management solution to actually monitor the specific SLA or customer.

The dynamic nature of networks and services puts increasing focus on change management. The expected time for changes has dropped from months to hours. We see operators and organizations realizing this and trying to reuse the change management process from the Information Technology Infrastructure Library framework [3], a set of best practices drawn from public and private sectors worldwide. Change management's goal is to ensure that standardized methods and procedures are used to efficiently and promptly handle all changes to minimize its impact on service quality, consequently improving the organization's daily operations.

2.3 Complex Service Structures

Services' complex structure is an underlying problem in network management solutions, according to one operator we interviewed. Often operators do not have true visibility of services across processes and systems. There is a discrepancy between how customer care manages service problems and how the assurance and repair organization manages physical resources.

With those background problems, operators are looking for service management solutions or even SLA management solutions. Generally speaking, the industry must solve several underlying problems before successfully deploying such systems:

Topology management: network topology, service topology, and the mapping between these.

Service management: formal but dynamic management of services, SLAs, and customers, across all processes and systems.

Service centric integration and modeling: use of service types and instances as keys in information systems, customer care systems, fault management systems, and so on.

2.4 Customer Interaction

Being an operator in the current telecom environment is far from what it was 10 years ago. Customers now compare on the open market such factors as quality, service, and price. Therefore, operators must stay in close contact with customers, keeping them apprised of service status, including problem resolution, and providing them with clear, easily interpreted bills. This communication level requires a network management solution that can map resources and alarms to services and problems in a way that customer care and the customer can understand.

Operators must prioritize work on the basis of service and customer priority, yet there is a big gap between customer care and the corresponding technical network management organization.

2.5 Pressure to Cut Operations Costs

Operators are trying to cut operational expenses for both operating the network and introducing new services and equipment. Previously, it was acceptable to spend a couple of months integrating new telecom equipment. Now, any integration must be complete within a week. The number of people managing the provisioning, assurance, and billing solutions is minimal. To cut costs, many operators also have ongoing projects to merge network operating centers for different geographical and technical domains into "super-NOCs."

Another major effort toward greater efficiency is automating network management activities, such as automated alarm correlation, trouble ticketing, and alarm enrichment.

2.6 Interface Management

Network management solutions are often huge software integration projects. A cost and complexity driver is managing interfaces, interface versions, interface documents, and so on. Current point-to-point integrations make the integrated network management solution sensitive to changes in interfaces and information. To make matters more difficult, telecom equipment vendors are not always keen to provide easily accessible management interfaces, which contrasts sharply to the norm for IP devices.

Contrary to rules and best practices, management interfaces often are not backward compatible. When equipment is upgraded, the previous integration work is often destroyed.

3 Ways to Improve

Given these problems and changes in the environment that will affect network management in the future, we believe the next generation of network management solutions must be based on principles different from the current solutions.

3.1 Service-Centric Network Management

At the core of a network management solution for the future lies a service model. Operators must have full control of the services provided to and used by customers. The service model must capture a service's semantics and its real-time status. Although service would be the model's primary focus, the model must still map other concepts such as resources, network topology, and customers. This model is far from trivial. It requires a strong formalism that can express relations and dependencies.

Figure 1 gives a flavor of how the model might work with a simple case. When two events arrive, correlation software maps them into the service tree and updates the service tree with the events' impact. In this case, one low-severity event and one medium-severity event arrive on different systems, making the overall severity for the service medium.

A modeling formalism for a service-centric view must be transformable. More specifically, it must be possible to pivot the "service graph" around the node of interest, so that the same model can be used to satisfy the district manager, the general manager, the technology domain manager, and the help desk manager at the same time. Current models with simple aggregations and so forth are far too primitive. Anyone who has used a service-management tool is familiar with the problem of services always being in a failed state because the models are too weak in expressing dependencies. This is not only a modeling problem; an even more challenging task is to maintain the model's actual instances. At all times, the model must be correct and mirror the network's actual state.

The most interesting attempt we have seen in modeling is the Distributed Management Task Force's *Common Information Model* [4]. *CIM* has several good aspects. It has a service-centric model, emphasizes relationships, and maps resources and network topology to the service model. However, CIM has weak formalism — Unified Modeling

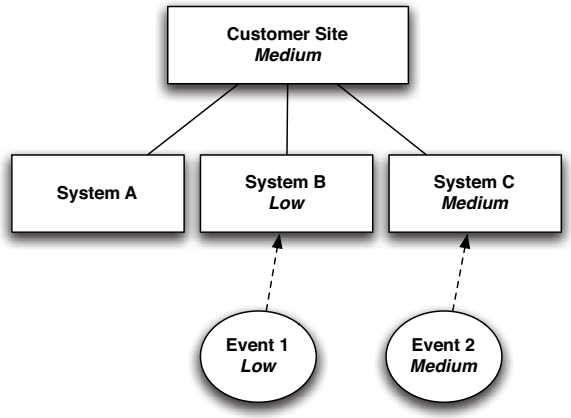

Figure 1: Service Impact

Language static class diagrams. We do not see UML classes as strong enough in expressing such items as semantics and interfaces. Also, CIM's model complexity and size has exploded. Modeling every aspect as classes yields a huge model that will not cope with the changing and dynamic nature of the future services and networks.

The IETF made an attempt to take SNMP one step forward with the *SMIng* data definition language, [5], which makes SNMP MIBs capable of holding objects, structured data types, and so on. SMIng has a pragmatic approach and would probably make a significant difference in the short term, although it has not created any footprints in the industry as yet. Attempts from the IETF's Network Management Research Group have the big advantage of being down to earth and well engineered. In the long run, however, something more powerful is needed.

From a solution point of view, service-centric network components are emergingÑfor instance, Cramer or Granite. These product examples are signs that the field is moving in the right direction. Topology must be a core component, however, and the current solutions and tools do not handle the dynamic nature of the topology changes. Typical implementations use an export, clean, merge, and load process to create an overall topology database. To be fair, the fault is not with the topology tools themselves but with the poor equipment interfaces [6].

3.2 Dynamic Network Management Strategy

Network management mechanisms and solutions must become much more dynamic to cope with the changing environment, heterogeneous networks, dynamic services, and customers that come and go. In practical terms, dynamic network management has four main requirements:

Integrate Network Elements. Interfaces for perfoming network management must be much better defined. This will require appropriate standards for content and communication. We foresee an approach totally different from current technologies: Strongly typed interfaces, with patterns for integration, will facilitate easier, but never automatic, integration among parts.

Optimize models. Topology and service models must be self-maintaning. This will require a fully integrated provisioning chain so that the models are always up to date. Also, network elements must publish dynamic interfaces for publishing the models.

Develop heterogenous networks and dynamic services. Service requests, service discovery, and service capabilities must be handled in a dynamic real-time fashion [7]

Cope with change. Networks change more quickly every day. Systems must be dynamic and able to manage change.

3.3 Knowledge Management

Operators we interviewed also stressed that they are too dependent on individuals and that their systems apply too little automation. Expert users with several years of network management experience are invaluable, but it's hard to reuse their knowledge because the processes are still carried out manually. The next generation of network management must apply knowledge management and expert-systems technology. This process will enable the solution to evolve, adapt, and capture the operator's knowledge and make the system self-learning and more automatic.

Such technology can capture network administrator usage patterns in real time and analyze them to produce a list of suggested automations [8].

The key is to have a system that is self-learning and self- adapting, so that it captures the expert users' behavior and provides tailored responses. Thus, rather than implementing every scenario using traditional development techniques, network administrators are actually adapting the solution as they work.

3.4 Challenges and Changes in the Environment

In addition to the problems we identified in our interviews with operators, we see many external factors that affect network management. These include (IP-based) services such as VoIP, managed voice, and streaming media; new technologies like IP Multimedia Subsystem; increased requirements from customers regarding avail- ability and quality; convergence of mobile and fixed networks; and ad hoc customers, services, and network access. Customers also expect roaming between opera- tors and access to networks and network technologies to occur without disruption [9].

We foresee a broker layer between users and network/service operators that will let users automatically receive the service that best fits their profile when they are mobile.

Users will pay the broker, who will pay the operator. This business model will put even more emphasis on how service providers express service capabilities and features.

3.5 Network management interfaces

We also anticipate great progress in the design of network management interfaces. For a long time, equipment vendors have been experimenting with different protocol technologies rather than providing easy-to-use, high quality interfaces.

Today's demand for ease of integration and automation will drive the industry to apply simple techniques like SNMP, but with a high degree of functionality and standards. We recommend the following best practices:

- Focus on functionality and quality rather than complex technologies.
- Provide an underlying model for topology and services
- Ensure backward compatibility. An upgrade or change of software should not affect the interface.
- Find ways that will let operators integrate equipment more smoothly into their overall management solution.
- Use dynamic approaches in interface technologies. Minimize the need for external data.
- Filter and correlate alarms before sending them. Send problem-oriented alarm states pinpointing the affected service rather then low-level symptoms.

We also see a strong need for improvements in modeling formalisms to express service models and more dynamic semantic interface definitions. An even more important issue is the quality of the models themselves, irrespective of the modeling formalism.

In many ways, network management problems have changed little since 1988, when SNMP was introduced. There is still no sense of how to model management information and no greater insight into which information is truly valuable to a management application. Progress requires investigating fundamental modeling questions: What characterizes a good model? Given a bunch of such models, what are the common structures, design patterns, ways of thinking, aggregation models, and so on? And given common denominators of good models, the problem becomes how to construct tools that let developers build such models easily. Is it even possible to develop a structured theory of network management that truly starts small and builds on real-world knowledge?

Telecom network management solutions need to shift perspectives from one of network element management to service management. Operators need a service view of their network, with automatic service-impact correlation. This requires some major changes in the underlying solutions: equipment vendors must improve the supplied management interfaces and network management solutions must implement a higher degree of automation and cor- relation with a service focus. One obstacle is the lack of models and

formalisms to describe topology and service structures. We're currently working to define a formal service modeling approach to enable the service layer.

References

[1] TeleManagement Forum, "Business process framework (eTOM)." GB921, version 7.3, July 2008.

[2] S. Chisholm and D. Romascanu, "Alarm Management Information Base (MIB)." RFC 3877 (Proposed Standard), Sept. 2004.

[3] itSMF, "Service Operation." Office of Government Commerce, ITIL Version 3 Publications, 2007.

[4] Distributed Management Task Force, "CIM Specification." Version 2.15.0, 2007.

[5] J. Schönwälder and F. Strauss, "Next Generation Structure of Management Information for the Internet," *Lecture notes in computer science*, pp. 93–106, 1999.

[6] R. State, O. Festor, and E. Nataf, "Managing Highly Dynamic Services Using Extended Temporal Network Information Models," *Journal of Network and Systems Management*, vol. 10, no. 2, pp. 195–209, 2002.

[7] M. D'Arienzo, A. Pescapè, and G. Ventre, "Dynamic Service Management in Heterogeneous Networks," *Journal of Network and Systems Management*, vol. 12, no. 3, pp. 349–370, 2004.

[8] M. Klemettinen, H. Mannila, and H. Toivonen, "Rule discovery in telecommunication alarm data," *Journal of Network and Systems Management*, vol. 7, no. 4, pp. 395–423, 1999.

[9] C. Åhlund, R. Brännstrom, K. Andersson, and Ö. Tjernström, "Port-based Multihomed Mobile IPv6 for Heterogeneous Networks,"

Paper B

Telecom Network and Service Management: an Operator Survey

Authors:
Stefan Wallin and Viktor Leijon

Reformatted version of paper originally submitted to:
12th IFIP/IEEE International Conference on Management of Multimedia and Mobile Networks and Services, 2009

Telecom Network and Service Management: an Operator Survey

Stefan Wallin and Viktor Leijon

Abstract

It is hard to know which research problems in network management we should focus our attention on. To remedy this situation we have surveyed fifteen different telecom operators on four continents to gather some feedback on what they desire and expect from the network management research community. Their input forms a foundation for future directions in network management research, and provides us with valuable insight into what the most urgent problems are in industry.

1 Introduction

Network management research covers a wide range of different topics, and it is hard for the individual researchers to prioritize between them. One factor to take into account is the requirements emanating from the telecom industry.

In order to get an objective view of what the industry considers important we have surveyed fifteen different companies, to gather their opinions on the current state of network management systems, as well as their expectations on the future.

As far as we can tell, there have been no previous surveys of this type for telecommunications network management. The process control and power industry areas seems to have a higher degree of industry feedback to the research [1, 2, 3], probably because of the human safety risks involved.

The results of this survey has strategic value both for researchers and solution vendors. It identifies areas where there is a strong need for further research and point to what changes are needed in order to stay competitive.

The contributions of this paper are:

- We present survey results from fifteen different companies, with a total of over 100 million customers, covering the current state (Section 3) and most important change drivers (Section 4).
- The respondents were then asked about their view on the future of OSS systems (Section 5) and what they expected from the OSS research community (Section 7.1).
- We conclude with a discussion of the focus areas identified in the survey: service topology and alarm quality in Section 7.2.

2 Method

We distributed the survey questions by e-mail to 20 operators of different size and on different continents. The individuals were selected based on their roles as network management architects or managers. 15 out of 20 operators answered. The respondents are a mix of fixed, broadband and mobile operators with a total customer base of over 100 million subscribers, see Table 2.1. The operators were classified by number subscribers into the categories [$<$ 10 M, $<$ 20 M, $<$ 100 M] to avoid giving out identifying information.

Some clarifying questions were sent over e-mail and a draft version of this paper was sent to all operators that provided answers. All questions except one were essay questions to avoid limiting the answers by our pre-conceived ideas. We have aggregated similar answers into groups, often using eTOM processes as targets. The number of answers for each alternative will not add up to the exact number of responding operators since the answers typically mentioned several different answers.

Table 2.1: Summary of responding operators.

Services	Customers	Region
Mobile and Broadband	$<$ 10”	North America
Mobile	$<$ 20”	Europe
Mobile	$<$ 10”	Europe
Mobile	$<$ 20”	South America
Mobile	$<$ 20”	North Africa
Managed OSS, Mobile, transmission	$<$ 20”	Europe
Mobile, broadband, transmission	$<$100”	Europe
Mobile, broadband	$<$ 10”	Europe
Mobile, broadband, transmission	$<$ 100”	Asia Pacific
Outsourced OSS, Mobile internet, broadband, 3G	$<$ 10”	Europe
Broadcast, virtual network, capacity	$<$ 10”	Europe
Mobile, broadband	$<$ 10”	Europe
Mobile, broadband, managed services	$<$ 10”	Asia Pacific
Full service carrier, wireline, wireless	$<$ 10”	Asia Pacific
Mobile and broadband	$<$ 10”	Asia Pacific

Figure 1: Change drivers for OSS.

3 Current Status of Network Management

The network and service management solution for a telecom operator is referred to as the OSS, the Operation Support System. Current OSS solutions typically belong to one of two types depending on maturity, either *Classic* or *Advanced*.

Classic OSS solutions covers service assurance and trouble ticketing. Element Managers are used to perform configuration management activities and performance management is only partially implemented.

Advanced OSS solutions have expanded the solution to include service management tools such as active and passive probes; and service models. General configuration management tools are used to some degree, spanning several different vendors.

Common to both types is that security is not generally covered by the OSS solution. IT and Telecom is typically still split into different organizations and the IT department manages the customer care and billing processes.

All of the operators said that they focused on making the OSS a proactive solution, but admitted that in reality most of the OSS work is reactive, responding to alarms and problems reported to customer care. The proactive activities were typically based on probes and statistics, where the data was used to predict problems such as capacity limitations, but these were not really integrated into the overall OSS solution.

An interesting comment from one of the operators was about "OSS culture", they had problems transitioning their network administrators from using the element managers to using the full OSS solution. This resulted in underutilization of the OSS and decreased the motivation to develop it.

4 OSS Motivation and Drivers

We asked the operators to identify the most important external drivers that motivates changes in their OSS. The answers are shown in Figure 1.

Increased focus on *services and service quality* was identified as the most important factor behind changes in the OSS. In order to understand this subject better we asked the operators to elaborate on how they viewed service management, one of the operators summarized it in the following way:

> *Services are not currently managed well in any suite of applications and requires a tremendous amount of work to maintain.*

The competitive market is pushing operators to offer more and more innovative services, including SLAs, which require the OSS solution to measure the service quality.

One operator described the experience with the two major alternatives to service monitoring, either using probes or mapping existing events and alarms to a service model. The latter approach failed since there was no good way to describe which alarms were really critical. They made the decision to use only probing for future services, stressing that future services will have service probing from the service deployment phase.

Some of the operators stressed the importance of standards for service models. The problem with models is that services are rapidly changing, therefore requiring a large amount of customization work. One operator expressed reservations about how detailed services can be:

> *Time and money will not be available to [develop] sophisticated approaches over a long period. Customers will have to accept limited quality assurance and quality documentation. Service levels will always be high level, if [they exist] at all.*

Another operator commented on how the use of service models is evolving:

> *Service models are becoming more and more important: currently [they are] not implemented in core processes but used as means to semi-document and analyze when evaluating impact of faults, new services, etc.*

As indicated by Figure 1 *cost reduction* is clearly another key factor. We asked the operators to further break down the cost drivers and the results are shown in Figure 2.

The first two items can be considered two sides of the same coin: Integration costs are high in OSS due to the introduction of new technologies and services. When a new type of network element is deployed it needs to be integrated into the OSS solution and while most solutions are based on well established products, there is still a high degree of customization needed to adopt the tools to user needs and processes.

In order to get a unified view of the solution, the resource oriented interfaces needs to be mapped into an overall service model. Operators are struggling with this challenge, the "information model development" in Figure 2. Finally the OSS itself is expensive due to the number of components that are needed. Even in the case where an OSS is built using only one vendor, it is still made up of a portfolio of modules which add up to a relatively costly software solution.

Returning to the change drivers (Figure 1) the next items are *network growth* and increased focus on *customer satisfaction*. While network growth is inherent in network

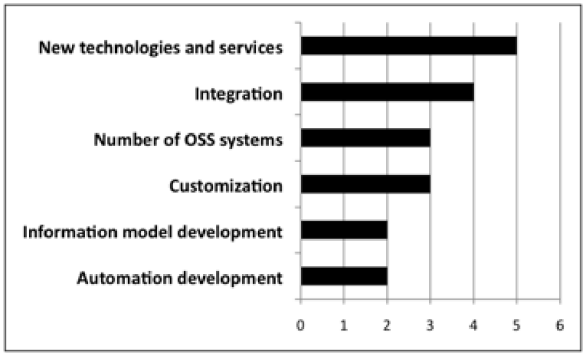

Figure 2: Cost drivers for OSS.

management, customer satisfaction has not historically been one a primary goals for OSS solutions.

To put the true business role of the OSS solution in focus, we asked if it was seen as a competitive tool or not. The answers were divided into two general streams:

- *Yes.* Motivations are the desire to decrease time-to-repair, decrease OPEX, and improve customer satisfaction by quicker response to new requirements and customer complaints. Two thirds of the responses fall into this category.

- *No.* These operators felt that it will be outsourced. The out-sourcing scenario was partly motivated by internal failures and a desire to give the problem to someone else.

5 The Future of OSS

5.1 Organizational Changes

The answers regarding future OSS changes are summarized in Figure 3, it is no surprise that Service management is seen as one of the most important upcoming changes. One of the larger operators predicted a "focus on service management - bringing this up to 40% from [the] current level of 5-10%".

Further, we see the strategic need for a service inventory to enable service quality, service provisioning and a service life-cycle view. Most of the operators are providing broadband services and this makes automatic service provisioning a must.

Service management was summarized in the following way by one of the operators:

> *Managing services must be the focus of the future development, while pushing network management into a supporting role, [...] service models [should be]*

Figure 3: OSS organizational changes.

> *constructed from auto-discovered infrastructure components that have auto-discovery deeply integrated. [...] [The] service models should be abstracted from physical and logical inventory to ensure there is a life cycle. [...] If the service models were to be constructed from scratch, with no base layer that is managed separately (auto-discovered or abstraction over inventory), that will most probably undermine the layered service level management solution.*

Common to all respondents is an increased focus on customer care, customer service, self-management, and self-provisioning. An interesting point made by one of the operators is the shift from eTOM verticals to a more supply chain based production model. Classical OSS systems have had separate solutions for fulfillment, assurance, and billing. The supply chain model will focus on the whole life-cycle from service definition to billing and retirement. This will remove the obstacles between eTOM verticals and help relieve the hand-over problem from "eTOM Strategy" to "eTOM Operations" areas.

Another trend is a radical transformation of the OSS from being a network-oriented function to becoming a customer-oriented organization. The surveyed operators mentioned features such as "more focus on customer service, customer experience, and self-management", this will result in OSS decisions being based more on share-of-wallet analysis.

5.2 Focus processes

The eTOM process framework defines the processes in several abstraction layers. We asked the operators to identify the three most important *level 2* processes (Figure 4).

The prefixes of the legend in Figure 4 refers to the eTOM vertical (**F**ulfillment, **A**ssurance, **B**illing) and horizontal (**C**ustomer, **S**ervice, **R**esource, **S**upplier) end-to-end processes. For instance F-S refers to Fulfillment-Service. Again, we see the focus on the service life-cycle through order handling, provisioning, service quality, problem management and billing.

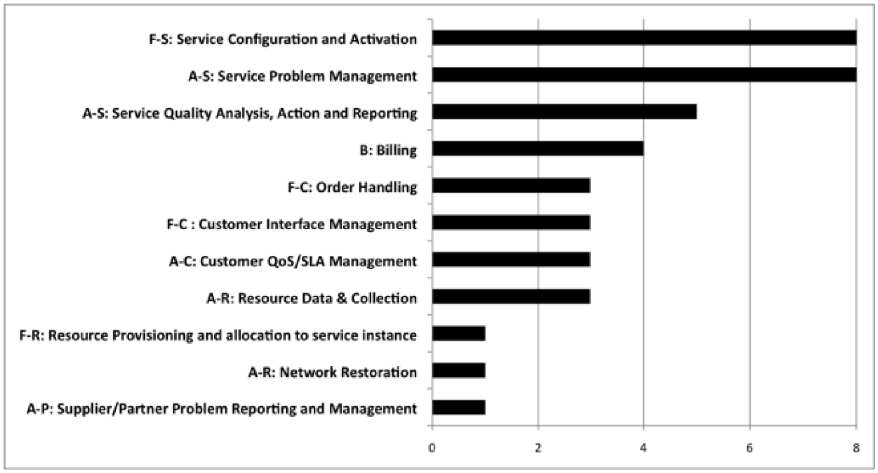

Figure 4: The most important eTOM Level 2 processes.

6 Standards and Research Efforts

6.1 Standards

The attitude towards standards was not very enthusiastic: "They are too complicated and are actually adding to the cost of ownership", in this case the 3GPP Alarm Interface was the main source for concern. Another operator had similar distrust for standards: "[In] alarm integration to the OSS, most of the vendors do not follow any one. We are pushing [our] internal standard to have useful alarm information for the end users". Some operators mentioned SNMP as a working protocol that is easy to integrate, however the lack of standard OSS interface MIBs is a problem, and the vendor MIBs vary in quality.

As important areas for future standardization efforts they mentioned "interfaces, data and semantic models, standardization of procedures" and "well defined top level common framework and common languages". We see from these comments that the current practice of using different protocols for different interfaces and having weak formal data models is a problem for OSS integrations. There is no accepted overall common framework which would enable unified naming of resources.

Surprisingly, none of the operators mentioned OSS/J [4]. On the other hand several operators considered the eTOM and ITIL [5] process standards to have real practical value. They used these process frameworks to structure the work and make it more efficient.

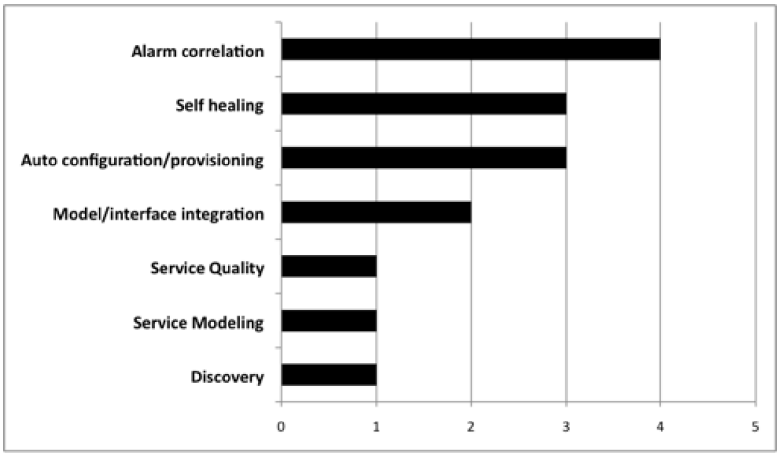

Figure 5: Research efforts.

6.2 Research efforts

In Figure 5 we can see the areas that the operators in this survey identified as the most important ones for future research.

It is well worth noting that for research alarm correlation is a key item, while it was not as prominent when the operators were asked to identify the key efforts and changes in the coming years. The opposite is true for service management and modeling, which were key items for more immediate change. One interpretation is that it is hard to foresee the challenges with the planned changes before we have attempted them. Many operators have struggled with alarm correlation for years without seeing any clear successes and are hoping for more research to help them. We might see the same thing happening with service management and modeling after a first wave of implementation efforts.

7 Discussion

7.1 OSS research

Historically a lot of research has gone into alarm correlation. Looking at the current correlation rates it is not clear how successful these projects have been, many correlation projects are facing challenges trying to capture operator knowledge and transforming it into rules. Other methods for finding rules based on rule discovery, knowledge management, data-mining and self-learning networks are interesting, but seem to require further investigation to be of practical use.

Having service models in place would be a key to unlocking many other functions. The current industry practice of large UML models with loose structure does not seem

able to cope with the requirements, more modular and semantically rich ways of doing service modeling are required.

Another basic area is that of formal interface definitions. The current integration costs can not be justified when we consider the fundamentally simple nature of the information that flows over the interfaces, integrating alarms should not be a complex and costly task. Semantically richer interface definitions would probably improve the situation, but this requires a focus on the semantics of the model and not only the syntax, protocol and software architecture.

Boutaba and Xiao [6] point to the following major enablers for future OSS systems:

- Policy-based network management, [7], [8]
- Distributed computing, [9], [10]
- Mobile agents, [11], [12]
- Web techniques, [13]
- Java, [14]

These are topics that we see in many network management efforts. However we see little correlation between these and those identified in this survey. Furthermore it partly illustrates the research's focus on software architectures rather then management.

7.2 Fulfilling operators future needs with the OSS

It is easy to talk about paradigm shifts but things change more slowly than we expect, nonetheless some major changes in OSS solutions are needed to help service providers to cope with the changing environment. Operators want to manage services rather than the network resources that are used to deliver services. This change in focus is driven by several factors; increased competition, more complex service offerings, distribution of services, and a market for Service Level Agreements.

One operator had this to say about the market for services:

> *The competition between companies pushes them to offer more and innovative services, they need to sell more services first, the want to go out before the others.*

Current network management solutions control and administer resources; physical resources are found, configured and monitored. However, from a customer point of view, services are bought, provisioned and billed.

One of the underlying challenges in a service centric solution is to maintain a model of the service topology. In contrast to the network resources services are abstract by nature, while it is often possible to automatically discover the network topology this is not necessarily true for the service topology.

Future solutions must rely on service repositories where different and systems publish their respective topology information with lightweight technologies, this will enable

operators to maintain a centralized view of the topology which will be the future OSS foundation. Note that this improvement has to come without significant additional integration costs or complexity.

Moving forward in the direction identified in this survey will fail if there is no way to share the information model between OSS systems. The feedback we get from operators who apply SID [15] is that it works well as a design base for the OSS system but not as a service model to maintain the dynamic business services. Service modeling must be dynamic and have well-defined semantics, the current practice of static models and informal documents does not cope with a changing environment.

While OSS solutions have primarily been network oriented it now needs to change focus to customer care, since operators see a huge possibility to reduce costs in an automated customer care. The number of employees in mobile network customer care greatly outnumbers the OSS staff, therefore, solutions that help automate customer care activities will be a priority in the coming years.

Operators will look for automated provisioning solutions including self-configuration of the network elements which helps avoid tedious parameter setting. The move from fixed-line services to broadband consumer services stresses this further since customers need to be able to buy, configure and troubleshoot their services with minimal support.

While we see these changes coming we need to realize that the integration of Telecom, IT and IP has not yet happened. Some of the operators have a somewhat naïve vision of a future when network administrators will only look at service views and SLA status.

None of our respondents reported positive results regarding the deployment of standards. Over the years we have seen great efforts to move from one protocol to another, from OSI-based solutions to CORBA and now Web Services. This journey is based on a desire to find a technology solution to an information problem, unfortunately OSS integration standards like OSS/J has not yet proven its cost effectiveness.

Alarm quality and alarm correlation is still an underdeveloped area, although research and industry initiatives go back decades [16] the current alarm flow at a standard network operations centre is fairly basic and often of low quality.

We did not get any real numbers on filtering and correlation rates, but the informal indications pointed to very low success rate which is consistent with what is reported by Stanton [17]. In many cases alarm messages go untransformed, unfiltered, and uncorrelated from the network elements to the network management system which leads to a chaotic situation that needs to be cleaned up before we can move into service and customer management. A representative answer was:

> *[around] 40% percent of the alarms are considered to be redundant as many alarms appears at the same time for one 'fault'. Many alarms are also repeated [...]. One alarm had for example appeared 65000 times in todays browser. Correlation is hardly used even if it supported by the systems, [current correlation level is] 1-2 % maybe.*

Some operators chose to completely ignore alarm correlation, they considered it too expensive and complex to get good results. These respondents instead pointed to probing,

statistics, and performance based solutions to get an overall picture rather then trying to automate root-cause analysis. It was also stressed that advanced alarm correlation projects are in many cases signs of bad alarm quality from the low-level systems.

Finally, we let an operator conclude this survey by pointing to ongoing challenging OSS improvements:

> *Significant work underway to move to a self-service enabled environment for the customer. Whilst this is to improve the customer experience, this is also expected to dramatically increase operational efficiencies. Other key improvements are, (1) [...] capability to enable growth and the rapid on-boarding of customers, (2) Improve flow-through provisioning and activations [...] to reduce manual intervention, improve service delivery timeframes and room for human error.*

8 Conclusion

We hope that this survey can form the basis for prioritizing among research topics. The most important conclusion is probably that there is a great potential to further network management research by working closer with service providers. There is a gap between the current research efforts which typically focus on new software architectures and protocols and the telecom companies that has other priorities.

It is worth noting that after decades of research, alarm correlation is still the most prioritized research area. This can partially be interpreted as a failure, since no solution seems to be ready. Instead we see a new set of research challenges emerging, connected to self-healing, service activation and provisioning.

If research is to support the future focus areas for service providers we need to find solutions for service and quality management. Another observation is the failure of alarms as an indicator of service status, where we see a trend towards probe based solutions.

The operators gave a clear message on their desire to move from network and resource management towards customer and service management solutions. This comes as no surprise, as the trend has been clear for some time, but the path there needs attention. A new brand of OSS Solutions that are based on the service life-cycle rather then separate OSS components for different processes is needed.

References

[1] M. Bransby and J. Jenkinson, "The Management of Alarm Systems," *HSE Contract Research Report*, vol. 166, 1998.

[2] N. Stanton, *Human Factors in Alarm Design*. Taylor & Francis, 1994.

[3] B. Hollifield and E. Habibi, *Alarm Management Handbook*. PAS, 2006.

[4] TeleManagement Forum, "OSS/J." Accessed 14th of August 2008. http://www.tmforum.org/ossj/.

[5] itSMF, "Service Operation." Office of Government Commerce, ITIL Version 3 Publications, 2007.

[6] R. Boutaba and J. Xiao, "Network Management: State of the Art," *Communication Systems: The State of the Art: IFIP 17th World Computer Congress, TC6 Stream on Communication Systems, August 25-30, 2002, Montréal, Québec, Canada*, 2002.

[7] J. Strassner, *Policy-Based Network Management: Solutions for the Next Generation*. Morgan Kaufmann, 2004.

[8] L. Lymberopoulos, E. Lupu, and M. Sloman, "An Adaptive Policy-Based Framework for Network Services Management," *Journal of Network and Systems Management*, vol. 11, no. 3, pp. 277–303, 2003.

[9] D. Tennenhouse and D. Wetherall, "Towards an active network architecture," *DARPA Active Networks Conference and Exposition, 2002. Proceedings*, pp. 2–15, 2002.

[10] T. Chen and S. Liu, "A model and evaluation of distributed network management approaches," *Selected Areas in Communications, IEEE Journal on*, vol. 20, no. 4, pp. 850–857, 2002.

[11] S. Papavassiliou, A. Puliafito, O. Tomarchio, and J. Ye, "Mobile agent-based approach for efficient network management andresource allocation: framework and applications," *Selected Areas in Communications, IEEE Journal on*, vol. 20, no. 4, pp. 858–872, 2002.

[12] R. Stephan and P. Ray, "Network Management Platform Based on Mobile Agents," *International Journal of Network Management*, vol. 14, pp. 59–73, 2004.

[13] G. Pavlou, P. Flegkas, S. Gouveris, and A. Liotta, "On management technologies and the potential of Web services," *Communications Magazine, IEEE*, vol. 42, no. 7, pp. 58–66, 2004.

[14] J. Lee, "Enabling network management using Java technologies," *Communications Magazine, IEEE*, vol. 38, no. 1, pp. 116–123, 2000.

[15] TeleManagement Forum, "Information Framework (SID) ." GB922 , Release 7.5, 2005.

[16] M. Steinder and A. S. Sethi, "A survey of fault localization techniques in computer networks," *Science of Computer Programming*, vol. 53, no. 2, pp. 165–194, 2004.

[17] N. Stanton, D. Harrison, K. Taylor-Burge, and L. Porter, "Sorting the Wheat from the Chaff: A Study of the Detection of Alarms," *Cognition, Technology & Work*, vol. 2, no. 3, pp. 134–141, 2000.

Paper C

Multipurpose Models for QoS Monitoring

Authors:
Stefan Wallin and Viktor Leijon

Reformatted version of paper originally published in:
21st International Conference on Advanced Information Networking and Applications Workshops, 2007, AINAW'07

Multipurpose Models for QoS Monitoring

Stefan Wallin and Viktor Leijon

Abstract

Telecom operators face an increasing need for service quality management to cope with competition and complex service portfolios in the mobile sector. Improvements in this area can lead to significant market benefits for operators in highly competitive markets. We propose an architecture for a service monitoring tool, including a time aware formal language for model specification. Using these models allows for increased predictability and flexibility in a constantly changing environment.

1 Introduction

Service operators face many new challenges in network management [1]. Among the most important trends is the increasing move towards service centric management. It is necessary for an operator to deliver predictable Quality of Service.

There are currently few useful solutions for dealing with QoS for a large number of services types, service instances and users.

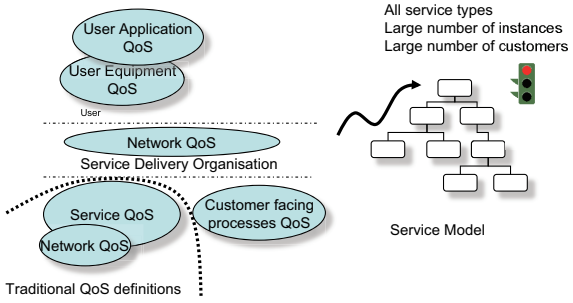

Figure 1: Overall Goal for QoS Monitoring

Figure 1 illustrates the overall QoS problem. To get a good general measure of QoS, multiple parameters need to be taken into account. It is fairly easy to have a state for a single service at a given time for one customer. But a large telecom operator needs to have an overall picture with support for different views. Different service views can include

geographical, customer-based, SLA service etc. With such service views an operator can connect technical service quality with business goals.

We are not trying to define individual QoS parameters for specific network technologies or services. Neither are we trying to define "good service quality". These areas are already well covered by individual standards, research and products. Instead we are trying to provide general capability for building large scale, multi purpose service models.

This paper defines the following components for a service management system:

- We give a simple definition of Quality of Service in Section 2.1.
- Using this definition in Section 2.2 we examine what kinds of scenarios a service monitoring tool must be able to handle.
- The outline for the architecture of a service monitoring system is given in Section 3.
- Finally, in Section3.2, we present a novel service modeling language, with built in capabilities for dealing with time and relationships between objects

2 Overview

2.1 Quality Of Service

There are several different interpretations of what Quality of Service means, each with different scopes.

In the IETF the term Quality Of Service typically refers to technologies to *achieve* "good quality". QoS in this sense is a means to prioritize network traffic, manage bandwidth and congestion, and to help ensure that the highest priority data gets through the network as quickly as possible. The IETF uses the following definition of QoS [2]: *"A set of service requirements to be met by the network while transporting a flow"*. Important solutions from this domain are IntServ [3], DiffServ [4], MPLS [5] and Policy solutions [6]. These are all important QoS solutions to achieve good service quality with Internet protocols. Similar solutions exist for other domains like fixed networks, 2G/3G networks [7] and specific services such as VoIP, customer care processes, etc. These QoS techniques are aimed at controlling and monitoring QoS in an objective way.

The ITU uses a more end-user focused definition of the term [8]: "the collective effect of service performance which determines the degree of satisfaction of a user of the service". Using this definition takes one step further in that it includes the subjective/perceived measurements by an end-user. Casas [9] presents a method to measure perceived Quality Of Service and the relationship between subjective and objective measurements.

The relationship between service quality and customer loyalty in general is discussed by Bloemer [8]. He also enhances the quality of service definition into a third level:

> *"service quality is a function of the difference scores or gaps between expectations and perceptions (P - E)".*

Why is an overall picture of quality of service needed?

SLAs: service providers want to sell SLAs where they promise a certain service quality. These promises should be expressed in a formal way that can be measured. SLAs often includes several kinds of QoS parameters ranging from technical parameters like jitter and delay to more indirect parameters like customer care or process metrics

Service quality: service providers are often overwhelmed with individual QoS parameters, but these are not integrated into an overall service view. It is hard to get information about current, past and future service quality.

Dynamic networks: services are carried over different networks like xDSL, WLAN, 3G. In order to support seamless roaming and still keep the perceived service quality we need calculations which span several different domains.

Quality Assurance: organisations are currently introducing QoS mechanisms in the traditional sense, but not always monitor the quality of the services.

QoS definition framework: new applications and services are constantly being defined. Each of these new services will have its own specific QoS parameters. There is a need for a common understanding about which these parameters are, and what their meaning is.

Governmental authorities have a growing need to monitor the quality of operators from a functional and service perspective.

For a more in-depth coverage of the motivations behind an overall solution for monitoring QoS see for instance Espvik [9] or Räisänen [10].

We conclude that the definition of QoS must be neutral and generic. It must encompass statements such as: – What is the quality of the Swedish GSM network? – What was the quality of the Swedish GSM network last Christmas Eve? – What is the quality of all the services provided to customer B? –How did a failure on a specific base station affect the GSM service in Stockholm at 10.00 AM yesterday?

Considering these requirements have lead us to adopt the following simple definition:

Definition 3 *Quality of Service is defined as a function of service parameters.*

These functions and the interpretation of the parameters are part of a specific model, and formulated by experts on the particular service. In some sense the function should represent the *"degree of conformance of the service delivered to a user by a provider in accordance with an agreement between them"* [11].

2.2 Use Cases

In order to make more concrete the kind of functionality a system for service modeling must have we present a few important use cases. Many of these use cases are represented by complicated models, and require the combination of several different areas of expertise. A service modeling language should be able to express them in the way the domain experts defines them.

2.2.1 An end-customer view

This is the traditional view in QoS, examining the quality of a single flow, for a single customer at a single point in time. This is the most basic requirement, and has been the focus of much previous research [12]. The requirement here is to be able to determine what QoS the customer is getting, and to compare that to the QoS the customer has purchased. How to compute the relevant parameters is service dependent.

2.2.2 A customer care view

If we instead view a customer care organization the picture changes slightly. The customer care representative is not as interested in the separate quality measures, since she may have responsibility for a hundred different types of service delivered to a million different customers. For customer care, an aggregate picture is required. Something that gives a picture of the total quality delivered. There is also a need for drill down capability, to explore the root cause for customer complaints.

2.2.3 A marketing view

A marketing organization has a very different focus, it needs to be able to predict customer loyalty, based on service price and service quality. This requires a model of customer behavior, and of how quality relates to loyalty [10]. This means that the same performance indicators could be used for marketing as for the end user view. But how they are viewed could be very different, and the model must be able to accommodate this.

2.2.4 A technical view

In contrast, a field engineer has a view that is relatively simple; all he wants to know is what he should fix next. He needs to be able to extract the highest-priority fault within his area of responsibility. The challenge here is to calculate a priority from the model. This requires that QoS problems be mapped to a single piece of hardware.

2.2.5 A what-if scenario

Another interesting possibility is to examine what-if scenarios. You should be able to experimentally change parameters and see their effect. If we cancel the lease on a backup

link, for example, what would the effect on customer satisfaction be? What would the financial effect be like?

3 Reference Architecture for a QoS Monitoring Solution

3.1 Overall

In order to be able to fulfill these use-cases, we are suggesting a three pronged solution consisting of 3 major components:

A modeling language with sufficient power to express the complex models needed for the use-cases. This language must be able both to express the way that parameters are computed from each other and to express the structure of the service model.

An analytic engine which can execute the modeling language and compute the values of all the parameters. This engine needs to have the appropriate interfaces, and be parallelizable so that it can be implemented in a scalable and fault-tolerant fashion.

Information visualization systems interfaces to extract and present the relevant data from the analytic engine. This can be accomplished by a combination of integration with report generators and a general, data driven interface.

It is imperative that the system has enough power to express the models, and that it is simple enough that integrating it into the support systems of an operator is feasible. This forces us to make a design trade-off between power and flexibility. There are already languages and system to create models such as Modelica [14] which is targeted at modeling physical systems. We have a different scope, but we note that just as in models of the physical world, the concept of time will be very important in our system.

3.2 A Language for Service Modeling

We use a tailor-made programming language for defining services and service level agreements. This enables us to create services using well understood methods of program construction. The language has two main purposes: first it will define the structure of the model, and second, it will define the relationship between parameters and determine how they will be computed.

We propose a simple, pure, functional language for defining calculation rules, with the following key features:

- The language is object oriented to facilitate the object oriented structure of a service model.
- We use type-inference to make the service modeling less error-prone and to facilitate composition of components.

```
class Cell:
   input errCount
   errRate = (errCount@NOW - errCount@(NOW-10m))/10
   linkErrors = sum l.errors (l in link)

class CellSL:
   properties maxOwnErrors, maxLinkErrors
   status = worst errStatus  linkStatus
   errStatus? = linearThreshold cell.errRate 0 maxOwnErrors
   linkStatus = simpleThreshold cell.linkErrors maxLinkErrors

def BronzeCellSL = CellSL( maxOwnErrors=>10, maxLinkErrors => 15)

Cell <=>* Link
CellSL => Cell
```

Figure 2: The Definition Layer

- Due to the nature of service modeling, the programming language must be able to treat time as an integral part of the syntax: all variables are seen as arrays, indexed by a time stamp.
- It is possible to use the time-index syntax to retrospectively change the value of variables.
- List comprehension and an extensive set of built-in functions provide the power needed to express complex models.

To make the language more concrete we present a simplified example taken from a model of a cellular network. The first class in Figure 2, `cell`, defines what properties we associate with a cell; and that it has a single measurement input (`errCount`). The definitions state how the parameters `errRate` and `linkErrors` should be computed from other parameters. Note the @ sign, which is used to indicate access to a time indexed value. The second definition, `CellSL`, defines a service level - a promise on the behavior of the underlying component - which encompasses the cell. It uses the parameters from the cell to form a view of the component. In effect saying "If we apply a Service Level on this cell, this is what the status would be like?".

Note that the service level is parametric with regards to the number of errors allowed. We provide the specialization BronzeCellSL, which gives specific values.

Finally we define the relationships between Cell and Links, Cell Service Levels and Cells. When relationships are defined it establishes implicit attributes: `Class1 <=>* Class2` gives the implicit *class2* attribute in `Class1` (list value) and the `class1` attribute in `class2`

Instantiation of the classes are separated from the definition. The example in Figure 3 shows the naïve case when objects are created one-by-one.

```
cell1 = new Cell
link1 = new Link
cellsl1 = new CellSL
connect cell1 link1
```

Figure 3: The Instantiation Layer

3.3 The Engine

The language requires a special runtime environment, which is responsible for marshaling inputs and outputs from the language and evaluating the expressions. All the services that the engine provides are related to variables, the main services are:

- getValue(variable,time)
- setValue(variable,time)
- subscribeVariable(variable,startTime,stopTime)

Since we define a purely functional language, there can be no side-effects of the computations except for the updating of variables. This means that the engine can utilize laziness to avoid computing unnecessary values, as well as caching to avoid re-computing values. The laziness and the possibility to implement parallelism in the engine will enable the engine to scale up to the size needed. The engine is designed to be able to handle systems of up to 2 million objects, with up to 5 million parameter calculations per minute. Although we focus on monitoring and visualizing the status of the services, we consider automatic actions to be an important part of a service management solution. These automatic actions include controlling the network or notifying operations. This will be possible by subscribing to the appropriate parameters.

3.4 The Visualization Tools

The open design of the engine makes it possible, and desirable, to build many kinds of interfaces towards the system to extract relevant information. The initial effort focuses on two major parts, a data driven data visualization tool, where you can examine objects on a simple dashboard to display their associated variables, and, drill-downs in the object hierarchy. The second part is the integration with a report generation tool, to allow periodic reports on the status of a system.

4 Related Work

Previous efforts on QoS monitoring have been focused primarily on frameworks, which we explore in section 4.1.1. There have also been some work on modeling, which we present in section 4.2.

4.1 Quality of Service Frameworks

There are several different approaches to managing and measuring service quality. We have divided them into three categories.

Control frameworks that actually try to manage resources in order to achieve a QoS Level

Single-flow measurement: focused on measuring a specific traffic flow.

End-to-end QoS Monitoring: trying to measure and monitor the overall quality of service.

4.1.1 Control Frameworks

The primary QoS techniques developed by the IETF are IntServ, DiffServ, and QoS techniques for MPLS [3, 4, 5]. Another class of QoS control frameworks rooted in the IETF are different flavors of Policy-based management (PBM) [6]. PBM provides a way to allocate network resources, primarily network bandwidth, QoS, and security, according to business policies. The European Commission has funded two programs focusing on QoS; AQUILA [13], [14] and TEQUILA [15]. Both attempt to define service definition and traffic engineering tools for the Internet to obtain quantitative end-to-end Quality. They are more focused on IP services and actually managing QoS where we are defining a framework for any service and limited to monitoring. Tequila tried to establish an IETF working group on formal service level specifications [16].

4.1.2 Single-flow measurements frameworks

Probes simulate and measure end-users behavior using the network as a black-box in order to estimate the actual delivered network service quality. Probes exists for mobile services as well as IP protocols, all have the limitation of measuring a single user at a defined location.

4.1.3 End-to-end Monitoring Frameworks

An obvious way to estimate the end-to-end QoS is of course to involve the user. MOS[17] (Mean Opinion Score) is probably the most common method for voice QoS in this area.

Eurescom funded a end-to-end monitoring effort eQOS [9]. eQOS gives an excellent background description of what we are trying to achieve. TAPAS [18] is a similar effort applying SLAs and QoS to application server solutions. It uses SLAng, see below, to define the SLAs. Services have a complex life-cycle, where monitoring QoS of a deployed service is only one phase. The EU FP6 project MobiLife [19] is studying packet-based services from an end-user viewpoint and addresses the whole life-cycle for services.

Service providers are looking for technical solutions for monitoring their operational service quality as well as to be able to sell and monitor customer specific SLAs. This has led new products in this area, for example HP OpenView SQM [20], Digital Fuel Service

Flow [21], Managed Objects SLM [22]. These products are quite successful in collecting events and measurements and monitoring SLAs. Most of the tools have weaknesses in the service modeling area; they deploy various UML flavors or simple object modeling techniques which allow for very little static analysis, for instance to be able to determine dependency graphs to facilitate lazy computation of parameters. Our work aims to improve service modeling and computational aspects such as time based calculations, and maintaining the state of a massive number of services and users.

4.2 Service Modeling

Probably the most extensive standards effort within service modeling is CIM [23]. CIM uses UML as the modeling language, and defines an XML mapping to exchange the models. The key strengths in CIM are the modeling guidelines and patterns that are used by the standard models and by enterprise extensions. The most important implementation is Microsofts implementation of CIM in their solution for management of Windows, "WMI" [24]. As pointed out by Microsoft in their System Definition Model [25] CIM can *"become unwieldy if used to describe the abstracted virtual constructs of a distributed system"*. CIM is aimed more at instrumentation rather then end-to-end service modeling. IETF is reusing the CIM model in the IETF Policy Framework WG, [26].

Many of the industry players behind CIM, are now joining up behind Service Modeling Language [27]. Based on the experience from CIM it does not start with large models.

An important part of any service model covering a defined QoS is of course the SLAs. SLAng [28] is a language focused on defining formal SLAs in the context of server products like J2EE and typical ASP environments with web services, server applications.

While CIM, SML and SLAng are rooted in the IT segment, TeleManagement Forum is trying to define a telecom related information model; the System Information Model, SID [[29] It is comparatively high level and models entities in telecom operators? processes. However, SID is being refined and moving closer to resources by incorporating CIM.

5 Conclusion and Future Work

We have shown a feasible architecture for a service monitoring framework, and described a new approach to service modeling using a formal language with suitable characteristics, such as an inherent notion of time. Our current focus is on developing and studying the formal properties of the service modeling language, and examining what kind of static analysis we can perform. We will also implement and evaluate a prototype of the framework. Another exciting angle is to study how service models are created and investigate what kind of generalizations are possible in the form of design patterns for models. We are also interested in finding characteristics for bad and good models respectively.

References

[1] S. Wallin and V. Leijon, "Rethinking network management solutions," *IT Professional*, vol. 8, no. 6, pp. 19–23, 2006.

[2] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick, "A Framework for QoS-based Routing in the Internet." RFC 2386 (Informational), Aug. 1998.

[3] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick, "A Framework for QoS-based Routing in the Internet." RFC 2386 (Informational), Aug. 1998.

[4] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Service." RFC 2475 (Informational), Dec. 1998. Updated by RFC 3260.

[5] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture." RFC 3031 (Proposed Standard), Jan. 2001.

[6] D. Verma, "Simplifying network administration using policy-based management," *Network, IEEE*, vol. 16, no. 2, pp. 20–26, 2002.

[7] 3GPP, "Quality Of Service (QoS) Concept and Architecture." 3GPP TS 23.107, version 5.13, 2004.

[8] J. Bloemer, K. de Ruyter, and M. Wetzels, "Linking perceived service quality and service loyalty: a multi-dimensional perspective," *European Journal of Marketing*, vol. 33, no. 11, pp. 1082–1106, 1999.

[9] T. Jensen, I. Grgic, and O. Espvik, "Managing Quality of Service in Multi-Provider Environment," in *Proceedings of Telecom*, vol. 99, 1999.

[10] V. Räisänen, W. Kellerer, P. Hölttä, O. Karasti, and S. Heikkinen, "Service Management Evolution," in *IST Mobile and Wireless Communications Summit*, 2005.

[11] ITU-T, " Framework of a service level agreement." Recommendation E.860, 2002.

[12] M. Galetzka, "User-Perceived Quality of Service in Hybrid Broadcast and Telecommunication Networks," in *5th Workshop on Digital Broadcasting*, 2004.

[13] T. Engel, H. Granzer, B. Koch, M. Winter, P. Sampatakos, I. Venieris, H. Hussmann, F. Ricciato, and S. Salsano, "AQUILA: adaptive resource control for QoS using an IP-based layered architecture," *Communications Magazine, IEEE*, vol. 41, no. 1, pp. 46–53, 2003.

[14] B. Koch and H. Hussmann, "Overview of the Project AQUILA (IST-1999-10077)," *Lecture notes in computer science*, pp. 154–164, 2003.

[15] A. Asgari, P. Trimintzios, M. Irons, G. Pavlou, R. Egan, S. den Berghe, T. Res, T. Ltd, and U. Reading, "A scalable real-time monitoring system for supporting traffic engineering," in *IP Operations and Management, 2002 IEEE Workshop on*, pp. 202–207, 2002.

[16] D. Goderis, Y. T'joens, C. Jacquenet, G. Memenios, G. Pavlou, R. Egan, D. Griffin, P. Georgatos, L. Georgiadis, and P. Vanheuven, "Service Level Specification Semantics and Parameters," *draft-tequila-sls-00. txt, Internet Draft, November*, 2000.

[17] ITU-T, "Mean Opinion Score (MOS)." Recommendation P.800.1, 2006.

[18] G. Lodi, F. Panzier, D. Rossi, and E. Turrini, "Experimental Evaluation of a QoS-aware Application Server," in *Network Computing and Applications, Fourth IEEE International Symposium on*, pp. 259–262, 2005.

[19] B. Mrohs, C. Rack, and S. Steglich, "Basic building blocks for mobile service provisioning," in *Autonomous Decentralized Systems, 2005. ISADS 2005. Proceedings*, pp. 82–89, 2005.

[20] Hewlett-Packard, "HP OpenView SQM." Accessed 14th of August 2008. www.managementsoftware.hp.com/products/sqm/index.html.

[21] DigitalFuel, "Service Flow." Accessed 14th of August 2008. http://www.digitalfuel.com/products/sla-management.aspx.

[22] Managed Objects, "Business Service Level Manager." Accessed 14th of August 2008. http://www.managedobjects.com/solutions/bslm.jsp.

[23] Distributed Management Task Force, "CIM Specification." Version 2.15.0, 2007.

[24] Microsoft, "WMI - Windows Management Instrumentation." Accessed 11th of Aug 2008. http://www.microsoft.com/whdc/system/pnppwr/wmi/default.mspx.

[25] Microsoft, "System Definition Model." Accessed 11th of Aug 2008. http://www.microsoft.com/windowsserversystem/dsi/sdm.mspx.

[26] B. Moore, E. Ellesson, J. Strassner, and A. Westerinen, "Policy Core Information Model – Version 1 Specification." RFC 3060 (Proposed Standard), Feb. 2001. Updated by RFC 3460.

[27] W3C, "Service modeling language," May 2008. http://www.w3.org/TR/sml/.

[28] D. Lamanna, J. Skene, and W. Emmerich, "SLAng: A Language for Defining Service Level Agreements," *Proc. of the 9th IEEE Workshop on Future Trends in Distributed Computing Systems-FTDCS*, pp. 100–106, 2003.

[29] TeleManagement Forum, "Information Framework (SID) ." GB922 , Release 7.5, 2005.

Paper D

Statistical Analysis and Prioritization of Telecom Alarms using Neural Networks

Authors:
Stefan Wallin, Viktor Leijon, and Leif Landen

Reformatted version of paper originally published in:
International Journal of Business Intelligence and Data-Mining

Statistical Analysis and Prioritization of Telecom Alarms Using Neural Networks

Stefan Wallin and Viktor Leijon

Abstract

Telecom Service Providers are faced with an overwhelming flow of alarms, which makes good alarm classification and prioritization very important.

This paper first provides statistical analysis of data collected from a real-world alarm flow and then presents a quantitative characterization of the alarm situation. Using data from the trouble ticketing system as a reference, we examine the relationship between the original alarm severity and the human perception of them.

Using this knowledge of alarm flow properties and trouble ticketing information, we suggest a neural network-based approach for alarm classification. Tests using live data show that our prototype assigns the same severity as a human expert in 50% of all cases, compared to 17% for a naïve approach.

1 Introduction

A medium-sized telecom network operations center receives several hundred thousand alarms per day. This volume of alarms creates severe challenges for the operations staff. Fundamental questions that need answers in order to improve the state of affairs are:

- Which alarms can be filtered out?
- How can we group and correlate alarms?
- How can we prioritize the alarms?

While extensive research efforts are focused on alarm correlation [1], the target for the work presented in this paper is *filtering* and *prioritization* of alarms.

Although all alarm systems support advanced filtering mechanisms, the problem is defining the filtering rules. Being able to filter out a high percentage of alarms would increase efficiency of the network management center since network administrators would only have to work with relevant problems.

Because there is such a high volume of alarms and tickets this kind of filtering and prioritization is of vital importance if operators are to determine which alarms are most critical to resolve [2]. Today, prioritization of alarms and trouble tickets is largely performed manually by network administrators who use a combination of their experience and support systems such as inventory and SLA management systems to determine the

priority of an alarm. This manual process makes the organization dependent on a few individual experts [3]. Furthermore, the priority information is typically only available in the trouble ticket system and not in the alarm system.

Two hypotheses are studied in this paper: that *statistic analysis techniques can be used to find alarm filtering strategies* and that *a learning neural network could suggest relevant priorities by capturing network administrators' knowledge*.

We start by define the inner workings of a telecom alarm flow (Section 2) and then describe how our data was extracted from the database (Section 3).

This paper takes four steps towards automatic alarm prioritization:

- We present some statistical properties of a real world alarm flow taken from a mobile service provider (Section 4).
- We show important properties such as that 11% of all alarms belong to easily identifiable classes of alarms which never give rise to actions from operators and that over 82% belong to classes where less than one alarm in a thousand generate an action.
- We describe the construction, training and validation of a neural network which successfully assigns priorities to incoming alarms (Section 5).
- Using statistical analysis we show that the neural network performs significantly better than a naïve but realistic alternative.

2 Defining the Alarm Flow

The operational activities at a service provider's Network Management Center are focused on managing a constant flow of alarms [3]. A primary goal is to resolve the most important problems as quickly as possible. A simplified process description for lowering error impact is:

- Group alarms that are related to the same problem.
- Associate the alarms with a trouble ticket to manage the problem resolution process.
- Assign a priority to the trouble ticket.
- Analyze and fix the problem.

Alarms are refined and distributed from the detection point in individual network elements, such as base stations, via subnetwork managers up to the overall integrated network management systems. Various interface technologies and models for alarms are used across these interfaces. X.733 [4] is the *de facto* standard for *alarm* interfaces and all later standard efforts are based on X.733 to some degree. It contains basic definitions of parameters in alarm notifications.

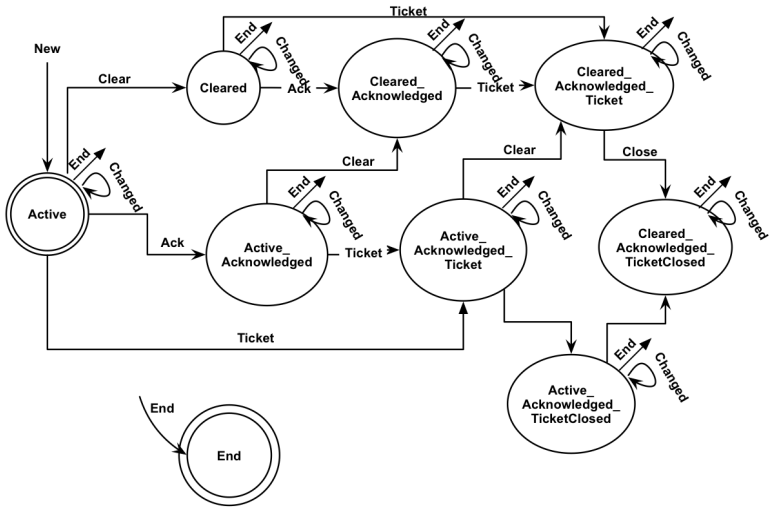

Figure 1: Finite State Machine for management view of alarms

The 3GPP Alarm IRP [5] defines alarms using a state focused definition. Furthermore, it models operator actions that can change the state of an alarm. The state where the alarm is acknowledged and cleared is the final state, and the life cycle of the alarm ends.

An X.733 compliant alarm has the following main attributes.

Managed Object: identification of the faulty resource. It points not only to the network element but also down to the individual logical or physical component.

Event Type: category of the alarm (communications, quality of service, processing error, equipment alarm, environmental alarm).

Probable Cause: cause of the alarm, where the values are defined in various standards.

Specific Problem: further refinement of probable cause.

Perceived Severity: an indication of how it is perceived that the capability of the managed object has been affected. Values are `indeterminate, warning, minor, major, critical and clear`.

Event Time: the time of the last event referring to this alarm.

Additional Text: free form text describing the alarm.

For the purpose of this study, we have defined a Finite State Machine for alarms as shown in Figure 1. It is a simplification and abstraction of major standards and typical telecom management systems.

From the resource point of view, the main events are `new` and `clear`, which moves the alarm into the `active` or `cleared` state. Note, however, that the cleared state does not imply that the network administrator considers the problem solved. This is managed by the trouble ticket process. In order to manage the problem, the user acknowledges and associates a 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. The mobile operator we studied used a priority in the trouble ticket system ranging from 1 to 6. Priority 1 is the most urgent and indicates a problem that needs to be resolved within hours, whereas priority 6 has no deadline. 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 `end` in the above state diagram. In many cases, this is automatically performed when the associated trouble ticket is closed. There is a short cut to bypass the trouble ticket process in order to end alarms that do not represent real problems.

Whether an alarm notification should be considered a *new* or *changed* alarm is a topic of its own. According to X.733, a notification with the same managed object, event type, probable cause and specific problem is considered to change an existing alarm. The three last parameters identify the type of alarm, and we will refer to the triple <`Event type, Probable cause`, `Specific problems`> as "alarm type". We will refer to alarms with the same managed object and alarm type as "associated alarms", and they will be grouped together under one main alarm, the changed alarm in the state diagram.

The X.733 definition of alarm type has created various vendor-specific mechanisms since the parameters are static and defined by standards. In real life, a vendor needs to be able to add new alarm types in a deployed system. The approaches to circumvent this differ between vendors, some use their own non-standardized probable cause values, and some use free text specific problem fields. A second problem is how to identify the managed object, different protocols and vendors use different naming schemes. The actual resolution varies from vendor to vendor. These two fundamental problems create major challenges for alarm systems and alarm analysis.

In the described alarm flow, network administrators need to answer two important questions: *Do I need to handle this alarm? What is its priority?* The operator we studied primarily uses the event time, managed object, and alarm type attributes in combination with their own experience and lookups in support systems to judge the alarm relevance.

3 Data Mining Process

Figure 2 illustrates the overall process in data mining [6]. Each step in this process is described below.

3.1 Data Set Selection

We used historical databases of alarms and trouble tickets as input. Two separate data sets were extracted at different times:

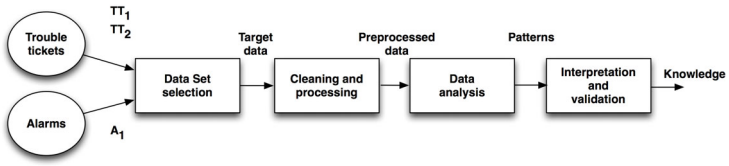

Figure 2: Data mining process.

1. $\{TT_1\}$: Only trouble tickets (85814 tickets associated with 260973 alarms)

2. $\{A_1, TT_2\}$: Alarms and corresponding tickets (3583112 main alarms, 16150788 associated alarms and 90091 trouble tickets)

The alarms were from approximately 50 different types of equipment from many different vendors.

The records in the trouble ticket database contain the most important fields from the alarms such as managed object, specific problem and additional text. In this way the trouble ticket database alone is sufficient to perform analysis and neural network training on alarms with associated tickets. This is the case for $\{TT_1\}$ which was used for training and testing of the priority network, as described in Section 5.

In order to perform full statistical analysis of the alarms as well as to train the neural network to judge if the alarm should be handled or not, we extracted a second data set $\{A_1, TT_2\}$.

3.2 Cleaning and Processing

The cleaning and processing of the alarms meant calculating *time dimensions*, and generating unique *alarm type identifiers*. It is not always easy to judge what identifies the true original time of the alarm detection since some equipment sets an absolute time stamp in the alarm, whereas in other cases it is not time stamped until it reaches the network management systems. Since we are studying the administrative processes around alarm management and not temporal alarm relationships, we simplify the data by using the time the alarm was created in the database as reference. This would not be a valid approach in the alarm correlation case where the time of alarm detection is relevant.

A fundamental problem is to define what identifies a unique alarm type. As described in Section 2 this is not as simple as applying the X.733 rule of event type, probable cause and specific problem. The data in our alarm database had free text alarm types embedded as part of the additional text field. This was implemented as part of every equipment integration into the overall network management system. We extracted this text and made it a column in the database. The data set contained about 3500 unique specific problem values.

Table 4.1: Distribution of priorities in A_1.

Severity	Frequency
Indeterminate	0.1%
Critical	17.5%
Major	22.8%
Minor	5.0%
Warning	54.6%

3.3 Data Analysis

The data analysis was done using both statistical analysis as described in Section 4 and as training/testing data for the neural network, as described in Section 5.

4 Quantitative Analysis of Alarm Flow

4.1 Basic Description of the Data

This is a quantitative analysis of the alarm set A_1, containing a total of 3 583 112 alarms, 12 567 676 associated alarms (on average 3.51 per main alarm) and 90 091 tickets. Among the alarms which are connected to tickets there is an average of 36 associated alarms, while unconnected alarms had only 3 associated alarms.

The distribution of severities is shown in Table 4.1 and shows us that the distribution of severities is very uneven.

Just under half of the alarms completely lack associated alarms, and 98.8% of them have less than ten associated alarms.

4.2 Alarm Types

In total there are over 3500 different alarm types defined (see Section 3.2), but most of them are very rare. Given the distribution shown in Figure 3 we see that some form of the Pareto principle seems to be at work here. We get 90% of all our tickets from the 26 most common alarm types. This indicates that there is a great deal to gain from improving automation for these 26 alarm types, or using methods such as improved alarm enrichment and correlation or perhaps helping semi-automate the ticket creation process.

On the other side of the scale, 10.6% of all alarms belong to alarm types that have never given rise to an actual ticket, and if we look at alarms which belong to an alarm type that results in a ticket less than once every 1000 alarms, we get 82.1% of all alarms. While this requires a detailed study, it does suggest that a great reduction in alarm volume can be achieved by exploring filtering on these alarms.

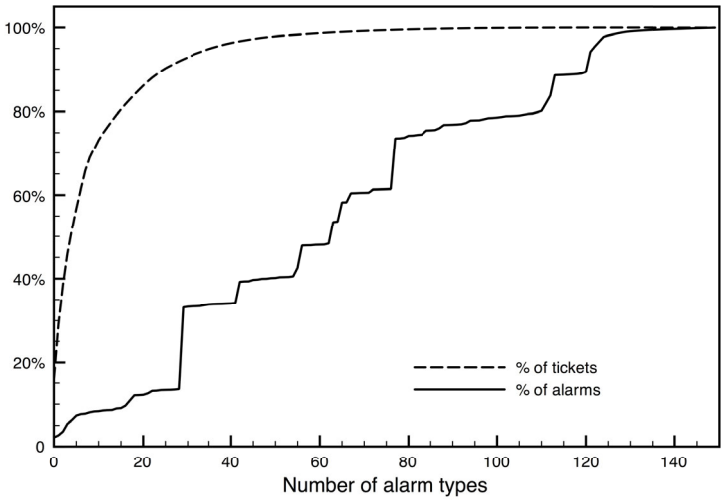

Figure 3: Cumulative distribution of tickets and alarms by alarm type.

Table 4.2: Number of alarms and tickets by weekday

Day	Num. alarms	Alarms/day	Num. tickets	Tickets/day
Monday	507 308	29 842	9 247	544
Tuesday	544 093	32 005	12 798	753
Wedensday	548 529	32 266	9 195	541
Thursday	562 218	31 234	9 761	542
Friday	527 041	31 002	8 405	494
Saturday	452 910	26 642	6 017	354
Sunday	440 705	25 924	4386	258

4.3 Temporal Properties

The material contains data from the May to September. A total of 120 days are covered with an average of 29859 alarms a day, or 1244 alarms per hour. In contrast, an average of only 116 tickets were created per day. The differences between the months are small.

One common wisdom in the telecom industry is that many alarms are caused by operator activity, that many the problems come from upgrades, reconfigurations and other events, implying that the operations themselves cause many alarms. To test this we calculate the average number of alarms and tickets for the weekdays (see Table 4.2). We can see that the busiest day (Wednesday) has 24.5% more alarms than the calmest day (Sunday), while having more than double the number of tickets.

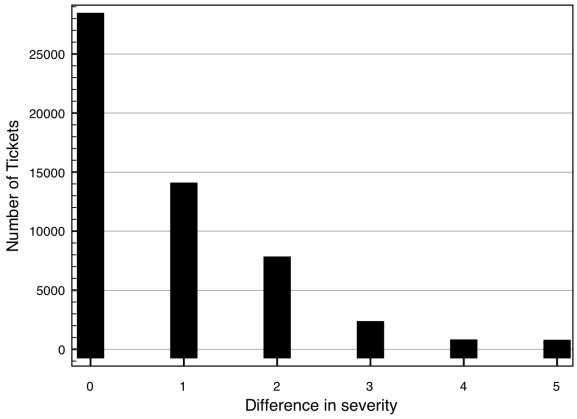

Figure 4: Differences in associated alarm severity for tickets.

4.4 Severities versus Priorities

This section studies the relationship between priorities and severities, and refers to data from data set TT_2 (see Section 2).

The number of alarms associated with a trouble ticket varies from 1 to 1161, an average of 5.2 with a standard deviation of 17. This is an indication that the relationship between alarms and tickets is complex. In an ideal world, alarms should indicate a problem and not individual symptoms. If this were true, the fan-out between tickets and alarms would have been much lower.

In Figure 4 we can see that the distribution of alarm severities associated with a single trouble ticket is low. If, for example, a trouble-ticket is associated with both a `major (2)` and a `warning (4)` alarm, it has a difference of two steps, as illustrated by Figure 4. The associated alarms have the same severity in more than 50% of the cases.

Discussions with the network administrators led to the hypothesis that if we looked at the maximum alarm severity associated with the trouble ticket, we would get good correlation. Figure 5 shows the analysis of this assumption. It clearly illustrates that we do not have a mapping between alarm severity and corresponding priority. For example, we see that priority 4 is distributed across all severities, and it is largest in the `warning` and `critical` severity. For further discussion on how badly severities and priorities correlate, see Section 5.4.

After studying the weak correlation between alarm severity and ticket priority, we looked for another correlation: the alarm type versus priority. We observe a strong correlation between some alarm types and their priority, but for other alarm types there is no correlation at all. So there is no direct and naïve alarm algorithm associating priority and alarm information.

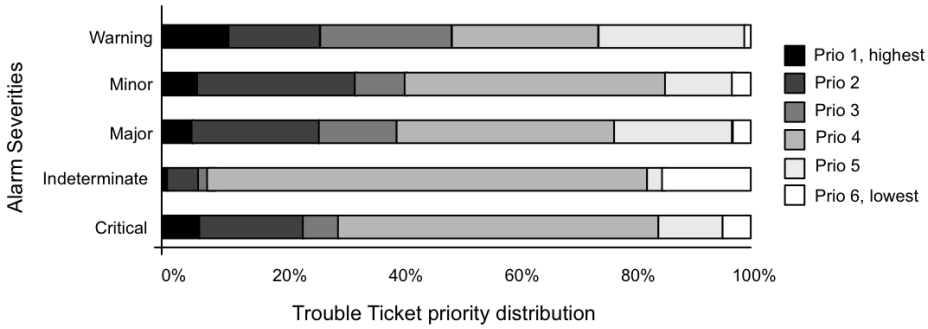

Figure 5: Ticket priorities per maximum severity.

5 Using Neural Networks for Prioritizing Alarms

5.1 The Architecture

Current solutions for *automatic* alarm prioritization are mainly based on service impact tools and expert systems which use information from external systems to modify alarms. This allows a prioritization algorithm to decide upon priorities since the alarm is bound to topology, service and business impact [7]. This type of solution have some intrinsic problems [8]:

- *Maintenance of service models*: formal models of network and service topology have to be maintained, which is complex and costly. Also, the change rate of network topology and service structures is challenging to handle.

- *Maintenance of impact rules*: correlation rules are typically expressed using Rete-based expert systems [9], which require extensive programming.

- *Capturing operators knowledge*: to write the rules for the expert system, the developers need to have input from the network administrators. However, experienced operators are often critical resources in the organization and cannot be allocated time to formalize rules.

The goal of our work is to set priorities in alarms at the time of reception by using neural networks. We let the neural network learn from the experienced network administrators rather then to building complex correlation rules and service models. As stated by [10]:

> Experienced decision makers do not rely on formal models of decision making, but rather on their previous experience. They use their expertise to adapt solutions to problems to the current situation.

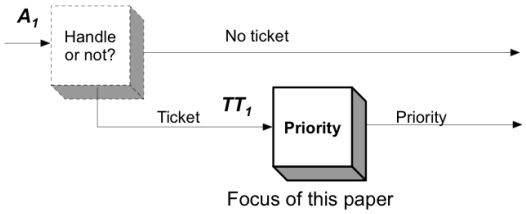

Figure 6: The two-step neural network training

The alarm management process can be viewed as a two-step procedure, see Figure 6. First of all, is the alarm important enough to create a trouble ticket? Secondly, if we create the trouble ticket what is the priority? We focus on the latter in this paper.

Deciding if an alarm deserves to be handled or not seems to be a more complicated question, based on some preliminary tests, information outside of the raw alarms is probably used in the decision process.

We have integrated a neural network architecture into an alarm and trouble ticket system. The neural network uses the manually assigned trouble ticket priorities and associated alarms as learning data. When the alarm system receives an alarm, it interrogates the trained neural network which generates a suggested priority to be put into the alarm information. The priority indicates if the alarm should be handled or not and, if so, the suggested priority.

The A_1 data set was used to train the network to judge if a trouble ticket should be created. This works since the alarm database contains a field for each alarm telling if it has a trouble ticket or not. The TT_1 data set was used to train the network to assign priorities to the alarms. The trouble ticket database contains all relevant attributes of the associated alarms.

We used the following alarm fields to construct the input to the neural network:

- Managed Object: the resource emitting the alarm
- Specific Problem: alarm type
- Additional Text: free text field with alarm information
- Perceived Severity: original severity as reported from the equipment
- Associated alarms count: the number of alarm notifications referring to the same alarm.

The selection of the above alarm attributes was based on discussions with network administrators to find the most significant attributes used for manual correlation and prioritization. `Additional text` and `Specific problem` are encoded using the soundex algorithm to remove the influence of numbers and convert the strings to equal length.

We used an open source *neural network engine* named `libF2N2` [11]. The lib2f2n2 library uses linear activation

$$f(x) = x$$

for the input layer and the logistic function

$$f(x) = 1/(1 + e^{-x})$$

for all successive layers. The neural network uses back-propagation [12] as the learning mechanism. It only supports iterative back-propagation, not batch back-propagation.

Two variables play a special role during learning: the *learning rate* and the *momentum* of the neural network. Learning rate indicates what portion of the error should be considered when updating the weights of the network. Momentum indicates how much of the last update that should be reused in this change. Momentum is an effective way to move a network towards a good generalization, but can also be a problem if the momentum moves us away from the optimum weights.

5.2 Assigning Priorities using Neural Networks

In Section 4.4 we showed that severities cannot be used as priorities. Our solution will, however, give network administrators a proposed priority based on their experience.

The priority of an alarm partially depends on its grouping with other alarms. The grouping in this case is performed by the network administrators when performing manual alarm correlation and creating the trouble ticket. The neural network, on the other hand, will analyze alarms one by one and assign a priority.

5.3 Test Configurations

The prototype was tested with different settings both for learning rate, momentum and neural network structure. The tests are outlined in Table 4.3. Since the learning is slow, approximately 3 minutes per epoch, we stopped the learning when the error rate stabilized. Each test was performed with data *not* used during the training and is randomly chosen from the data-set. About 10% of the data set was used for training.

`Layers`, `Neurons` and `Output` in Table 4.3 describe the architecture of the neural network. The number of neurons used in each hidden layer is given by the `Neurons` field of the table. The 6 neuron `Output` (Test 1, 2, 3 and 5) was a mapping where each neuron represented one priority and the one that got the highest result was the proposed priority. The 1 neuron `Output` (Test 4) was a scaled priority where 0,1 represented priority 1 and 0,25 represented 2 and so on. `Training Error` is the average of the mean square error of the last epoch in the training. `Testing Error` is the average error of each prioritized alarm. A priority error of one, e.g., assigning priority 3 instead of 4, equaled 20% in the 6 neuron output and 15% in the 1 neuron output.

The tests show us that the number of layers are more important than the number of neurons in each layer. The extremely low error on Test 3 shows that 4 layers is a better alternative than 3. Test 5 shows that with only 2 layers the application does not

Table 4.3: Test Results: L = Layers, N = Neurons, O = Output, LR = Learning Rate, M = Momentum, Tr E = Training Error, Te E = Test Error

Test	Epochs	L	N	O	LR	M	Tr E	Te E
1	1200	3	200	6	0,01	0,3	3.1%	18,1%
2	680	3	100	6	0,03	0,3	2,4%	17,7%
3a	100	4	50	6	0,03	0,2	6,1%	16,0%
3b	1000	4	50	6	0,03	0,2	4,7%	16,9%
4	300	3	70	1	0,05	0,2	31,8%	12,8%
5	1000	2	100	6	0,05	0,2	3,1%	30,0%

learn to prioritize. Adding more neurons does not make the neural network prioritize better. Notice how the error drops when decreasing the number of neurons between Test 1 and Test 2. Although the `Testing Error` drops considerably when using the 1 neuron `Output` the high `Training Error` makes us reluctant to use it.

In Test 3 we can see that we do not necessarily get a better result from longer training. More work is needed to find a suitable method of when to stop training. We have no direct correlation between `Training Error` and `Testing Error`. The high `Training Error` on Test 4 is accompanied by a low `Testing Error` and in Test 5 we have the opposite relation.

Figure 7 shows how mean square error descends during training for test 1. All tests, except test 4, produced similar graphs.

5.4 Results

Having identified approach 3b, see Table 4.3, as the most promising one, we decided to statistically evaluate the success of this algorithm. For all the confidence intervals below we have used a standard t-test.

We compared the neural network to four competing approaches:

1. The neural network approach as described for 3b.

2. The trivial severity approach, simply scaling the severity into the priority by multiplying the severity by 1.25.

3. Always selecting the statically optimal priority optimizing for “least average error”, computed from the reference. The statically optimal choice turned out to be 3.

4. Selecting the priority at random for each alarm, but using the same distribution of priorities as the reference.

The average errors and variances for the methods, together with the 99% confidence interval on their mean errors are given in Table 4.4.

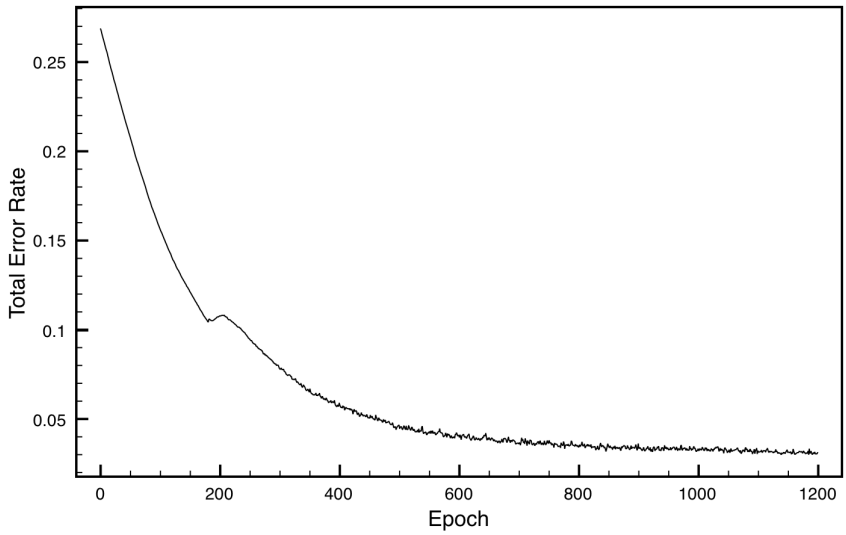

Figure 7: Mean square error decrease during learning

Table 4.4: Average error and variants for the methods.

Method	Average Error	Error variance	Confidence interval	Difference from A
1	0.8001	1.3450	[0.7938 . . . 0.8064]	-
2	1.9054	0.9463	[1.8979 . . . 1.9129]	[1.0973 .. 1.1132]
3	1.1922	0.4063	[1.1881 . . . 1.1963]	[0.3973 .. 0.3869]
4	1.6161	1.4241	[1.6080 . . . 1.6234]	[0.8235 .. 0.8084]

We have plotted the relative frequency of the errors in Figure 8. A negative error errs on the side of caution, judging an alarm to be more serious than it actually is. A positive error on the other hand is an underestimation of the importance of the alarm. For the purpose of this study we consider both types of error to be equally bad. We can see that while the neural network is the only method centered around zero, the others generally overestimate the seriousness of the alarms slightly.

Having seen what appears to be a distinctive advantage for the neural networks, we apply t-tests to try to determine how the effect of the other methods compares to A. We did this by comparing the error pairwise for each alarm in the test set and computing the 99% confidence intervals. The results are presented in the final column of Table 4.4.

It is spectacular how well the "always pick priority 3"-method does. This is because it guesses in the middle, so it is often wrong, but usually just a single step. This rule is, of course, worthless from a prioritization point of view.

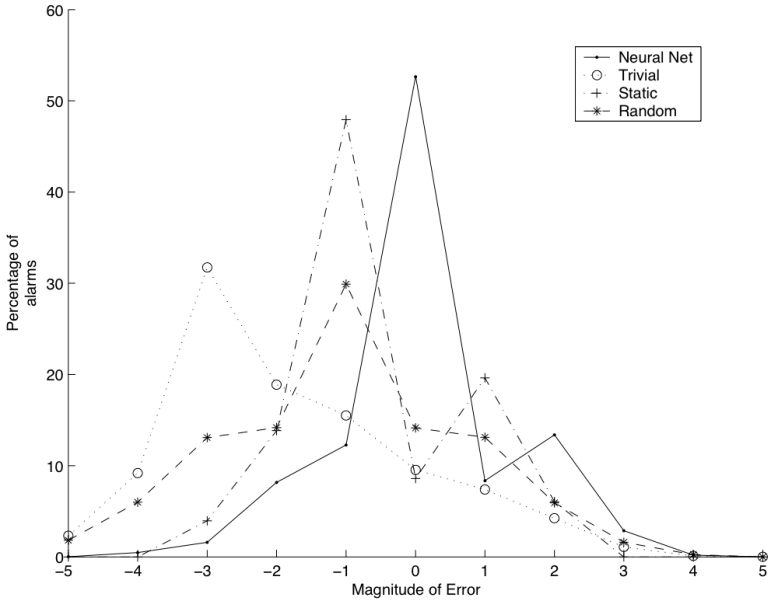

Figure 8: Distribution of errors (%)

The comparisons are all to the advantage of the neural network. It has a statistically secure advantage over the other methods. This suggests a distinct advantage of using neural networks for alarm prioritization. This becomes an even stronger conclusion when one considers the distribution of errors in Figure 8.

6 Related Work

Data mining, or knowledge discovery in databases, is being used in different domains such as finance, marketing, fraud detection, manufacturing and network management [13]. The major categories of machine learning used are rule induction, neural networks, case-based reasoning, genetic algorithms, and inductive logic programming [6]. The following problem types are typically addressed:

- *Classification*: the training data is used to find classes of the data set. New data can then be analyzed and categorized. This is the main theme of our work, where we are looking for ticket priorities based on the alarm data.

- *Prediction*: this focuses on finding possible current and future values such as finding and forecasting faults in a telecommunication network. This is covered well by related research efforts.

- *Association*: attempts to find associations between items such as dependencies between network elements and services [14].

- *Detection*: focuses on finding irregularities in the data and seeks to explain the cause. Within the telecom sector a common application is to detect churn and fraud.

Bose et. al. [6] performed a study to analyze the usage of data mining techniques across domains and problem types. They found that 7% of the usage was in the telecom sector and almost evenly spread among classification, prediction and detection.

Gardner et. al. [15] illustrate the classification category. They use a self-organizing map, a Kohonen network [16], to categorize alarms. In contrast with conventional ANN networks, a self-organizing map does not require a correct output as training data. The primary application is analysis and classification of input where unknown data clusters may exist. The network is in a sense self-learning. This is in contrast to our prioritization scenario where we *have* a complete output definition.

Most research efforts related to alarm handling focus on correlation [17, 18, 19, 20, 10]. This mainly falls into the *prediction* and *detection* categories above. Alarm correlation refers to the process of grouping alarms that have a reciprocal relationship [21]. The aim is "*the determination of the cause*" [8]. Wietgrefe et. al. [22] uses neural networks to perform the correlation. Common for these efforts is that they look at the stream of alarms and tries to find the root cause or the triggering alarm. For example, in the Wietgrefe study, the learning process is fed with alarms as inputs and the triggering alarm as output.

We are not trying to find the root cause of the alarms, neither to group them. In contrast, our problem is in some sense a simpler one, but overlooked; *to prioritize the individual alarms*. The *main* input of our analysis is the manual alarm prioritization in trouble tickets along with the alarms. Previous efforts have mostly focused on the alarm databases themselves. The use of the trouble ticket database in the learning process makes the solution adapt to expert knowledge. Training a network with root cause alarms is a fundamental challenge since it is hard to find a true output set.

Sasisekharan et. al. [23] combine statistical methods and machine learning techniques to identify "patterns of chronic problems in telecom networks". Network behavior, diagnostic data, and topology are used as input in the solution. This solution covers the challenging aspect of problem prediction. It is more focused on data-mining techniques and uses topology as input. It shows a strength in combining several approaches.

Levy [24] also combine data mining and machine learning in an implementation of an alarm system. They come to the same conclusion as we do regarding the Pareto distribution of the alarms: "there is a lot of value in the ability to get an early warning on the 10% of causes that create 90% of field failures". This further emphasizes the foundation for the work presented in this paper where we are focusing on pinpointing the relevant alarms to be handled.

Klemettinen [25] presents a complementary solution to alarm correlation, using semi-automatic recognition of patterns in alarm databases. The output is a suggestion of rules,

users can navigate and understand the rules. This is in contrast with neural networks solutions like ours, where there is no explanation to users why our network suggests alarms to be handled, and the suggested priority.

7 Conclusion and future work

The network operator that was the source for our data indicated that priority estimates that were within one step of the true value would be useful, something we manage for most of the alarms as shown by Figure 8. This is a statistically significant improvement for operators compared with the currently available alarm severity.

We have presented a solution that adjusts automatically to network administrators by learning from the trouble ticket database. This is an efficient use of human expert knowledge compared to traditional approaches using rule-based systems which has the underlying problem of converting human knowledge to rules. It also avoids the complex problem of having a complete set of rules and topology information in an ever-changing environment.

This solution has several benefits for the service providers:

- Priorities are available immediately as the alarms arrive.
- Captures network administrators' knowledge without disturbing their business-critical work.
- Adapts to changing behavior and changing network topologies.

Further, we have provided some characterization of the alarm flow, showing the long-tail behavior of alarm types, providing an opportunity for further study of the possibility of exploiting this from two sides, both from alarm filtering at the "non-important" side and for alarm automation/enrichment at the "important" side.

The work presented in this paper was run using a historic database of alarms and trouble tickets. The next step is to deploy the test configuration in a running system to study how it adapts continuously. We will also apply the tests using data from a different operator.

References

[1] M. Steinder and A. S. Sethi, "A survey of fault localization techniques in computer networks," *Science of Computer Programming*, vol. 53, no. 2, pp. 165–194, 2004.

[2] J. Wilkonzon and D. Lucas, "Better alarm handling- a practical application of human factors," *Measurement and Control*, vol. 35, no. 2, pp. 52–55, 2002.

[3] S. Wallin and V. Leijon, "Rethinking network management solutions," *IT Professional*, vol. 8, no. 6, pp. 19–23, 2006.

[4] ITU-T, "X.733: Information technology - Open Systems Interconnection - Systems Management: Alarm reporting function," 1992.

[5] 3GPP, "3GPP TS 32.111: Alarm Integration Reference Point (IRP)," 2004.

[6] I. Bose and R. K. Mahapatra, "Business data mining, a machine learning perspective," *Information & Management*, vol. 39, no. 3, pp. 211–225, 2001.

[7] S. Wallin and V. Leijon, "Multi-Purpose Models for QoS Monitoring," in *21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07)*, pp. 900–905, IEEE Computer Society, 2007.

[8] R. Sterritt, "Towards autonomic computing: effective event management," in *Proceedings of the 27th Annual Software Engineering Workshop*, pp. 40–47, NASA Goddard/IEEE, 2002.

[9] C. L. Forgy, "Rete: a fast algorithm for the many pattern/many object pattern match problem," *IEEE Computer Society Reprint Collection*, pp. 324–341, 1991.

[10] M. C. Penido G, Nogueira J.M, "An automatic fault diagnosis and correction system for telecommunications management," in *Proceedings of the Sixth IFIP/IEEE International Symposium on Integrated Network Management*, pp. 777–791, 1999.

[11] "libF2N2 Feedforward Neural Networks." Accessed 10th of August, 2007. http://libf2n2.sourceforge.net/.

[12] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations* (D. E. Rumelhart and J. L. McClelland, eds.), pp. 318–362, MIT Press Cambridge, MA, USA, 1986.

[13] R. J. Brachman, T. Khabaza, W. Kloesgen, G. Piatetsky-Shapiro, and E. Simoudis, "Mining business databases," *Communications of ACM*, vol. 39, no. 11, pp. 42–48, 1996.

[14] C. Ensel, "Automated generation of dependency models for service management," in *Workshop of the OpenView University Association*, (Bologna, Italien), 1999.

[15] R. D. Gardner and D. A. Harle, "Alarm correlation and network fault resolution using the Kohonen self organising map," in *Global Telecommunications Conference (GLOBECOM'97)*, vol. 3, 1997.

[16] T. Kohonen, *Self-Organizing Maps*. Springer, 2001.

[17] P. Fröhlich, W. Nejdl, K. Jobmann, and H. Wietgrefe, "Model-Based Alarm Correlation in Cellular Phone Networks," in *Fifth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 1997.

[18] G. Jakobson and M. D. Weissman, "Alarm correlation," *Network, IEEE*, vol. 7, no. 6, pp. 52–59, 1993.

[19] G. Liu, A. K. Mok, and J. E. Yang, "Composite events for network event correlation," in *Proceedings of the Sixth IFIP/IEEE International Symposium Network Management, 1999*, pp. 247–260, 1999.

[20] D. M. Meira, *A Model For Alarm Correlation in Telecommunications Networks*. PhD thesis, Federal University of Minas Gerais, November 1997.

[21] R. D. Gardner and D. A. Harle, "Methods and systems for alarm correlation," in *Global Telecommunications Conference (GLOBECOM'96)*, vol. 1, 1996.

[22] H. Wietgrefe, K.-D. Tuchs, K. Jobmann, G. Carls, P. Fröhlich, W. Nejdl, and S. Steinfeld, "Using neural networks for alarm correlation in cellular phone networks," in *International Workshop on Applications of Neural Networks to Telecommunications (IWANNT)*, 1997.

[23] R. Sasisekharan, V. Seshadri, and S. M. Weiss, "Data mining and forecasting in large-scale telecommunication networks," *IEEE Expert: Intelligent Systems and Their Applications*, vol. 11, no. 1, pp. 37–43, 1996.

[24] D. Levy and R. Chillarege, "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.

[25] M. Klemettinen, H. Mannila, and H. Toivonen, "Rule discovery in telecommunication alarm data," *Journal of Network and Systems Management*, vol. 7, no. 4, pp. 395–423, 1999.

Paper E

SALmon - A Service Modeling Language and Monitoring Engine

Authors:
Viktor Leijon, Stefan Wallin, and Johan Ehnmark

Reformatted version of paper originally published in:
4th international Symposium on Service-Oriented System Engineering

SALmon - A Service Modeling Language and Monitoring Engine

Viktor Leijon, Stefan Wallin and Johan Ehnmark

Abstract

To be able to monitor complex services and examine their properties we need a modeling language that can express them in an efficient manner. As telecom operators deploy and sell increasingly complex services the need to monitor these services increases.

We propose a novel domain specific language called SALmon, which allows for efficient representation of service models, together with a computational engine for evaluation of service models. This working prototype allows us to perform experiments with full scale service models, and proves to be a good trade-off between simplicity and expressive power.

1 Introduction

Operators want to manage services rather than the network resources which are used to deliver the services. This change of focus is driven by several factors; increased competition, more complex service offerings, distribution of services, and a market for Service Level Agreements [1].

A result of this transition is an increasing need to predict, monitor and manage the quality of the service that is delivered to the end users. However, the complexity of understanding and modeling services is a serious obstacle.

We want to find a way to model Services, Service Level Agreements and the structure underlying them.

Service modeling is intrinsically hard, since we need to express calculations, types and dependencies. Current UML-based approaches tend to hide this without really providing the expressive strength needed. On the other hand, using traditional object-oriented programming languages gives the expressive strength but creates a gap between the model and the domain experts. Time-dependent calculations are often complicated or unnatural to express in these languages.

An implementation challenge is to manage the *volume* of service types and instances. Service providers have large infrastructure and service portfolios. There can be several million cells, edge devices, areas and customers.

Managing a large number of object instances with calculated Key Performance Indicators, including indicators which are calculated over time intervals, have computational challenges which are not addressed in current solutions. Time is an inherent dimension in service monitoring and SLA management for several reasons. We need to be able to manage late arrival of data, there may be delays between the collection of a key performance

indicator and its introduction into the SLA system for instance due to batching. The actual time-stamp must be used in the overall calculation of status which may require recalculation. Operators also want to make "time-journeys", looking backwards and forwards to understand how service quality has developed. Furthermore, SLAs contain time variables in the form of requirements on time-to-repair and availability measurements.

It is vital to be able to provide different views for different users. Naive attempts to model services use a tree structure where Key Performance Indicators are aggregated upwards in the tree. However, different roles in the organization require different types of aggregation views: per customer, per site, per area, per service, and ad-hoc grouping of service instances.

The main components of our solution are a dedicated service modeling language and a run-time environment for calculating the service status. The language is an object-oriented functional language tailored to the domain-specific requirements.

This paper makes the following main contributions towards a useful service monitoring engine:

- We give an overview of the design considerations that went into SALmon, a novel language for writing service descriptions (Section 2).
- This language has been implemented in the form of a prototype implementation of a calculation engine that we discuss in Section 3
- Finally we examine a few typical scenarios and how they can be handled in our system (Section 4).

2 The Modeling Language

We employ a tailor-made programming language for defining services and service level agreements. This enables us to create services using the well understood methods of program construction. The language has two main purposes: first, it will define the structure of the model, and second, it will define the relationship between parameters and determine how they will be computed.

The language is a simple functional language for defining calculation rules. Calculations are associated with properties in object to facilitate the object-oriented structure of a service model.

Due to the nature of service modeling, the programming language must be able to treat time as part of the normal syntax: all variables are seen as arrays indexed by a time stamp. It is possible to use the time-index syntax to retrospectively change the value of variables.

List comprehension and an extensive set of built-in functions provide the power needed to express complex models. To make the language more tangible we present a simplified example taken from a model of a GSM network, see Section 2.3. The language has two fundamental layers: the Definition Layer and the Instantiation Layer.

2.1 Definition Layer

The definition layer defines the classes and calculations in the model. Core concepts that we want to represent as classes are Services and SLAs. Classes have inputs, anchors, attributes and properties:

Inputs define a time-indexed variable that is mapped to an external data source. Typical external sources are probes, alarms, performance data and trouble-tickets.

Anchors label connections to other class instances and hence provide the basis for building structures from service objects.

The definition layer only defines the name of the anchor and its multiplicity, so that an anchor is defined to have either exactly one anchored instance *or* zero or more.

Properties are values that can be left undefined in a class definition to yield an abstract base class. Properties can be defined or redefined in sub classes to model differentiated service levels.

Attributes define calculation rules for parameters in a strict purely functional language.

The calculation rules have knowledge of which instance of the attribute is being evaluated, and can use that together with attributes, properties and other anchored objects to calculate values.

Code reuse is facilitated through an inheritance system where subclasses can override and redefine attributes and properties

The definition layer cannot create new objects, only define classes. The sources for inputs are not defined here. Different systems can feed the same input and using undefined inputs will result in undefined results.

2.2 Instantiation Layer

The instantiation layer creates instances of the service classes, assigns properties and establishes connections between instances through anchors.

The anchoring of instances creates a directed graph: $G = (V, E)$ where the vertices V are service objects and edges E are connections to anchors. The graph may be connected or disconnected. Common special case for service models are trees and forests.

SALmon has dedicated constructs to ease instantiation and anchoring of instances. Since we are working with a large amount of service instances and relationships, there are dedicated iterator constructs to simplify the process.

2.3 Example

We illustrate our language with a simple model with service objects for a mobile network. The purpose of the model is to provide SLAs for mobile voice services in dedicated areas.

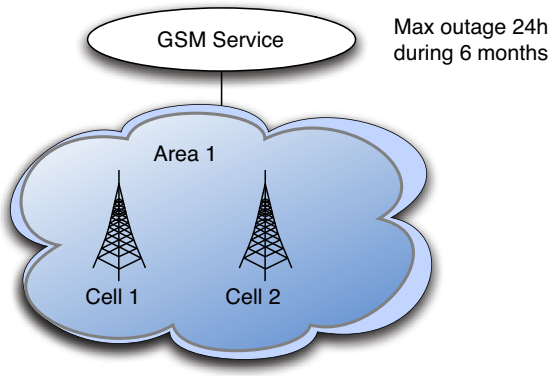

Figure 1: Sample Model

The input data source is trouble-tickets which cover both technical problems derived from alarms and customer complaints. A sketch of the sample models is in Figure 2.3.

Four classes are defined in Figure 2. The first class, `GSMCell`, defines a single input: the number of open trouble tickets associated with the cell. The boolean attribute `ok` is true only when the number of open tickets is zero.

The second class, `GSMArea` defines an anchor point for cells. It also defines another attribute `ok` which depends on the `ok` attribute of all anchored cells. When the area is instantiated it is anchored to the cells that cover an important area for an SLA customer, such as an enterprise main office.

The third class, `GSMService` aggregates areas into a general service perspective.

The fourth class `ServiceLevel` contains rules to calculate downtime and conformance to a service level agreement defined by properties. This represents the service sold to the end-customer, and downtime is calculated as the total time with open tickets.

We define two different service levels in Figure 3, `GSMServiceLevel1` and `GSMServiceLevel2`, which are subclasses to `GSMServiceLevel` where the properties have been fixed.

We are now ready to show the instantiation of a small service in Figure 4. It builds a service level `service1` which monitors a `GSMArea` called `area1` made up of two cells, `cell1` and `cell2`. This example corresponds to a customer who has bought a service level agreement for their main office with a maximum outage of 24 hours per six month period.

Service monitoring needs to be integrated with external tools such as alarm and trouble ticket systems to notify operators about problems. This is handled by allowing external systems to subscribe to attributes. This mechanism also allows us to separate the presentation in the user interface from the design of the calculation engine.

```
class GSMCell
  input openTickets
  ok = (openTickets == 0)
end

class GSMArea
  anchor* cells
  ok = allTrue cells.ok
end

class GSMService
  anchor* areas
  ok = allTrue areas.ok
end

class GSMServiceLevel
  anchor service

  property OutageMeasurementPeriod
  property MaxOutageTime

  okSLA = downtime < MaxOutageTime
  downtime = totalFalseTime service.ok@(NOW, NOW-OutageMeasurementPeriod)
end
```

Figure 2: Classes for Areas and Cells

```
def GSMServiceLevel1 = GSMServiceLevel(MaxOutageTime => 24h,
                                OutageMeasurementPeriod => 6 months)

def GSMServiceLevel2 = GSMServiceLevel(MaxOutageTime => 72h,
                                 OutageMeasurementPeriod => 6 months)
```

Figure 3: Service Levels for GSM Service

```
create GSMCell cell1
create GSMCell cell2
create GSMArea area1
create GSMServiceLevel1 service1

create area1.cells cell1
create area1.cells cell2
create service1.areas area1
```

Figure 4: Model Instantiation

3 Prototype Implementation

We have implemented an early prototype of the SALmon language runtime and interpreter using the Java™ J2SE Framework [2] and the ANTLR parser generator [3].

3.1 Classes

In the current implementation service models can be built from the basic building blocks:

Classes with inputs, anchors, properties and attributes.

Class inheritance where base class attributes can be overridden.

Property values can be fixed through a declaration similar to class inheritance.

3.2 Expressions

Inputs and attributes can both be seen as *lists* of time-stamped values. In this subsection we will refer to inputs and attributes as *time variables* viewed as lists of tuples (V, T) where V is the value and T is the time-stamp. With this view we abstract the fact that the values of inputs are available as semi-static data from external sources while attributes are calculated on demand by the runtime engine.

The expression for an attribute evaluates using the other available time variables, namely

- Inputs of the same class.
- Attributes of the same class.
- Attributes of other instances connected through an anchor.

The evaluation is performed by time-indexing. The current implementation restricts time indexes to constants or constant functions of the `NOW` parameter. Intervals of a

```
class Service
  anchor system

  // Pass on the status attribute of the instance anchored to system at the
  // time of the evaluation.
  currentStatus = system.status@NOW

  // Request the status of the last day and return the worst one.
  dailyStatus = worstOf system.status@(NOW, NOW-1day)
end
```

Figure 5: Time variable evaluation in attribute expressions.

time-variable can also be retrieved by specifying a time range. Examples of how time variables are evaluated are given in Figure 5

The need to handle lists of values arises as a consequence of two things: anchors aggregating multiple sub-service instances *and* processing time-intervals of inputs or attributes.

The list comprehension is provided through the common higher order list processing functions `map`, `fold` and `filter`:

map applies a unary function on all items in a list and returns a list of the result.

fold reduces a list into a single value by recursively applying a binary function on a list, for example when summing a list of numbers.

filter takes a list and returns only the values accepted by a predicate or unary boolean function.

The function arguments of higher order functions can be supplied either as an anonymous function or a named helper function. Helper functions depend only on their explicit arguments, and as such can be considered as purely functional. All functions are call by value. Basic operations for arithmetic, boolean logic and comparison are also implemented.

3.3 Execution engine

The runtime implementation provides basic functionality to load definition layer source files, create instances of classes, associate inputs with external data sources, connect instances through anchors, and request values of attributes of created instances.

All computations of the runtime begin with the request of an attribute value. The expression of the attribute is internally computed within a stack-based machine which computes a function after evaluating its arguments. This makes the attribute the fundamental runtime calculation unit.

The default state of the runtime is a resting state. As a request for an attribute at a certain time-stamp may depend on the calculation of other attributes, this will result in what can be considered a directed graph of calculation units where non-connected units can be calculated independently and hence in parallel.

In the prototype all data mapped to inputs reside in a database. Attaining satisfactory database performance is one of the main issues under investigation.

4 Scenarios

This section illustrates how to apply SALmon for fulfilling a few typical requirements on Service Monitoring systems.

4.1 Service Levels

Service Levels are modeled as normal classes which conform to "best practice" from ITIL and TM Forum standards and have a standard set of attributes (e.g. Figure 2).

They are associated with the operational objects through an anchor. It will often be desirable to have a high level SLA which aggregates the various SLAs into a single unit. This is easily expressed in SALmon since we use classes to model SLAs and Service Levels.

A common feature of SLAs is that they are measured on a periodic basis. The example below illustrates how outage during the current month can be calculated by summing the parameter `h` over the last month:

```
outage = sum h NOW month(NOW) 1m
```

4.2 Outage and Service Hours

Calculations need to be affected by time windows. For example, it might be okay to take a piece of equipment out of service if the customer is informed, or an SLA might only apply during office hours.

Figure 6 shows how to use a time filter to calculate outage. The time filter defines a longer service window in between 7 p.m. and 4 a.m.

4.3 What-if scenarios and Goal-Seeking

One might want to see how a certain action or change of parameter value changes the overall Service Quality. The result should show the affected service components and the resulting calculated values.

To solve this requirement, we have the ability to take a snapshot of the state of the system, making a copy-on-write version. Simulation input can then be applied to the copy in order to study the effects.

```
// Checks if outage is longer than window.
outageOk = binThreshold((timeFilter NOW) - outageDuration)

// The percentage of remaining time.
timeRemain = ((timeFilter NOW) - outageDuration) / (timeFilter NOW)

// Helper function which returns the correct service window.
timeFilter t =
     if hour(t)<4 or hour(t)>19 then
       4h
     else
       2h
```

Figure 6: Service Hours Calculation

A straight-forward example is to simulate a big network outage. The simulation input is then driven by alarms and/or work orders in the trouble-ticket system that will affect inputs in the service models.

The opposite of "what-if" is *goal-seeking*. If an operator wants to increase the measured key performance indicators of a service, how do we find the needed changes in the low level inputs? The solution to this is to translate the model into an expression that can be evaluated in a logical framework which provides goal-seeking mechanisms. Important in this application domain is that we only need good-enough proposals, not the exhaustive list or necessarily the optimal one. The goal-seeking can be done interactively with human intervention until a good-enough change is found.

4.4 Different views

It is important to be able to provide different views for different roles such as customer care, technical maintenance, and marketing.

The "core" model is a tree and the other views are variants of the tree structure like grouping cells in another way then areas and offices, for example by type and revision.

4.4.1 Modeling of service tools

The example below shows a model which mimics the OpenView Operations Service Navigator tool [4]. It is a simple, yet useful tool to model services in a tree structure. Every node has an alarm status. Furthermore, propagation rules state how severities should propagate to parents and calculation rules specify how a parent node should calculate its alarm state based on its children.

```
class SNNode
   anchor* children
   property propRule, calcRule
   property name
   ownStatus = OK
   status =  snFunc propRule ownStatus children
```

5 Related Work

One of the most important sources for service and SLA modeling is the SLA handbook from TM Forum [5]. It provides valuable insights into the problem domain but not to the actual modeling itself.

TM Forum has also defined an accompanying service model, SID, "System Information Model" [6]. SID is comparatively high level and models entities in telecom operators' processes. However, SID is being refined and moving closer to the resources by incorporating CIM [7].

The Common Information Model, CIM, has an extensive and feature-rich model including a modeling language *MOF* (Managed Object Format). Key strengths in CIM are the modeling guidelines and patterns. However, CIM faces some major challenges since the UML/XML approach tends to create unwieldy models. It is also aimed more at instrumentation than end-to-end service modeling.

Some of the major players behind CIM are now working on the "Service Modeling Language", SML [8]. SML is used to model services and systems, including their structure, constraints, policies, and best practices. Each model in SML consists of two subsets of documents; model definition documents and model instance documents. Constraints are expressed in two ways, XML schemas defines constraints on the structure and contents whereas Schematron and XPath are used to define assertions on the contents.

An interesting feature is that SML addresses the problem of service instantiation by providing XSLT discovery transforms to produce instances from different sources. Other attempts exist to specify service models as component interaction with UML collaborations [9]. This kind of service modeling serves purposes closer to the design of systems than service models for QoS metrics.

A simple and pragmatic model for a general service model is given by Garschhammer et al. [10]. This work serves as a guide for modeling and identifies several important research areas.

SLAng [11] is a language focused on defining formal SLAs in the context of server applications such as web services. It uses an XML formalism for the SLAs. SLAng identifies fundamental requirements needed in order to capture SLAs but differs from our current effort in that it "focuses primarily on SLAs, not service models in general".

When it comes to programming languages with an inherent notion of time, Benveniste et al. [12] give an overview of the synchronous languages. These languages have the concepts of variable relationships and of computing values based on the previous value

of a variable. However, they have no notion of retaining values after the computation, and they have a discretized notion of time.

Perhaps most closely related to SALmon is the notion of stream data managers [13], which take a more database oriented approach to the problem. This makes their syntax less suitable for service models, and means that they have a stricter view on time progress. However, a lot of the underlying work may be reused in the current setting.

6 Conclusion and Future Work

We have demonstrated a language for writing service models, and shown the design of a prototype calculation engine. Further, we conclude that the proposed system is a good match against real-world scenarios.

In the future, the scaling and caching that is made possible by the parallel structure of the language should be further examined. This work has been started [14], but deserves more attention.

The database layer should be updated to handle large numbers of concurrent but simple requests for input data at specific times or intervals. Since the requested data might be for a single primitive data type the overhead for each request is critical.

Finally, the user information visualization should be examined. With the ultimate goal of being able to manage large service structures, it is desirable to present relevant information in an efficient manner.

References

[1] S. Wallin and V. Leijon, "Multi-Purpose Models for QoS Monitoring," in *21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07)*, pp. 900–905, IEEE Computer Society, 2007.

[2] SUN Microsystems, "J2SE 5.0." Accessed 14th of July, 2008. http://java.sun.com/j2se/1.5.0/.

[3] T. Parr, "ANTLR parser generator." Accessed 14th of July, 2008. http://www.antlr.org/.

[4] HP, *HP OpenView Service Navigator*. URL: http://h20229.www2.hp.com/products/servnav/index.html, 2008.

[5] TM Forum, "SLA management handbook," tech. rep., TM Forum, 2004.

[6] TeleManagement Forum, "Information Framework (SID) ." GB922 , Release 7.5, 2005.

[7] Distributed Management Task Force, "CIM Specification." Version 2.15.0, 2007.

[8] W3C, "Service modeling language," May 2008. http://www.w3.org/TR/sml/.

[9] R. Sanders, H. Castejon, F. Kraemer, and R. Bræk, "Using UML 2.0 collaborations for compositional service specification," *ACM/IEEE 8th International Conference on Model Driven Engineering Languages and Systems (MoDELS)*, 2005.

[10] M. Garschhammer, R. Hauck, H. Hegering, B. Kempter, I. Radisic, H. Rolle, H. Schmidt, M. Langer, and M. Nerb, "Towards generic service management concepts a service model based approach," *Integrated Network Management Proceedings, 2001 IEEE/IFIP International Symposium on*, pp. 719–732, 2001.

[11] D. Lamanna, J. Skene, and W. Emmerich, "SLAng: A Language for Defining Service Level Agreements," *Proc. of the 9th IEEE Workshop on Future Trends in Distributed Computing Systems-FTDCS*, pp. 100–106, 2003.

[12] A. Benveniste, P. Caspi, S. Edwards, N. Halbwachs, P. Le Guernic, and R. de Simone, "The synchronous languages 12 years later," *Proceedings of the IEEE*, vol. 91, no. 1, pp. 64–83, 2003.

[13] A. Arasu, B. Babcock, S. Babu, J. Cieslewicz, M. Datar, K. Ito, R. Motwani, U. Srivastava, and J. Widom, "STREAM: The Stanford Data Stream Management System," tech. rep., Stanford, 2004.

[14] V. Leijon, P. A. Jonsson, and S. Wallin, "A declarative service modeling language with efficient caching," in *Domain Specific Languages (DSL' 09)*, 2009. **submitted**.

Tryck: Universitetstryckeriet, Luleå

ISSN: 1402-1757
ISBN 978-91-86233-34-1

Luleå 2009

www.ltu.se