MACHINE PROTECTION DIAGNOSTICS ON A RULE BASED SYSTEM

Timmy Lensch[a], Yuri Nechaev*[a], Winfried Schütte[a], Victor Soloviev†[a], Marcus Walla‡[a], and Matthias Werner[a]

[a]Deutsches Elektronen-Synchrotron DESY, Hamburg, Germany

Abstract

Since commissioning the high-brilliance, 3rd-generation light source, PETRA-3 in 2009 [1] the accelerator operation has become routine. To guard the machine against damage a Machine Protection System (MPS) was built [2]. Alarms and beam information are collected by the MPS and can be used to analyse beam losses and dumps. The MPS triggers a visual diagnostic software [3], which is used to analyse the hardware dump cause. The diagnostic software is based on a Domain Specific Language (DSL) architecture. The MPS diagnostic application is designed with a server-client architecture and written in Java. The communication protocol is based on TINE [4]. We characterise the data flow of the alarms and the DSL specification and describe the composition from the delivered structure to a single, human understandable message.

MOTIVATION

For an ordinary operator it is often difficult to identify the cause of a beam loss in the MPS Client application, see Fig. 1. Numerous small coloured squares representing mixed devices are horizontally sorted according to its position in the storage ring. Unrelated conditions can easily confuse the operator as to the cause of a beam loss event. Expert analysis is often required. Thus a rule-based logic which effectively automates the expert's knowledge was implemented in the MPS Diagnostic application. It deduces from the collection of distributed devices across the entire PETRA storage ring an easy and comprehensible message of the beam loss cause. As a communication protocol for transferring information between the dedicated hosts the TINE protocol [5] and its dedicated TINE Event Archive service [6] (EAS) is utilised, where raw event data is stored. All involved applications are written in Java.

INTRODUCTION

The MPS Diagnostic software is a collaboration of DESY groups for instrumentation (MDI) and controls (MCS). The purpose of the MPS Diagnostic software is to allow a human operator to determine the cause of a beam loss with a useful and comprehensible tool. A short introduction to the hardware and server follows.

The MPS hardware is a fast technical interlock system which is distributed over the whole PETRA storage ring. If

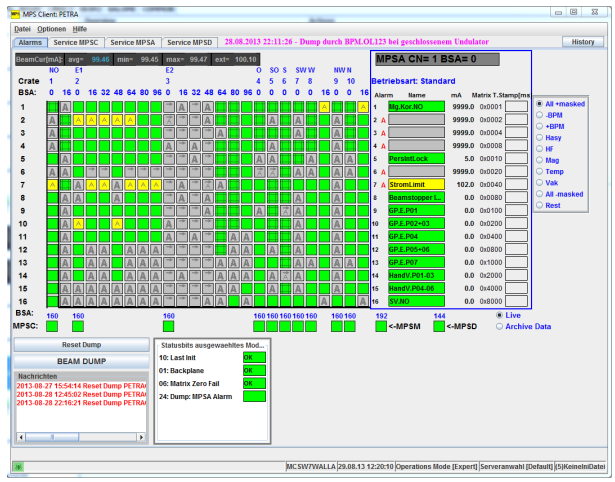

Figure 1: MPS Client application with the MPS alarm state on the left side. On the right side you see a description of the involved modules. At the bottom are buttons to manually dump the beam.

an interlock alarm of certain devices such as Beam Position Monitors (BPM), vacuum shutters, radio frequency (RF) system or magnet power supplies indicates an unsafe condition the MPS generates within a few hundred nanoseconds a dump trigger which is then sent to the RF system. The RF then shuts down the beam during several orbits in approximate 400 μs. The beam is then dumped.

The protection part of the MPS hardware is independent of its diagnostic feature and furthermore is also independent of the function of the field bus and server connection.

Additionally, the MPS measures the effective beam current to detect fast beam losses. Due to the MPS hardware synchronisation on the timescale of sub-microseconds it can determine if a beam loss forced an alarm (e.g. BPM) or if the alarm occurred first and forced the MPS to dump the beam. A typical situation is beam loss due to an RF failure. This is immediately detected by a BPM because of orbit deflection. The BPM then triggers an alarm to the MPS which dumps the beam. This dump is only a consequence of the prior beam loss.

To allow diagnostic investigation on a beam-loss event all data provided by the MPS hardware are transmitted to the MPS server as a snapshot and is provided for post-mortem analysis [7]. The MPS server fills the internal buffer with the new event and stores the data in the event archive. As a unique identifier for the event archive the time stamp of the event is taken.

* Retired 2013.
† Retired 2012.
‡ marcus.walla[at]desy.de

ISBN 978-3-95450-139-7

The devices can be distinguished by temperature, BPM, magnet power supplies, RF, vacuum (getter pumps (GP) and vacuum shutters (SV)), undulator, or personal interlock (PI). The raw data array carries information about the alarm states of a specific MPS alarm device, where aggregated alarms[1] can also be handled.

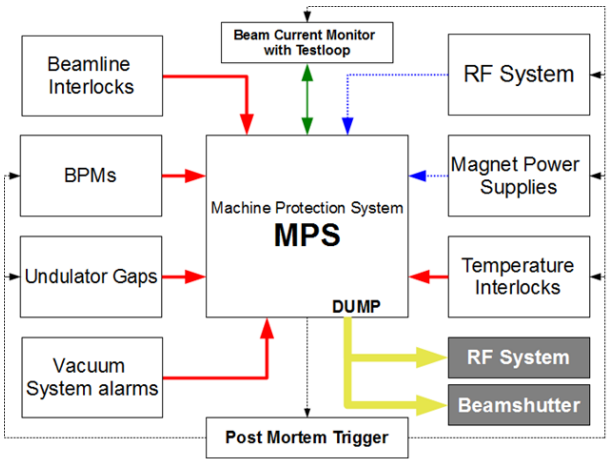

Figure 2: Overview of the MPS system. The red lines indicate alarm signals which causes a beam dump. The blue lines are only for analysing beam losses.

Figure 3 presents the communication overview for all relevant elements.

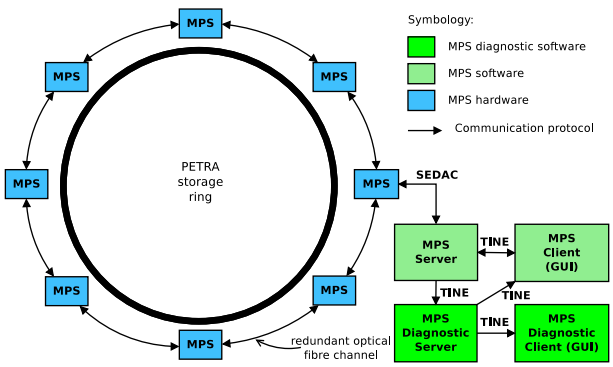

Figure 3: Overview of the communication. SErial DAta aCquisition (SEDAC) is a proprietary field bus for communication between devices invented by DESY in 1976.

ANALYSIS AND DESIGN

As a basis for further discussion, we consider that the accelerator has already lost the beam as a precondition and the corresponding buffer in the MPS server and event archive is filled with the raw event data array. Under this condition the MPS server delivers the data for post-mortem analysis.

[1] An example of an aggregated alarm is the temperature interlock.

RAW DATA STRUCTURE

The delivered raw data structure from the MPS server consists of different items. For diagnostic purposes only a subset is relevant: the collection of all involved alarm inputs, the detected time stamp when the beam loss was detected, and the time correlation when the beam loss event occured and the causative alarms. Moreover a flag informs the diagnostic implementation about whether the MPS dumped the beam which caused the beam loss or if the alarm are just a consequence of the beam loss. The MPS alarms from the raw data structure corresponds to an array of meta informations published by the MPS server. This published array represents the corresponding group for each alarm. This associative structure is used to identify which MPS alarm belongs to which group and adds the alarm to the group. The following summary shows the important group elements:

- BPMs
- Temperatures and water flow sensors (DDW)
- Fast beam shutter (SSK), undulator gap, undulator failure, and vacuum interlock
- Vacuum GP, SV, and hand valve
- Corrector, chopper, and thyristor magnets
- Personal interlock, manual dump key, software dump, screen monitor (SM), and wiggler gap
- RF high voltage, thyristor, and blanking produce a failure
- Fast kicker magnets
- Beam current limit

Additionally, each MPS alarm implies a specific alarm code. Hence a subset of all relevant alarms reduces the number of elements to a minimum for an event. The following, very general list classifies all possible events [8]:

- alarm was just before beam loss (i.e. RF failure); beam was not lost due to MPS dump, but BPM alarm can be a consequence of a beam loss
- dump condition was applied and is an initial alarm
- dump was cause by simultaneously triggered alarms; MPS evaluates the chronologic order
- beam loss without alarm

Only alarms matching one of these conditions are evaluated by the diagnostic application.

ISBN 978-3-95450-139-7

MPS EVENT

To create an MPS event the following pseudo code is executed:

```
create MPSGroup collection
loop over all EMPSGroup

  create MPS alarm collection
  loop over all MPS alarm elements

    when MPS alarm is in EMPSGroup & alarm != 0
      add MPS alarm element to collection

  when MPS alarm collection has elements
    create MPSGroup(group, MPS alarm collection)
    add MPSGroup to group collection

create MPS event
add group collection
add evaluated beam parameters
```

Figure 4 shows the class diagram of the `MPSEvent` with the related classes `MPSGroup` and `MPSAEingang`.

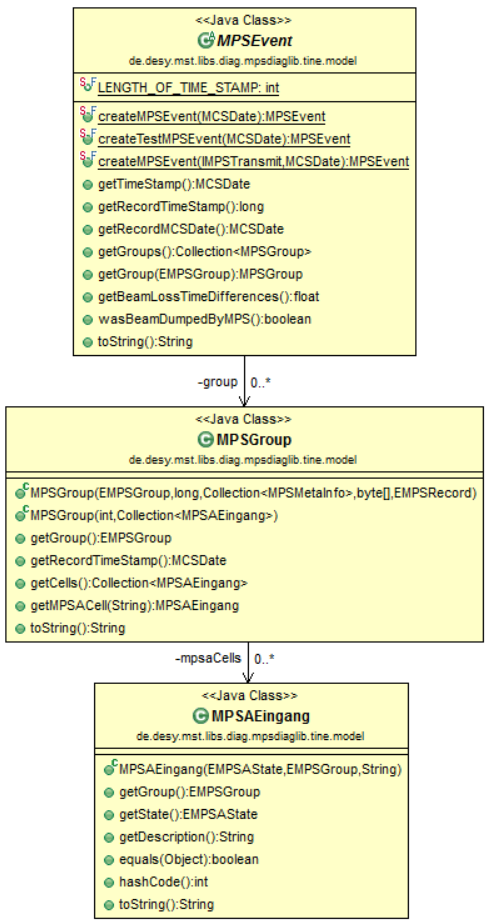

Figure 4: Class diagram of `MPSEvent` and related classes.

RULE BASED LAYER

The expert's knowledge is specified with the help of a domain specific language (DSL) architecture on a rule based system [9]. This style of development allows concentrating about **what** the application accomplishes instead of **how** it is accomplished. It is a declarative language specification. An internal DSL can be quickly implemented and is sufficient. The rules will not change very often. Therefore an explicit separation between the fast Java program and the rules is not required. One disadvantage is that the DSL structure has syntax fragments separated from the general purpose language.

IMPLEMENTATION

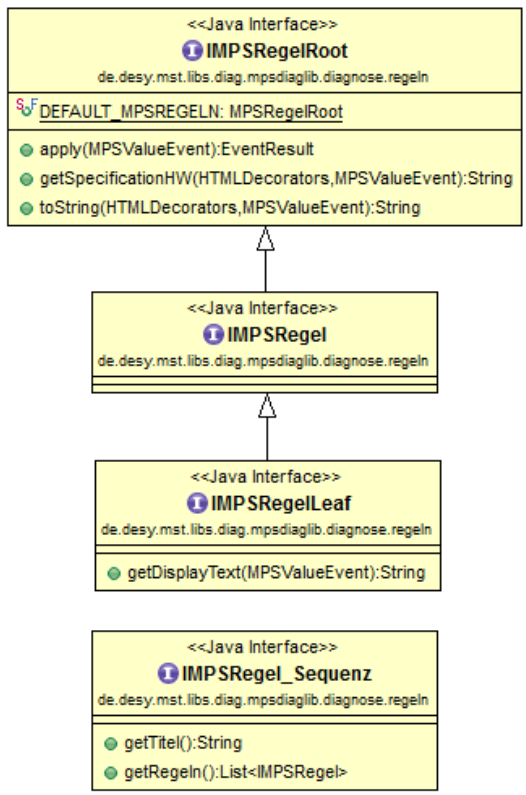

Figure 5: Class diagram of the rule based system.

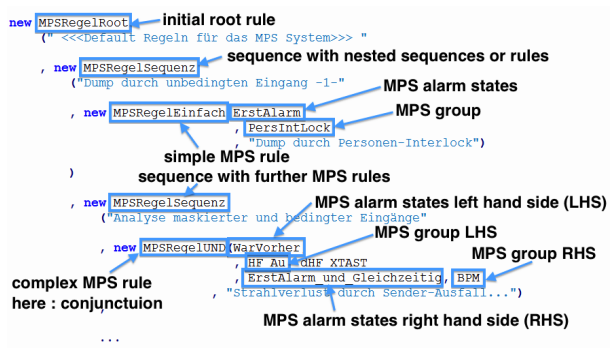

Figure 6: Extraction of the implemented MPS rules.

ISBN 978-3-95450-139-7

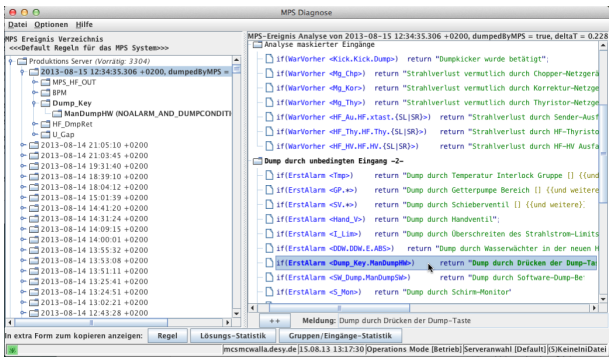

Figure 7: MPS Diagnostic application displaying a manual beam dump on August, 15th in 2013.

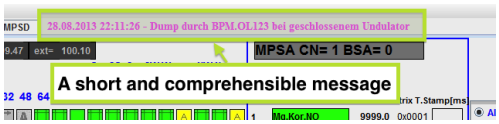

Figure 8: Cause of a beam loss in the MPS Client application.

The rule system consist of an initial root element which is constructed with a list of MPS rules. An element in the rule list can be a sequence with further rules or a leaf rule. This data structure reflects the logic of the leaf nodes.

The sequence must be verified in its defined order. Figure 5 represents a class diagram of the data structure. Figure 6 shows an extract of a concrete definition of the MPS rules. Constructing the compressive and human readable message is then trivial. Figure 7 shows a snapshot of the application.

A big advantage of using the abstract DSL layer is that modifying the order of sequences, adding, deleting or extending the rule system is very easy and comfortable. The proper logic is explicit and not hidden. A disadvantage is that only a Java developer has the knowledge to modify the DSL specification. But that can be improved for each developer after a short training period. Since the rule based system returns the cause in a short message it can be read via the TINE interface and displayed in the MPS client application, see Fig. 8.

TEST SCENARIOS

Writing source code and test cases is going hand in hand [10]. The unit test used gives a developer the opportunity to verify each class in a test case and provides the safety that the code segment executes as expected. As a low-level test we decided to use the JUnit test framework [11] intensively, and an easy-to-use environment for writing test scenarios at the method level.

A big advantage of using the DSL architecture and the rule-based system is its testability. It allows one to prepare predefined `MPSEvents` and forward them to the rule system. The outcome of the test must be defined with the predefined event otherwise the code has a failure.

STATISTICS

Besides preparing an event-based analysis, the MPS diagnostic application can also examine all events and count their occurrences over all defined rules. By investigation of all collected events from the 8th February 2011 until the 23rd August 2013 the total number is 3,336. Furthermore 342 events are unresolved (10.25 %) [12]. The PETRA storage ring operation provides service periods. By subtracting the service time from all unresolved events it yields to a total number of 85, so 2.7 %, which is for the initial iteration step a good result.

CONCLUSION

- The number of detected events which can be resolved is 97 % which is for a first iteration a very good result.
- An operator is now able to see a short and comprehensive message of the last beam loss in the MPS client application and can provide a screenshot in the online logbook.
- The workload of the experts is released.
- The work of the operator is strongly improved.

REFERENCES

[1] K. Balewski et al., “Commissoning Results of Beam Diagnostics for the PETRA III Light Source”, DIPAC09, Basel, Switzerland, 2009

[2] T.-J. Lensch, M. Werner, “Commissioning Results and Improvements of the Machine Protection System for PETRA III”, BIW10, New Mexico, US, 2010

[3] Y. Nechaev, M. Walla, private communication

[4] Piotr Bartkiewicz, Philip Duval, Steve Herb, Honggong Wu, “The TINE Control System, Overview and Status”, FOPA03, Tennessee, US, 2007

[5] `http://tine.desy.de`

[6] Jaka Bobnar, Igor Kriznar, Reinhard Bacher, Philip Duval, Mark Lomperski, “New TINE JAVA General Purpose Diagnostic Applications”, TUZ01, Ljubljana, Slovenia, 2008

[7] T.-J. Lensch, W. Schütte, M. Walla, private communication

[8] T.-J. Lensch, M. Werner, “Cause Identification of Beam Losses in Petra III by Time Correlation of Alarms”, MOPD90, Hamburg, Germany, 2011

[9] M. Fowler, “Domain Specific Languages”, Addison Wesley, 2011

[10] K. Beck, “Test-driven development: an example”, Addison Wesley, 2003

[11] `http://junit.org`

[12] T.-J. Lensch, M. Walla, private communication

ISBN 978-3-95450-139-7