ON AUTOMATED MONITORING OF SLAs

Mike Tsykin
Fujitsu Australia Limited
1230 Nepean Hwy. Cheltenham 3192 Australia

James P. Bouhana
Performance International Inc.
955 Mass Ave. PMB 345 Cambridge MA, 02139 USA

The growth of distributed systems has made the reporting and management of SLAs very difficult. However, the need for doing so became all the greater. This paper proposes an approach to the automated monitoring, problem detection, alerting and reporting of Service Levels for large, multi-platform Open Systems. It also surveys the available products and describes a sample implementation.

1. INTRODUCTION

In managing performance and capacity of large-scale systems (consisting of hundreds or more nodes), there can be many problems. However, there is a general agreement that given adequate tools and people, problems in measuring and reporting of performance metrics can be overcome. Unfortunately that is not the case where prediction of service levels is concerned. The reason is that, when the prediction of service levels is discussed, it is usually assumed that:

1. Service levels are represented by End-To-End Response Time (ETE RT)
2. Very short-term prediction is required

Both assumptions are natural. Indeed, ETE RT is the prime metric for determining service levels. As for its' prediction, the common perception is that the transaction-based ETE RT is operationally useful only if it can warn the system / network managers that they should expect immediate problems.

However, Service Level Management should be considered within the framework of Service Level Agreements (SLAs). Compliance with these is assessed over a period of time (frequently, a month) and metrics used are statistical in nature (averages and percentiles are common). Therefore, it is very useful to be able to assess how well an organization is 'tracking' against an SLA. Even more useful is to know what level of service is required in the future if a given SLA is not to be breached.

The objective of this paper is to show that it is possible, given the right tools and data, to develop a custom predictive infrastructure. To prove the point, the authors start with describing the process necessary for a successful implementation. Then, they discuss approaches to both short and long-term Service Level prediction and briefly look at the existing products that may be useful.

2. PROCESS

2.1 Overview

The process of integrated, automated service level management (including predictive Performance Monitoring and Capacity Planning) is shown in Figure 1 below. Brief descriptions of separate stages are given below. For full treatment, please refer to [1] and [2].
However, the 'Predict Future Problems' part constitutes the subject of the paper and will be dealt with in detail.

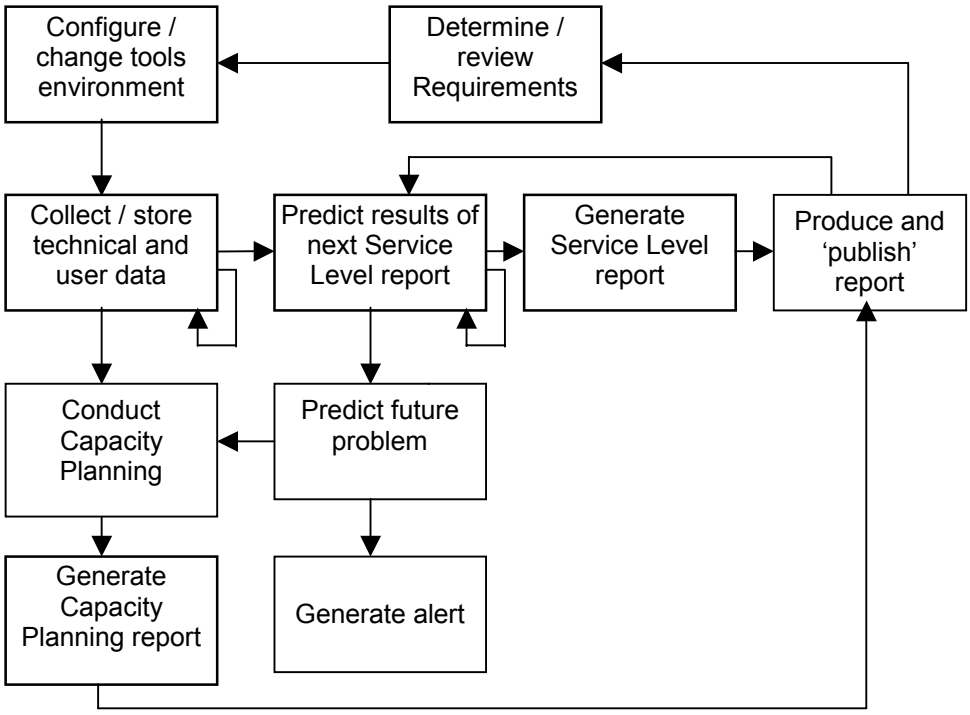

Figure 1. Automated Service Level Management process

Separate stages are briefly described below.

- Once-only
  - **Determine requirements**. This stage necessarily occurs before the Capacity Planning commences, then as required. Subjects to be addressed are:
    - Frequency of reporting. Regular, usually monthly for Service Level Reporting
    - Reported items and metrics. Determined by contractual requirements and availability of data. Usually, the following metrics are used:
      - ⇒ Availability (calculated as absence of reported problems)
      - ⇒ Help Desk statistics (incidents by class, resolution times, etc.)
      - ⇒ Response time
      - ⇒ Resource utilization (CPU, disk, memory, communications lines, etc.)
      - ⇒ User data (usage projections, major changes, etc.)
    - Format of reports. For preference Web-based, or at least electronically delivered on a CD-ROM
    - Distribution list
  - **Create / modify tools environment**. The entire process relies heavily on automation, therefore on the usage of tools. Tools environment must be always kept up-to-date. For instance, every new system placed under management must have the necessary tools installed, as a condition for the service delivery. Equally, certain events (operating system upgrades, etc.) may result in tools upgrade requirement
- Continuous
  - **Collect / store data**. Objective of this process is to collect the necessary data for Service Level Reporting and other related tasks (e.g. Capacity Planning). Varies from fully automated for all (or, at least most) platforms to manual for user forecast data. Storage of all data

is in a set of automated Performance Databases
  - **Predict results of the next Service Level Report**. This is an automated process which forecasts the probability of Service Levels being met for the next reporting cycle
  - **Predict future problem**. This process compares the predicted Service Levels with required ones, in order to forecast the service level problems BEFORE THEY OCCUR. This is an automated process
- Cyclical (monthly)
  - **Generate Service Level Report**. The process consists of two components:
    - Fully automated: generate graphs for all the necessary metrics and insert into the report template (Word or Web-based)
    - Manual: write the report
- Ad-hoc and related processes
  - **Generate Alert**. This occurs when a problem was forecast. It is an automated process
  - **Generate Capacity Planning Report**. This is a manual process, but it relies on the availability of all the necessary data from the performance databases and use of advanced tools. Thus, man-power requirements are dramatically reduced
  - **Produce and Publish Reports**. This is an automated process, consisting of:
    - Web publishing, or
    - Automated delivery of a Word document (e.g. E-mail) or 'burning' of a CD-ROM.

2.2 Role of automation

Obviously, automation of the process plays a critical role: without it, no meaningful management can be possible. However, it is useful to consider the extent of its applicability. In other words: is it sensible to attempt to automate the ENTIRE process?

The authors' experience indicates that it is not. The reason is the inherent complexity and changeability of distributed environments. Automation inevitably relies on 'stored knowledge', be it in the form of rules or 'hard-coding'. Where such knowledge does not exist, it cannot be stored. The authors are yet to be convinced of the utility of 'self-learning' algorithms.

However, the question begs to be asked: why reinvent the wheel? Why attempt to re-implement the best heuristic decision-making engine available: the human brain? Why not automate the mundane and repetitive and let the brain do what it does best: solve the problems? These are the rhetorical questions, of course and no argument is expected, nor presented. They are posed merely to set the stage for further discussion.

3. SHORT-TERM PREDICTION

3.1 General

In this case, the requirement is to predict the unpredictable – which is to say that the short-term resource consumption in a computer system under heavy load may vary widely. There are too many variables, too much non-linearity and too little data. Given a decent scale, this becomes a daunting proposition. What's more, this is a gray area, not addressed by either conventional Capacity Planning or Performance Monitoring. In other words, a rolling 'gap' of 24 hours exists between the past events addressed by Performance Monitoring and the future ones, addressed by trending, be it in Performance Monitoring or Capacity Planning. It is arguable whether this gap is of 24-hour duration or more, but it certainly is there. This is well expressed by Netuitive Inc. in refer to Figure 1 below (presented here with kind permission of Netuitive, Inc., 2001).

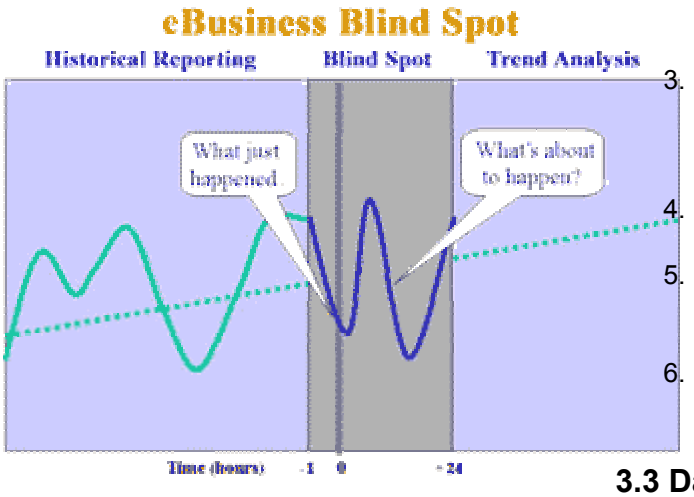

Figure 2. The Gap (with permission of Netuitive Inc., 2001)

As the result, businesses are deprived from any insight into their immediate future – the most critical period in the management of Service Levels.

It seems to be too much to ask to be able to predict performance problems. However, every small step is in the right direction: a predicted problem may be dealt with automatically, or brought to the attention of a human expert. Therefore, even a prediction of the probability of a problem occurring is useful – it may be left to the expert or a rule-based system to decide whether to address it or ignore.

It must be said that processes such as these are well enough understood for longer time intervals. Therefore, the subsequent discussion assumes very short (minutes) intervals.

3.2 Algorithm

The algorithm is deceptively simple. It is repetitive, meaning that a procedure is to be repeated regularly in time. It consists of six steps and relies upon the necessary data being available.

1. Extract the data for n preceding time intervals
2. Examine the data, looking for defined relationships. "Defined" does not necessarily mean a set formula, for instance a linear regression. This may well be a type of a curve to be fitted or a continuum of 'curves' to be examined according to some criterion of quality.
3. Select the appropriate relationship and apply a criterion of quality to decide whether the relationship is 'strong' enough for extrapolation.
4. If yes, extrapolate to m intervals (m < n)
5. Predict a problem. For instance, assess whether an extrapolated value exceeds an exception threshold
6. If yes, take action

3.3 Data

This is obviously the key to the entire problem. Given appropriate data, almost any relationship may be discovered. However, in order to be useful, the approach must work with commonly available data (for instance, derived from SAR or similar).

3.4 Relationships

The authors spent considerable time looking for functional relationships between common data items and were not able to discover combinations that were consistently useful. The level of noise appears to play a significant part. Surprisingly, no predictive relationship was discovered between a datum and its preceding values either. In other words, techniques such as smoothing did not prove to be useful.

However, time series analysis yielded some results. In retrospect, this is reasonable since it is rare for an exception to occur 'out of the blue', usually a system 'builds up' to it over several preceding intervals. However, values of a variable for each interval seem to be statistically independent from the previous intervals. That may well not be true at extremes (i.e. for heavily loaded systems), but seems to hold under the normal conditions.

3.5 Experiment

In order to test the validity of time series analysis, an experiment was conducted.

A time series of CPU busy values was captured for a lightly, but intermittently loaded Windows NT 4.0 system. The load was artificially imposed, but even so, only relatively low CPU busy numbers were achieved. Time resolution

was 1 minute. Tivoli Distributed Monitoring was used for capture and Softek Reporter (Fujitsu OCMM V4) was used for analysis.

In this case, the CPU values for 10 minutes were approximated by an exponential curve. The coefficient of correlation between the results of the approximation and the actual values was used as a criterion of quality – perhaps, not the most rigorous approach mathematically, but the one that was found to be reliable and useful over the years. The value achieved in the sample case was 0.71, which is just above the arbitrary threshold of quality (r = 0.7). Therefore, the results were extrapolated for the subsequent 4 minutes.

To validate the approach, the projected values were compared to the actual. The results are presented in Figure 4. The coefficient of correlation between the actual and projected values is high at 0.98. However, this is not meaningful given the shortness of the series (only 4 samples).

This was allowed to run at 1-minute intervals over the period of several hours. Several other exceptions were predicted, but the frequency of successful prediction was not calculated at this stage. However, the approach works sufficiently well to be investigated further

3.6 Influence of the length of the row

Authors investigated one more question: how long should be the row used to determine the parameters in the example above. In order to answer that, the chances of a value falling into a given interval (say, 90%) were calculated, depending upon the preceding N values (N=1:10) falling into the same interval. Results are presented in Figure 5. It shows that the chances are highest for one preceding sample and gradually decline as the length of the row increases. This is not really unexpected, but it is useful to have it confirmed. Of course, the results will be different for different systems.

3.7 Future research

The following needs to be done:

- The approach must be shown to work successfully (or not) on a significant number of systems with significant actual load
- Different time resolutions and different metrics must be used
- Multi-variate analysis must be performed

The appropriate project has now started. Results may be available by the time of the Conference. If so, they will be presented.

4. TREND-BASED PREDICTION

4.1. Introduction

4.1.1 OVERVIEW

The concept of Quality of Service is becoming more significant as hardware and software architecture become increasingly complex. Especially in distributed systems it often difficult for end-users to pinpoint problem areas in an end-to-end system that is not performing well at one or more intermediate points. Therefore customers of systems have established Service Level Agreements (SLA's) in order to track performance for individual systems and for sets of interconnected systems.

SLA's are often formulated as a performance objective over some specified time interval. Some examples are end-to-end response time, transaction throughput, reliability, and system or application availability. SLA's may be stated informally as a desirable target or they may be more formal in the sense of imposing either a reward or a penalty contingent upon the degree of SLA compliance.

There are several prerequisite activities and capabilities that must be enabled in order to assess ongoing SLA compliance. Some of these are listed below:

- **Measurement:** The metrics that define the SLA must be able to be measured at each point that is a component in the overall SLA criterion.
- **Storage:** The overall SLA metric (and it constituent components) need to be archived at intermediate sample points within the SLA interval so that tracking can be done.
- **Tracking:** The current value and trend of the SLA metric needs to be evaluated in order to assess the risk

of not meeting the SLA target by the end of the SLA time interval.

- **Risk Reduction:** If SLA compliance is at risk, then a proactive risk reduction plan needs to be deployed in order to increase the chance of meeting the SLA target.

The remainder of this section presents quantitative objective functions for evaluating intermediate SLA compliance.

4.1.2 TERMINOLOGY

The formulation of a SLA can in some cases appear to be complex, reflecting the detail and complexity of the situation that the SLA addresses. A relatively simple SLA metric is that of availability, and that is what will be used for illustrating the concepts in this paper. However, even the metric of availability can have several variations, such as: overall system availability (end-to-end), sub-system availability, or application availability (independent of system). For now, however, it will suffice to think of **availability** in terms of a single system and define it as 100% minus the percentage of time that the system is unavailable for production work.

The **SLA interval** is the time span over which SLA compliance is evaluated. Usually this is a long-range interval of several days to one month, although it may be quite short (several minutes) for SLA's that are specific to transient peak periods. The techniques presented in this paper require that the SLA interval be long enough to evaluate and act upon the SLA objective functions either manually or via automated tools.

We will formulate the availability **target SLA** as 95% availability over a one-month period and calculate it as the simple average of daily availability. The **time granularity** for calculating daily availability is often dictated by the periodicity with which a tool measures the underlying SLA metric. We will assume a one-minute measurement interval, although the time granularity is not critical to the basic techniques of using objective functions.

A **remainder SLA** is calculated based upon an intermediate SLA (i.e., prior to the end of the SLA interval) and it represents the required performance level necessary to achieve the target SLA by the end of the SLA interval. For example if average achieved availability to date is 92% on day 15 of a 30-day interval, then the remainder SLA is 98% average availability for the remaining 15 days.

4.2 SLA objective functions

4.2.1 PURPOSE & CALCULATION

The achieved value-to-date of a SLA metric can be calculated at any intermediate point from the start of the SLA interval. The purpose of a SLA objective function is to calculate the remainder SLA. The remainder SLA states the performance objective that must be achieved in order to comply with the target SLA by the end of the SLA interval.

Evaluating the SLA objective function for an availability SLA metric is relatively easy since both daily and overall availability are bounded (by 0% and 100%). For some other SLA metrics the objective function calculation may not be exact (see Section 4), but it usually is possible to calculate a bounded range for remainder SLA if historical SLA metrics have been archived.

For the availability example, define the following notation:

N = number of days in SLA interval (N=30)
n = number of days elapsed to date
A = target SLA for SLA interval (A=95%)
a = achieved availability to date
R = remainder availability for achieving target SLA

In terms of achieved availability to date, the target SLA formula is

$$A = (na + (N-n)R) / N$$

from which we derive the SLA objective function for remainder availability:

$$R = (NA – na) / (N-n) .$$

Note that although daily availability is bounded by 0% and 100% there is no such restriction on remainder availability. For example if average achieved availability to date is 90% 20 days into a one-month SLA interval, then the remainder function yields a value of 105%

required average availability over the remaining 10 days in order to achieve a SLA target of 95%. The 105% availability target is clearly infeasible, so the SLA objective function has revealed that the SLA target cannot be meet. In fact, the SLA objective function would reveal that the target SLA is infeasible after day 16 (for an average of 90% availability to date), in which case the remainder SLA would be 100.7%.

The purpose of calculating the SLA objective function is not to discover the infeasible dates beyond which a SLA target cannot be met. The real value lies in knowing either that SLA achievement is likely or that it is in jeopardy of being missed. These messages are conveyed by the SLA Objective Function charts and tables, which are discussed next.

4.2.2 OBJECTIVE CHARTS & TABLES

The value of remaining SLA as evaluated by the SLA Objective Function can be charted to visually depict the SLA compliance risk at any point.

Figure 6 shows the objective function for the availability example. The x-axis denotes how many days have elapsed to date and each data series represents a level of achieved availability to date (from 85% to 100%). The height of the bars denotes the average remainder availability from that day forward (to the end of the 30-day SLA interval) for exactly meeting the monthly availability target of 95%. Only the range of 50% to 100% of the objective function is charted (since anything over 100% is infeasible and a remainder availability of less than 50% is unlikely to be attempted even if it is mathematically feasible for SLA compliance).

The constant-height gray series is at the 95% attained availability level, and 95% is also the remainder availability regardless of the day-to-month progress (since 95% is also the target availability).

The bars in front of the gray series represent attained availability to date in excess of the target of 95%. Thus the SLA Objective Function evaluates to a remainder availability that is less than 95%.

The bars behind the gray series represent attained availability to date less than the target of 95%. Thus the SLA Objective Function evaluates to a remainder availability that is greater than 95% and in some cases greater than 100% (which is the infeasible region indicated by the red bars). The yellow bars indicate remainder availability that is equal to or greater than the baseline availability level. The yellow bars demarcate the challenge region in which proactive controlled SLA actions are likely in order to achieve a remainder SLA level of performance that exceeds what is customarily achieved.

A tabular form of the SLA objective function is shown in Figure 7 which can be viewed as a two-dimensional projection of the remainder SLA values onto a table. The red and yellow areas have the same meaning as before, denoting the infeasible region and the challenge region. One use of the table is to track the daily changes in achieved availability to date and resulting remainder availability. That is, at the start of each day one cell would be marked that corresponds to the number of days elapsed so far and the availability to date.

It is also useful to speak of a **baseline SLA,** which is the nominal level of performance that is usually achieved. In a well-managed system that has a realistic target SLA the baseline SLA will be higher than the target. Knowing the baseline SLA level is useful in assessing how challenging a remainder SLA is. Remainder SLA's that are in excess of the baseline will be more challenging to fulfill and may require that controlled SLA activity be initiated.

The term **controlled SLA** is the performance level that can be achieved by deferring planned activities (e.g., maintenance, load sharing) that might detract from achieving the target SLA. Using the example where availability is the SLA metric, discretionary activities such as preventive maintenance might be deferred in order to achieve the target SLA. In one project the term "managed availability" was used, which was defined as 100% minus uncontrolled unavailability. That is, managed availability is what could likely be achieved if *all* planned unavailability events were deferred.

4.3 Tracking

There are two other charts that assist in tracking the day-to-day progress of achieved availability and remainder availability. Figure 7 shows the achieved availability (vertical bars) for each day of the month and the average month-to-date availability (line series). Threshold lines are also shown for target

availability (95%) and baseline availability (98%).

Figure 8 shows the remainder availability required for 95% monthly availability for the same daily availability shown in Figure 7. Again, threshold lines are also shown for target and baseline availability.

In practice the intention is that the SLA Objective Function values shown in Figure 8 ould influence the results shown in Figure 7, especially for those days following day 15 in which the remainder availability exceeded the baseline value. Note that in this case, the example has happy ending – the daily availability metrics rise significantly after day 20 and the resulting monthly availability comes in at 95.1%.

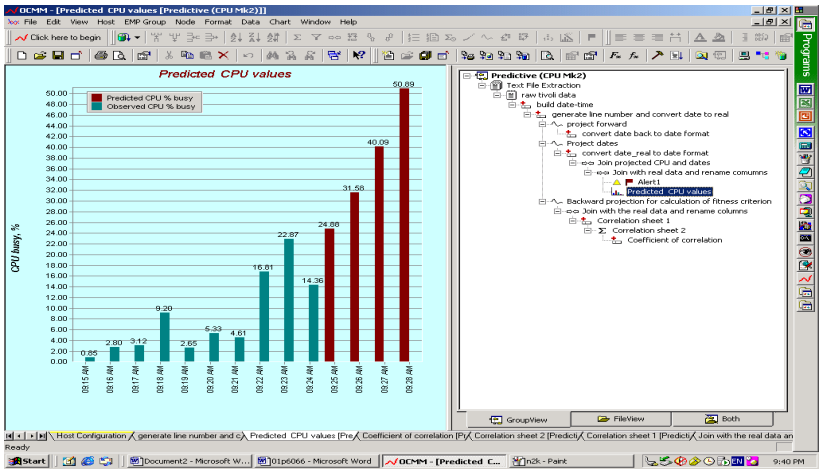

Figure 3. Experimental EMP Group

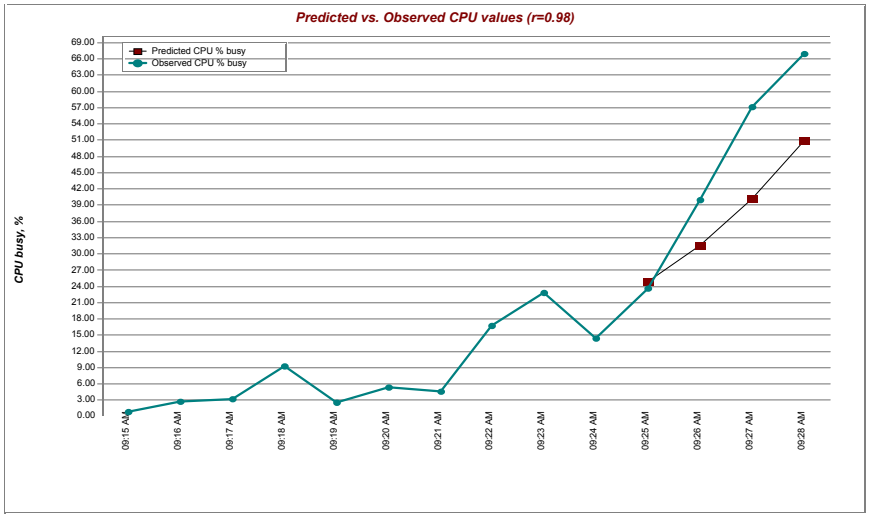

Figure 4. Validation of results

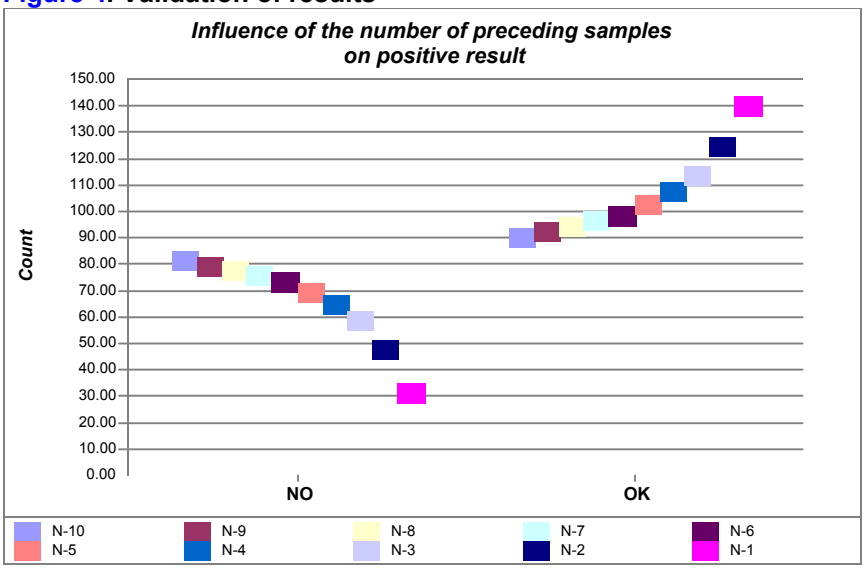

Figure 5. Influence of the length of the row

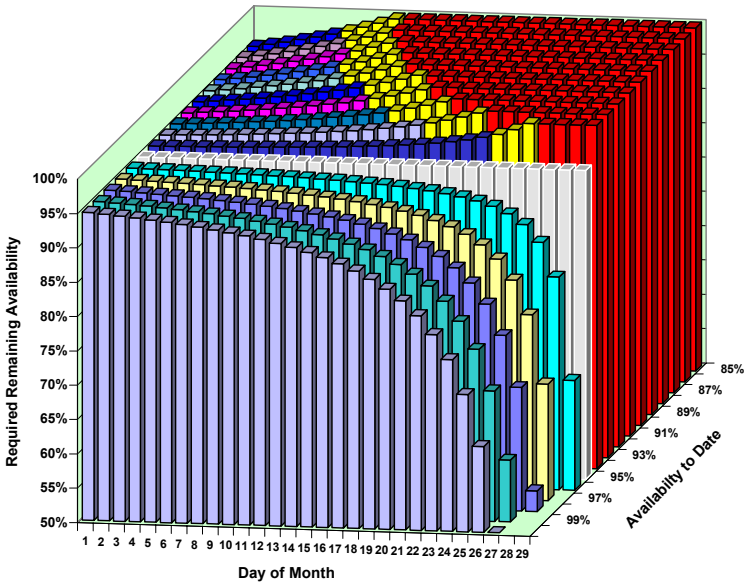

Figure 6. SLA Objective Function for Monthly Availability = 95%

Availability to Date	Days Elapsed 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	Availability to Date
8.0%	95.3%	95.7%	96.1%	96.5%	97.0%	97.5%	98.0%	98.6%	99.3%	100%	101%	102%	103%	104%	105%	106%	108%	110%	112%	115%	118%	123%	128%	135%	145%	160%	185%	235%	385%	85.0%
86.0%	95.3%	95.6%	96.0%	96.4%	96.8%	97.3%	97.7%	98.3%	98.9%	100%	100%	101%	102%	103%	104%	105%	107%	109%	111%	113%	116%	120%	125%	131%	140%	154%	176%	221%	356%	86.0%
87.0%	95.3%	95.6%	95.9%	96.2%	96.6%	97.0%	97.4%	97.9%	98.4%	99.0%	100%	100%	101%	102%	103%	104%	105%	107%	109%	111%	114%	117%	121%	127%	135%	147%	167%	207%	327%	87.0%
88.0%	95.2%	95.5%	95.8%	96.1%	96.4%	96.8%	97.1%	97.5%	98.0%	98.5%	99.1%	100%	100%	101%	102%	103%	104%	106%	107%	109%	111%	114%	118%	123%	130%	141%	158%	193%	298%	88.0%
89.0%	95.2%	95.4%	95.7%	95.9%	96.2%	96.5%	96.8%	97.2%	97.6%	98.0%	98.5%	99.0%	100%	100%	101%	102%	103%	104%	105%	107%	109%	112%	115%	119%	125%	134%	149%	179%	269%	89.0%
90.0%	95.2%	95.4%	95.6%	95.8%	96.0%	96.3%	96.5%	96.8%	97.1%	97.5%	97.9%	98.3%	98.8%	99.4%	100%	101%	102%	103%	104%	105%	107%	109%	111%	115%	120%	128%	140%	165%	240%	90.0%
91.0%	95.1%	95.3%	95.4%	95.6%	95.8%	96.0%	96.2%	96.5%	96.7%	97.0%	97.3%	97.7%	98.1%	98.5%	99.0%	100%	100%	101%	102%	103%	104%	106%	108%	111%	115%	121%	131%	151%	211%	91.0%
92.0%	95.1%	95.2%	95.3%	95.5%	95.6%	95.8%	95.9%	96.1%	96.3%	96.5%	96.7%	97.0%	97.3%	97.6%	98.0%	98.4%	98.9%	100%	100%	101%	102%	103%	105%	107%	110%	115%	122%	137%	182%	92.0%
93.0%	95.1%	95.1%	95.2%	95.3%	95.4%	95.5%	95.6%	95.7%	95.9%	96.0%	96.2%	96.3%	96.5%	96.8%	97.0%	97.3%	97.6%	98.0%	98.5%	99.0%	100%	101%	102%	103%	105%	108%	113%	123%	153%	93.0%
94.0%	95.0%	95.1%	95.1%	95.2%	95.2%	95.3%	95.3%	95.4%	95.4%	95.5%	95.6%	95.7%	95.8%	95.9%	96.0%	96.1%	96.3%	96.5%	96.7%	97.0%	97.3%	97.8%	98.3%	99.0%	100%	102%	104%	109%	124%	94.0%
95.0%	95%	95%	95%	95%	95%	95%	95%	95%	95%	95%	95%	95%	95%	95%	95%	95%	95%	95%	95%	95%	95%	95%	95%	95%	95%	95%	95%	95%	95%	95.0%
96.0%	95.0%	94.9%	94.9%	94.8%	94.8%	94.8%	94.7%	94.6%	94.6%	94.5%	94.4%	94.3%	94.2%	94.1%	94.0%	93.9%	93.7%	93.5%	93.3%	93.0%	92.7%	92.3%	91.7%	91.0%	90.0%	88.5%	86.0%	81.0%	66.0%	96.0%
97.0%	94.9%	94.9%	94.8%	94.7%	94.6%	94.5%	94.4%	94.3%	94.1%	94.0%	93.8%	93.7%	93.5%	93.3%	93.0%	92.7%	92.4%	92.0%	91.5%	91.0%	90.3%	89.5%	88.4%	87.0%	85.0%	82.0%	77.0%	67.0%	37.0%	97.0%
98.0%	94.9%	94.8%	94.7%	94.5%	94.4%	94.3%	94.1%	93.9%	93.7%	93.5%	93.3%	93.0%	92.7%	92.4%	92.0%	91.6%	91.1%	90.5%	89.8%	89.0%	88.0%	86.8%	85.1%	83.0%	80.0%	75.5%	68.0%	53.0%	8.0%	98.0%
99.0%	94.9%	94.7%	94.6%	94.4%	94.2%	94.0%	93.8%	93.5%	93.3%	93.0%	92.7%	92.3%	91.9%	91.5%	91.0%	90.4%	89.8%	89.0%	88.1%	87.0%	85.7%	84.0%	81.9%	79.0%	75.0%	69.0%	59.0%	39.0%	-21%	99.0%
100.0%	94.8%	94.6%	94.4%	94.2%	94.0%	93.8%	93.5%	93.2%	92.9%	92.5%	92.1%	91.7%	91.2%	90.6%	90.0%	89.3%	88.5%	87.5%	86.4%	85.0%	83.3%	81.3%	78.6%	75.0%	70.0%	62.5%	50.0%	25.0%	-50%	100.0%
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	

Figure 7. Tabular Form of SLA Objective Function

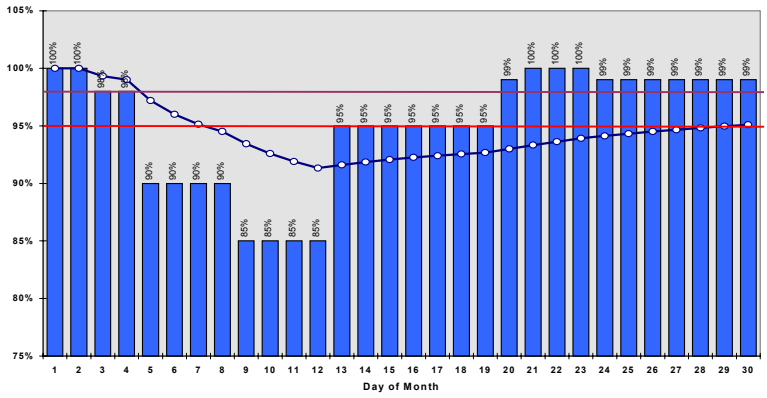

Figure 8. Daily Achieved Availability & Running Average

5. EXISTING TOOLS

Great many tools claim the predictive capability, but authors remain somewhat sceptical about many of them. However, several claims seem to be well-founded.

Softek Reporter (OCMM V4) has demonstrated its utility in near-real-time predictive monitoring. (Reference or Case Study??) Other tools, from C/C++ to Excel and SAS® may be also used, with custom code or macros. In addition, 'out-of-the-box' tools also exist.

The only tool that claims a near-real-time predictive capability is Netuitive Analyst from Netuitive[tm] Inc. The Authors were not able to assess Netuitive Analyst personally at this point, but the marketing material appears to be promising.

A number of tools claim the trend prediction capability. One good example is Neugents[tm] from Computer Associates. There are also others. Authors have no direct experience with these tools at this point in time.

In short, the availability of predictive Service Level Management tools is very limited. This reflects the stage of the development of the concept itself – mere beginning.

6. SUMMARY

This paper looked at the approaches to the prediction of the ability to meet SLA requirements, both short and long-term. An experiment was conducted to demonstrate that even the short-term prediction is feasible, given the right tools. A brief review of the tools available on the market was also presented.

Authors encourage the readers to 'have a go'. It really is not THAT hard!

7. DISCLAIMER

All brand names are Registered Trademarks and Requested Trademarks of their respective owners.

8. LITERATURE

1. M. Tsykin
On Automated Service Level Reporting
CMG99 Proceedings

2. M. Tsykin
Automated Service Level Reporting: Experience of Implementation
CMG2000 Proceedings