REDESIGN OF ALARM MONITORING SYSTEM APPLICATION BeamlineAlarminfoClient AT DESY

S.Aytac, DESY, Hamburg, Germany

Abstract

The alarm monitoring system 'Beamline-AlarminfoClient' is a very useful technical-service application at DESY, as it visually renders the locations of important alarms in some sections (e.g. fire or other emergencies). The aim of redesigning this application is to improve the software architecture and allow the easy integration of new observable areas including a new user interface design. This redesign also requires changes on server-side, where alarms are handled and the necessary alarm information is prepared for display. Currently, the client manages alarm data from 17 different servers. This number will increase dramatically in 2014 when new beam lines come into play. Thus creating templates to simplify the addition of new sections makes sense both for the server and client. The client and server are based on the Tine control system and make use of the Tine-Studio utilities, the Alarm Viewer and the Archive Viewer. This paper presents how the redesign is arranged in close collaboration with the customers.

INTRODUCTION

BeamlineAlarminfoClient (BAiC) is a visualization of emergency alarms in different areas. Currently it is used in the experimental halls PETRA III, FLASH and Photon-Science (PS) with 17 areas to be monitored.

The start of the first project was in 2004 and only used for monitoring FLASH area. Then PS alarms were added and at finally PETRA III. Since 2012 the FLASH extension project has been running and separated to FLASH-I and FLASH-II with independent FEL sources. In 2014 the PETRA III extension project began, with additional halls in the north of the storage ring and one in the east. The near future will give us ~30 areas to be monitored.

Aim of the Software Project

The functionality of the application is mostly defined by the customers and is regarded as an additional diagnostic for the technical-service personnel at DESY.

The followings alarms are monitored and displayed:

1. gas (concentration, magnetic- & exhaust valves)
2. fire
3. water
4. emergency call
5. emergency stop
6. common errors

The GUI is split into main and area views (outlined in Figure 1). If an area alarm is identified, a popup window of this area becomes visible, the active alarm is listed in the area table, and the alarm location toggles on the area plan. When the alarm is cancelled the popup becomes invisible and the alarm is listed in the main view alarm table of the last 72 hours.

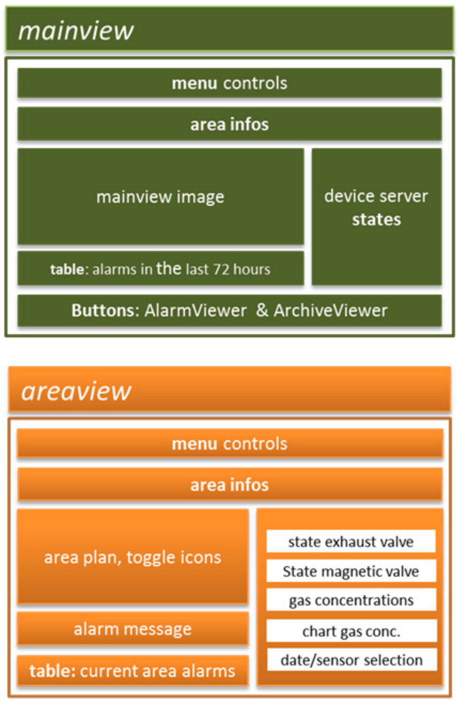

Figure 1: BeamlineAlarminfoClient GUI functionality of main and area view.

The Tine-Tool *Archive Viewer* is used for trending gas concentrations and *Alarm Viewer* for long time archiving of emergency alarms.

Motivation

Due to an unexpected increase in monitored areas, there is now a unique source code for every area server in the previous project. In 2013 a redesign of the project led to more maintainable software.

The main objective of this redesign is to reduce the project-code changes to a minimum of entries resulting from a new area as well as to reduce the communication links between server and client. In addition an upgrade of the graphical user interface is planned.

SYSTEM ARCHITECTURE

Every building displays one or more client application. The locations of these displays are at the entry of every building. Hence they are readily visible for technical-service personnel.

ISBN 978-3-95450-146-5

The client uses the tine control system [1] to get data from the server and communicates with each of the *n* servers. Thus each server has *m* number of clients to provide with data (as outlined in Figure 2).

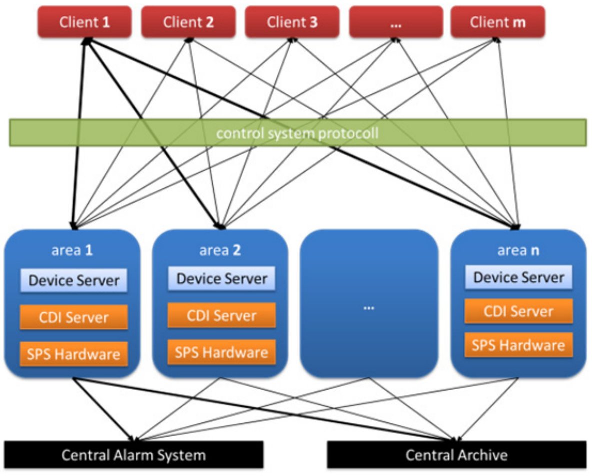

Figure 2: Communication between different server layers and the client.

Server

Before the client can display any data, this information has to be provided from a device server. Consequently the device server development depends on what alarm information the client needs to display. Due to the dependency of client and server development the main client requirements were discussed with the customers prior to any server development.

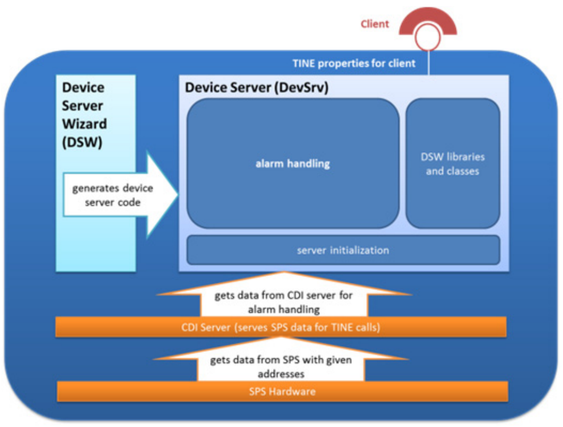
Client
TINE properties for client
Device Server Wizard (DSW)
generates device server code
Device Server (DevSrv)
alarm handling
DSW libraries and classes
server initialization
gets data from CDI server for alarm handling
CDI Server (serves SPS data for TINE calls)
gets data from SPS with given addresses
SPS Hardware

Figure 3: Device server architecture.

The server has priority because without data the client has nothing to display. In addition most logical operations concerning alarm handling are implemented at the server side. In Figure 3 the simplified server architecture is shown.

With the device wizard server [2] a basic source code for a device server is generated. Accordingly the programmer focus is on developing the handling of alarms.

During alarm handling (see Figure 4) SPS hardware data is checked for pre-defined emergency alarms. This data check occurs during the update() method of the device server. First the device server receives data from the CDI server, which maps the SPS hardware data. This data represents the value of emergency alarms.

Alarms are divided into alarm sensor states (digital alarm) and values of gas concentration (analog alarm). The server extracts the alarm information and checks digital alarms and, if needed, the analog alarms (if a gas alarm is set).

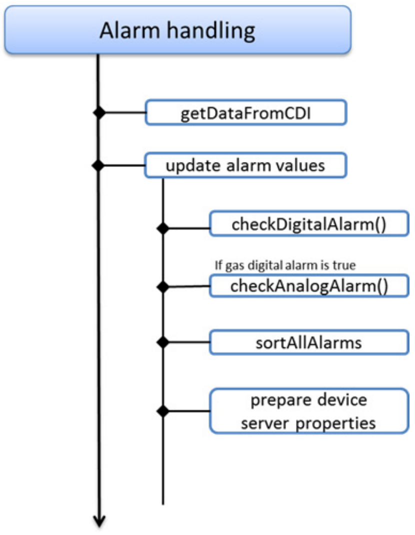

Figure 4: alarm handling.

All active alarms are saved into a list and sorted by their timestamp, and made available through property calls (active alarms). The cancelled alarms are listed separately and made available as an additional property.

Client

The client GUI is divided into jddd panels [3] and other java implementations. Jddd is useful for creating GUI components quickly, like displaying states of alarms with their icon location on the area plan. The main implementation is in rich-client java code.

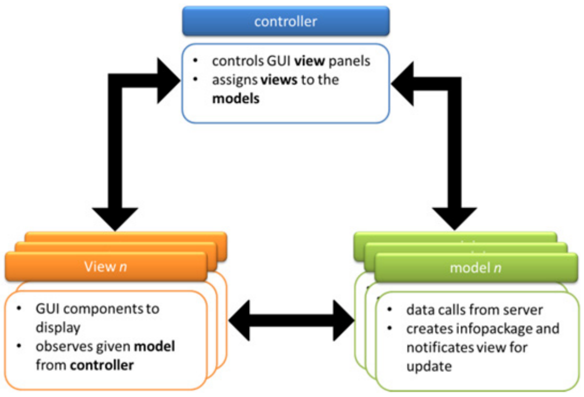
controller
controls GUI view panels
assigns views to the models
View n
GUI components to display
observes given model from controller
model n
data calls from server
creates infopackage and notificates view for update

Figure 5: MVC pattern.

ISBN 978-3-95450-146-5

The GUI uses the model-view-control (mvc) architecture for receiving and displaying data from servers. Figure 5 shows the functions of the mvc components. The controller controls all views and models and for every area an instance of a model and view is created. An array list of area server is used in the controller class to initialize area views and models.

Communication

As previously mentioned some device server properties are utilized in simple jddd panels. However this is not discussed in this paper. Instead we focus on the tine call links used in the main rich-client java code. Only one call back data link is defined to receive a property labelled *infobyte*. This property contains information about the CDI server state, existing active alarms, and whether or not an area alarm info text has been edited. Only when a flag is set in the received infobyte property, is the corresponding property called from the server. This process is shown in Figure 6.

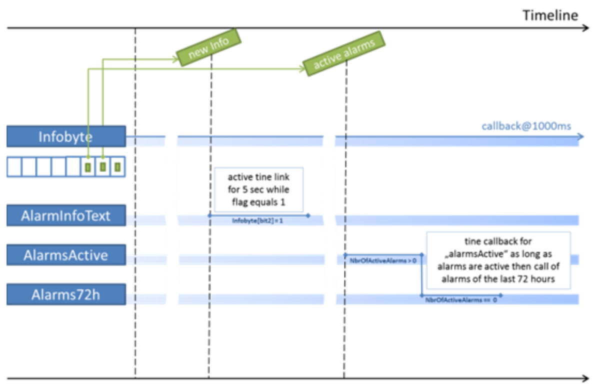
Timeline
new Info
active alarms
callback@1000ms
Infobyte
active tine link for 5 sec while flag equals 1
Infobyte[bit2] = 1
AlarmInfoText
AlarmsActive
NbrOfActiveAlarms > 0
tine callback for „alarmsActive" as long as alarms are active then call of alarms of the last 72 hours
Alarms72h
NbrOfActiveAlarms == 0

Figure 6: Client/Server communication, tine calls.

STATUS / SUMMARY

At the end of this year the goal is to start with the alpha test of this project. First tests with some parts of server and client, primarily the correct extraction and representing of the alarm states, have already passed successful.

On upgrading the GUI the view became similar to the previous one, so that better recognition for the technical service is possible.

The server/client communication was reduced to one tine call back link. Now only one server project exists with initializing files for a new area. Only the files have to be created or updated and only one project source code has to be edited. Parallel to this, new areas can be trivially added to clients.

As long as no new application functionalities are demanded on customer side the project maintenance has been dramatically simplified.

REFERENCES

[1] TINE; http://tine.desy.de
[2] Josef Wilgen, "First Experiences with a Device Server Generator for Server Applications for PETRA III", PCaPAC08, Ljubljana, Slovenia
[3] jddd; http://jddd.desy.de