ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA189

AUTOMATING OPERATION STATISTICS AT PETRA-3

P. Duval, M. Lomperski, H. Ehrlichmann, D. Haupt, R. Bacher, DESY, Hamburg, Germany
J. Bobnar, Cosylab, Ljubljana, Slovenia

Abstract

The quoted machine availability of a particle accelerator over some time range is usually hand-generated by a machine coordinator, who pores over archived operations parameters and logbook entries for the time period in question. When the machine is deemed unavailable for operations, 'blame' is typically assigned to one or more machine sub-systems. With a 'perfect' representation of all possible machine states and all possible fatal alarms it is possible to calculate machine availability and assign blame automatically and thereby remove any bias and uncertainty that might creep in when a human is involved. Any system which attempts to do this must nevertheless recognize the de-facto impossibility of achieving perfection and allow for 'corrections' by a machine coordinator. Such a system for automated availability statistics was recently presented [1] and we now report on results and improvements following a half year in operation at PETRA-3 and its accelerator chain.

INTRODUCTION

A particle accelerator facility has an operations schedule (potentially 24/7) where the facility is obligated to supply users or experiments with beam. Any unanticipated deviation from this operations schedule is regarded as non-availability. Quite naturally, machine coordinators strive to present a perfect score of 100% availability at the weekly operations meeting. Traditionally a machine coordinator will scour the machine data, spreadsheets, logbook entries, etc. to obtain the *official* availability of the facility over the period in question.

We are motivated to generate this availability number automatically for several reasons. First and foremost, we can remove the human element entirely if the official availability is generated entirely automatically. Secondly, we can free up a significant amount of time spent by the coordinator calculating such a number by hand. Finally, we can monitor the availability on-line during operations.

Furthermore, the information at our fingertips will also allow us to automatically calculate the meantime between failures (MTBF) for any chosen time range.

Finally, the online information goes a long way in helping those parties responsible for a particular subsystem to identify and repair operations issues in the context of the *whole machine*.

REQUIRED SERVICES

Automatically calculating machine availability over a selected time range requires three central services. There must be a machine state server which correctly defines all possible declared states of a facility. There must be a central alarm system with a clear definition of what constitutes a fatal alarm. There must also be an archive system which keeps a history of the state and fatal-alarm information. The criteria which identify which states designate official operations (as opposed to, say, machine studies) should also be in place. For example, a fatal alarm of the RF system during *machine studies* does not constitute a failure of the accelerator facility, which tacitly suggests that the circumstances under which a fatal alarm does in fact lead to a failure of the facility must be established. Only then can we calculate a meaningful value for the meantime between failures.

Machine State Server and Periphery

The possible states of an accelerator facility are defined by the machine coordinators and the facility itself will be in *some* state at any given time. Theoretically the choice might be as simple as *running* or *not running*, but is generally more complicated. The state of a machine will be declared to the state server and the machine will be assumed to be in that state until another state declaration is made. The set of all possible machine states is completely configurable.

An additional declared state *problems* is used to identify real failures of the machine. Thus, there must be some service which officially declares this state.

In practice, the actual declaration of a machine state is governed by the following schema:

- The de-facto state is entered into a calendar well in advance (i.e. *machine studies*, *test run*, *user run*, *maintenance*, etc.)
- The real state is declared by the afore-mentioned service, which also makes use of a set of predefined rules. These rules make use of the de-facto state as well as the current machine conditions (e.g. *preparing*, *out-of-specs*, *problems*, etc.)
- The operators can manually intervene and set the declared state to whatever is deemed appropriate.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA189

Alarm System

The principal ansatz concerning availability is that "if the machine is not available then there must be at least one fatal alarm in one of its subsystems." And if we are treating *problems* as a declared state then a corollary to this ansatz is that "if we are in the *problems* state then there must be at least one fatal alarm".

It's now easy to discern a number of consistency checks which must be made to ensure a robust system. If the state is *problems* and there are no fatal alarms then this is by definition wrong and needs to be investigated and fixed. Furthermore, if there is a fatal alarm then the state must be *problems*. If this is not true, then this is likewise wrong.

Ensuring that the control system alarms reflect the true state of the machine is a painstaking procedure and is a task which generally falls on the machine coordinator to undertake and complete.

Archive System and Bean Counting

Our goal is to be able to specify any particular time range and obtain state and availability information as well as failure statistics. This is in principle easy to realize. As we are never interested in a time granularity smaller than a second or two we need only count the seconds spent in any one state and archive this number. In a similar fashion, we count the seconds where an alarm subsystem has at least one fatal alarm and archive this number. To determine the MTBF over a time range we need to count the time we are officially in an operations mode as well as the time we are in a failure state. The difference in the archived values at the end points specified by the selected time range provides us with all we need to know. As the archived data represent nothing more than counts (the cumulative number of seconds in a state) we often refer to this as *bean-counting*, a moniker which effectively represents its inherent simplicity.

The calculation of the total time spent in a state is in fact just that simple. The MTBF over a time range is likewise simply the total time spent in officially declared operations mode divided by the total number of failures (plus 1). In addition to these simple numbers we would also like to know, for instance, the duration of a failure. This includes of course the time spent in the original problems state which identifies the beginning of the failure as well as any subsequent time spent in preparing for the next operations, etc.

The Operation History Viewer shown below in Figure 1 and available in the TINE Studio suite [2] in fact makes use of such archived *bean* counts and allows the user to select any time range and examine the machine state and availability history.

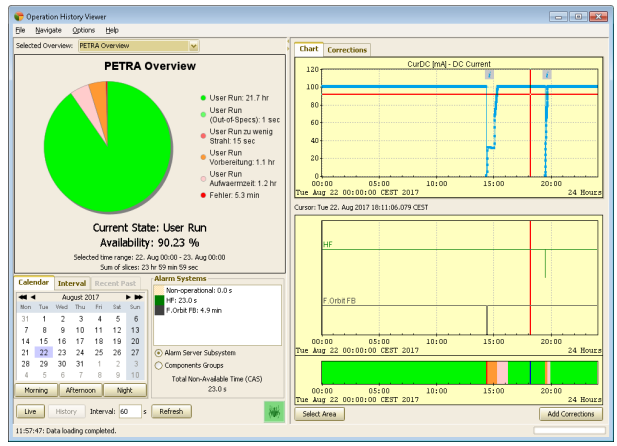

Figure 1: Operation History for the PETRA-3 showing data from August 22, 2017.

In addition to the traditional pie-chart display of the total amount of time spent in each machine state, any subsystem of the facility which was not 100 % available over the selected time is noted and presented in a trend chart where periods of non-availability are easily recognized. The fatal alarms (the *blame*) at any given time are likewise easily viewed. In the above figure, we also see that two corrections were made on this day, as indicated by the 'i' annotation icons in the machine history plot. Clicking on an annotation icon will display the nature of the correction.

RESULTS

The devil, however, is always in the details. The simplicity of the above technique is muddled by the sheer complexity of ensuring the validity of the alarm information. As long as the *problems* state accurately reflects a failure of the facility, the availability of the machine itself is deduced from the archived state counts. However, the availability of a particular subsystem will depend on the ability of the alarm system to identify fatal alarms.

Generating the meantime between failures for the machine as a whole likewise depends exclusively on the state information, whereas the MTBF for individual subsystems is once again tied to the alarm system. Furthermore, we also need to address the possibility of false starts and irrelevant failures, where the brief appearance of either an apparent operational run or of a fatal alarm is automatically withheld from the calculations.

Corrections

The trial-and-error period involved in ensuring that the availability statistics are correct is expected to be long and drawn out. To this end it is important to be able to *correct* the raw statistics displayed by the Operation History Viewer above. We do not correct the actual stored state data (the *bean* counts). Instead we provide a corrections database for both the machine states and the subsystem

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA189

availability, which is then optionally applied to the statistics displayed in the application.

A machine coordinator can use the same application to correct known false information, for instance if the state change trigger declaring *problems* was somehow missed, etc. An operator can select a region in the state bar (lower right side in Fig. 1), or double click on a colored area to select the entire colored area, and then apply a state correction to the selected region.

Correcting a state to or from *problems* brings up the issue of correcting the availability. Since the *problems* state automatically implies that the machine was not available so long as it is in this state, then there is likely to be incorrect stored alarm information as well. Namely, we perhaps missed a fatal alarm somewhere (e.g. we know from the logbook that there was unscheduled downtime even though the declared state claimed we were in a user run) or perhaps we recorded a fatal alarm when there wasn't one (e.g. we had a happy user run even though the RF system claimed a fatal alarm). If on the other hand the stored alarm information is correct then the declaration of *problems* was itself somewhere in error.

Figure 2 below shows the application of one of the corrections noted in Figure 1. The pie chart shows uncorrected data and the region of the initial correction is marked. In addition the annotation associated with the correction has been expanded (by clicking on the 'i' icon):

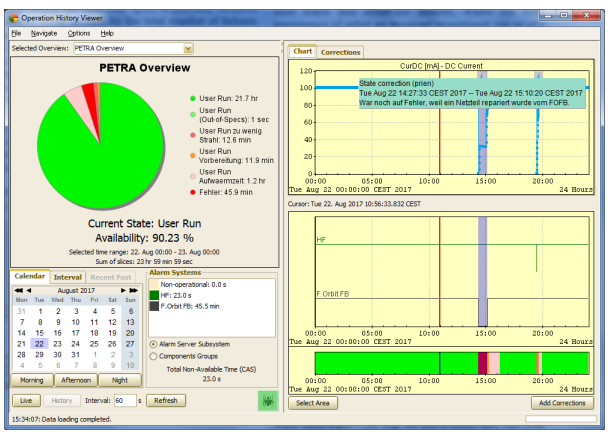

Figure 2: Applying a correction to the data shown in Figure 1.

We note in passing that the appearance of state corrections has become the exception rather than the rule. Most time ranges shown in the operation history viewer do not in fact show any state corrections.

Failure Statistics

Most recently we have added the ability to determine the meantime between failures to the operation history system. The MTBF is closely related to the machine availability and to a large extent involves making use of the same stored data that is used in calculating the availability. There are, however, other considerations at play here. In addition to the MTBF we would also like to see the duration of a failure as well as the number of failures within any selected time range. A number of possible machine states are precluded from either availability or MTFB calculations (e.g. *machine studies*, *test run*). For the remaining states however, a distinction sometimes needs to be made as to the reason the machine is in a particular state. For instance, '*preparing for operations*' is not yet '*officially operating*' and will count in the duration of a failure state if it follows on the tail of a failure state. Yet, it is not itself a failure state. Furthermore, in order to avoid false starts or irrelevant failures, a configurable window (default = 60 seconds) of applicability is applied to both operations states and the *problems* state. Namely, a declared operations state or *problems* state must be active for at least the applicability window or the state change is not allowed into the statistics.

Figure 3 below shows the failure statistics pertaining to the same time range shown in Figures 1 and 2, above.

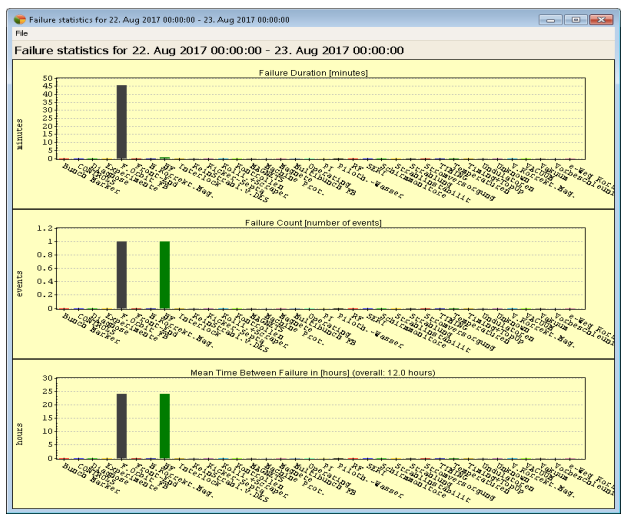

Figure 3: The Failure Statistics Breakdown for PETRA on August 22, 2017.

Currently, the failure statistics display does not incorporate corrections. This issue is currently being addressed.

CONCLUSIONS

The ability to automatically display operation and availability statistics for a facility over any particular time interval and/or monitor the same on line is worthwhile and relatively easy to implement to first order. We have shown here a technique for providing these statistics not only for the facility at large but for its component subsystems as well. In addition, the failure statistics can also be automatically determined and provided to the machine operators and subsystem managers. The techniques described here hinge on the proper identification of fatal alarms and assigning them to the reason(s) for the non-availability. The operation history depends as well on the absolute correct declaration of the proper state of the facility, including a declaration of the

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA189

failure state, *problems*. If these two points are met then the rest is simple *bean* counting and archiving.

We cannot understate how difficult it sometimes is to ensure that the identification of fatal alarms is in fact correct. This is often an iterative process spanning months if not years. Realizing this, we have added the ability to post-correct the raw data providing the automated statistics. Thus a machine coordinator can ensure that the displayed statistics for any time period is officially correct and at the same time do his part in iterating the system toward perfection.

We expect this to remain an ongoing project for some time. To be useful, this system absolutely requires an engaged machine coordinator who not only knows the systematics of machine operations but is willing to identify inconsistencies, both in the state declaration and in the setting of fatal alarms, and trace them back to their source.

The system presented here has been in use in the PETRA 3 accelerator complex since the early part of this year (2017). As might be expected, a number of issues were encountered and dealt with. Some issues involved improper state identification. Others involved applying corrections. The official state declaration has now been fine-tuned to meet the machine coordinators' expectations, and as operators have gained intuition using the Operations History Viewer, the number of reported issues concerning the latter has substantially declined.

The most recent addition to the operation history system, i.e. that of calculating and presenting the failure statistics, is currently being commissioned, but is also showing every sign of coming to fruition.

If we are persistent in our efforts, then the automated availability calculation can not only be trusted but can be monitored on-line, for example at the beginning and end of a shift. Once the automatic calculation can be trusted, then we can regard the official availability as an *honest* assessment, as we have effectively removed any *human element* in the calculation which might subconsciously exaggerate or minimize downtime (and with the side-effect that the *human* involved is free to engage in other activities).

REFERENCES

[1] P. Duval, M. Lomperski, H. Ehrlichmann, J. Bobnar, "Automated Availability Statistics", in *Proc. PCaPAC 2016*, Campinas, Brazil, October 2016, paper WEPOPRPO18.

[2] P. Duval, M. Lomperski, and J. Bobnar, "TINE Studio, Making Life Easy for Administrators, Operators and Developers", in *Proc. ICALEPCS 2015*, Melbourne, Australia, October 2015, paper WEPGF133.