# Status of the *CS* Control System Framework

**Dietrich Beck*** **and Holger Brand**
**GSI-Darmstadt, EE, Planckstr. 1, D-64291 Darmstadt**

## Introduction

The *CS* framework is in use since a few years at about 12 experiments at five institutes. Instead of supporting a maximum number of process variables, the main focus of *CS* is to provide a basis for control systems requiring a high flexibility and performance. A detailed description of *CS* is given in [1]. This text aims at describing the status of *CS* with the upcoming version 3.10.

*CS* follows the concept of a classical three layer architecture. The front-end layer comprises of low-level hardware drivers as well as high-level drivers for instruments. The components of the front-end layer are not application specific and have a high re-usability. The application layer provides application specific modules and tasks like a sequencer, which is typically experiment specific. The third and top layer incorporates the back-end and typically contains graphical user interfaces (GUIs) and an optional Supervisory Control and Data Acquisition (SCADA) back end. The components of all layers are linked together and exchange information by sending events. Since the syntax of events is standardized, components may easily be exchanged and a control system can be re-configured on-the-fly.

## Main Ingredients of the *CS* Framework

*LabVIEW*

LabVIEW is used as the only programming language. Its learning curve is rather fast and it provides a complete set of tools from drivers for low-level hardware up to the design of high level GUIs.

*Object Oriented Approach*

An object oriented approach has been implemented within the *CS* framework. This approach eases standardization of components and reusability. As an example, each hardware device type is represented by a class. During run-time, an object of that class is created for each physical device of that type. Of course, classes exist also for GUIs, sequencers and state machines. The object oriented approach has been developed and implemented within the *CS* framework. *CS* does not use LabVIEW Object Oriented Programming (LVOOP), which has been introduced with LabVIEW version 8.20.

*DIM*

DIM [2] is a communication layer for distributed mixed environments and has been developed at CERN more than 16 years ago. Such a communication layer allows using event driven communication, scaling to large systems by distributing the tasks and remote access. The main idea behind DIM is the concept of named services. A server may publish services to which clients subscribe ("observer pattern", "one-to-many") and it may receive commands from clients ("command pattern", "many-to-one"). Both patterns allow for event driven communication and avoid polling.

* Corresponding author: d.beck@gsi.de

*Supervisory Control And Data Acquisition*

Supervisory Control And Data Acquisition (SCADA) functionality is provided by using the LabVIEW Datalogging and Supervisory Control (DSC) module. This module makes SCADA feature like alarming and trending available for *CS*. Moreover, it serves as an OPC client or even as an OPC server.

## Status of the CS Framework

Technically, changing the communication layer to DIM has been the major change during the year 2006. By this, a real publisher-subscriber mechanism became available making the design of many applications simpler and straighter forward. Moreover, DIM as communication layer allows for a much better scaling to large systems. It was demonstrated, that one million DIM services can be used for a distributed control system. Furthermore, it has been shown that five thousand objects, that may represent complex devices, can be created within one distributed system. Another step is the change of the development platform from LabVIEW version 7.1 to LabVIEW version 8.20. The main difference here has been the introduction of a project manager as well as new LabVIEW libraries (lvlibs). The release of *CS* version 3.10 based on LabVIEW 8.20 is planned for late summer 2007, an alpha version is already available. The long term stability of a *CS* based system is typically a couple of hundred hours. On MS-Windows based PCs, this requires binaries and shared libraries to be installed locally as well as rebooting a PC after the installation of software updates.

Organizationally, the main web-site containing all documentation as well as related links has been moved to a Wiki-Web [3]. A dedicated site at SourceForge [4] serves for downloading, bug tracking, and feature requests. As source code control system, we have set up a Subversion [5] repository at GSI. The *CS* framework is available under the terms of the GPL license.

## Experience Using the *CS* Framework

*Feedback from Experiments*

In general, users of operational control systems are satisfied with *CS* based control system. Typically, difficulties arise when updating to new versions of *CS*. Up to *CS* version 2.10d4, the migration to a new version has been fairly easy, since the developers of *CS* took reasonable care of backward compatibility. This situation is different with the migration to the new version 3.10. In February 2006 it was decided during a workshop, to change not only the communication layer but also to clean up the *CS* code itself. Naturally, full backward compatibility can not be maintained and the migration of user code to version 3.10 can take a few weeks to months of time. On the one hand, this is a long time. On the other hand, this is justified since most experiments took the opportunity to redesign their application taking advantage of the new features especially of the communication layer of *CS*.

*CS* is very flexible when new features are required from the users. As a result, there are always many possibilities to solve a problem. New *CS* users often have difficulties in identifying the best solution path. Especially inexperienced users have difficulties with developing control systems in a short time. *CS* based control systems are event driven, multi-threaded, operated in a distributed environment and use an object oriented approach. Where experienced programmers become productive after one or two weeks of learning time, programming beginners need significant more time to digest the underlying principles.

*View of CS Developers*

One of the ideas of the *CS* framework is to have a rather large abstraction for classes for hardware. Such classes are not experiment specific and have a high degree of reusability. Especially the experiments of the ion trap community follow this approach. But other *CS*

users often develop devices classes that are too experiment specific. Such classes can barely be reused. In the future, the following measures may assist in obtaining reusable code. Reducing the time required to maintain classes can be achieved by limiting the multiplicity of hardware device types. The compliance with coding conventions is encouraged by assigning a responsible maintainer for each class, consequent usage of features coming with LabVIEW 8.20, enforcing correct wiring of VIs by using "required" connectors and regular code reviews. Another measure is the introduction of base classes for certain categories of hardware like power supplies or arbitrary function generators. Such base classes can save a lot of work, enforce standardization, provide standardized operating panels and encourage collaboration between different experiments.

**Conclusion and Outlook**

The *CS* framework is successfully applied at about 12 experiments at GSI, CERN, MSU, University of Mainz and University of Greifswald. The new HITRAP facility at GSI will partly be controlled via a *CS* based control system. On the long term, the MATS facility at FAIR/GSI has already chosen to use *CS*. By this, the *CS* framework needs to be supported at least for the next fifteen years. Most users are pleased with the present features and performance of *CS* and new functionality is not required. At present, changes similar to the one from version 2.10d4 to 3.10 are not envisaged and it is expected that backward compatibility is guaranteed from now on.

---

[1] D. Beck et al., Nucl. Instrum. Meth. A 527 (2004) 567-579.
[2] C. Gaspar and M. Dönszelmann, Proc. IEEE Eight Conference REAL TIME '93 on *Computer Applications in Nuclear, Particle and Plasma Physics*, Vancouver, Canada, 8.-11. June 1993.
[3] http://wiki.gsi.de/cgi-bin/view/CSframework/WebHome
[4] https://sourceforge.net/projects/cs-framework/
[5] http://subversion.tigris.org/