PROCESS CONTROL: OBJECT ORIENTED MODEL FOR OFFLINE DATA

Ch. Gerke, T. Boeckmann, M. Clausen, J. Hatje, H. Rickens, DESY, Hamburg, Germany

Abstract

Process control systems are primarily designed to handle online real time data. But once the system has to be maintained over years of continuous operation the aspects of asset management (i.e. spare parts) and reengineering (loading process computers and field bus processors with consistent data after modification of instrumentation) become more and more important. One way to get the necessary information is data mining in the running system. The other possibility is to collect all relevant information in a database from the beginning and to build up configuration files from there. For the cryogenic systems in the XFEL, the planned x-ray free electron laser facility at DESY in Hamburg, Germany, EPICS will be used as the process control software. This paper presents the status of developing our device database which is to hold the offline data. We have chosen an approach representing the instrumentation and field bus components as objects in Java. The objects are made persistent in an Oracle database using Hibernate. The user interface will be implemented as a plugin to the control system studio CSS based on Eclipse.

INTRODUCTION

Figure 1 schematically shows the relational database used as a central store of information. Using a database makes it easy to store all information in one common location and also to ensure that there is no redundant information. In this way all applications can share consistent data which stay consistent after modifications which have to be applied over the years, because they must be modified only in a single place.

We will explain two applications in a little more detail:

- EPICSORA, a program to develop the EPICS [1] database text file, which must be loaded to the process computers.
- The IO configurator, a tool to administer the field bus configuration data and create the field bus configuration files to be loaded to the field bus devices.

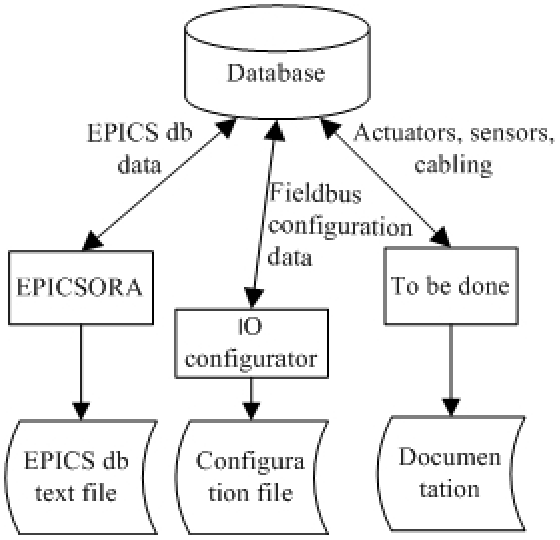
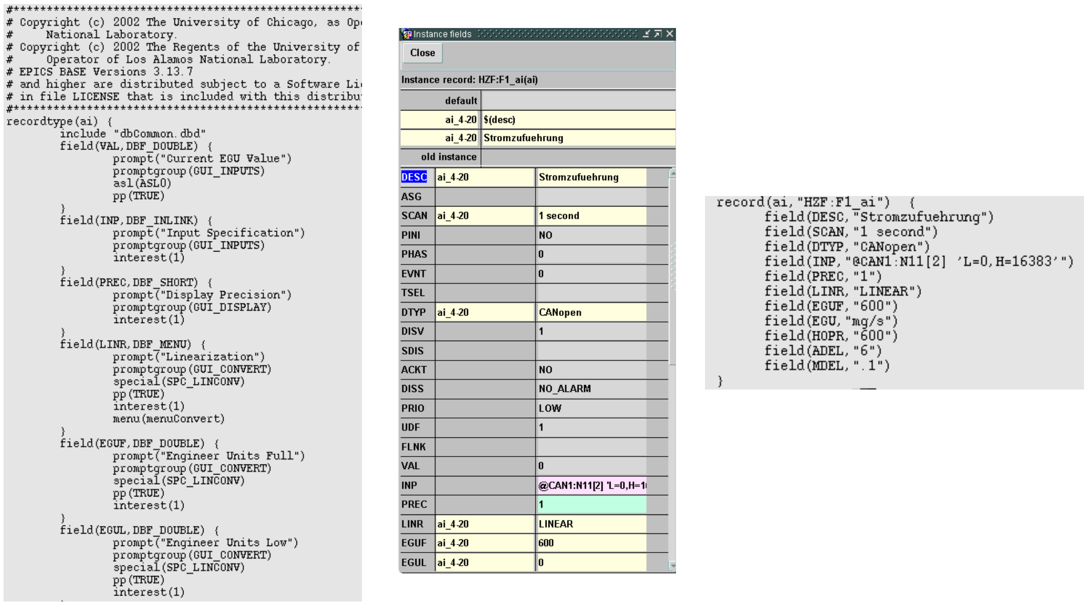


Figure 1: Database and associated programs.

These two programs are being developed with quite different tools due to the fact that EPICSORA development started in 1999 already whereas we starting



```
#*****************************************************
# Copyright (c) 2002 The University of Chicago, as Op
#     National Laboratory.
# Copyright (c) 2002 The Regents of the University of
#     Operator of Los Alamos National Laboratory.
# EPICS BASE Versions 3.13.7
# and higher are distributed subject to a Software Li
# in file LICENSE that is included with this distribu
#*****************************************************
recordtype(ai) {
        include "dbCommon.dbd"
        field(VAL,DBF_DOUBLE) {
                prompt("Current EGU Value")
                promptgroup(GUI_INPUTS)
                asl(ASL0)
                pp(TRUE)
        }
        field(INP,DBF_INLINK) {
                prompt("Input Specification")
                promptgroup(GUI_INPUTS)
                interest(1)
        }
        field(PREC,DBF_SHORT) {
                prompt("Display Precision")
                promptgroup(GUI_DISPLAY)
                interest(1)
        }
        field(LINR,DBF_MENU) {
                prompt("Linearization")
                promptgroup(GUI_CONVERT)
                special(SPC_LINCONV)
                pp(TRUE)
                interest(1)
                menu(menuConvert)
        }
        field(EGUF,DBF_DOUBLE) {
                prompt("Engineer Units Full")
                promptgroup(GUI_CONVERT)
                special(SPC_LINCONV)
                pp(TRUE)
                interest(1)
        }
        field(EGUL,DBF_DOUBLE) {
                prompt("Engineer Units Low")
                promptgroup(GUI_CONVERT)
                special(SPC_LINCONV)
                pp(TRUE)
                interest(1)
```

Instance record: HZF:F1_ai(ai)

default	
ai_4-20	$(desc)
ai_4-20	Stromzufuehrung
old instance	

DESC	ai_4-20	Stromzufuehrung
ASG		
SCAN	ai_4-20	1 second
PINI		NO
PHAS		0
EVNT		0
TSEL		
DTYP	ai_4-20	CANopen
DISV		1
SDIS		
ACKT		NO
DISS		NO_ALARM
PRIO		LOW
UDF		1
FLNK		
VAL		0
INP		@CAN1:N11[2] 'L=0,H=1
PREC		1
LINR	ai_4-20	LINEAR
EGUF	ai_4-20	600
EGUL	ai_4-20	0

```
record(ai,"HZF:F1_ai")  {
        field(DESC,"Stromzufuehrung")
        field(SCAN,"1 second")
        field(DTYP,"CANopen")
        field(INP,"@CAN1:N11[2] 'L=0,H=16383'")
        field(PREC,"1")
        field(LINR,"LINEAR")
        field(EGUF,"600")
        field(EGU,"mg/s")
        field(HOPR,"600")
        field(ADEL,"6")
        field(MDEL,".1")
}
```

Figure 2: EPICSORA takes the default values for record fields from the *.dbd files. As output it delivers the fields with modified values as a *.db file.

to work on the field bus IO configurator just this year.

EPICSORA uses Oracle Forms and Stored Procedures. On the other hand Hibernate [2] is used to make JAVA objects persistent in the Oracle database for out device database and the IO configurator.

EPICSORA

Input Output Controllers (IOC) are the working horses in EPICS. They read and write values from the process field, perform calculations on these values. They provide channel access servers which offer these values for e.g. display or archive clients.

Hardware addresses and parameters to convert raw values into physically meaningful values are defined in records. Other records describe calculations or pid-loops.

Record fields are described in data base definition (*.dbd) files. They hold default values for each field of a record. Fields which must take other than default values are listed in database (*.db) files. An example for dbd and db files is displayed in figure 2.

The IOCs are loaded with *.dbd and *.db files.

Prototypes

Usually there are several records which share the same field values, maybe only the record name and the referred input or output locations are different. There are also groups of records that are used often, e.g. analogue input, pid-calculation and analogue output for analogue control loops. Or a set of records can describe a more complex object like a section of an accelerator.

For these reasons we have introduced the concept of prototypes and instances. Whereas instances define values for all fields in the *.db file prototypes define values only for some of these fields, namely those which are the same for several instances. So we have the following hierarchy: All fields have default values defined in *.dbd, some fields get other values in the prototypes, finally some field values are set at the instance level.

The prototypes serve two purposes:

- They define values for some record fields that take the same value for several instances.
- They impose a structure on the records in the *.db file by grouping them and possibly defining links between them.

At the current stage of development EPICS does not support such information on structure, this information is preserved at the instance level in the database but not in the *.db file for the IOCs. Therefore the approach populating the database by mining for the information in the *.db files [3] cannot yield information on a structure of the records. On the other hand our approach cannot be easily applied to *.db files of the IOCs configured in the past.

Typically prototypes are used when control loops are defined or when input records are associated with their archive records. But they can also be used for more complex applications like representing objects which are instances of prototypes composed of other prototypes ("composite prototypes").

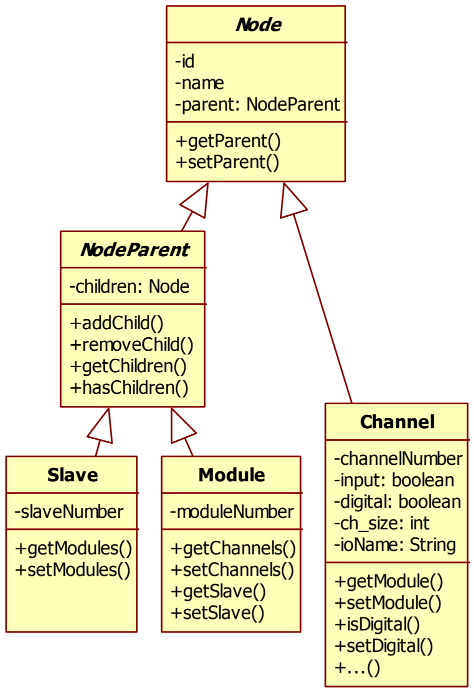


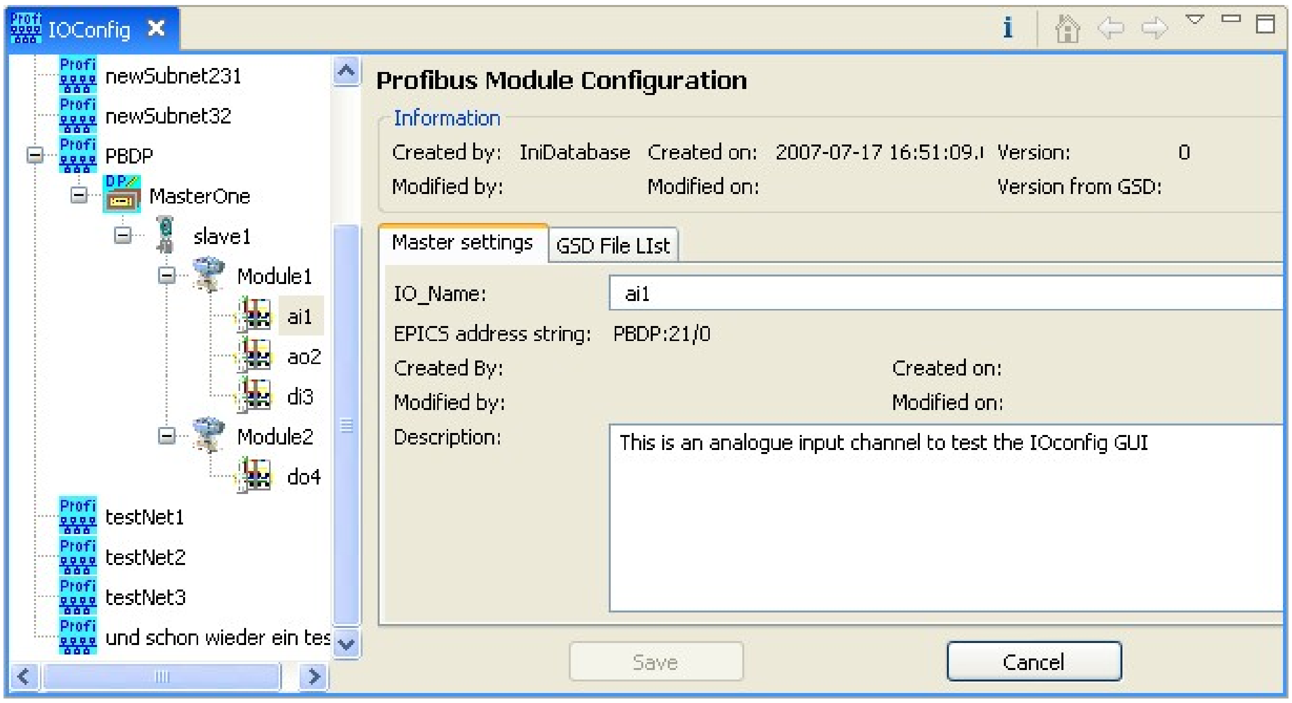
Figure 3: Schematic view of classes representing the field bus model. Slave, module and channel are specific to the Profibus field bus.

Prototypes also can have parameters. In this case their fields can define rules on how the field values for the record instance depend on the parameters. If the parameters are smartly chosen the task to create the records for quite a complex object may just require selecting an adequate prototype and the value for one parameter.

IO CONFIGURATOR

The field bus systems we have in mind (CAN, Profibus…) show a clear tree structure. There is some kind of bus controller hooked up to the IOC. From here a line leads to the field where we find remote IO stations. Our signals from the field are connected to these stations.

In objected oriented language we have some classes which describe these different types of field bus elements. The generic properties like the element's name and tree properties like "having a parent" are defined in the abstract classes "Node" and for all elements except the leafs on the tree "NodeParent", see figure 3.



Profibus Module Configuration

Information

Created by: IniDatabase Created on: 2007-07-17 16:51:09.ı Version: 0

Modified by: Modified on: Version from GSD:

Master settings | GSD File LIst

IO_Name: ai1

EPICS address string: PBDP:21/0

Created By: Created on:

Modified by: Modified on:

Description: This is an analogue input channel to test the IOconfig GUI

Figure 4: CSS view on the field bus tree (left) and the property window for the selected object (right).

The tree properties of the objects make it easy to display the structure of the bus next to the properties masks. In figure 4 we show the current status of our development of a GUI for the Profibus IO configurator in CSS [4]. It uses the JFace [5] Tree Viewer and Sash Form for the split window.

The tree properties are also used to calculate the address strings which can be used by EPICSORA to refer to the hardware channels. For this purpose we have introduced the notion "IO_name" which corresponds to the name given to an actor or sensor in the R&D drawing. In figure 4 it is displayed in the field labelled "EPICS address string". EPICSORA refers to this value if the INP or OUT field of the corresponding record contains the function $IONAME(x) with x being the required IO_name as a string parameter. Thus EPICSORA neither has to care about the layout of the bus nor must the IO configurator have knowledge about the EPICS records.

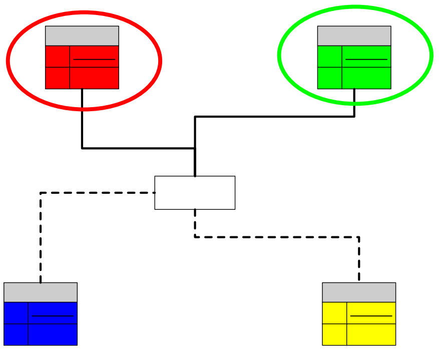


Figure 5: Different aspects of the process control system are linked by the name of the signal involved; "IO_name".

CONCLUSION

EPICSORA and the IO configurator are tools being developed to help engineers to supply the process control computers and controllers with configuration data. Although they are based on quite different programming techniques they communicate quite easily with each other. This is accomplished through the field IO_name in the database.

Thus in the development for the control software of the XFEL cryogenic system it will be possible that different developers work in parallel on different aspects. At the same time they can refer to foreign aspects using symbolic names.

REFERENCES

[1] Experimental Physics and Industrial Control System, http://www.aps.anl.gov/epics

[2] Christian Bauer and Gavin King, "Java Persistence with Hibernate", Greenwich, CT, 2006. http://www.hibernate.org

[3] Dohan, D.A., "Component/Connection/Signal Modelling of Accelerator Systems", PAC 2005.

[4] Jan Hatje et al., "Control System Studio (CSS)", this conference, ID 1341 – MOPB03

[5] JFace, http://wiki.eclipse.org/index.php/JFace