THE INTEGRATED ALARM SYSTEM OF THE ALMA OBSERVATORY

A. Caproni†, E. Schmid‡, European Organisation for Astronomical Research in the Southern Hemisphere (ESO), Garching, Germany

Abstract

ALMA is composed of many hardware and software systems each of which must be properly functioning to ensure the maximum efficiency. Operators in the control room, follow the operational state of the observatory by looking at a set of non-homogeneous panels. In case of problems, they have to find the reason by looking at the right panel, interpret the information and implement the counter-action that is time consuming so after an investigation, we started the development of an integrated alarm system that takes monitor point values and alarms from the monitored systems and presents alarms to operators in a coherent, efficient way. A monitored system has a hierarchical structure modelled with an acyclic graph whose nodes represent the components of the system. Each node digests monitor point values and alarms against a provided transfer function and sets its output as working or non nominal, taking into account the operational phase. The model can be mapped in a set of panels to increase operators' situation awareness and improve the efficiency of the facility.

ARCHITECTURE PRINCIPLES

During the development study, we have found that each monitored system has a hierarchical structure that can be modelled with an acyclic graph whose nodes represent the components of the system [1]. Each node, or component, of the monitored system can be working properly or be in a non-nominal state. In the latter case, the error could or could not generate an alarm to catch the attention of the operator or engineer. In fact, depending on the particular operational phase, an error could be safely ignored without distracting the operators. This is for example the case of the failure generated by a non-operational antenna during the maintenance. This case shows that having an error does not correspond 1-to-1 to an alarm: the monitor points in input to a component must be elaborated against a user provided heuristic to decide case by case if a non-nominal value in one or more of them is enough to produce an alarm for the operator. The model graph in the right side of Figure 1, shows the nodes that are working well in green and those in a non-nominal state in orange or red. Such information can be used to map the information in the model in the panels for engineers and operators as shown in the left side of the same Figure 1.

† acaproni@eso.org
‡ eschmid@eso.org

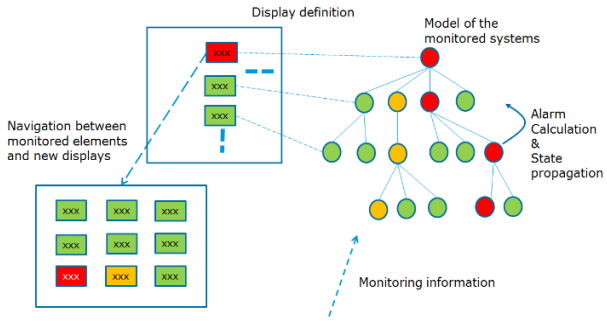

Figure 1: The schema of the alarm system.

The inputs of the IAS come in the form of values from monitor points like a temperature sensor, or alarms generated by other alarm sources such as specialized software systems like for example the ALMA Common Software (ACS) or the control system of a power plant. The set of inputs is therefore very heterogeneous: the IAS must be able to elaborate each type of input independently of its format and the software system who provides it. In the scope of this document, we will call the input to the IAS Integrated Alarm System Input/Output (IASIO), regardless if they are the values of monitor points or alarms, and without distinction of the software source that produces them.

Distributed Alarm System Unit

The core of the IAS [2] is a distributed software system composed of Distributed Alarm System Units (DASU) that concurrently evaluate the IASIOs in input and, if appropriate, produce one or more alarms. Sometimes when the translation of IASIOs into alarms is very complex or several IASIOs must be correlated, the DASU produces an intermediate value instead of an alarm. We call such temporary value a synthetic parameter. Typically, a DASU represents the IAS model of a particular subsystem of the observatory. For more complex subsystems, it can also make sense to break them down into a hierarchy of DASUs, most likely following the natural hierarchy of the subsystem.

The output produced by a DASU being an alarm or a synthetic parameter can be, in turn, the input to another DASU, as shown in Figure 2. This design follows the principle that the software systems that produce the IASIOs in input to the IAS can or cannot be completely separated. It is a central concept of this architecture the concept that the values coming from the remote systems and those produced by the DASU are indistinguishable.

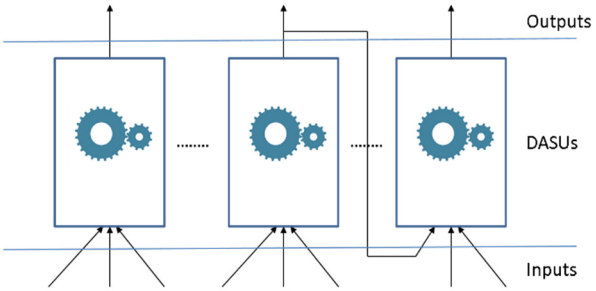

Figure 2: The DASU collaboration diagram.

It is natural to associate one DASU to each of the monitored systems but it is not the only possible way to connect the DASUs: the number of DASUs and their interconnections represents a convenient decomposition of the real system allowing to model the observatory.

If the IASIOs in input to the DASUs are produced by external software systems, they must be converted to the proper format before being processed by the DASU. Such a conversion is shown at the bottom of Figure 3 and is a conceptual representation of this task that in the real system will likely be done by a dedicated software tool.

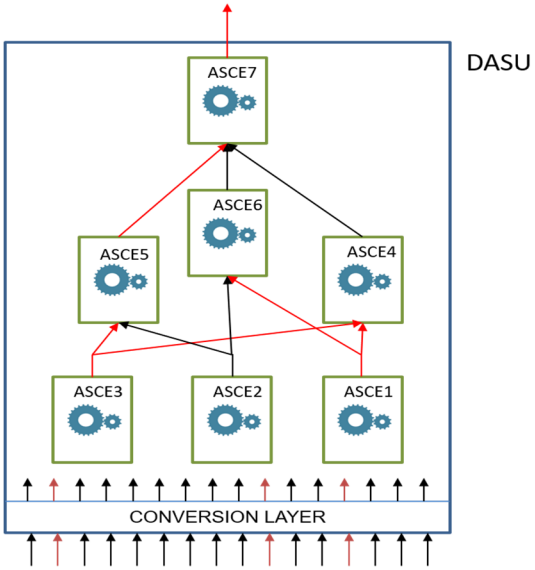

Figure 3: Internal schema of the DASU.

Often, there is a big number of inputs to a DASU and it can be challenging and error prone to correlate all the possible values of the inputs to generate the output. For this reason, the DASU is internally composed of one or more Alarm System Computational Elements (ASCE) that are software components that perform the evaluation of the inputs, to produce in the output alarms or synthetic parameters.

Alarm System Computational Elements

An ASCE is a software component that effectively makes the correlation of the inputs to produce an output. It takes as input one of more IASIOs input to the DASU, or the output of other ASCEs running in the same DASU, allowing to reduce the complexity of the computation of the entire DASU.

Each ASCE has a rule to transform the inputs into the output that can be an alarm or a synthetic parameter that represents an intermediate step of a complex computation. We call such rule a Transfer Function (TF), in analogy with neural networks where a neuron transfers the values of all its inputs to a single output. More formally, we can say that the output O of an ASCE E with n inputs, $I_1 ... I_n$, is the result of applying the transfer function $\sum$ to its inputs:

$$O_E = \sum_{i=1}^{n} I_i$$

The heuristic of the transfer function of each ASCE is provided by the operators or the engineers that have a deep knowledge of the system; the IAS will provide a set of predefined transfer functions to cope with the most common use cases. This freedom to provide user defined TFs to the IAS gives great flexibility, but user errors in the Transfer Function could endanger the IAS at run time, so parameters like execution time must be constantly monitored at run-time. Each TF must be tested before being used in production possible with the help of a simulator.

The association of TFs to ASCEs is defined in the Configuration Database (CDB); the same TF is reusable by any number of ACSEs. The connection of ASCEs is acyclic. The number and connection of ASCEs allows to model the hierarchy and interdependencies of real equipment.

THE CORE OF THE IAS

The core of the IAS is composed of the DASUs and ASCEs we already saw together with other components like the Configuration Database (CDB) and the Back-Stage Database (BSDB), and the IASIO data structure.

All these components collaborate to evaluate the inputs provided by the remote systems against the model and defined rules, and ultimately generate a number of alarms, either set or cleared. Clients of the IAS, like operator GUIs, are notified of these alarms and display them together with additional context information, such as the values provided by the monitored components.

The Integrated Alarm System Input/Output

The IAS architecture is based on the uniformity of the monitor points and alarms produced by external subsystems (after a proper conversion) and alarms and synthetic parameters produced by the IAS itself. This allows the DASU and the ASCE to have between their inputs also values calculated by IAS components as you can see in Figure 2.

Inputs from external systems contain several fields, including an identifier and the current value. The identifier uniquely identifies a IASIO inside the core of the IAS and is not the identifier of the monitor point to the monitored control software.

The value is the actual value of the monitor point or alarm: it can be a numeric value, an array of values, a bit

mask, a pattern, an alarm or something else. The IAS initially provides a basic set of types and will be extended following a bottom up approach to support the data types provided by the ALMA monitored control system.

A value provided by a controlled system has an associated validity that reflects possible problems that arose while the control software retrieved the value of a monitor point. The validity at this stage only depends on the monitored system. When a value coming from remote system is translated into a IAS data structure and injected into the core, its validity is updated taking into account possible problems during the computation or the delivery to the core, like for example network problems.

The value of a monitor point is sent to the core on change to immediately notify of updates of its value. It is also sent at regular time intervals to reinforce its validity. If a value does not arrive in the expected time frame, the core marks it as invalid and it will be properly displayed in the panels to let the operator know that the values he/she is looking at may not reflect the actual situation.

Identifier

IASIOs, Computational Elements and DASUs and more in general all the IAS components, have an identifier that allows to uniquely identify them at run time. The identifiers are defined in the CDB and are composed of three parts:

- A unique identifier, ID.
- The identifier of the parent.
- A stringified representation of the identifier.

The ID is a string that uniquely distinguishes one object from another. The identifier of the parent is the unique *identifier* of the parent, making the identifier a recursive data structure: the chain of *ID*s allows to quickly identify who owns an object and where it. The stringified version of the identifier consists of the *ID* plus the identifiers of all its parents. Its purpose is to improve performance at run-time. The recursive data structure, and the related parent identifier, is very useful for debugging: it says for example which ASCE produced an IASIO and in which DASU it runs.
The Table 1 below describes a possible assignment of parent IDs of IAS components.

Table 1: Parent Identifiers of IAS Components.

IAS Component	Parent
Value from a remote software system	The ID of the remote software system.
IASIO	• The ASCE that produced it • The plugin that generated the IASIO for the value received from a remote software system.
DASU	Nothing.
ASCE	The DASU where it runs.

PLUGINS

Each monitored system has its own control software. The control software of the ALMA array has been developed in collaboration with ALMA partners and we have full control over it. However, other control software systems in the facility are proprietary, possibly a with limited access that does not allow to deploy software. However, all control software systems provide a way to get monitor point values and alarms that can be forwarded to the IAS and presented in the panels after processing.

We call plugins the software components that interface with remote control software to collect and send monitor point values and alarms to the core of the IAS.

As we said earlier, the value of each monitor point must be sent on change and at regular time intervals. Apart of collecting and sending the values to the core, the plugins filter up front the alarms from flickering and other noise so that the values received by the IAS are stable and reliable enough to let the IAS present coherent alarms in the panels.

The IAS provides a library of the most commonly needed filters to be applied by the plugins to the values; it also provides a library to send the values to the core of the IAS so that developing a plugin ultimately means developing the part related to the monitored software system, reusing the tools provided by the IAS for the filtering and sending.

We have identified the following use cases:

- The plugin can be deployed in the control software of the monitored system: it can directly access the monitor point values
- The plugin cannot be deployed in the control software of the monitored system but the control software offers an API to get monitor points.

A plugin can easily retrieve monitor point values and alarms when it is possible to deploy it in the workspace of a monitored control software. In that case, it can run as a component of the control software or as client that can directly access the internals of the system though a specific API.

Proprietary control software could be closed in this respect and forbid to deploy software inside the control system. Some of them provide an API to access the monitor point values and alarms or, in the worst case, provide web pages and logs that could be parsed.

In both cases, running a plugin must have no side effect in the monitored system, nor in the alarm system. In particular, a plugin must not introduce instability in the monitored system if for example the integrated alarm system is not running or the transport framework not available. The other way around is also true, the integrated alarm system must not be affected by a misbehaving plugin but it must clearly report the problem in the panels.

Another key factor is the network configuration. Often a control software runs inside a shielded private network being inaccessible to plugins unless they run in the control software private network itself. In that case, it is often

possible to established an outgoing connection from the control software to the integrated alarm system network.

There are many possible scenarios for getting monitor point values and alarms from the heterogeneous remote monitored system of ALMA. Most likely the network configuration needs to be updated and there is the need to buy specific products to access monitor points and values from proprietary control software. In any case a specific solution must be found for each case as it is not possible to generalize a plugin to cover all possible cases.

BACK-STAGE DATABASE

To decouple plugins from the integrated alarm system, the plugins sends monitor point and values to a back-stage database (BSDB) instead of establishing a direct communication with the core components IAS as shown in Figure 4. The deployment of the BSDB may or may not be that of Figure 4 as it can or cannot be in the same servers of the IAS or distributed between several servers [3].

In case the BSDB is not available, the plugins stop sending data until the BSDB will be available again. All the monitor point values and alarms collected when the BSDB is unavailable are lost forever. This is not a big deal as each value will be resent after a defined time interval: when the BSDB will be available again, the updated value will be sent automatically and propagated up to the operator panels.

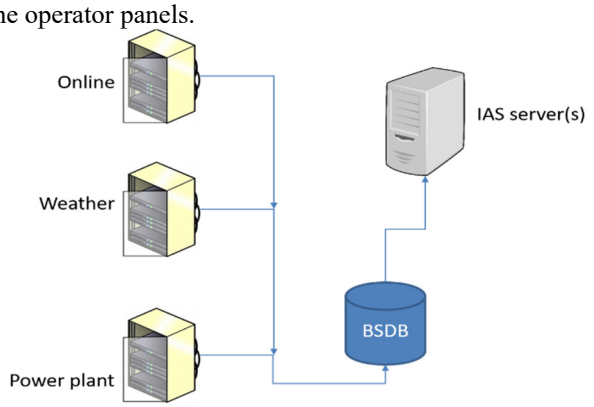

Figure 4: Communication between plugins and IAS core.

In the selection of the transport system we considered also the fact that each monitor point value and alarm has a validity: when a value is too old to be reliable it is marked as invalid to make the operator aware that what he/she sees in the alarm panel may not reflect the real state of the underlying components. There is no point to save monitor point and values for long time in the BSDB.

At the same time, the back-stage database must be fast enough to ensure the delivery of monitor point and alarms in a short time: a non-nominal state detected in a remote monitored system must be notified to operators not later the 2 seconds later.

The BSDB of the IAS adopts the Apache Kafka distributed platform for collecting and propagating monitor point values and alarms collected by the plugins to the core and the alarms and synthetic parameters produced by ASCEs and DASUs to other components of the core and up to the web server and operator/engineering panels.

Kafka offers many advantages being very fast, distributed and replicated. Kafka also automatically discards the records older than a certain age, a concept that matches the design of the IAS.

DATA FLOW

Figure 5 shows a logical view of the data flow inside the core of the IAS for the simplified case of a single monitored system and one DASU with 2 ASCEs.

The external software system on the top left side produces monitor points and alarms that are collected and sent (1) by the plugin to the temporary queue *T*. Monitor point values and alarms stored in T need to be converted into IAS data structures before being injected in the core.

Such a conversion happens in (2) where dedicated software tool extracts the records from the temporary *T* queue, and translates them into IASIO objects to be finally stored into the *IOs* queue.

The number and deployment of the *T* and *IOs* queues depends on the framework adopted, but in principle there should be one *T* queue for each remote software system to reduce the load on the converters; for the same reason, more than one *IOs* queue could be deployed.

Figure 5 also shows one DASUs with two ASCEs. The inputs of the ASCEs are IASIOs that come from the *IOs* queue (4). The output produced by the DASUs, of IASIO type, are in turn stored into the *IOs* queue (3) to be propagated from there to other DASUs (not shown in the picture). From the picture it is clear that monitor point values delivered by plugins, after conversion, and IASIOs produced by ASCEs are indistinguishable and processed the same way.

Data extracted from *T* and *IOs* queues are immediately discarded: the standard way a client has to be informed about IASIOs updates is by subscribing to the events produced by the publishers attached to the *IOs* queue(s).

Apart of DASU and ASCE, there are other consumers connected to the IO queues like the one to store IASIOs in the Long Term database (LTDB) for permanent storage, or the consumer to inject IASIOs in the web servers to be finally visible in the operator and engineering panels.

OPERATOR AND ENGINEERING PANELS

The primary function of the integrated alarm system is to show the high-level status and alarm information panel of all the main monitored systems of the ALMA observatory like for example the ALMA Antenna Array, the weather stations, the power plant, allowing the operator to identify the area of a problem in case of a non-nominal situation.

In the system displays, each element summarizes the overall status of a system with a proper colour coding. Once an element is an abnormal status, the operator will be able to navigate to a new display in order to see the

status of the sub-elements and identify the root cause of the problem. Depending on the complexity of the monitored element, it might be necessary to have several levels of displays showing more details of each element.

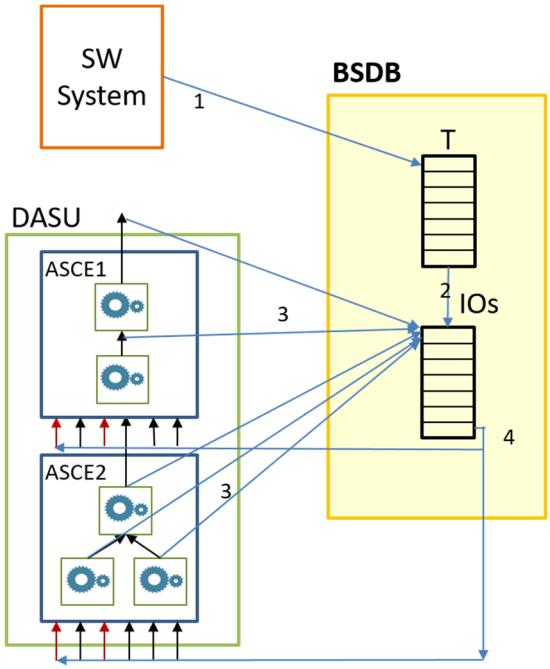

Figure 5: IAS data flow.

The panels of the alarm system are primarily displayed in the ALMA control room for the telescope operators. The operators in the control rooms are those who detect an alarm and start the counter action and are the only ones allowed to send commands to the alarm system like for example acknowledging or shelving alarms for a given time intervals.

Engineers, usually outside of the control room, are also interested in displaying alarm panels with dedicated views to the equipment they are responsible for.

Other users may want to check the state of the observatory from their premises in Europe, the USA or Japan. Such users are allowed to check the state but cannot interact with the alarm system.

To allow users to open alarms panel from the control room and many other places, the alarm panels are displayed in a browser once the user credentials have been checked.

Internally, the web server gets the IASIOs produced by ASCEs directly from the BSDB and forwards them to the panels with the proper layout and colour coding, as shown in Figure 6.

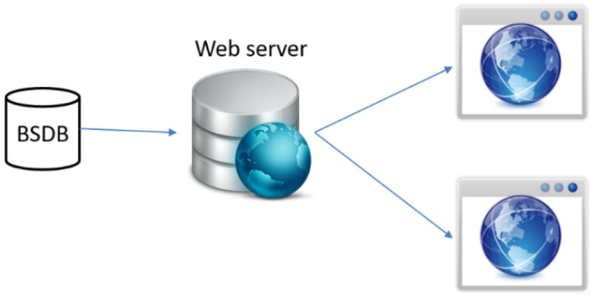

Figure 6: Operator and engineer panels.

Operator actions, like acknowledging and shelving alarms are handled entirely by the web servers and recorded in the long-term database.

The Integrated Alarm System will provide also other panels for the management of the alarm system itself. Such interfaces could be developed as web application or be dedicated Python or Java tools.

Displaying alarms for a complex environment like the ALMA observatory is not an easy task. Sometimes, especially for engineering panels, a simple tabular view could suffice but for the operators in control room there is the need to more complex and interactive panels. For example, a panel displaying an alarm that requires an urgent intervention in one of the 66 antennas shall clearly show the antenna but also in which of the 200 possible pads the antenna is located and how to reach the pad. For this kind of panels the geolocation is an important factor.

For such a complex task, we made a workshop [4] with the participation of human machine interaction experts from Inria Chile and with the involvement of operators, engineers and software developers. During the workshop, we identified the most important systems to monitor that will always be shown in a panel independently if are in a nominal or non-nominal state. In case of alarm, the entry in the main panel shows the system in non-nominal state and with a proper color coding. To understand the root cause of the problem, the operator must click over to open a sub-panel that shows a detailed view of the monitored system.

Inria human machine interaction experts prepared a detailed report of the outcome of the workshop and provided mock-up interfaces one of which is in Figure 7. It shows the antenna and the technical buildings located at the ALMA Observatory facility at 3000m a.s.l. and the facility building located at 5000m a.s.l. at the right side. Items in a non-nominal state are those in red. The final design of this interface can substantially differ from that in picture Figure 7.

Apart of operator and engineer panels there will be several technical panels and tools to investigate the behaviour of the alarm system at run-time and offline. Such panels allow, among the others, to identify cheating alarms and other noise [5].

ALARM CONFIGURATION AND DOCUMENTATION

The Integrated Alarm System is primarily an application for the operators in the control room. They have the responsibility to check the alarms and start the counter action but they are also those who better know the internals of the observatory for finding the best possible counter actions. This consideration makes the operators, together with engineers, the best suitable person to prioritize and provide documentation for the alarms.

Figure 7: Mock-up interface for the alarms generated by the building infrastructure of the ALMA observatory.

We are going to introduce the new tool following a top down approach. We identify few important and reliable alarms and display only them; when they prove to work as expected and the operators trust the new alarm system we will increase the number of displayed. The operators must help finding the documentation but also defining the transfer function to manipulate the inputs.

The integrated alarm system will offer to operators a way to write and keep up to date the documentation of each alarm with the possibility to add text, pictures and video clip. One possible solution could be to let the alarm panel provide each displayed alarm with a link to a documentation editable page like for example a wiki page.

CONCLUSION

To increase the efficiency of the ALMA observatory we found that the operators in the control room need an alarm system showing not only the alarms produced by the array control software but also those coming from other control software like the weather station or the power plant. Those alarms increase the situational awareness of the operators and allow them to tackle a problem possibly before it affects observations.

Such tool, the integrated alarm system, must be able to accept in input monitor point values and alarms coming from a set of heterogeneous control software systems. The IAS elaborates the inputs and produces the alarms, displayed by a set of panels, to operators and engineers in the control room or seated in the offices in their premises.

A set of plugins, each of which closely depend on the monitored system, retrieves monitor points and alarms from the control software and, after removing any noise with a proper filtering, sends them to the core of the IAS. The values provided by the control software need to be converted into a uniform data type to be elaborated by the core.

The core elaborates the inputs against a model to produce a set of alarms to be shown to operators and engineers. We have found that each monitored system has a hierarchical structure that can be modelled with an acyclic graph whose nodes represent the components of the system. Each node can be working as expected or in a non-nominal state. The IAS maps each node to an element in the alarm panels. We defined the IAS software architecture mapping the nodes into DASUs and allowing to decompose the elaboration of a great number of inputs in the ASCEs by applying to them a transfer function.

The IAS provides the most important transfer functions to transform the inputs into alarms, but it is possible to define custom functions to implement specific rules whose definition must be found in close collaboration with operators and engineers who have the deepest knowledge of the system.

During a workshop with human machine interaction experts from Inria Chile, and in collaboration with developers, operators and engineers, we have identified the

most important elements to show in the control room and created mock up panels for display panels.

The IAS is currently under development, available under LGPL license. It is hosted in Github [6] [7].

REFERENCES

[1] E.Schmid, “Integrated Alarm System for the ALMA Observatory, ESO-287159”, Design report, 2016.

[2] A.Caproni, E.Schmid, Integrated Alarm System Architecture, ESO-293482 Design report, 2017.

[3] A.Caproni, E.Schmid, “Integrated Alarm System Design, ESO-299387, Design report”, 2017.

[4] Technical report – Integrated Alarm System UI Front-End Workshop, Inria, Chile. 2016.

[5] B.R. Hollifield and E. Habibi, “Alarm management: seven effective methods for optimum performance”, ISA Research Triangle Park, NC, USA: 2007.

[6] Integrated Alarm System website, `https://integratedalarmsystem-group.github.io/`

[7] Integrated Alarm System on github, `https://github.com/IntegratedAlarmSystem-Group`