ConSys – A NEW HIGHLY OBJECT ORIENTED CONTROL SYSTEM

K. T. Nielsen[+], J. S. Nielsen, T. Worm,
ISA, Institute for Storage Ring Facilities in Aarhus, University of Aarhus, Denmark

1 ABSTRACT

The ConSys control system[1,2] is developed for control of the storage ring facilities at ISA. A strong object-oriented approach has been used resulting in a highly flexible and modular design, where front-end and client computers have identical core software. All machine dependent information is stored in a database. The result is a site and machine independent system that is easy to maintain. The control system is implemented on PC´s running Windows NT. It follows the standard model, as well as the publisher/subscriber model.

2 THE ABSTRACT MODEL

A very limited set of object types defines and implements the core functionality of the control system: *Devices* are the interface between the controlled hardware and the control system. *Data servers* register at devices and serve as data transmitters between the *device* and the clients. Before a *client* can access parameters on the control system it must send a *data request* to the system. The *data request* must include an *address* specifying the parameter. For all data in the control system, a *ConSys data* object is used as base class. All data transmitted between computers by the *transport* object must be descendants of the ConSys data object.

2.1 Parameters

An important feature of a control system is the basic data unit used in system. Many control systems uses a full device data block as the basic access unit[3], resulting in a large number of objects types to maintain. However, for simplicity a small number of basic object types is desirable. These may afterwards be combined into logical groups reflecting the hardware structure. When the operators are operating the machine, they tend not to operate on say a full power supply, but on for example the current setting. The ConSys control system acts on single parameters. A parameter may be a current setting on a magnet, or an on/off bit for a power supply. As the value of a parameter is transmitted as a descendant object of the data value object, a great flexibility in the contents of a parameter is possible. Most parameters are of simple types, like floating and binary values, but could as well be a full spectrum from a data acquisition. As a rule the parameters on the system tend to be as fine-grained as feasible.

2.2 Addressing

A central concept in the control system is the generalised address space. As the access method of the individual hardware devices may vary greatly, the control system must provide a uniform and flexible way of accessing parameters. To create a homogeneous control system it is important to hide the details of parameter access, and only make use of this knowledge in the devices. The address object contains the information needed to access a parameter in the device. The address is fully qualified; there is no need for more information to determine the location of a parameter. The base address object only contains information about the computer and the device where the parameter is located. Any additional information is provided by descendant classes.

2.3 Devices

The device object provides the abstraction between the hardware instrument drivers and the control system. The device includes the real time database with the parameter storage needed by the given device. The base object defines abstract methods for reading and writing. An address object included with these methods locates the parameter in the device. The object furthermore includes an abstract method for parameter subscription. Data servers may call this method in order to register for subscription at the device. The device signals the registered data servers whenever data has changed in the device.

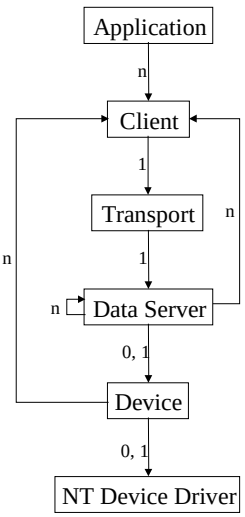

Figure 1: ConSys Object relations

An important concept is a virtual device - a device without direct connection to hardware. In the simplest case, a virtual device is just a storage device for persistent information. A more advanced use of the virtual device is intelligent data handling - that might be automation of operation or as special conversions between say physical parameters and direct controls. In the lather case the virtual device utilises the ability to connect to any parameter on the control system, just as any other client application on the system.

2.4 Data Servers

Obtaining and transmitting data to the client is done by data servers. A data server may register itself at a device. The device will signal the data server when data may have changed. It is the responsibility of the data server to decide whether the data should be sent to the client or not. When a writing operation takes place, it is the responsibility of the data server to ship the data to the device. The data server may also act as an agent for the client, combining information from one or more devices on behalf of the client. As the data server does not assume a connection to a device, a data server may start other data servers up or act as a client to do complicated tasks. As opposed to a device, a data server provides no persistent storage.

To provide the control system with initialisation information about the data servers a data request object is used. The data request includes information about the type of data server to load, and from which library to get it. The object may also contain a reference to an instance of the address object. This allows the data server to access a device. The request also contains a handle that uniquely identifies data received and transmitted by the data server. Descendant objects may add additional information used by the data server, such as criteria for transmission of data to the client.

2.5 The ConSys Client

Applications interface to the control system through the ConSys client object. This object provides a small but powerful set of methods to be used by the client program to gain access to the control system data. On creation the client is supplied with a data request. Based on the data request the client will create all necessary connections. For most applications only two methods provided by the client is needed - one for writing data to the system and one to handle data arriving from the control system. Internally in ConSys the data servers and devices are important users of the client object.

2.6 Data objects

The base of all data transmitted in the control system is the ConSys data object. Although the object can be transmitted in the system, it is the two descendant objects that are used. The first descendant object is a message object, which acts as base object for all messages. The other descendant is the data value object. This object is the base object for all parameter information exchanged in the system. In order to distinguish the origin of the data in the control system, the data value objects have a unique handle attribute defined by the client in the request. All data arriving as a result of a given request is tagged with this handle. The handle is especially used when the request results in access to more than one parameter. Likewise data sent to the control system must include a handle in order to let the system send the data to the correct destination.

In figure 1 the relations between the important objects in ConSys control system are drawn. An application may create as many client objects as it wants, depending on the structure of the application. Each client creates a transport layer, which in turns creates a single data server. The specific type of the data server depends on the request specification. The data server may create as many helper data servers or new client objects as it needs. It may also connect to a single device. A device may create as many client objects it wants. If the device is a virtual device it will not connect to a device driver. If the device is controlling hardware, it connects to a single device driver. *Figure 1 illustrates the fact that the control system is its own best client, with data servers and devices acting freely as clients on the system in parallel with access from other applications.*

3 IMPLEMENTATION

In order to take advantage of the huge amount of good and cheap software and hardware available for PC‘s, Windows NT is used as platform for ConSys. To reduce development time and cost the ConSys only runs on the Windows NT platform. Note that this is true for front-end computers as well as client computers. Having chosen the same operating system on all machines has given a possibility for a very high degree of symmetry in the system. In fact it has been possible to use the same core code on front-ends and clients, which has greatly increased simplicity, reliability and flexibility of the system.

To a large extent ConSys uses the possibility to set process and thread priorities. As the overhead of having many threads running on a system is insignificant, the control system uses a large number of threads at many different priorities to optimise the response time. The result is that the system can take a very high load before the user starts to see a significant performance penalty.

3.1 Name space

An identification triplet defines parameters in logically related groups: *machine.name.sub-name.* The machine identifies the accelerator to which the parameter belongs. The name identifies a group of related parameters, i.e. parameters controlling a given piece of equipment. The sub-name identifies the individual parameters for the piece of equipment. Although the triplet usually reflects the hardware structure, there are no assumptions made by the control system.

3.2 Establishing a data connection.

For accessing single parameters, simple requests for each data type exist. Requests include an address and information needed to dynamically create the associated

data server. To access more than one parameter from a single client a packet request is implemented. The packet request has a list of requests. This list can contain request of any type - for example other packet requests to create nested structures. The packet request is closely related to the packet data server. The packet request specifies creation of a packet data server. When this special data server is created it splits the request list into separate packet requests depending on destination front-end. For each of these single front-end requests a new client is created. If one of these clients detects a request containing requests for the local computer it creates the specified data servers. All other clients contain request for remote computers. These clients create a transport to the specified machine and send the request to the ConSys kernel on that machine. On the remote machine, yet another packet server is created, now only containing local requests on the remote machine. This packet server collects data from the local data servers on the front-end and routes them back to the original data server on the calling machine.

ConSys data is transmitted between computers using an instance of the transport object. The base transport object only has a few general methods used to handle the communication. It can be created in two modes - either as a server or as a client. When a client in ConSys wants access to a remote computer, it creates an instance of the transport in client mode with a specification of the name of the remote computer. During the creation, the transport object tries to establish a connection to the specified computer. On a successful connection, ConSys data objects can now be send trough the transport using its serialisation operators for reading and writing.

3.3 Database

All machine dependent information for the control system is stored in an ODBC compliant database – at ISA Microsoft SQL server 7.0 is used. For performance reasons the object-oriented structure of ConSys cannot be represented directly in a relational database, instead generalised tables with free fields are used.

In general, the database tables used to construct ConSys objects of a given type contain the attributes of the base object and a number of generalised fields used for the construction of descendants to the base object. Helper tables in the database describe the actual use of the general fields for a given object. With the helper tables, it has been possible to develop an object oriented database editor reflecting the object structure of ConSys.

As an example, a parameter definition in the parameter table includes the informations needed for the correspondent address object: An address type field identifies the address object associated with the parameter, computer name and device identification number fields contain information needed to initialise the base part of the address object. The additional information needed to initialise the given address object is stored in a series of general-purpose integer and string fields in the parameter table. The actual use of these fields is described in a helper table for the address objects in the control system.

The parameter table furthermore specifies the interpretation type of the parameter. The interpretation type is a combination of three related parameter attributes - the data type, the conversion type and the display type. The data type identifies the data objects known by the control system. Each data object type has at least one conversion object type and one display type. The ConSys devices on the front-ends may utilise the conversion objects to convert between native values and physical values. The display type gives the client application knowledge on how to display the parameter data. The valid combinations of data types, conversion types and display types are hardwired into the ConSys code and reflected in the database by a table maintained by the programmers.

3.4 Dynamic object load

An important mechanism in the control system is the ability to dynamically create instances of objects. By supplying the system with the name of the class and the dynamic link library, the system is able to create an instance of the class. This method is used to create the various components of the control system. The transport layer, database interface, and the devices are created at start-up of the control system. Likewise the data servers are dynamically created as specified by the client request. This ability to dynamically add and configure the system without re-compilation has proved a great advantage.

4 CONCLUSION

The object-oriented approach has resulted in a highly flexible and modular design, where front-end and client computers have identical core software. The limited number of base objects, the central database for all machine dependent information and the possibility of loading and creating object instances dynamically has lead to a site and machine independent control system that is easy to maintain.

REFERENCES

[1] J.S.Nielsen K.T.Nielsen and T.Worm: “ConSys – A new control system for ASTRID and ELISA”, TUP46G, Proc. From the 6'th European Particle Accelerator Conference, EPAC98, Stockholm 1998, p 1679.

[2] www.isa.au.dk/ConSys/

[3] Jie Chen, Graham Heyes, Walt Akers, Danjin Wu, Wiliam A. Watson III: “CDEV: An Object-Orientated Class Library for Developing Device Control Applications”, ICALEPCS 1996 proceedings

+ Now at Scanvaegt International A/S