GSI

Architectural Design Specification

1.0

SHIPTRAP Control System

Distribution list:

Name (alphab.)	**Department**
Faouzi Attallah	Atomic Physics/GSI
Wolfgang Quint	Atomic Physics/GSI
Gerrit Marx	Atomic Physics/GSI
Christophor Kozhuharov	Atomic Physics/GSI
Klaus Poppensieker	DVEE/GSI
Dietrich Beck	DVEE/GSI
Holger Brand	DVEE/GSI
Jürgen Neumayr	SHIPTRAP/Munich

Author:
Dept.: DVEE
Name: Dietrich Beck
Tel.: 2520
Signature:

Inspector:
Dept.: DVEE
Name: K. Poppensieker
Tel.: 2782
Signature:

File: G:\beck\shiptrap\documents\archsp.doc
Date: December 6, 2001

Status: <doc status>
File: Project file Sect. <nn.nn>
File directory <directory>

Document Management

History of changes

Version	Status	Date	Person resp.	Reason for Change
0.0	created	11 Sep 2001	Dietrich Beck	
1.0	update	6 Dec 2001	Dietrich Beck	alpha version of framework

Persons authorized to make changes

Dietrich Beck DVEE/GSI
Klaus Poppensieker DVEE/GSI

Document was created using the following tools:

WINWORD 2000

Contents

1 Introduction

1.1 Purpose of the document

The purpose of this architectural design specification is to define the architecture of the SHIPTRAP control system and to describe the behavior between the product components. Moreover, these design specifications will be updated during the implementation of the control system and finally serve as a documentation of the software.

1.2 Validity of the document

The architectural design specification does apply to the SW and the HW of the SHIPTRAP control system. Only HW devices like the DS345 function generator, that are directly connected by the software, are regarded as being part of the control system. Other HW like an einzel-lense in a CF150 crosspiece are not regarded as being part of the control system.
The design specification is based on the SW requirements specification. However, for the first versions of this document this design specification was/is still based on the preliminary requirements.

1.3 Definitions of terms and abbreviations

- AFG: arbitrary function generator
- Configuration Data: the requested parameter values to be set at the HW, the operational mode of the experiment, the HW devices and processes to be used, the scan range, etc….
- CDD: Configuration Data for the Dynamic parameters
- CDS: Configuration Data for the Static parameters
- Device: a HW device like an AFG
- Device Driver: an independent process that is the interface of the control system to the instrument driver of a device.
- DSC: Data logging and Supervisory Control (the DSC Module is an additional module for LV replacing BridgeVIEW)
- Dynamic Parameters: “Dynamic parameters” are the parameters that can _not_ be implemented with the help of OPC servers in a straightforward way, like loading a waveform to an AFG generator.
- HW: hardware
- NI: National Instruments
- OPC: OLE for Process Control: By this, lots of data items (that have simple types) can be exchanged between the nodes of distributed systems. Typical examples are values for voltages, DIGI I/O, …
- Static parameters: In general, “static parameters” are the parameters that can straightforwardly be implemented with the help of OPC servers.
- SW: software
- VI: virtual instrument

1.4 Relationship with other documents

The document is related to the SW design specifications as a basis. Later, the architectural design specifications will be refined by the detailed design specifications. These documents can be accessed via the SHIPTRAP control system home page http://www-wnt.gsi.de/shiptrap/.

2 Product architecture

The control system will be developed with LV from NI. It consists of two parts. There is a general part that is not SHIPTRAP specific and another part that is SHIPTRAP specific.

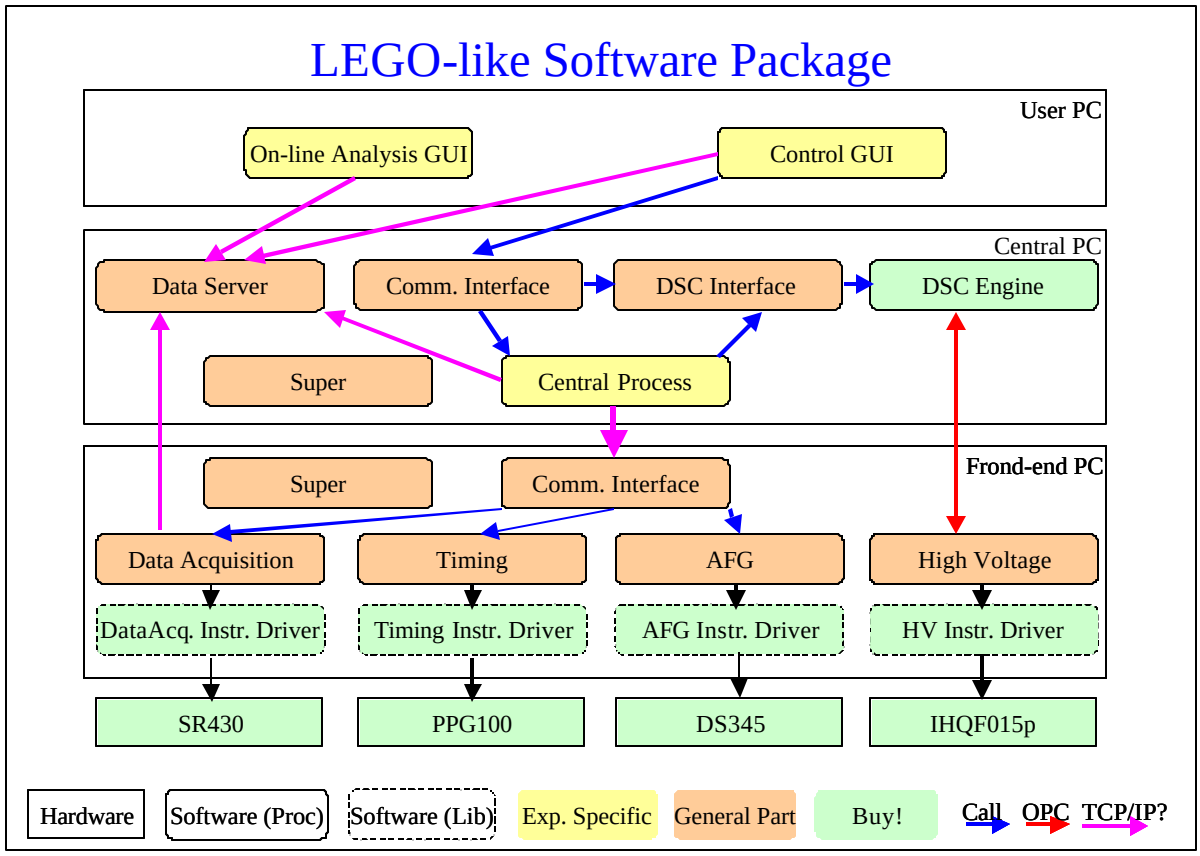
2.1 Distributed Computing

In principle, the control and data acquisition could be done on one single PC. However, it may be good to separate functions and to distribute certain tasks to different nodes.

First, there is(are) the user node(s). In principle, those nodes can be anywhere at GSI or in the world. These are the nodes where the GUIs run on.

Second, there is exactly one central node. Here run the central process, the DSC engine, the DSC interface process, the super process, the data handler and the data server. Also, some HW devices may be connected to this node.
Third, there is(are) the front end node(s). Here, one typically will host processes that either consume lots of resources and/or processes that are safety relevant and should have a node for their own.

2.2 Product structure



The processes that are SHIPTRAP specific are the

1. Control GUI, the
2. On-line analysis GUI, the
3. Central Process(es), and the
4. Data Handler.

Everything else belongs to the general part that is not SHIPTRAP specific. The idea behind the general part is to provide a framework that allows for the implementation of control systems for (almost) any experiment. The whole control system is event driven.

2.2.1 Event Driven Processing

The event driven properties of the control system are implemented in two ways.

One possibility are “calls”. Here, a caller sends an event to another process, the callee. A callee may be any process that does basically nothing but waiting for calls. After receiving a call, the callee will perform the requested action. If requested, the callee will return an answer. Communication by calls does only consume very little CPU time and avoids polling.

Instead of waiting for a call, a process can listen to an OPC tag for a requested parameter value to change its value. This can be done by the read tag VI from NI without using CPU time. Waiting for a change of an OPC tag is always an asynchronous process, since no answer is returned to the callee. Instead, the process will update the actual parameter value which is a second OPC tag.

2.2.2 Control GUI

Here, the experimentalist configures the parameter values that have to be sent to the HW and configures how the measurement will be done (scan range, parameter to scan, used devices, etc…). Also, the control GUI allows to monitor the actual parameters and serves as the user interface for the alarming. When the configuration of the requested parameters (configuration data) has been done, this configuration data will be sent to the central process and the DSC interface process for initialization of the hardware via the device drivers. After initialization, the measurement is started and stopped from the control GUI. From the control GUI, one can also change parameter values of devices on-line and during a measurement.

The control GUI will be implemented with LV.

2.2.3 On-Line Analysis GUI

The on-line analysis GUI serves to view/check the performance of the experiment and the incoming data on-line. It can, as an example, be implemented with LV but one could also think of something different like Root.

2.2.4 Super

The super process is similar to the one used for the ISOLTRAP experiment. It manages loading and unloading of device drivers and other processes. Moreover, it maintains lists the processes currently loaded. There will be a super process on all nodes with a DSC engine and/or processes.

The super process will be implemented with LV.

2.2.5 Central Process

The central process serves to control the devices with dynamic parameters. It receives the configuration data for the dynamic parameters, loads the required device drivers, sets the requested values and configures the data module at the data server. When a measurement is started, it changes the scan parameter, triggers the read out of the data via a data acquisition device, etc… . The design of the central process itself is not yet drawn up, since it depends on the requirements of SHIPTRAP which are not yet clear. The DSC engine is linked to the central process via the DSC interface process.

The central process class is written in LV.

Comment: In principle, there can be more than one central process or sub-processes depending on the requirements of the experiment. An example for such a sub-process may be a vacuum control process that may run on a front end node. For simplicity, only one central process is shown in the figure in Sect. 2.2.

2.2.6 The DSC engine

The DSC engine is a process within the environment of the operation system. The DSC engine maintains so called tags which are complex process variables with a value, alarm limits, etc… . Basically the DSC engine serves to handle devices with static parameters. Moreover, it handles the alarming and trending of the parameters of the control system.

The DSC engine is a process provided by the DSC module from NI.

2.2.7 The DSC Interface Process

The DSC interface process serves for various things. First, it is used to set the tag values for static devices according to the configuration data. Second, it serves as an interface when the central (or another) process wants to change values of static parameters. Third, the central process can check if all values requested for the static parameters have been applied via the DSC interface process.

The DSC interface process class is implemented with LV.

2.2.8 Communication Interface

A communication interfaces can be used to transfer events/call from one node to another. For example, if the central process will call a (device)process for a dynamic device on its local node, it will not use the communication interface. However, if the central process calls a (device)process for a dynamic device on a remote node, this call will go via the communication interface on the remote node. One has to watch out, that the communication must have the capability of serving multiple calls at the same time.

The implementation will be done with LV.

2.2.9 Front End DSC Engine

A front end DSC engine may be needed when devices with static parameters are installed on remote nodes different from the central node. Like this, one avoids that device drivers have to read/write from/to a remote DSC engine. Especially, device drivers do not have to reconnect to their tags, when the DSC engine on the remote node restarts after a shut down.

The parameter values of devices with static parameters are only set by the front end DSC engine. The problem of a front end DSC engine is that alarms and other properties can not be transferred between different DSC engines. But this will be implemented in future versions by NI.

An alternative would be if device drivers on the front end nodes are directly connected to the DSC engine on the central node via DataSocket.

The DSC engine is a process provided by the DSC module from NI.

2.2.10 Devices and Drivers

Everything from the instrument driver, to the HW device is straight forward according to standard NI solutions. As an example, the HW device may be represented by an AFG DS345 from Stanford Research. This is connected by the HW interface, a GPIB card, to the computer. For the GPIB card, the driver provided by NI is used. In many cases, an instrument driver is already provided by NI. For devices where this is not the case, one has to write an instrument driver according to the recommendations by NI. Up to this point, everything can be off-the shelf. The device driver sitting on top of all this is self-written and interfaces the instrument driver to the rest of the control system. The device driver is called from the central process or via a change of a tag value of the DSC engine. The device driver is an instance of a device driver class and a process within the LV environment. Each hardware device is represented by a device driver instance. As an example, if there are three DS345 AFGs, there will be three instances (processes) instantiated from a DS345 class.

A device driver class will be implemented in LV.

2.2.11 Data Server

The data server is used by the data acquisition device driver to write its data to. It represents an interface for further data processing. It basically buffers the data in memory for a limited time. The data can then be retrieved via a TCP/IP connection. For further data processing one can either use

LV or something completely different like MBS. Also multiple connections should be possible via a publisher and subscriber model. There are two types of connections, “hard” and “soft” connections. If the internal buffer of the data server is full and a “hard” connection is established, the data server will no longer accept the upload of new data and a measurement will stop. For a “soft” connection the same situation will cause a disconnect.

The data server is a process within the LV environment and written in LV. Typically, the data server will run on the same node as the data acquisition device driver.

2.2.12 Data Handling

The procedure how to handle the data is still an open question, since no clear requirements have been given yet. In any case, the data handling is not a general part of the control system but specific for the SHIPTRAP experiment. However, many different scenarios for the data handling are open, since the data server serves as an interface for almost any data handling procedure.

Possible scenarios

1. A data handler gets the data from the data server. Then, the data handler writes the data to the backup medium and to a network disk. The on-line analysis GUI then operates on the open file on the network disk where the data handler continuously writes the data to.
2. A data handler gets the data from the data server. Then, the data handler writes the data to the backup medium and to the network disk. The on-line analysis GUI gets the data as an additional subscriber from the data server.
3. The on-line analysis GUI gets the data from the data server and also writes the data to the network disk. Here, an optional data handler can be used as an additional subscriber to the data server to get the data and store it to the backup medium.
4. MBS gets the data from the data server and writes the data to the backup medium as well as to the network disk. The data are then shuffled to the on-line analysis GUI buy MBS as well.

It is proposed to use scenario 2 since this does not restrict the number of on-line analysis GUIs (0-n) and has the advantage that there is no need to operate on the open file where the data is continuously stored. But it has the disadvantage that one does not have an instant feedback, if the storing of the data really works. The data handler will be implemented with LV.

2.3 Behavior of the product components

Here, a typical procedure is given to demonstrate the behavior and the relationship of the product components.

2.3.1 Power-Up

After power up of the PCs there is the following situation. All PCs are running, nobody is logged in.

2.3.2 Startup

Here, one has to start the Super process and the DSC engines on the central node as well as on the front end nodes. Some device drivers and special control processes for safety sensitive HW are started. The question is, if all these processes can be started automatically. It is not straight forward to start those processes as services in Windows. Maybe, NI will find a solution.

2.3.3 Control GUI Start and Setting up Configuration Data

The user starts the control GUI. Then, he/she has to login and sets up the configuration data for the dynamic parameters (CDD) of the control system and the configuration data for the static parameters (CDS) of the control system. The configuration data contains the devices to be used, the parameter values to be set at the device drivers and the configuration of the central process itself.

2.3.4 Initializing the Hardware and the Central Process

The CDD are transferred to the central process, while the CDS are transferred to the DSC engine on the central node via the DSC interface process. The central process then loads all device drivers required to set the parameter values for the dynamic parameters and sets the values. Moreover, the central process itself is configured by setting the scan range, the operational mode, etc… . The device drivers required to set the parameter values for the static parameters are instantiated via the Super process. Who tells the Super process to instantiate the device drivers and sets their values?

2.3.4.1 Example: Loading and Initializing a PPG100

The pattern generator PPG100 is a device which has dynamic parameters. The central process instantiates the PPG100 device driver via the Super process. Then, the central process resets the PPG100 device to its default values and transfers the pattern data to the device driver. For transferring the pattern data, the central process and the device driver communicate via the "call and answer mechanism" (see sect. 2.1).

2.3.4.2 Example: Loading and Initializing a HV power supply

A HV power supply is a device which has static parameters. The HV power supply device driver is loaded (by whom?) via the Super process. The requested values are then set by setting the values corresponding tags in the DSC engine via the DSC interface process. The device driver of the HV power supply gets the requested value by the "wait for changed value mechanism" (see sect. 2.1) and will set the requested values at the hardware according to the tag values.

2.3.4.3 Example: Loading and Initializing a MCS

An MCS is used for data acquisition and is a device with dynamic parameters. Its device driver is instantiated by the central process via the Super process. Moreover, the device driver gets the information to which data object of the data server the acquired data will be transferred to.

2.3.5 Starting a Measurement

Once the hardware has been configured, the user can enter the start command in the control GUI. That command is then sent to the central process. Then, the central process has to prepare everything for the start. That is

1. the loop with the scan values
2. the data server
3. the data acquisition device
4. the pattern generator

2.3.6 Doing a Measurement

Within the scan loop the central process has to do the following for each cycle.

1. set the next scan value
2. start the data acquisition device
3. (enable external triggers for the pattern generator)
4. start the pattern generator
5. wait for the pattern generator to finish
6. (disable external triggers for the pattern generator)
7. tell the data acquisition to process the data. The device driver of the data acquisition then
    a. reads the data from the hardware
    b. puts the data to the data object at the data server
8. clear the data acquisition device
9. go back to 1.

2.3.7 Stopping a Measurement

The measurement is stopped whenever the user gives a stop command at the control GUI. That command is transferred to the central process. The central process does the following.

1. (disable external triggers for the pattern generator)

2. stop the pattern generator
3. stop the data acquisition device

There are two modes for stopping a measurement. First, a measurement can be stopped immediately without waiting for the present scan to finish (BREAK). Second, a measurement will continue until the present scan is finished. After that, the measurement is stopped (STOP).

2.3.8 De-Initializing the hardware and the central process

After a request by the user, the control GUI will send the de-initialize command to the central process and the DSC interface process. Those two processes will stop and unload all running devices drivers that have been loaded an initialized earlier (see Sect. 2.3.4). Then, the central process itself will terminate. The DSC interface process will continue running.

2.3.9 Shutdown

After de-initializing the hardware and the central process, the DSC engines, the communication interfaces, the data server and the super process will still run. The shutdown procedure will force those processes to terminate.

2.3.10 On-Line Analysis GUI

The on-line analysis is separated from the control GUI and does not get the init/start/stop commands. So, the on-line analysis GUI connects to the data server upon start and gets the data. An identifier of the current measurement and a data event number is coming with the data (see Sect. 2.6). By this, the on-line analysis GUI knows when a new measurement has started.

It is not yet clear, how the on-line analysis will be implemented.

2.3.11 Trending and Alarming

For each process, there exist three status tags, three error tags and a watchdog tag. The event loop, the periodic loop and the state machine loop have each a status and a error tag. The watchdog tag gives information about the watchdog status for a device. The tags have to be created for each process that exists in the control system. They will be present even if the process has not been instantiated. The processes write their status as well as their error information to these tags via the DSC interface process. Like this, historic trending and alarming is all done by the DSC engine. However, for this method a strict naming conventions for the tags has to be used.
Once an error has occurred, the ongoing action will discontinue and the alarm will be transferred to the control GUI where the user has to acknowledge it.
Note: Additional tags can be created and used by user classes.

2.3.12 Security

There may be more than one instance of the control GUI running. However, only one GUI can be the active one (control mode) that is allowed to set controls. All other control GUI just serve to display the parameters values presently set (monitor mode). For normal users: If another GUI wants to switch to control mode, the first GUI must switch to monitor mode first. For administrator users: An administrator can always switch any GUI from monitor to control mode, independently whether another GUI is in control mode. In this case, the other GUI that was in control mode, is switched to monitor mode.

Buy using different user levels one would also have an additional security for safety critical controls, like the one for venting a vacuum chamber.

There may be more than one instance of the on-line analysis GUI running.

Security will be implemented using the features of the DSC engine.

2.3.13 Non-Functional Characteristics

Windoze is not a real time operation system. Nevertheless, short reaction times smaller than 10ms can be achieved when using the “call and answer” mechanism on one node. Reaction times can be a factor of up to 10-100 longer, when using the “wait for changed value” mechanism and/or the two

communicating processes are running on different nodes. For time critical sequences or procedures, HW modules like the PPG100 are used that have a time granularity of about 100ns. However, one has to keep in mind that some hardware devices may also take their time. For example, it may take a few 100ms for the MCS SR430 to read out its data, clear the internal buffer and get ready for the next cycle.

2.4 Reason for decisions relating to the product architecture

2.4.1 Decision

The decision for this design is strongly based on the experience with the control system for ISOLTRAP. There have been a few feasibility studies showing that such a system can also be implemented with LV. An "ISOLTRAP control system" like system has the advantage of being event driven and very flexible concerning changes of the sequences, devices used and procedures. Device driver processes following the idea of the ISOLTRAP system are well suited for devices with dynamic parameters.

Additionally, it is required to have alarming, trending and security features. Since it would take a long time to implement those features by ourselves, the DSC engine from NI will be used which provides those features already. Moreover, the DSC engine is ideally suited for the handling of static parameters from OPC servers. Here, the combination of an OPC server and the DSC engine already has some functionality of a device driver and there is no need for the implementation of a software device driver.

2.4.2 Advantage

Both approaches are quite complementary. On the one side, there is the high flexibility of the ISOLTRAP type system for changes even during run-time. On the other side, there is industry standard, alarming, security, trending and an off-the shelf solution in the case of the DSC engine. A merge of the two approaches makes an ideal system. For each HW device one can choose the approach that is best suited. For setting new parameter values, one can change a parameter value for ANY device, even if it is connected via the DSC engine, with the "call and answer" mechanism thanks to the DSC interface process. That makes the system extremely transparent from the GUI as well as from the central process.

Although the principle relationships between the components will not change, the architecture itself is quite flexible depending on the requirements of SHIPTRAP, that may change with time. For large systems, one can have more processes like the central process. For small systems one can have two scenarios. First, one can think of systems entirely without DSC engine depending on the "call and answer" mechanism. Even devices with static parameters can be implemented with device driver for dynamic parameters. For this scenario one does not have DSC features like alarming, trending and security but there is the need for the implementation of a simple error logging. Second, one can think of systems which have no devices driver for devices with dynamic parameters and that have no sequences like the one required for the cycles and scans for SHIPTRAP. Then, one can do with the DSC engine alone and there is only a simple central process for communication with the DSC engine.

2.4.3 Disadvantage

There are two disadvantages of the present design. First, the use of the DSC module(s) is a cost factor. Its about 3700.- Euro for the DSC module and one(!) run-time system. Second, one has to have the some know-how of the event oriented "call and answer" mechanism that is needed on the one side, and the DSC engine on the other side. Also, it takes some time to learn how the G++ toolkit from Vogel Automatisierungstechnik works.

2.4.4 Other Solution Paths

For a control system like the one needed for SHIPTRAP, one could think of having system of ISOLTRAP type alone and without a DSC engine. This would imply the need of programming the functionality of the DSC engine, which would take too long.

The other extreme, the DSC engine alone, has other disadvantages. Here, not all hardware devices can be programmed as being stupid devices like DACs or ADCs. One would need to implement a protocol above the DSC engine. There are two disadvantages. First, communication will be slower. Second, one needs workarounds for implementing device drivers having dynamic parameters.

One could also do without the G++ toolkit from Vogel Automatisierungstechnik. As an advantage, one does not depend on this toolkit. As a disadvantage, one has to invest a few man months to implement the basic features of this toolkit.

2.5 Interfaces

The control GUI and the on-line analysis GUI have user interfaces. These interfaces depend strongly on the requirements of the experimentalists which are not clear yet. The interface to the central process and the DSC interface process is realized by a call from the control GUI via the communication interface running on the central node. The configuration data for dynamic and static parameters are both transferred to the central node by the call itself or via a data server. The control GUI also allows to change single process variables via the communication interface.

The central process, sub-processes of the type of the central process and device drivers communicate by sending and receiving calls (LV queues), signals (notifications) and listening/writing to OPC variables (DSC engine tags). All these processes can also connect themselves to a data server for uploading or downloading data.

The DSC engine is interfaced by listening/writing to its tags and the alarm and security mechanisms. For high level processes, the DSC interface process is used for passing the data. Low(er) level processes like device driver can listen to tags via the OPC-events from the G++ toolkit.

An OPC Server communicates with a DSC engine via the tags. Each tag at the OPC server has a corresponding tag in the DSC engine.

Data can be uploaded/downloaded to/from the data server via TCP/IP connections. The installation, removal of data modules as well as operations on data modules is done via the “call and answer” mechanism. For SHIPTRAP, the data elements for the data acquired do have the following structure.

- data identifier
- number of data element (to know if the data are complete and when to clear the display for example in the on-line analysis GUI)
- THE data

A data handler connects to the interface to the data server via TCP/IP via a “hard” link. In the scenario shown in the figure in sect. 2.2 the data handler just writes the data to files.

On the local node, the communication interface does the communication via the “call and answer” mechanism. To/from a remote node, the communication is done via TCP/IP.

2.6 Data model

The configuration data created by the control GUI is based on a database. The data model of this data base depends on the requirements of SHIPTRAP and is not yet clear.

2.7 Technical constraints

For the support of HW drivers, we restrict ourselves to the Windoze operating systems. Another option would be LV RT.

3 Product components

This is just an overview of the product components. The functions and non-functional features are described in the document with the detailed design specifications.

3.1 Experiment specific

1. Control GUI: User interface for the controls but not for the on-line analysis.
2. On-Line Analysis GUI: The on-line analysis GUI displays the data of the current measurement.
3. Central Process: A SHIPTRAP specific process that is the heart of the system and an image of the experiment. Loading/initializing/unloading of device drivers, scanning etc. are done here.
4. Data Handler: Storing of data to data storage device.

3.2 General part, not experiment specific

1. Super: Shuts- down and restarts the control system (DSC-engine, data server, communication interface, etc…). Does some management concerning loading and unloading of device drivers and processes.
2. DSC Engine: Alarming, trending, safety. The processes/functions do error handling and status reporting via the DSC engine. The DSC engine provides complex process variables, the tags. The tags are also used by devices with static parameters for the “wait for changed value” mechanism.
3. DSC Interface Process: The DSC interface process serves as an interface for other processes to the DSC engine.
4. Communication Interface: allows for transparent communication with processes on remote nodes.
5. Front End DSC Engine: A front end DSC engine has the task of making the processes on the front end nodes independent of the status of the DSC engine on the central node. The functionality of the front end DSC engine is the same as for the DSC engine on the central node. It is not yet clear whether one needs also a front end DSC engine interface process.
6. Device Driver: A device driver is the interface between the instrument driver from LV and the control system.
7. Data Server: Provides data buffer and an interface for transferring data between processes.