Deliverable 7.2

EISCAT_3D Radar Control and Monitoring Subsystem Report

Jussi Markkanen and Assar Westman

15-March-2009

Table of Contents

1. Introduction

This report concerns the global, user-level, Control and Monitoring (C&M) of the EISCAT_3D radar. At existing EISCAT radars, the user-level C&M software system is called EROS ("EISCAT Real-Time Operating System"). The main thrust of this report is to argue that the present-day EROS could, and indeed preferably should, be used as the starting point also for EISCAT_3D C&M.

This report will not attempt to define in any way the actual C&M commands for the individual subsystems of EISCAT_3D (for the present EISCAT, the EROS command reference runs for about 60 pages). For one thing, at this stage of EISCAT_3D development, there are not even nearly sufficient details available to allow such issues to be realistically addressed. More importantly, from the point view of developing the user-level C&M system, it does not matter much what the low-level control programs of devices and subsystems actually do in detail, as long as the radar subsystems are "EROS-conforming". An EROS-conforming subsystem or device has these properties

- Engineering-level C&M software for the subsystem exists.
- The software is able to access all the necessary aspects of the hardware.
- The software provides programmable C&M in a form that allows interfacing to EROS in some standard way.

Long experience has shown that it is normally both fast (a few days' work) and safe (no adverse side effect) to integrate a conforming subsystem into EROS. Therefore, it is unnecessary, and even wasteful, to go into details about EROS commands for a subsystem too early. Even at the time when the project actually starts placing tenders for the various subsystems to the industry, it will not be necessary to have a detailed

EROS-level interface designed. The requests for quotations should merely request that the engineering subsystems be EROS-conforming. Clearly, this approach means that we have resigned to accept a bottom-up, hardware-driven development of the C&M system. The approach has worked well for existing EISCAT radars. And while more powerful than existing EISCAT radars in many ways, EISCAT_3D is still only a radar, and does not seem to place qualitatively new requirements for the overall C&M system.

The work description of this work package stipulates a literature study to inspect *the current state of the art in real-time control systems architecture and software* in order to find if an existing control system *meets the specific requirements of the EISCAT_3D project*. We have not done such a study. Very early on it became clear to us that the in-house EROS system can be evolved to meet also the needs of the EISCAT_3D radar. We give our justification for this claim in section 3 of this report. And once it is accepted that EROS can do the job, the argument for actually using EROS for the job becomes, to our mind, overwhelming, and any further search for a suitable C&M becomes moot. This is the subject of section 2. In sections 4 and 5 we explain the strongly networked architecture of present EROS and point out how the system should be made even more networked for EISCAT_3D. In section 6 we elaborate, with examples, what we do, and do not, want concerning the engineering-level C&M software. Section 7 summarizes the migration path from present EROS to a baseline C&M system for EISCAT_3D.

2. Benefits of having an EROS-type system for EISCAT_3D

Our basic assumption is that EISCAT_3D C&M system ultimately needs to be maintained and operated by EISCAT, even in the case that much of the hardware would be built and maintained by an external (even commercial) consortium. That is, we assume that it is EISCAT staff, together with the scientific user community, that is mainly responsible for the actual EISCAT_3D operation. Our second assumption is that at the time of EISCAT_3D, there will be other, older EISCAT radars like the EISCAT Svalbard Radar, still in operation. If these premises hold, there is strong motivation to evolve the EISCAT_3D control and monitoring system from the existing EROS.

EROS supports many kinds of users via a scriptable command language

The user-level C&M software is crucial for the usability of a radar system. The C&M software is the radar's user interface, and is clearly mission-critical. With a scientific radar, it is particularly important that users with different levels of technical radar expertise can make use of the system. In the future, we anticipate welcoming many EISCAT users with less radar experience than is typical in the present day. This will make a well-tested and well-known (well-documented) user interface even more important in the future. It must be possible to offer simple ways of operating for some users, by providing ready-made "packages" of operation, while also making it possible for experienced users to have full access to all features of the radar. The way this essential flexibility is achieved in EROS is by the use of a scriptable command language, called ELAN, which can control all aspects of the radar. It is possible for experienced users, including EISCAT staff members, to write ELAN command scripts for their technically less experienced colleagues. The ever-expanding library of such scripts can be either used as such, or can be used as starting point for smaller or larger modifications. This flexibility should be provided by any EISCAT_3D C&M system. EROS, of course, has been built from the ground up to facilitate full programmatic control of the EISCAT radars.

The familiarity with EROS helps in maintenance and development

The radar downtime should be kept in the minimum. Software problems will always occur, but the more quickly they can be solved the better. In a scientific context, the user requirements for the software can change rapidly. A new piece of hardware may come available, and must be integrated into the system. An existing piece of hardware can develop a fault that still allows meaningful operation to continue, if only the control software can be suitably hacked. All these things mean that it must be possible for the software to be adapted quickly, cheaply, and without fuss, to whatever will be required by the hardware situation on the one side, and by the user wishes on the other side.

If EISCAT operates the radar, it seems necessary that EISCAT has good in-house understanding and complete in-house control of the C&M software. The idea of starting to fill web-forms of a commercial software service contract in order to get some obscure bug fixed, or some curious user request to fulfilled, while the paying scientist with her rocket on the launch-pad is biting her nails, is too nightmarish for us to contemplate. The EROS system has been built entirely in-house, and is very familiar to us. EROS is implemented as a rather lightweight, loosely coupled system of interacting processes, where most faults will be well isolated and relatively easy to locate and correct. Even substantial problems can be repaired in-flight when an experiment is going on, without taking the system down. To preserve the flexibility of software service and development also for the EISCAT_3D, it seems desirable that all essential parts of the overall C&M software not only be designed but actually implemented by EISCAT staff, based on the existing EROS core.

A related advantage in adapting EROS as the C&M system also for the future EISCAT radars is that the regular software development and maintenance work going on at the existing EISCAT radars, will then automatically benefit the new system also. The present radars will act as a good testing environment for the software of the new system.

EROS provides unified control of all EISCAT radars

A key aim in EROS design has been to provide a uniform interface for the radar hardware across all EISCAT radar systems and sites. The purpose is that from the user's point of view, all EISCAT radars look and feel as much the same as possible, and feel part of a single system. This must be seen beneficial also in the future, only, to reign-in on the increasing hardware complexity brought by the EISCAT_3D, more so. As a relatively general-purpose, command-based radar control environment, EROS is quite good in supporting this kind of unifying approach. The devices and subsystems may be different from site to site and from system to system, but they all are still controlled by just some commands. Often both the command name and command parameters can be kept intact between systems, and only the internal implementation will need to be made system-dependent. In the very least, the command name can be kept intact, and only some of the command parameters need to be adapted for specific system. This kind of unification would be lost if one would use EROS at the old radars but something else at EISCAT_3D.

3. EROS is able to do the job

EROS has proved to be a capable and rather convenient system for controlling and monitoring of all current EISCAT radars, but it is still also under continuous development. Over the 25+ years lifetime of EROS, there has been one complete pre-implementation from scratch, and in addition several major re-writes. As smaller modification, support for new EISCAT facilities has been recently added to the EROS system. One recent EROS module is for EISCAT's ionospheric heater facility, another one is the E3D module added for the EISCAT_3D demonstrator array (WP 7.1). The two new modules are still far from feature-complete, but for the older sites, at this time we do not know of any major complaint about missing features. It is probably fair to say that the current version of EROS, EROS 5, is a reasonably mature, but in no way frozen, C&M system.

The envisioned EISCAT_3D radar, though advanced in terms of antenna and transmission hardware and receiver signal-processing performance, does not be so much qualitatively different from the present systems that it could no more be handled by EROS, by adding a few new modules. Things that *would* make life difficult for EROS would be anything that would necessitate a "hard" real-time operating system: if it for instance would be required that the EROS core should be able to guarantee that some critical events unconditionally take place in a precisely specified instants of time (say, specified within a small fraction of a second), no matter what else might be happening in the system; or should always happen with precisely specified delays after a multitude of different kinds of trigger events. We would not, for instance, want to use an EROS-type system to run any kind of automated weapons system. But there are no such needs in the present EISCAT systems, nor does it appear to be in the EISCAT_3D.

Nevertheless, EROS *is* a real-time system in the sense that it can serve multiple asynchronous command sources simultaneously, and is able to respect the user's wish about the timing of events, to some degree, most of the time. In most cases, it is not a problem if events are delayed a little, as long as the order of the events is maintained. But importantly, in addition to being able to take care of these kind of soft real-time needs, EROS is capable of serving some crucial, hard real-time needs where events *must* happen at the specified time to be useful at all. The EROS core sets up, in its soft and leisurely fashion, but well in advance in time, suitable trigger events for separate, external, precision-clock-based hardware subsystems. These subsystems will then start at the specified fractional microsecond, based on the clock time, and will run locked to the clock-signal thereafter. This arrangement makes it possible for EROS to control, admittedly in a somewhat limited way, those events in the transmission and reception that need to occur at a specified fraction of a microsecond. It appears that this combination of soft and hard real-time control is still sufficient also for the EISCAT_3D.

Thus, we find no fundamental reason why the present, well-tried, in-house EROS could not be scaled up to meet the quantitatively increased, but qualitatively still pretty much unchanged, control and monitoring needs of the EISCAT_3D system as well.

4. EROS control of an EISCAT radar

EROS view on an EISCAT radar

Figure 1 shows an overview of an EISCAT radar system, from the EROS point of view, at a typical radar unit (radar site). Basically, a radar is a signal processing factory, consisting of smaller or larger collection of functional subsystems forming the "signal path" to generate, receive, process and store the signal; and a timing layer to make sure that critical events on the signal path happen at precisely determined instants of time. For the electromagnetism-based signal, the large value of the signal speed necessitates that the timing in several places be controlled to sub-microsecond accuracy in the present radars, and down to sub-nanosecond accuracy in the time-steered EISCAT_3D. Depending of the radar, the subsystems on the signal path may include modules for generating transmission; antennas with their direction controllers; analogue receivers; digital receivers and other digital signal processing equipment; data storage facilities; and various computing machinery for data analysis.

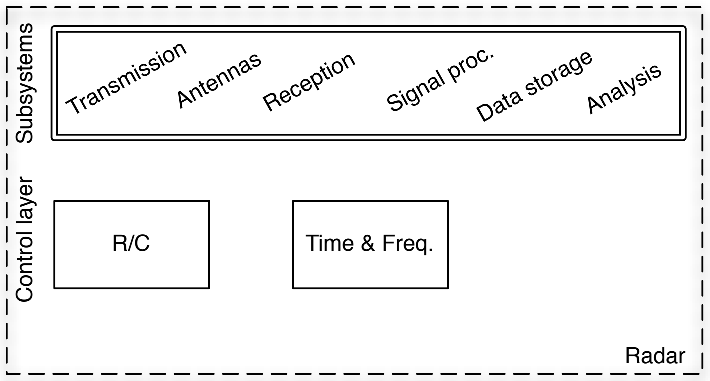

Figure 1. EROS view of an EISCAT radar. *A typical radar unit that EROS needs to control consists of a set of subsystem forming the signal path, and a control layer that provides hard real-time control of the signal path.*

All the hardware subsystems on the signal path can, and in most cases will, include some "engineering-level" C&M software, in addition to the hardware proper, so that EROS preferably does not need to know about the fine details of the hardware. Although, in the present EISCAT systems, this wish is not always granted, and the EROS kernel performs some pretty low, device register bit-level operations, too. The timing layer consists of GPS-based timing subsystem, and one or more programmable radar-controller (R/C) modules. The timing system maintains the time, and also generates the various clock signals needed in a phase-coherent radar. The timing system also generates START- and RESTART trigger signals to the radar controllers, at precisely controlled instants of time. After being started, the radar controller executes a loaded program sequentially, locked to the high-precision clock, to issue high time-resolution and high time-accuracy commands to the signal path.

Three types of EROS commands

Sitting on top of the radar hardware, the EROS system supports three basic functions. First, EROS acts as a tool of convenience, the user interface, for a radar engineer in accessing individual subsystems interactively, for instance, to do some hardware testing. To this purpose, EROS supports a large set of device-specific commands. Second, EROS provides the means to coordinate the multitude of radar subsystems for the purpose of running radar measurements (called "radar experiments" in EISCAT). This involves configuring the signal path and performing coordinated actions to get the signal flowing on the path. To this purpose, in addition of the device-specific commands, EROS also supports commands and command options to create and manage the real-time structure of an experiment.

As indicated, EROS is a command-based system; a software engineer's viewpoint could be that EROS is just a run-time system for the EISCAT Radar Command Language, ELAN. A typical experiment requires tens of ELAN-language commands to be given, often in a periodic manner, over sometimes very long periods of time. Therefore, in almost all cases, EISCAT experiments are implemented as programs written in the ELAN language. After an experiment program is started via a special EROS command, it typical proceeds without much interactive control afterwards, until the program is ultimately stopped (see Figure 5 for the general case).

The third central task of EROS, in addition of supporting (local) "device commands" and (general) "control commands", is to support (networked) "communication commands". This last, crucial, function takes place largely behind the scenes, and provides the distributed environment needed for running EISCAT experiments.

The distinction between local and remote system in current EROS

In the current version of EROS, shown in Figure 2, there is still a rather sharp and visible distinction between the "local" radar and "local EROS" on one hand, and the "remote" radar and "remote EROS" on the other hand. The distinction concerns the amount of available monitoring information and command feedback; at present, the amount of visible information is quite different at local and remote sites. When working on the EROS console terminal at the local site, it is possible to send interactive EROS commands to any remote site (the arrow labelled "8" in Figure 2). Also the command script that is executing at the local EROS can contain commands to be executed at remote sites. The remote commands allow a limited form of monitoring of the remote radars from the local site. There are special commands to display state information from the remote systems. However, much of the automatically generated monitoring and command-feedback information is only visible on various screen windows at the local radar.

At least for the more complicated or critical experiments, with present EROS an operator needs to be actually present at the participating radar site in person. But it is a specified aim for the EISCAT_3D that the radar sites must be able to be operated unmanned.

At present, in an emergency, or for small-scale software maintenance work such as re-starting the whole EROS system, it is possible to project an EROS console display from a remote site to the location where it is needed, using standard VNC (Virtual Network Computing) technology, but we do not consider this as a viable long-term solution to the monitoring problem. We think that for EISCAT_3D, it is necessary to provide full, uniform access to any remote EROS system from any local EROS system (and some, more restricted, access even from some non-EISCAT, non-EROS systems). That is, we want the distinction between "local" and "remote" EROS largely to vanish in the future.

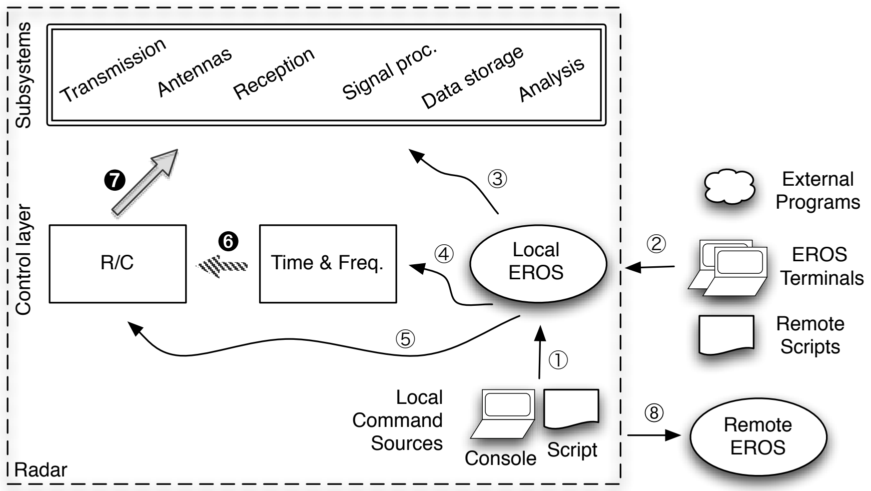

Figure 2. EROS control of current EISCAT radars. *Local EROS receives commands either from local console user or from locally executing script (1), or via internet from other locations (2). The commands either affect directly the radar subsystems on the signal path (3), or prime the hard real time system (4)-(5), to emit the most time-critical commands (6)-(7). It is also possible to send control-and-monitoring commands to other EISCAT radars, both interactively from the console and from the experiment script (8). Presently, the local console has a privileged view of the local radar's status, not available elsewhere.*

Future EROS must provide full console access from everywhere

For the EISCAT_3D, we intend to get rid of the distinction between the fully capable "Local Console" on one hand, and the restricted view offered by a "Remote Terminal" on the other hand. Instead, in the future, all EROS Terminals must have full Console access, as indicated in Figure 3. To achieve this does not seem to require any fundamental changes in EROS internal architecture, and should probably be implemented in some future versions of EROS irrespective of EISCAT_3D developments.

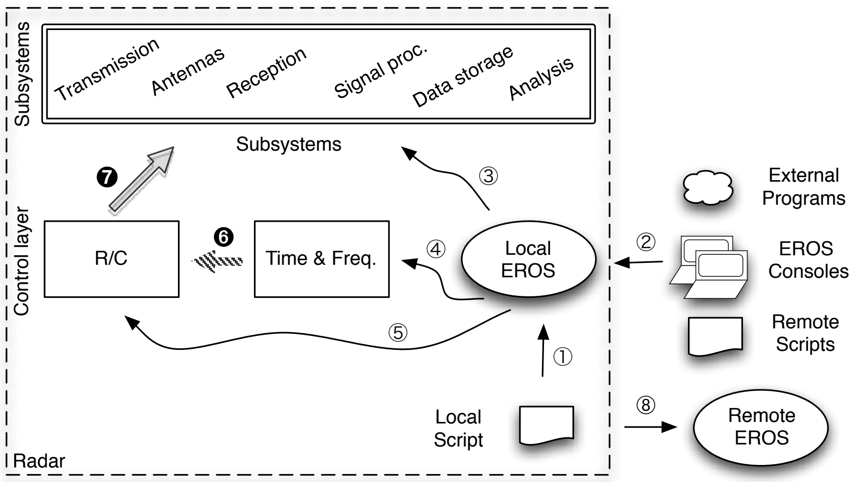

Figure 3. EROS control of future EISCAT radars. *The next major EROS update will remove the privileged status of the Local EROS Console, basically by removing the Local Console itself, so that EROS will run without any controlling terminal. All interactive users, whether "local" or logging in over the internet, will in principle be able to receive the same level of service.*

Thus, in the future, any part of EISCAT system (that is under EROS control at all; and almost everything ought to be), can be accessed from anywhere within the system, in a uniform manner. It will be possible to have a multiple controlling consoles for a single experiment, though it still will probably to be a bad idea to have multiple remote scripts to issue commands to the same local EROS.

It will also be possible for EROS to run with no Console at all; this option will come together with a change in the EROS boot strategy. In the present system, EROS at a given radar must be booted up by an interactive command by a local user (or a user accessing the site over VNC), and that site, and the particular terminal there, will host the local EROS Console. In the future system, each "local EROS" will be booted up automatically when the computers responsible for running EROS are booted up, and will always be available. Each such local EROS also contains a communication server program, the "EROS Engine". The local EROS will not create any visible windows anywhere during the boot. Instead, a client program, the EROS Console, which can be run "anywhere", including the local radar, connects to the EROS Engine of the radar of interest and creates whatever EROS windows are desired. The current EROS implementation already basically makes use of client-server architecture for communication, so the changes required to implement the above EROS boot-up and inter-radar communication strategy are not large.

5. The internal structure of EROS

Figure 4 shows the top-level internal structure of EROS at a radar site. EROS consists of a set of concurrent, interacting processing, executing on a set of computers. All EROS systems have a core of three processes, called the ENGINE process, the EXPDOER process, and the MONITOR process, and, depending on the particulars of the site, a smaller or larger set of other involved programs and processes, which is labelled by the trendy term CLOUD in Figure 4. The CLOUD, typically distributed to many computers, has access to the radar's various functional subsystems.

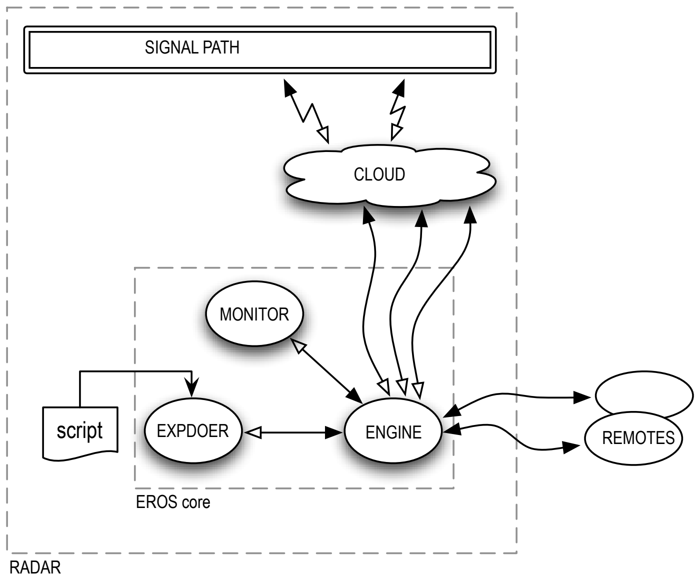

Figure 4. Top-level internal structure of EROS. *EROS is implemented as a core of three interacting Tcl/Tk interpreter processes, all executing in a single UNIX workstation, and a radar-site-depend "cloud" of other programs and processes living in many type of computers available to the core via the local area network.*

Between (some of) the processes, there are bi-directional communication channels for the exchange of commands and messages. The ENGINE process is the central element of EROS. It acts as a communication hub and maintains the EROS state variables at the local radar. All EROS-level commands directed to the radar's subsystems, whether they come from a local script, from interactive consoles, or from some external user-programs, ultimately go through the ENGINE. This allows ENGINE to perform the important task of a command flow sequencer, in order to ensure command integrity so that only a single command source acts on a subsystem at a time. The ENGINE process also collects the command responses and sends them over to the command sources when appropriate. It also collects the status streams coming from the MONITOR and from the CLOUD, and pushes the data onto currently logged-in interactive EROS consoles.

The task of the EXPDOER process is to support the execution of an experiment script. The ELAN language script executes in the EXPDOER process, and in addition, EXPDOER works together with the ENGINE and MONITOR processes when an experiment state change is required. As indicated in Figure 5, EROS allows an experiment to run as a state machine. At present, only one experiment can be running at a time at a given site, but this could change for EISCAT_3D. An experiment is either stopped, or is executing in some BLOCK. From experiment programmer's point of view, a BLOCK is a subroutine in the ELAN script which it is possible to jump to, from anywhere within the script, using a GOTO command. The GOTO command can be coded into the script itself, possible behind some conditional, or, crucially, can be given interactively at any time. Typically, a BLOCK defines some functional mode in the experiment, so that the interactive GOTO command can be used to reconfigure the experiment as external conditions require.

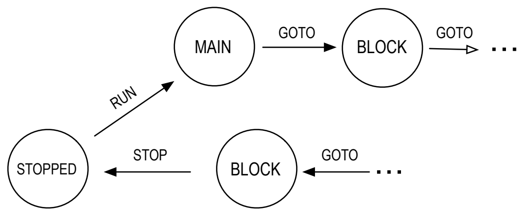

Figure 5. The structure of a radar experiment under EROS. *On the top level, an EISCAT experiment runs as a state machine. A state, or "block", typically provides a dedicated mode in the experiment, and executes in some kind of infinite loop until the state is changed to another block by either an external or an internally generated GOTO command.*

The block structure allows a limited but useful form of interactive control of a running experiment. Often one can roughly know at experiment preparation time what to expect when the experiment is run; normally, one does not need to prepare for "everything". Then the block-based form of control suffices, and is convenient to use even via the present command-line-based user interface. However, if in the future radar users would require a large part of the radar features and facilities to be at their fingertips for quick, arbitrary, interactive adjustments, a graphical command interface would probably come in handy. We comment about the possibility of an EROS graphical user interface in the last section of this report.

The bubble marked by CLOUD in Figure 4 refers to the networked software that EROS core uses to actually get access to the radar subsystems, and takes various forms. In the existing EISCAT systems, the most common entity in the CLOUD is another Tcl/Tk-interpreter-process of the same type as the ENGINE process, but running on a VME UNIX computer, which has direct access to radar hardware via the VME bus. In this case, the most typical way to access the hardware is by the cloud process executing a UNIX command-line command in the VME computer, using the EXEC UNIX system call. In the present EROS, there are a rather large number of such simple, stand-alone, one-task-only programs to control the hardware, originally developed by the EISCAT engineering teams when the hardware was originally built and tested. In these cases, the principal function of EROS is just to provide a convenient networked, scriptable interface to these well-tested commands. The C-language has been used to write the commands.

The advance in using C-language is that there is a standard way to extend the Tcl/Tk language by adding new C-coded commands. There is some processing overhead in using EXEC, and in those cases where more efficient access is needed, the command's C-code can be compiled directly into the Tcl/Tk interpreter. Either way, once a C-coded command-line command and its source code are available, it normally is straightforward and fast to make the command's functionality available in EROS. The most difficult task has often been to find a command syntax for the ELAN command that would be "natural" from the experiment writer's point of view. That view might not be the same view as the hardware engineer has, and some iteration is normally required.

Besides having a Tcl/Tk interpreter process, running in the hardware-access computer, to act as communication server for the EROS kernel, some dedicated, custom-made programs are also used in present EISCAT systems to facilitate networked access to the radar hardware. One such program is the GADGETBOX server. The GADGETBOX system allows EROS core to access the hardware by simply EXECing a certain command-line command locally, the system taking care of all the networking. The GADGETBOX was written in-house and is a rather general-purpose system for controlling hardware over local-area network, and is expected to prove useful in EISCAT_3D environment also. The GADGETBOX system uses UNIX System V STREAMS and the Transport Layer Interface (TLI) to implement the communications. Another standard approach is to use the Remote Procedure Call (RPC) networking interface. In present EISCAT, an in-house piece of software uses RPC to get access (engineering access, and on top of it, EROS level access) to the ESR 32m antenna control hardware. And for simple needs, simple solution suffices. A SOCKET communication server which it took only a couple of hours to write, running in a small Linux computer, allows EROS core communicate with the DDS system of the EISCAT_3D demonstrator array. The point here is that EROS clearly can work happily with many kind, possibly with any kind, of client-server networking solution between the EROS core and the CLOUD.

For EISCAT_3D system, it is expected that a large number, possible thousands, of networked processors will be embedded in the antenna arrays. Then the availability of network addresses may become an issue, especially, if one wants to make individual nodes accessible from anywhere in a straightforward way (for maintenance purposes, for instance). It is best that the EISCAT_3D networking be built from the beginning to use the long (128 bit) IPv6 internet addresses. But apart from the addressing issue, we do not really share the concerns expressed in WP 7 objectives, that *Both the global control and monitoring of the whole facility and the low-level control and monitoring of the thousands of distributed antenna elements within the individual arrays place large demands on the support systems and software.* If necessary, a dedicated computer, or a shallow hierarchy of computers, can be used to access the arrays' computers, with the EROS core communicating directly with the top of the hierarchy only.

6. EROS-conforming radar subsystems

We did state in the Introduction that for the EISCAT_3D the various radar subsystems should be delivered EROS-conforming, meaning that it should be possible immediately after delivery to control the subsystem programmatically from a local computer. (And we have a mild preference for that local computer to be running some flavour of UNIX.) EROS will then be able to arrange the networked access one way or another. In the minimum, the subsystem development teams should provide full control of their hardware via command-line commands. In addition, a direct C-language interface should be provided, for example, by making available the source code of the command-line commands. When the EISCAT_3D subsystems are going to be put out to tender, these requirements should be explicitly mentioned.

There is one further requirement that we ideally would like to place concerning EROS-friendliness. This concerns software simulation of the subsystem. It has turned out to be very useful both for radar experiment development and for EROS system development that EROS can run in a simulation mode on any (UNIX) workstation, without any access to the radar hardware. For some subsystems, the simulation is implemented via the very same command-line programs that implement the hardware access, by making use of dedicated simulation options of the commands. In these cases, the EROS core does not even know whether it is operating in simulation mode or in real-time mode. Obviously, the more complete and faithful the simulation is in terms of hardware actions, the more useful the simulation is for experiment development, but even a minimal "echo-type" response can already be very helpful.

During the design study, we have met both god and bad EROS-conformity of the subsystems. What we definitely do *not* mean by "programmatic control of a subsystem from a local computer" is that there merely exists a graphical Windows program that we can talk to only by typing commands at that computer's keyboard! This was the case with the DDS system used for the demonstrator array (WP 7.1). There is no way to have EROS control of such a system; if the subsystem's vendor does not provide EROS-conformity, someone else must provide it anyway. In this case, we acquired a separate Linux computer with a lot of IO lines, and started building a communication interface to the DDS chip from bit- and pin-level up. That work may still be finished, some day. For the evaluation-level DDS equipment used for the demonstrator array, the mess was both understandable and manageable. But would similar EROS-non-conformity happen during construction of the actual EISCAT_3D system, serious and unpredictable delays in C&M development could easily occur.

The correct solution was offered by the Luleå group, responsible for the demonstrator array's receiver front-end. The group provided us with a server program running in a Linux computer that had direct access to hardware, and a simple, well-thought-of and well-documented command-line command to access the server. The access command could directly by executed by the EROS core. In this happy case, it took us less than a working day to add the receiver front-end control to EROS, and that time already includes all the time needed to arrange for the EROS infrastructure of a new local radar site, in the sense of Figure 3, for the demonstrator array.

7. From EROS 5 to EROS 3D

During the design study, with the needs of EISCAT_3D in mind, EROS was updated from version 4 to version 5, by solving two critical, long-standing problems. First, the GOTO command that is required to handle an experiment as a state machine (Figure 5) was implemented in a proper way. This was non-trivial, for the underlying Tcl/Tk language, similarly with many other modern interpreted programming languages, does not directly support that kind of feature. Second, the inter-process communication was improved, to solve certain situation which could (and often did) lead to deadlocks under EROS 4. With these enhancements now in place, we have all the currently "known unknowns" in the EROS implementation under control.

One major EROS update, call it EROS 6, is needed before the system can be considered powerful enough to serve as a base-line control and monitoring environment for EISCAT_3D. The update will implement the changes outlined above (change from Figure 2 to Figure 3), which intend to make EROS boot-up automatic, and intend to offer full console access to any of the radar units available from anywhere (obviously, with some qualifications). We consider these enhancements essential for systems such as the EISCAT_3D sites that should be able to operate mostly unmanned.

There are places for other enhancements, of course. Some of those would be very visible to radar users, some less so. Foremost, it is certainly legitimate to wonder whether the command-line oriented EROS user interface, now more than 25 years old, still can usefully serve also in the new era of EISCAT_3D. A new generation of radar users might well view a command-line interface as a horribly repulsive anachronism. Aesthetics aside, typing-in a lot of commands with numerous parameters, as might be needed for handling the antenna arrays with their multiple beams for instance, could well become prohibitively cumbersome. Wouldn't it be more exciting to build something totally new?

We have sympathy for that wish, and will not rule out a general revamp of the user interface in the longer term. The command-line interface will always be maintained, but we can imagine ways that might facilitate more efficient interactive input built on top of that. One idea for a point-and-click command interface would be to elaborate on Figure 6, taken from EROS on-line documentation. The Figure shows the present EISCAT Svalbard Radar as seen by EROS. It is possible to point and click on the various radar subsystems in the Figure to produce documentation for them. Conceivably, one could use a similar strategy to produce commands instead. But if one starts along that slippery road, one is soon called for to provide the real-time flow control commands graphically, too, and then things can easily become messy. There exist graphical process control programming systems, and some (LabView) are even used for isolated applications within EISCAT; but we have doubts. Building a good graphical user interface along these lines would be a big and notoriously problematic task, and we cannot help recalling how the software landscape is littered with exciting—and expensive—large-scale projects that never came to anything useful. By sticking with the old command-line workhorse at least initially, we believe the risk of such blatant waste is minimized.

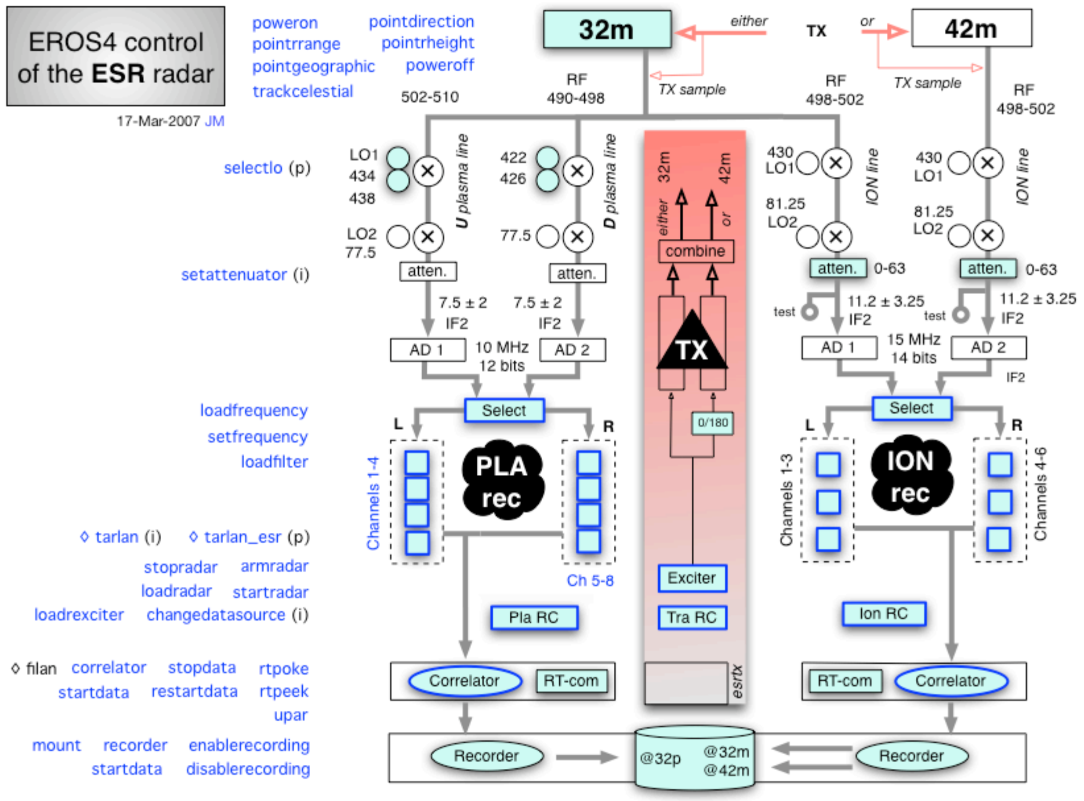

Figure 6. EROS view of the EISCAT Svalbard Radar. *The graphic, in its native form on a web page of EROS on-line documentation, allows point-and-click access to the documentation, see [http://www.sgo.fi/~jussi/eiscat/erosdoc/esr_radar.html]. Interactive EROS commands to the radar subsystems could be generated in an analogous manner.*

Rather than attempting to create a GUI for general radar control, a more realistic alternative is to provide graphical interface for monitoring only. At the moment, interactive radar monitoring involves issuing EROS commands to display the required information onto the screen. With the increased number of hardware components in the new system, having an automatically updated overview of the hardware state on the screen would be useful. Providing a GUI for such limited purpose is not a problem in principle, the Tcl/Tk system has excellent GUI facilities. An initial GUI for radar monitoring is planned together with the EROS 6 update.

In addition of user-interface enhancements, EROS would benefit of stronger use of several new enhancements that have been taking place in the underlying Tcl/Tk system in recent years, such as namespaces, threads, new data structures, and native support of non-string data types.

We expect that EROS 6 will be implemented and will start benefitting the existing EISCAT radars long before EISCAT_3D hardware enters the construction phase. Provided that the EISCAT_3D subsystems are delivered EROS-conforming as we have suggested, they can be brought under EROS control as soon as they become available. With EROS 6 in place at the time, we do not expect incorporating the new hardware to be much more complicated than what it was to bring the demonstrator array into EROS control. It might be appropriate to upgrade EROS version name to EROS 7, or maybe to EROS 3D.