CONTROL SYSTEM INTEROPERABILITY, AN EXTREME CASE: MERGING DOOCS AND TINE

P. Duval, A. Aghababyan, O. Hensler, K. Rehlich, DESY, Hamburg, Germany

Abstract

In controlling large facilities one is rarely able to manage all controllable elements via a common control system framework. When the standard framework must deal with numerous 'foreign' elements it is often worthwhile to adopt a new framework, rather than 'disguising' such components with a wrapper. The DOOCS[1] and TINE[2] control system frameworks fall into this scenario. Both systems have a device server oriented view, which made early mapping attempts (begun in 2000) immediately successful. Transparent communication, however, is but a small (albeit important) part of the control system merger currently taking place. Both systems have well-established central services (e.g. archiving and alarms), and possess a general 'culture' which might dictate to a large extent how something is actually 'done'. The long term goal of the DOOCS/TINE merger is to be able to make use of any tool, from either the DOOCS or TINE toolbox, on any control system element.

We report here on our progress to date, concentrating on the REGAE accelerator, and plans for the XFEL accelerator (to begin commissioning in 2015).

INTRODUCTION

'Interoperability' is a bit of a trendy word these days and it is important to be clear at the outset what we mean by 'control system interoperability'.

Any control system framework will likely provide interfaces to popular scientific and engineering software such as MatLab and LabView as well as popular user utilities such as Python, Java, .Net, and the like. If these interfaces are not native to the software in question then one speaks of 'interoperability' with regard to allowing the control system to interface ('interoperate') with such external software packages. In this paper, however, we refer to 'interoperability' as being that between the different control system frameworks themselves.

Since circa 1990 control system frameworks have been typically recognized by their names rather than, say, 'the control system they use at KEK'. Likewise there has been a strong tendency for institutes to adopt an existing controls framework, rather than 'inventing their own'. The most popular of these is EPICS[3]. There are nonetheless a large number of institutes which base accelerator control on something else, for example TANGO[4], ACS[5], STARS[6] or, our primary focus here, TINE[2] and DOOCS[1].

Consequently when the primary control system is not, for instance, EPICS it often occurs that, over the course of operations, some provision must be made to interface to exotic EPICS elements which invariably creep into the system. This is in fact one of the primary motivations for pursuing interoperability. Experiments and test equipment from other facilities can suddenly introduce timelines, not to mention complexity, which necessitate seamless, rapid, and robust integration of foreign components into a control system. Epics2tine [7] is one of the first attempts to do this systematically. Since then, a number of translation interfaces and gateways such as tango2tine, epics2tango, etc. have been available.

In this vein, a doocs2tine translation layer was embedded directly into the DOOCS libraries in the year 2000. This constituted the primary step in the eventual control system merger now taking place.

Below we will first discuss what the interoperability between control system frameworks might mean in general and then give specific details concerning what it means to merge two relatively distinct control system frameworks. We note here that this goes far beyond the simple ability to 'trade data'.

CONTROL SYSTEM FRAMEWORK INTEROPERABILITY

There are in principal three ways to go concerning the interoperability between two distinct control system frameworks [8]. If System A refers to the primary control system framework, then each of these interoperability methods amounts to translating requests from System A into System B language, obtaining results, which are then translated back to System A language. This can be achieved by a stand-alone gateway process, by incorporating the translation layer directly within the System A client-side API, or by incorporating the translation layer within the System B server-side API. The relative merits of these approaches have been discussed before [8]. Solutions such as the Joint Controls Project (JCOP) [9], Control System Studio (CSS) [10], or java DOOCS Data Display (*jddd*) [11] focus on the second method listed above. We note here that the third method, server-side translation layers, being the most invasive is also the most demanding, as the introduction of any new software (the translation layer) on the front-end elements places these critical components at new risk. Nevertheless, it is precisely this third method which allows a control system merger to take place in the first place and is the key to the DOOCS/TINE merger we now describe below.

MERGING DOOCS AND TINE

Device Servers versus Databases

Control system frameworks have a general perspective concerning the accelerator control points. Some, such as,

ISBN 978-3-95450-124-3

EPICS or VISTA [12], have a database view of the controllable elements, where one thinks of 'getting' or 'setting' (or 'monitoring') some item in a database. Others, such as TANGO, TINE, and DOOCS, have a device server view of the controllable elements, which are regarded as devices at some location. Here one thinks of calling the methods of some device. That both TINE and DOOCS both have a device server perspective makes the task of merging the two considerably less daunting. DOOCS and TINE also have a three-tier naming hierarchy to identify a 'device' along with a property name to identify a 'method'. Unlike DOOCS, however, TINE elements can also take on a '*property server*' view, whereby a server does not represent an interface to a device collection so much as a service with properties, each of which in turn might refer to a different collection of keywords. We shall come back to this point below.

Request-Response Translation

The request-response translation between DOOCS and TINE is straightforward as long as both systems agree on the contents of the data being transferred. The early doocs2tine layer in fact concentrated on ensuring that the set of data types used in DOOCS were matched in TINE and vice versa. Besides the standard primitive data types, both systems also provide compound data types for atomic transfer (e.g. a name, a float, and an integer value). Such data types must of course exist in both systems. TINE also allows user-defined structures, which are not directly supported in DOOCS and presents a potential problem. However, the individual fields of a TINE structure are accessible via the normal DOOCS API.

At this point in the merger (~2001), all DOOCS servers are now 'visible' and accessible to TINE clients and all TINE servers are visible and accessible to DOOCS clients. That is, we now have the ability to 'trade data', and in a systematic way. In fact, the full gambit of the efficient transport techniques available in TINE (e.g. asynchronous communication, contract coercion [13]) are now available in DOOCS via the TINE protocol.

Culture Shock

In practice, although both systems offer rich client programing, Servers in a DOOCS-centric facility such as FLASH are usually accessed via *ddd* or *jddd* [11] panels, which are 'simple' clients with data acquisition and display widgets. Servers in a TINE-centric facility such as PETRA III are usually accessed via rich clients written in java, using RAD (Rapid Application Development) tools such as ACOP [14]. A successful merger implies that a client developer can remain in his culture of expectations and be unaware of the idiosyncrasies of either framework.

The panel approach tends to place the burden on the server developer to provide data 'ready to display', which is not a bad thing. It also tends to decouple the panel developer from making data update decisions. In the early days, a *ddd* panel would synchronously poll a TINE server even though a more efficient asynchronous communication was available. In addition, TINE server developers have been known to overload specific method calls, delivering differently encoded data based on the requested data type and input. A panel application accessing such a method will only access the 'default' method call.

Such considerations really only provide caveats to the client application developer and do not impact per se on a merger of the two systems. What does impact more strongly is the inherent control system browsing within the panel builders and other browsing tools. Here naming conventions and cultures along with browsing logic play a strong role in meeting expectations.

As noted above, TINE also supports '*property servers*'. Browsing such servers requires querying the keywords of a property as opposed to querying the properties of a device, as is the case with device servers. Although the naming hierarchy remains the same, such browsing logic must be incorporated in the relevant DOOCS utilities in a DOOCS-centric system with TINE property servers.

Infrastructure

Assuming we have addressed request-response mapping and the culture shock aspects of client applications communicating with a mixture of DOOCS and TINE servers, can we claim to have merged the two control systems? We have of course achieved something remarkable, but the answer to this question remains a resounding 'no'. What still needs to be considered is the infrastructure aspects behind the frameworks.

Archiving

An accelerator control system will have an archive system, an alarm system, naming services, and security to go along with the general culture and behavioral aspects and expectations of a user within either a DOOC-centric or TINE-centric facility.

Both DOOCS and TINE provide a local history subsystem, where the history of specific properties can be acquired directly from the servers, and there are utilities in both DOOCS and TINE which can access and display this information. However, each utility is expecting functionality which may or may not be present depending on the pedigree of the server. At the time of this writing, the expectations of either culture are approximately only 50 per cent met, with archive reading utilities often making '*if that didn't work, then try this*' decisions. We will not discuss the TINE Central or Event archive systems nor the DOOCS DAQ system at this juncture, except to note that these additional add-on services do not reflect on the merger status.

Alarms

Alarm mapping was introduced in 2009 and is by and large successful. We note that DOOCS servers 'push' alarm information to a central server, whereas TINE servers set alarms which are then 'pulled' by a central server. The alarm mapping consists then of DOOCS servers setting alarms for access via the TINE central

ISBN 978-3-95450-124-3

alarm server and for the TINE central alarm server to push selected alarms to the DOOCS central alarm server. The alarm utilities of either system can then be used to view alarms.

Naming Services

Naming servers for both DOOCS and TINE are similar in that the address of a specific device server, based on its context and server name are resolved centrally with the results being returned to the caller. Device and property information is then obtained directly from a specific server, meaning that the server must be on-line to receive that latter information. The principal complication to this scenario occurs when the device server in question is not a device server residing on a single host but is instead a device group. In DOOCS such configurations are handled administratively, whereas in TINE they are usually handled via plug-and-play. The group server mapping is done seamlessly as long as the proper information is provided within a DOOCS server's configuration file.

Security

Security can be a real show-stopper. DOOCS security is based on a *unix*-style *gid* and *uid* (group ID and user ID) access mask of the caller, whereas TINE security is based on the caller's user name and/or the network address. Where *gid* and *uid* information is unavailable, DOOCS servers attempt to match the caller's user name with available NIS or LDAP information in order to ascertain it. This approach works fine except in the case where a TINE middle layer server is attempting to issue a command to a DOOCS server. In such cases the user name of the caller is then the TINE Middle-Layer FEC (Front End Controller) name, which is definitely not a user name to be found in any NIS or LDAP table. Thus commands from such a Middle Layer are rejected. To overcome this difficulty, TINE servers now note whether a specific call is directed at a DOOCS server and if so supply the original user name of process in the command request.

Turing Tests

On could speak of undergoing Turing tests at various levels in order to determine the state of a merged system. Would a client programmer using his favorite development tool be able to distinguish between a DOOCS server and a TINE server? Do utility applications such as alarm or archive viewers behave differently depending on the flavor of the framework being used? Do remote process control applications, such as front end watchdogs, depend in any way on which kind of server process is being monitored?

The tacit goal is of course to be able to answer 'no' to all of the above questions. In reality an expert will always be able to detect differences. However the degree to which these Turing tests are being passed is sometimes remarkable, particularly as concerns the lay user.

To be sure, a browsing tool suddenly indicating a *property server* is a dead giveaway that the target must be a TINE server, as would be a target property indicating a structure data type. Alarm viewing applications on the other hand do not readily distinguish between DOOCS and TINE alarms. And although archive functionality mapping is not yet complete archive viewing applications likewise do a remarkably good job displaying data. One can now, for instance, drag and drop from a *jddd* panel into the TINE archive viewer. Framework independent remote process control is currently being addressed.

Status

FLASH is a DOOCS-centric facility but has long had native TINE servers in control, notably for the magnets. PETRA-III is a TINE-centric facility but likewise makes use of native DOOCS servers, notably in the vacuum sub-system. The 'exotic' cases here have over the years had their (mostly minor) issues, but could always be dealt with on a special basis.

The Relativistic Electron Gun for Atomic Exploration (REGAE) facility at DESY provides an excellent test bed for determining our progress in the DOOCS/TINE merger as it consists of a good mixture of TINE and DOOCS servers, as well as a good mixture of TINE rich client applications, *jddd* panels, and MatLab applications in the control room. In REGAE, virtually all DOOCS servers are communicating only via the TINE protocol, even when contacted by a *jddd* panel.

After an initial period of 'growing pains', operations in the REGAE control room have been smooth for well over a year, demonstrating the current success of the merger. This bodes well for the X-ray Free Electron Laser (XFEL) project currently underway at DESY.

REFERENCES

[1] DOOCS; http://doocs.desy.de
[2] TINE; http://tine.desy.de
[3] EPICS; http://www.aps.anl.gov/epics
[4] TANGO; http://www.tango-controls.org
[5] ACS; http://www.cosylab.com/solutions/ICT/ACS/
[6] STARS; http://pfwww.kek.jp/stars
[7] "An EPICS to TINE Translator", Z. Kakucs, et al., PCaPAC 2000 Proceedings.
[8] "The Babylonization of Control Systems", P. Duval et al., ICALEPCS 2003 Proceedings.
[9] JCOP; http://en-dep.web.cern.ch/en-dep/internal/JCOP/
[10] "The CSS Story", M. Clausen et al., These proceedings.
[11] jddd; http://jddd.desy.de
[12] VISTA; http://www.vista-control.com
[13] "The TINE Control System Protocol: How to Achieve High Scalability and Performance", P. Duval and S. Herb, PCaPAC 2010 proceedings.
[14] ACOP; http://public.cosylab.com/acop/site/

ISBN 978-3-95450-124-3