AUTOMATED AVAILABILITY STATISTICS

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

Abstract

The availability of a particle accelerator or any large machine with users is not only of paramount importance but is also, at the end of the day, an oft quoted number (0 to 100%) which represents (or is taken to represent) the overall health of the facility in question. When a single number can somehow reflect on the maintenance, operation, and engineering of the machine, it is important to know how this number was obtained. In almost all cases, the officially quoted availability is generated by hand by a machine coordinator, who peruses the operation statistics over the period in question. And when humans are involved in such a calculation there might be a latent tendency to avoid the stigma of low availability. So, not only might there be scepticism at 'impossibly high' availability, but comparing quoted availability from one machine with another might turn out to be virtually meaningless.

In this paper we present a method for calculating the machine availability automatically, based on the known (and archived) machine states and the known (and archived) alarm states of the machine. Ideally this is sufficient for a completely automatic determination of the availability. This requires, however, a perfect representation of all possible machine states and a perfect representation of all possible fatal alarms (those leading to down time). As achieving perfection is ever an ongoing affair, the ability for a human to 'post-correct' the automated statistics is also described.

INTRODUCTION

A particle accelerator facility has an operations schedule (potentially 24 hours/day 7 days/week) 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 pore over 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.

REQUIRED SERVICES

The necessary ingredients to device such a system for automatically calculating machine availability over a selected time range consist of three central services. There must be a machine state server which correctly defines all possible declared states in which a facility can be in at any time. There must be a central alarm server with a clear definition of what constitutes a fatal alarm. It should also be realized that the condition of a fatal alarm is inextricably bound to the machine state, as we shall see below. There must also be a central archive server which keeps a history of the state and fatal-alarm information.

Machine State Server

The possible states of an accelerator facility are defined by the machine coordinators and the facility will be in *some* state at any given time. Theoretically the choice could be as simple as *running* or *not running*, but is generally more complicated. For completeness, the TINE [1] state server also recognizes the state *unknown* if there is no proper declared state. Otherwise 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.

There exists the question as to what to do about *problems*. Either *problems* is a valid machine state which is likewise declared or *problems* is an attribute assigned to one of the other defined valid machine states. For instance: “this is a declared *user run* but we have *problems*”. The TINE state server can handle either option, but we point out that the current configuration treats *problems* itself as a valid declared machine state. This implies that some service must determine that we cannot be in an operational state and officially declare the state *problems*.

Central Alarm Server

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”.

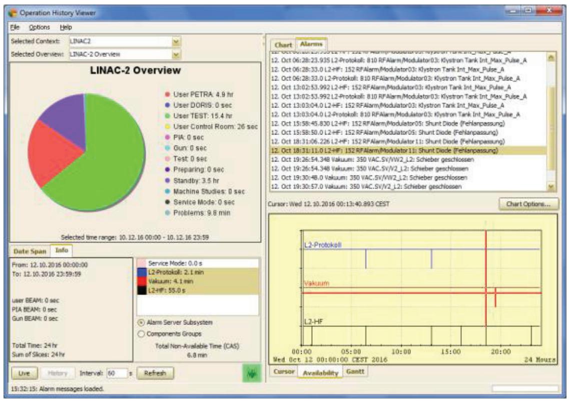
We perhaps begin to see a number of consistency checks unfolding before our eyes. If the state is *problems* and there are no fatal alarms then this is by definition wrong and needs to be investigated and fixed. Furthermore, if there is a fatal alarm then the state must be *problems*. If this is not 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.

Central Archive Server and Bean Counting

Our goal is to be able to specify any particular time range and obtain both state and availability information. This is in itself actually 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. And in a similar fashion, we must only count the seconds where an alarm subsystem has at least one fatal alarm and likewise archive this number. 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 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.



Selected Context: LINAC2
Selected Overview: LINAC-2 Overview

User PETRA: 4.9 hr
User DORIS: 0 sec
User TEST: 15.4 hr
User Control Room: 26 sec
PIA: 0 sec
Gun: 0 sec
Test: 0 sec
Preparing: 0 sec
Standby: 3.5 hr
Machine Studies: 0 sec
Service Mode: 0 sec
Problems: 9.8 min

Selected time range: 10.12.16 00:00 - 10.12.16 23:59

From: 12.10.2016 00:00:00
To: 12.10.2016 23:59:59

user BEAM: 0 sec
PIA BEAM: 0 sec
Gun BEAM: 0 sec

Total Time: 24 hr
Sum of Slices: 24 hr

Service Mode: 0.0 s
L2-Protokoll: 2.1 min
Vakuum: 4.1 min
L2-HF: 55.0 s

(●) Alarm Server Subsystem
( ) Components Groups

Total Non-Available Time (CAS)
6.8 min

Interval: 60 s

15:32:15: Alarm messages loaded.

```
12. Oct 06:28:23.935 L2-Protokoll: 810 RFAlarm/Modulator03: Klystron Tank Int_Max_Pulse_A
12. Oct 06:28:33.0 L2-HF: 152 RFAlarm/Modulator03: Klystron Tank Int_Max_Pulse_A
12. Oct 06:28:33.0 L2-Protokoll: 810 RFAlarm/Modulator03: Klystron Tank Int_Max_Pulse_A
12. Oct 13:02:53.992 L2-HF: 152 RFAlarm/Modulator03: Klystron Tank Int_Max_Pulse_A
12. Oct 13:02:53.992 L2-Protokoll: 810 RFAlarm/Modulator03: Klystron Tank Int_Max_Pulse_A
12. Oct 13:03:04.0 L2-HF: 152 RFAlarm/Modulator03: Klystron Tank Int_Max_Pulse_A
12. Oct 13:03:04.0 L2-Protokoll: 810 RFAlarm/Modulator03: Klystron Tank Int_Max_Pulse_A
12. Oct 15:58:45.830 L2-HF: 152 RFAlarm/Modulator05: Shunt Diode (Fehlanpassung)
12. Oct 15:58:50.0 L2-HF: 152 RFAlarm/Modulator05: Shunt Diode (Fehlanpassung)
12. Oct 18:31:06.226 L2-HF: 152 RFAlarm/Modulator 11: Shunt Diode (Fehlanpassung)
12. Oct 18:31:11.0 L2-HF: 152 RFAlarm/Modulator 11: Shunt Diode (Fehlanpassung)
12. Oct 19:26:54.348 Vakuum: 350 VAC.SV/VW2_L2: Schieber geschlossen
12. Oct 19:26:54.348 Vakuum: 350 VAC.SV/V2_L2: Schieber geschlossen
12. Oct 19:30:48.0 Vakuum: 350 VAC.SV/VW2_L2: Schieber geschlossen
12. Oct 19:30:57.0 Vakuum: 350 VAC.SV/V2_L2: Schieber geschlossen
```

Cursor: Wed 12.10.2016 00:13:40.893 CEST

Figure 1: Operation History for the PETRA-3 linac showing data from Oct. 12, 2016.

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.

RESULTS

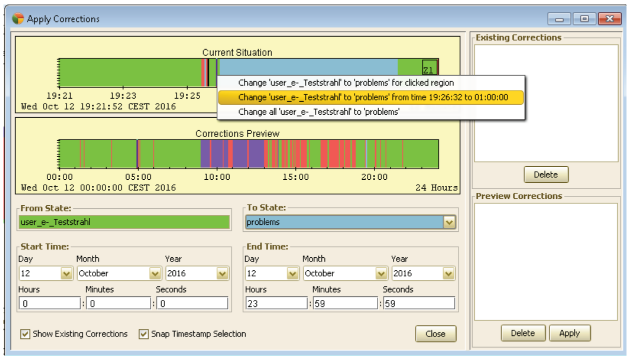
The simplicity of the above technique is alas muddled by the sheer complexity of ensuring the validity of the alarm information. For example, a fatal alarm appearing in the RF system even though there is a perfectly good user run will destroy the availability calculation. And of course the state declaration must correspond with the true state of the machine. Any such inconsistencies will be spotted by the machine coordinator calculating the availability statistics and if they are religiously forwarded to those persons responsible for generating the alarms or declaring states, then eventually the availability results generated automatically by the above techniques will coincide with those calculated by hand by the machine coordinator.

Corrections

The trial-and-error period involved in ensuring that the automatically generated availability statistics are correct is expected to be long and drawn out. In fact any future upgrade to the accelerator facility is likely to bring new use cases and more trial and error with it.

During this trial-and-error process it is more important than ever to be able to *correct* the raw statistics displayed by the Operation History Viewer above. We do not correct the actual stored state data (i.e. the *bean* counts). Instead we provide a corrections database for both the machine states and the subsystem 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. A right-click over the states history display (the pie chart in Figure 1) will offer the option to examine or apply corrections in the form of a new popup window, as shown in Figure 2.



Wed Oct 12 19:21:52 CEST 2016

Change 'user_e-_Teststrahl' to 'problems' for clicked region
Change 'user_e-_Teststrahl' to 'problems' from time 19:26:32 to 01:00:00
Change all 'user_e-_Teststrahl' to 'problems'

Wed Oct 12 00:00:00 CEST 2016

From State: user_e-_Teststrahl
To State: problems

Start Time: Day 12 Month October Year 2016 Hours 0 : Minutes 0 : Seconds 0
End Time: Day 12 Month October Year 2016 Hours 23 : Minutes 59 : Seconds 59

[✓] Show Existing Corrections [✓] Snap Timestamp Selection

Figure 2: The state correction popup window.

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

ISBN 978-3-95450-189-2

declaration of *problems* was itself somewhere in error. In fact, a close scrutiny of Figure 1 will reveal that over the 24 hour period that was Oct. 12, 2016 the PETRA3 linac was in the *problems* state for 9.8 minutes whereas there were only 6.8 minutes of *non-availability*. This is clearly inconsistent and needs to be both corrected and investigated.

Apropos correcting availability, there is a second correction popup window designed to correct the availability counts and either assign blame to or remove blame from some subsystem, and this brings up two points. Point one is that the operation history application must itself check the time range of any *problems* state (with applied corrections) against that of the overall non-availability (with applied corrections) and warn the application user if these are not synchronized. Point two is that if the machine coordinator sees fit to apply any correction whatsoever he should inform the responsible parties that he needed to do so and request that steps be taken to fix the problem at the source. This latter turns out to be very important feedback for server programmers who might otherwise not have the overall picture of operations statistics in mind.

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. The devil as usual is in the details. The technique described here hinges on the proper identification of fatal alarms (defined as those leading to or responsible for non-scheduled downtime) 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. 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. With the ability to post-correct the information displayed in the TINE Studio Operations History Viewer, it should not require much extra effort for a machine coordinator who is already calculating operations and availability statistics to see this through. To expedite the iterative improvement necessary a notification system will be added to the corrections dialog, so that the persons responsible for alarms and state declarations can be informed of those inconsistencies which led to a correction.

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] TINE website, `http://tine.desy.de`

[2] P. Duval, M. Lomperski, and J. Bobnar, "TINE Studio, Making Life Easy for Administrators, Operators and Developers", in *Proc. ICALEPCS'15*, Melbourne, Australia, Oct. 2015, paper WEPGF133.