UPGRADE OF ALARM SYSTEM AT PF-AR ACCELERATOR CONTROL

Takuya Nakamura*[A)], Kazuro Furukawa[B)], Takashi Obina[B)], Kay Kasemir[C)]
[A)]Mitsubishi Electric System and Service Co.,Ltd.,
2-8-8 Umezono, Tsukuba, Ibaraki, 305-0045
[B)]High Energy Accelerator Research Organization (KEK),
1-1 Oho, Tsukuba, Ibaraki, 305-0801
[C)]Oak Ridge National Laboratory (ORNL),
Oak Ridge, TN 37831, U.S.A.

Abstract

Alarm surveillance program at KEKB and PF-AR, was built by SAD script, and use for accelerator operation in a long term. SAD(Strategic Accelerator Design) is a computer program complex for accelerator design. It is used for accelerator design, simulations and operator interface program. However, this alarm surveillance program has often suspended when many alarm notifications occurred. Recently, a collection of software and control tool called CSS (Control System Studio) to the EPICS community is created, for which the development and implementation is progressing in several laboratories. CSS includes an alarm surveillance system, and we tried to implement it to solve a problem of former alarm surveillance system for PF-AR. In this paper, we will report the CSS alarm system for PF-AR.

PF-AR 加速器制御におけるアラームシステムの更新

1． はじめに

KEKB、及び PF-AR 加速器では、加速器の制御ツールとして Python や SAD(Strategic Accelerator Design) といったスクリプト言語を主に利用している。これらのスクリプト言語を用いて、サーバー計算機上で実行されるプログラムや、運転用の OPI(Operator Interface) パネルを作成している。また運転用の OPI パネルとしては、MEDM というツールも併せて利用されている。近年、EPICS コミュニティ [1] に CSS(Control System Studio) [2] [3] と呼ばれる制御ツールを集めたソフトウェアが登場し、様々な研究所での開発や利用が進んできている。CSS には加速器の制御で利用されるツールが色々と用意されており、GUI ツール、データ収集ツール、アラームシステムなどのツールが用意されている。CSS は Java Eclipse platform の環境で動作し、Linux、Windows、Macintosh などの OS で動作するため、様々なシステムへの導入が容易である。また開発環境と実行環境が同じ環境であり、それぞれの連携が容易である事などから、各研究所での利用が広まっている。

KEKB と PF-AR のアラームシステムは、SAD スクリプトによって加速器の状態を常時監視する OPI パネルが作成され、運用されてきた。しかし、アラームの発生状況によっては OPI パネルが停止するという不具合がしばしば発生しており、新しいアラームシステムへの移行が検討されていた。そこで今回、PF-AR のアラームシステムの更新案として CSS のアラームシステム [4] が検討され、動作評価試験、及び新システムへの移行が行われた。ここでは、今回導入された CSS アラームシステムについて報告する。

* nakataku@post.kek.jp

2． 従来のアラームシステム

2.1 概要

KEKB と PF-AR の運転で使用していたアラームシステムは、SAD によって作成されたプログラムが利用されていた (図 1)。このプログラムは、PF-AR では約 1000 点、KEKB では約 25000 点のレコードに接続して値をモニターしている。EPICS 標準の機能として、各レコードはそれぞれが持つ値についてアラーム状態を判断する事ができる。例えばアナログ値を扱う Analog Input レコードでは、アラームを判定する上限と下限のしきい値の設定があり、レコードの値がしきい値を超えていないか判断する。アラームの情報はレコードの値と共に、モニターしているプログラムに通知される。SAD プログラムがモニターしているレコードのアラームの判断については、各レコードが持つ EPICS 標準のアラーム機能は利用しておらず、独自の判定式を用いている。この独自のアラーム判定式を使う事で、一つのレコードに対して複数のアラーム状態を定義する事ができる。例えば機器のリターンコードの値を数字で扱うレコードの場合、数値の上限や下限の超過ではなく、ある特定の数値ごとにアラームかどうかを判断する事ができる。

2.2 問題点

KEKB、PF-AR の運転において、長らく SAD によるアラームプログラムが運用されていたが、アラームの発生状況によってしばしば停止するという問題が起きていた。それは加速器のメンテナンス中に多数のレコードがアラーム状態になった時や、IOC の再起動によるレコードとのコネクションが切断した時など、アラーム状況が多数変化するときに発生していた。

また、このアラームプログラムが SAD によって作成されている事による問題点もある。SAD は KEK で開発された加速器設計の為の計算プログラム言語であり、

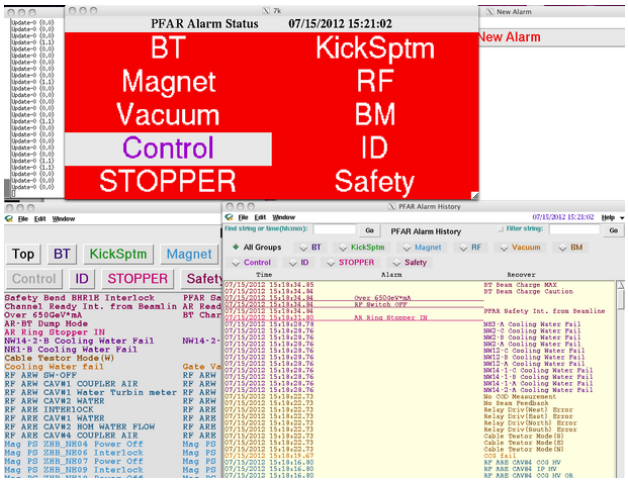

図 1: SAD アラームプログラム

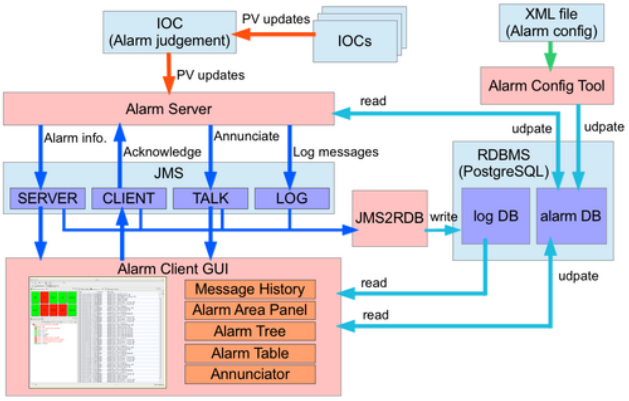

図 2: CSS アラームシステム構成

広く一般に利用されている言語ではないため、他のプログラム言語ほど情報や文書が整備されていない。そのため SAD に習熟した者でないとアラームプログラムのメンテナンスは難しく、保守性が悪くなってしまう状況となっている。これらの問題を解決するため、新しいアラームシステムへの更新が検討された。

3. 新しいアラームシステム

近年、EPICS コミュニティに CSS と呼ばれる制御ツールを集めたソフトウェアが登場し、各研究所での開発や利用が進んでいる。CSS の制御ツール群にはアラームシステムも含まれており、今回 PF-AR 加速器での動作評価試験、及び新システムへの移行が行われた。

3.1 ソフトウェア構成

CSS のアラームシステムは、単一のプログラムではなく、いくつかのソフトウェアを組み合わせる事で構成されている (図 2)。

CSS ではアラームシステムの根幹をなすプログラムが用意されており、Alarm Server(アラームサーバー)、Alarm Client(アラームクライアント)、Alarm Config Tool(アラーム設定ツール)、JMS2RDB(アラーム履歴保存) といったプログラムがある。これらのプログラムについては、次章にて詳細を説明する。

CSS アラームシステムを構成する各プログラムは、JMS(Java Message Service) を利用してお互いに情報を通信しており、そのソフトウェアとして Apache ActiveMQ [5] を使用している。またアラームの設定や現在値の保存には RDBMS(Relational Database Management System) を利用している。CSS アラームシステムがサポートしている RDBMS には、Oracle、PostgreSQL、MySQL があり、我々のアラームシステムでは PostgreSQL を採用した。

3.2 JMS (Java Message Service)

JMS とは、Java プログラムがネットワーク上でのメッセージ通信を行うサービスである。CSS のプログラムは出版/購読型のモデルで非同期のメッセージ通信を行い、Alarm Server、Alarm Client、JMS2RDB が利用している。

3.3 PostgreSQL

アラームシステムの情報管理には RDBMS が用いられており、我々は運用経験のある PostgreSQL を選択した。PostgreSQL ではアラーム監視の設定情報やアラームの現在値、またアラームの履歴などの情報が管理されている。当初 PostgreSQL 9.0.4 で運用を開始したが、その後サーバーの移行やアップデートが行われ、現在は PostgreSQL 9.1.3 での運用となっている。

3.4 アラーム判定 IOC

SAD のアラームプログラムでは、独自のアラーム判定式を用いてレコードの値を判定しており、各レコードが持つ EPICS 標準のアラーム情報は使用していない。一方、CSS アラームシステムの Alarm Server は、EPICS 標準のアラーム情報を元に動作する仕組みとなっている。Alarm Server がこれまで監視していたレコードに直接接続した場合、これまで利用していた柔軟性のあるアラーム判定を実現する事ができない。そこで、監視対象のレコードを持つ IOC と Alarm Server との間に、アラーム判定を行う専用の IOC を新たに立ち上げる事とした。

アラーム判定 IOC にはアラーム判定の計算を行う CALC レコードを用意し、その CALC レコードの入力フィールドの一つに、従来モニターしていたレコードを接続する。CALC レコードは与えられた入力を元にアラームの判定を行い、正常状態の時は 0、アラーム状態の時に 1 という計算結果となるよう調整し、また、レコードのアラーム設定についても、計算結果が 0 の時には正常、1 の時にアラーム状態となるよう整備する。

Alarm Server は、このアラーム判定 IOC 上の CALC レコードに接続する事により、柔軟性のあるアラーム判定を実現する事が可能となる。

4. アラームシステムのソフトウェア

4.1 Alarm Server

Alarm Server は、レコードの EPICS 標準のアラーム情報を元にアラームの監視を行う。監視するレコードの名前やアラーム発生時の動作などを規定した設定情報

は、RDBMS を用いて管理している。また、レコードの現在のアラーム状態についても、RDBMS にて管理している。

Alarm Server は、レコードのアラーム状態に変化が生じたとき、JMS へのメッセージ送信と、RDBMS のアラーム状態情報の更新を行う。この時、Alarm Client は JMS を通じてアラーム状態の変化を検知し、様々な GUI を通じてユーザーにアラーム情報を通知する。

4.2 Alarm Client

Alarm Client は、ユーザーにアラーム状態を通知するソフトウェアであり、その GUI には 5 種類のツールが存在する。グループごとのアラーム状況を通知する Alarm Area Panel、ツリー構造でのアラーム表示を行う Alarm Tree、現在のアラーム一覧を表示する Alarm Table、音声によるアラーム通知を行う Annunciator、アラームの履歴を表示する Message History といったツール群が用意されている (図 3)。

中でも Annunciator は音声によるアラーム通知を行うため、オペレーターは画面を見なくてもアラームを知る事ができる。そのため、加速器の運転で他の画面を操作している時でも、耳からアラームの通知を受ける事ができ、とても有用なツールの一つである。

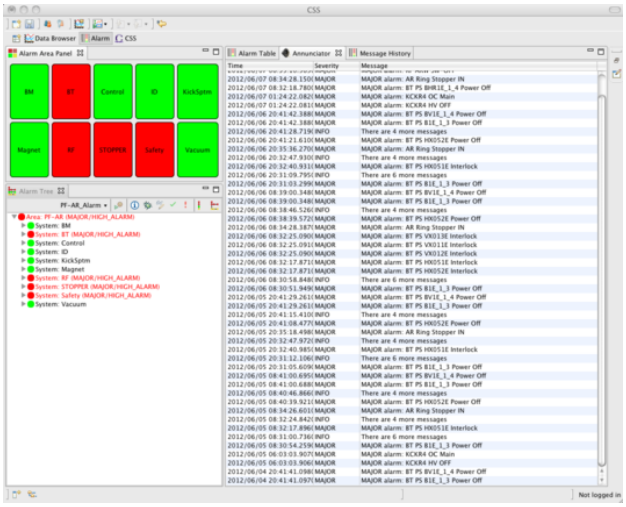

図 3: CSS Alarm Client

また Alarm Client では、アラームサーバーが利用する設定値の変更を行う事ができる。Alarm Client から設定値の変更を行ったとき、RDBMS にある設定値を保存しているテーブルの更新を行い、JMS を通じて Alarm Server に設定変更実施の通知を送信する。JMS から設定変更の通知を受けた Alarm Server は、RDBMS から設定値を読み直して新たな設定値での監視を再開する。

4.3 Alarm Config Tool

Alarm Config Tool は、アラーム監視の設定値を大量に変更する事ができるツールである。XML の形式でアラーム監視の設定情報を記述したファイルを用意し、RDBMS に保存されているデータを直接更新する。Alarm Client の項で述べた通り Alarm Client からも設定値の変更を行う事が出来るが、大量のレコードの設定を一つずつ変更するには非効率であり、Alarm Config Tool の利用が効果的である。

4.4 JMS2RDB

JMS2RDB は、JMS を通じて送受信されるメッセージの履歴を RDBMS に保存するツールである。このツールはアラームシステムだけでなく、あらゆる CSS ソフトウェアの JMS メッセージを対象に RDBMS への保存を行う。

Alarm Server は、アラーム状態の変化を検知したときに、JMS にメッセージを送信する。JMS2RDB は Alarm Server から送信された JMS メッセージを受信し、その内容を RDBMS に保存する。Alarm Client の Message History ツールは、JMS2RDB が保存したデータを検索する事で、アラーム履歴の表示を実現している。

5. 独自のアラームクライアント

CSS の Alarm Client には有力なツールが揃ってはいるが、これまで使用してきた SAD のアラームプログラムとの違いも多く、PF-AR の運転にそのまま導入するには適していない。例えば、現在のアラーム情報を表示する Alarm Table については、レコードが所属しているグループの情報が表示されないという点がある。またアラーム履歴を表示する Message History では、アラームの説明文が表示されなかったり、新規のアラームが発生したときに情報が自動で更新されないといった違いもある。これらのツールを導入する為には PF-AR の運転に適した動作を行うよう調整が必要となるが、簡単な設定変更などでは対応できず、ソフトウェア内部の修正が必要である事がわかった。

そこで、CSS の Alarm Client ツールを修正するのではなく、PF-AR の運転形態に即した、Python による独自のクライアントプログラムを作成する事とした。今回構築したアラームシステムから情報を取得するには、Apache ActiveMQ と PostgreSQL との通信機能を備えている事が必要である。Python には様々なモジュールが用意されており、Apache ActiveMQ と PostgreSQL との通信を行うモジュールを利用してプログラムを作成した。ただし、Apache ActiveMQ の初期設定では Python から直接接続できなかったため、STOMP(Simple Text Orientated Messaging Protocol) [6] という通信プロトコルを使う事にした。Apache ActiveMQ の設定を変更し、STOMP でのメッセージ通信にも対応するよう調整し、Python と Apache ActiveMQ 間でのメッセージ通信が可能となった。

従来のアラームシステムで実装していた機能を実現するようプログラムを作成した。グループごとのアラーム状況と最新 10 個のアラーム情報とを合わせて表示する Alarm Status(図 4)、アラーム状況の変化に対応して自動でアラーム履歴情報を更新する Alarm Message History(図 5)、現在の全てのアラームを表示する Current Alarm(図 6) の 3 つの Python プログラムを作成した。これらのプログラムは、Apache ActiveMQ からアラーム変化の通知を受けた後、PostgreSQL から情報を取り出して表示する事によりアラームクライアント機能を実現している。

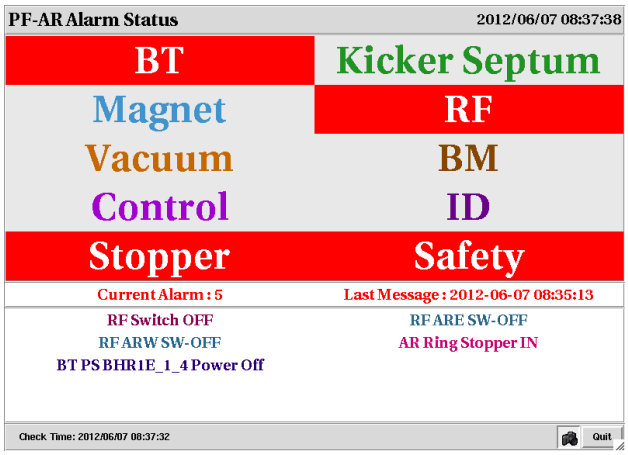

図 4: Python Alarm Status

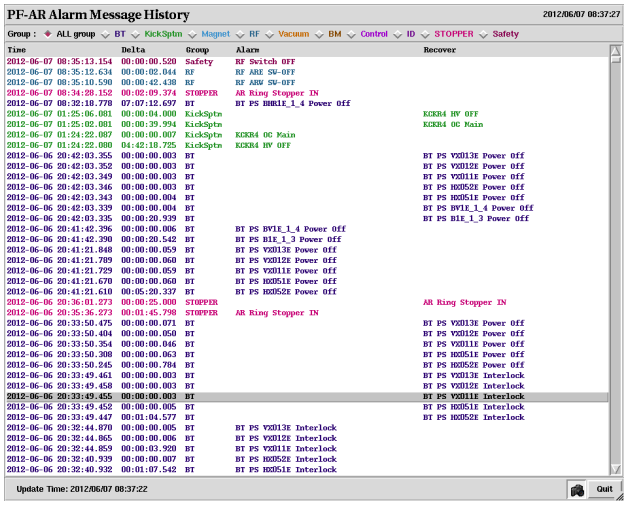

図 5: Python Alarm Message History

6． 運用実績

今回構築したアラームシステムは 2011 年 10 月から実運用を開始し、2012 年 7 月まで安定した運用を続けている。PostgreSQL は Alarm Server とは別の計算機で運用するようシステムの変更を行ったが、特に問題もなくスムーズに移行を行う事ができた。なお、PostgreSQL を運用している計算機のトラブルにより、アラームシステムが一時利用できないという状況が発生した。しかし、その計算機が復旧して PostgreSQL が正常に立ち上がると、Alarm Server は自動的に PostgreSQL との再接続を行いそのまま運用が再開された。

運転員のアラームの監視についても、Python で作成したプログラムと CSS Alarm Client の Annunciator による音声通知とを組み合わせて利用する事により、素早く適切なアラームの判断を行えるようになった。

また現在、CSS の他のツールについても評価を進めており、GUI ツールである BOY や、データアーカイバーである Archiver の動作試験を実施している。CSS の複数ツールの連携を含めたオペレーターインターフェースについては、EPICS 利用プロジェクト間でも共通化を図ろうと努力している [7]。

図 6: Python Current Alarm

7． まとめと謝辞

今回構築したアラームシステムは非常に安定して動作しており、将来的には SuperKEKB のアラームシステムへの導入も検討している。そのためには、より多数 (数万点以上) のレコードを監視したときのパフォーマンスを試験、評価したい。また、アラームシステムに続いて他の CSS ツールの活用を進めていき、加速器制御システムのさらなる安定運用の実現を目指したい。

今回のアラームシステムの導入に関して CSS 開発者の一人である Kay Kasemir 氏の多大な協力を受け、安定したシステムの構築が実現できました事を感謝いたします。

参考文献

[1] EPICS Community
http://www.aps.anl.gov/epics/

[2] Control System Studio
http://cs-studio.sourceforge.net/

[3] Control System Studio (CSS) at KEK
http://www-linac.kek.jp/cont/epics/css/

[4] Kay Kasemir, et al.,“ The Best Ever Alarm System Toolkit ”, Proceedings of ICALEPCS2009, Kobe, Oct. 12-16, 2009.

[5] Apache ActiveMQ
http://activemq.apache.org/

[6] STOMP
http://stomp.github.com/

[7] N.Kamikubora, et al.,“ オペレータ視点の J-PARC 加速器統合画面開発プラン ”, Proceedings of the 8th Annual Meeting of Particle Accelerator Society of Japan, Tsukuba, Aug. 1-3, 2011.