Upgrade and standardization of real-time software for telescope systems at the Gemini telescopes

William N. Rambold*[a], Pedro Gigoux [a], Cristian Urrutia [a], Angelic Ebbers[b], Philip Taylor[c], Mathew J. Rippa[b], Roberto Rojas [a], Tom Cumming[b]
[a]Gemini Observatory, c/o AURA, La Serena, Chile; [b]Gemini Observatory, 670 N. A'Ohoku Place, Hilo, HI USA; [c]Observatory Sciences Ltd, William James House, Cowley Rd., Cambridge, UK

ABSTRACT

The real-time control systems for the Gemini Telescopes were designed and built in the 1990s using state-of-the-art software tools and operating systems of that time. Since these systems are in use every night they have not been kept up-to-date and are now obsolete and very labor intensive to support. Gemini is currently engaged in a major upgrade of its telescope control systems. This paper reviews the studies performed to select and develop a new standard operating environment for Gemini real-time systems and the work performed so far in implementing it.

Keywords: Telescope Control, Real-Time Software, EPICS, Software Upgrade, Software Development

1. INTRODUCTION

The Gemini Observatory consists of twin 8.1-meter diameter optical/infrared telescopes, which provide full sky coverage from their locations on Mauna Kea in Hawaii (first light 1999) and Cerro Pachón in Chile (first light 2000). Most of the control systems for these telescopes were developed in the late 1990s using the technology and techniques of that time. The control architecture for both telescopes identical, and is grouped into three general areas: Telescope Facility Systems, Instrument Systems, and Adaptive Optics (AO) Systems as shown in Figure 1.

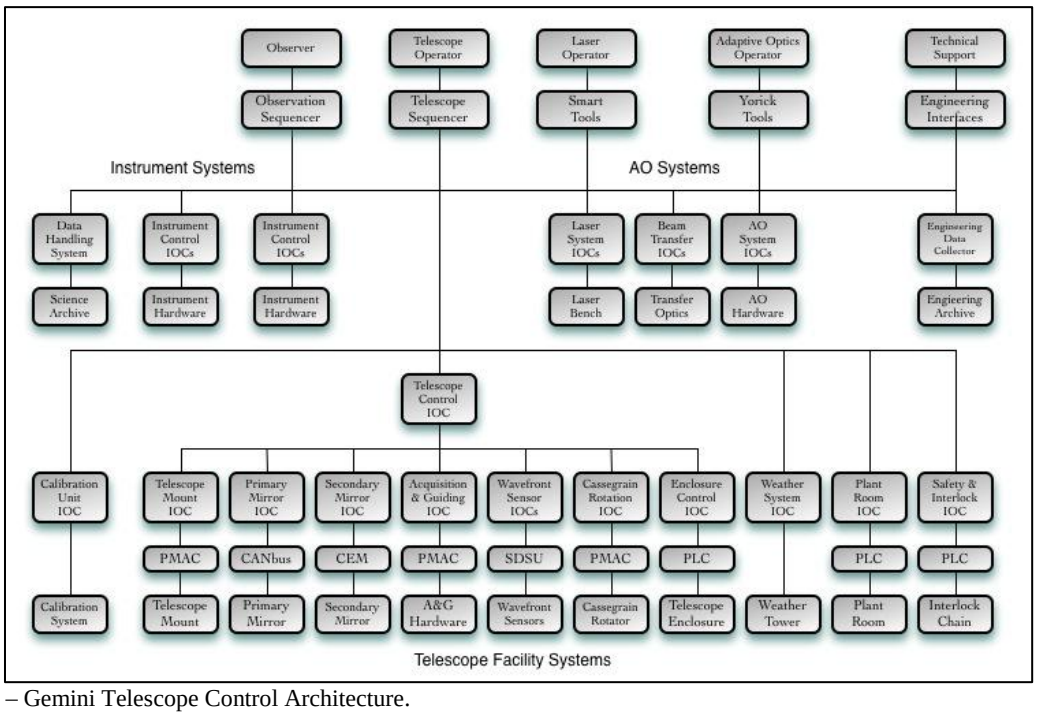

Figure 1 – Gemini Telescope Control Architecture.

*w.rambold@gemini.edu; phone 56.51.220.5600; fax 56.51.220.5650; www.gemini.edu

This architecture includes both the hardware and software necessary to operate the Observatory Control System (OCS)[1]. The Experimental Physics and Industrial Control System (EPICS)[2] was adopted as the standard control framework in which to run the facility subsystems, specifically for real-time control. A Standard Instrument Controller[3] based on Versa Module Europa bus (VME) hardware was developed to ensure conformity between externally developed subsystems.

The Gemini software model makes a clear distinction between the low level software that directly controls and coordinates the telescope and instrument hardware and the high level software that interfaces with the users and sequences this hardware to perform queue scheduled science observations. All low-level Gemini software applications are classified as real-time even though the application may not have any hard real-time performance requirements. While the high level software has evolved over the years the real-time control software has, with the exception of minor modifications to add features and fix problems, remained relatively unchanged. The real-time software development environment used to maintain these systems is likewise rooted in the 1990s. It has evolved organically over time, now consisting of many different build processes; code repositories; development operating systems; customized hardware drivers; and a distributed support code base unique to each application.

The current real-time systems are becoming more difficult and time consuming to maintain as tools become obsolete and the code and development environment variants increase. This paper describes a project that has been initiated to reduce long-term operational costs by upgrading and standardizing these real time software systems, tools and development processes.

2. FEASIBILITY STUDY

A study to determine the feasibility of upgrading the real-time telescope systems was undertaken during 2011 and 2012. This study involved gathering detailed information and statistics on all Gemini real-time systems and software development processes as well as reviewing similar upgrade projects at other institutions. The objectives of this study were to investigate ways to reduce ongoing real-time software development and support effort by at least one FTE while re-using as much hardware, code and user interface infrastructure as possible; reduce licensing costs by using open-source tools wherever possible; and improve software development efficiency by adopting new standards based on successful practices developed by the telescope and accelerator community.

2.1 Situation Analysis

2.1.1 *System Inventory*

In order to decide on the most appropriate upgrade strategy it was necessary to understand the current state of our systems. A complete review of the 40+ real-time systems was performed and the results documented for subsequent analysis. This established, for each system, a baseline record of: basic details (system description and photographs), hardware configuration (architecture; processor board; peripheral boards), software configuration (boot image; operating system and version; hardware drivers; custom records, drivers, and support modules), and an overall system assessment (stability, fault history, performance issues; and maintenance issues).

2.1.2 *Review of similar upgrades at other institutions*

Gemini is not the first facility to upgrade legacy real-time systems in an operational environment. Institutions performing similar upgrades were contacted to see what lessons could be learned from their experience.

The recent experience of the Keck Observatory TCS Upgrade project[4] was particularly relevant as their original controls software environment was very similar to Gemini, using EPICS software developed under Solaris for VME single-board computer targets running VxWorks. The VME hardware has been phased out and the Keck TCS software now runs under EPICS 3.14 on Linux PCs running RedHat Linux MRG. TDCT was adopted as the database schematic editor. The existing Motif based MEDM user interfaces were converted to caQtDM.

Diamond Light Source is a mature, operational EPICS site and has retained VxWorks IOCs for older systems[5]. Newer systems use PC based IOCs running Red Hat Linux for non real-time applications and RedHat MRG using the PREEMPT_RT patch where real-time performance is required. They use spreadsheets and a custom XML based system for database generation, although Visual DCT is used to create some templates. EDM is used for user interfaces, but there are plans to migrate from EDM to CSS/BOY in the future.

The Canadian Light Source (CLS) is one of the first major accelerator facilities to adopt a fully open source control system[6]. CLS uses EPICS 3.14, Real Time Embedded Multiprocessor System (RTEMS) running on single board computers, and Linux for development, operator interface computers and servers. A custom database tool was developed to create the underlying EPICS databases. CLS has retained EDM for most of the user interfaces.

2.1.3 *Hardware Platforms*

The choice of hardware platform has a major impact on the software architecture so this was investigated first. Each real-time system was evaluated to see if it was practical and advisable to maintain the current hardware for at least 10-years. In all cases the hardware was deemed old but stable, with no hardware related performance issues. In the past few years a concerted effort has been made to obtain spares for these systems, this has resulted in a large inventory of parts to keep these systems going. In cases of obsolete components without adequate spares a replacement strategy is already being implemented. Given the requirement to retain as much existing hardware as possible, and the investment already made in spares for these systems, the decision was made not to include a base hardware upgrade as part of the overall system upgrade process.

2.1.4 *Control Framework*

The EPICS framework for the facility has become deeply imbedded in every aspect of observatory control, from low level hardware control to the execution of observing sequences. EPICS has evolved since the early 1990's and a large and active user community has continued to support advances in both hardware, software and operating systems[7]. Many of the operational issues experienced with this control framework stem from different versions of EPICS (especially the Channel Access component) operating simultaneously on the same network. Given that EPICS is well supported and adequate for our telescope control needs the decision was made to retain it as the core real-time control framework. An upgrade of all EPICS systems to a current stable version would increase system stability and make it easier for them to be maintained current in the future.

2.1.5 *Real-Time Operating System*

When the Gemini Project was being designed and developed through the 1990's, EPICS only supported the VxWorks operating system[8]. The baseline technology decisions, namely EPICS, VxWorks and VME, were made early on to reassure project committees that the implementation was possible and based on current technologies[9]. Choosing VxWorks has resulted in initial and recurring licensing fees in excess of $100,000 USD and has imposed similar costs on its external system developers. Most Gemini systems still use older VxWorks kernels without long-term support. To upgrade to the most recent VxWorks kernel would require new development and run-time licenses, which would add in excess of $200,000 to operating costs over the next 10-years. Given that there is very little need for hard real-time performance in any of the existing systems, and that there are no plans to develop new VxWorks based systems in the future, exploring alternates to VxWorks would provide an opportunity for substantial cost savings for both Gemini and its system developers.

2.1.6 *Support Code*

The Gemini telescope sub-systems and instruments have been, and continue to be, developed by different institutions. As a result, independent copies of common device support routines, drivers and libraries were shipped with each of these systems. Over time these copies have diverged, either because they were originally customized to fit a specific system need, customized over time to add new features, or because bug fixes were only applied to some of them.

There are now many unique copies of drivers and support routines such as drvSerial, Allen-Bradley PLC, Delta-Tau PMAC, Xycom-240, VMIC, SDSU/ARC, etc. Resolving these differences and creating a common code base for all real-time applications would reduce the effort required to maintain existing systems and would provide a standard base for future development. A common code base would also allow bug fixes and EPICS upgrades to be applied to all systems in a coordinated manner.

2.1.7 *Development Tools and Processes*

The various software development tools, frameworks and processes currently in use were identified and documented to identify the issues that need to be addressed in a system upgrade.

All real-time VME-based software is currently developed on Solaris machines and the infrastructure is fundamentally based on Sun hardware. Cross development is done with the GNU toolkit and the standard VxWorks build tools. Gemini

has standardized on CentOS Version 6.5 Linux for all software development so it will be necessary to upgrade the remaining Solaris based tools to run under this operating system.

Gemini real-time applications are built using a large number of EPICS database schematics. Since EPICS is used not only for the low-level hardware interfacing functions, but for all high-level sequencing and control of each subsystem as well, extensive use is made of hierarchical schematics. All schematics are currently built and maintained using the proprietary CAPFAST tool from Phase Three Logic. To ensure availability in the long term and avoid ongoing licensing costs a replacement, royalty free Linux based tool is needed. The new tool will need to be compatible with the existing legacy of hierarchical schematic files.

Most of the telescope system user interfaces were developed using the EPICS Display Manager (DM). More recent systems have interfaces developed in Extensible Display Manager (EDM) but there are still a large number of DM screens to support. Both DM and EDM have very limited functionality and have been supplanted by newer channel access clients. If we are to keep all channel access clients and servers at the same version these tools will have to be replaced. There are a large number of engineering screens in DM and EDM, so ideally any upgrade path should be at least semi-automated.

There is currently no standard software configuration control process for real-time systems. Both the Concurrent Versions System (CVS) and Apache Subversion (SVN) repositories are used, and there are many different development paths for applications and their associated channel access clients. A variety of custom scripts, processes, and build environments are used to develop and deploy code. This lack of standardization results in developers requiring specialized knowledge about each system and increases the time needed to make even minor modifications. A standardized development framework and set of processes would resolve these issues and reduce the effort required to develop and maintain software.

2.1.8 *Summary*

The situation analysis showed that an upgrade of our current real-time systems would achieve the objective of reducing software maintenance effort by increasing system stability and standardizing systems and processes.

2.2 Proposal Development

2.2.1 *Work breakdown and analysis*

To develop an appropriate and feasible upgrade strategy the various upgrades identified above were grouped into manageable work packages as shown in Table 1.

Table 1 – Real-time system software upgrade work packages.

WBS	Description
Software Development Environment	Create a real-time software development environment based on the latest operating systems, packages and tools.
Common Code Base	Replace obsolete software packages and merge divergent software modules to create a common, stable code.
Telescope Facility Systems Upgrade	Upgrade the IOCs for all core Telescope Facility Systems to use the new environment and common code base.
AO Systems Upgrade	Upgrade the IOCs for all AO systems in the North and South to use the new environment and common code base.
Science Instrument Systems Upgrade	Upgrade the IOCs for all instruments in the North and South to use the new environment and common code base
Develop New Control System Standards	Develop real-time control software standards and architectures for future real time system development.
Project Oversight	Apply best practice project management processes to ensure project success and effective use of resources.

To estimate the effort that would be required to implement these upgrades a Work Breakdown Structure (WBS) was developed to the third level for each work package. Time estimates were obtained from the various domain experts

involved along with an estimate of the relative value of each task to the overall upgrade objectives. The results are summarized in Table 2.

Table 2 – Effort and value estimates for proposed work packages.

Work Package	Effort (hours)	Value for Effort	Outcome
Software Development Environment	960	High	Ensures a stable and maintainable infrastructure with set of tools and processes with a lifespan of at least 10 years.
Common Code Base	1400	High	All systems use the same drivers and support code, improve ability to keep systems current.
Upgrade Facility Systems	2400	High	Long term investment in stability and maintainability of core systems.
Upgrade AO Systems	640	Medium	Critical AO systems are more stable. Many upgrade aspects are already covered by other projects.
Upgrade Science Instruments	1760	Low	Older instruments are more stable and easier to maintain. Requires replacing specialized hardware and re-writing hardware dependent code.
Upgrade System Standards	1440	High	Reduced number of different system architectures to maintain in the future.
Project Oversight	1440	High	Effective planning and use of resources.

2.2.2 *Scenario development and selection*

The total effort for all work packages was estimated at 10040 hours, which equates to 6 FTE or 2 FTE per year for a three year-project, considerable more that the 0.8 FTE per year that Gemini can support with existing staffing levels. To reduce the internal effort required the WBS was re-evaluated to identify tasks that could be contracted out as a self-contained work package or performed by a term software engineer working under general guidance of in-house system experts. Based on the value and effort analyses a set of de-scope and outsourcing scenarios were also developed to see if it was possible to address at least the major issues with the resources available.

Table 3 shows the effort distribution for each of the scenarios evaluated: 1 – do all of the work in-house; 2 – do all of the work, contracting out as much as possible; 3 – remove instrument upgrades; 4 – remove instrument and AO upgrades; 5 – remove instrument, AO and standards upgrades.

Table 3 – Upgrade scenario analysis.

Scenario	Internal Effort	Contract Effort	Term Hire Effort	Total Effort
1	10040	0	0	10040
2	6648	592	3392	10152
3	4040	592	3392	8024
4	3400	592	3392	7384
5	1960	592	3392	5944

Based on this analysis all of the high value work packages could be accomplished with an internal commitment of less than 0.8 FTE over three years with the addition of a two-year term position

The outcome of the study was a recommendation to upgrade the core telescope facility systems using a modern and standardized software infrastructure, plus developing new standards for the future, over three years using a mix of in-house, contract and term hire effort.

3. UPGRADE PROJECT

The feasibility study proposal was approved in 2013 with a formal kickoff at the end of the initialization phase in January 2014. The development environment and common code base work will be completed first to provide a solid foundation for the subsequent system upgrades. The high-level project plan and current status are shown in Figure 2.

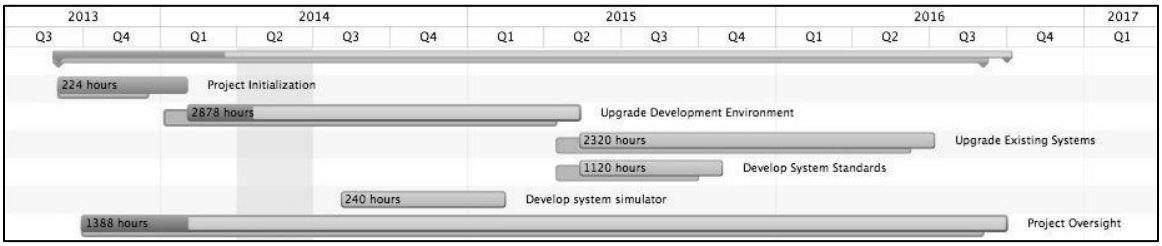

Figure 2 – Proposed real-time upgrade project.

3.1 Development Environment

The development environment work package was identified early as one that could be contracted out to an external developer. During the initialization phase formal work package and product descriptions were developed and a contract placed with Observatory Sciences Limited to deliver the package.

Work on the development environment started in February 2014 and the first conceptual design and tradeoff study review was held in April. The SDE is now in the final design stage with a review scheduled for late June. The complete environment is scheduled for delivery in August 2014.

3.1.1 *Environment Components*

The new environment consists of an interrelated set of packages and components designed to allow efficient software development, as shown in Figure 3.

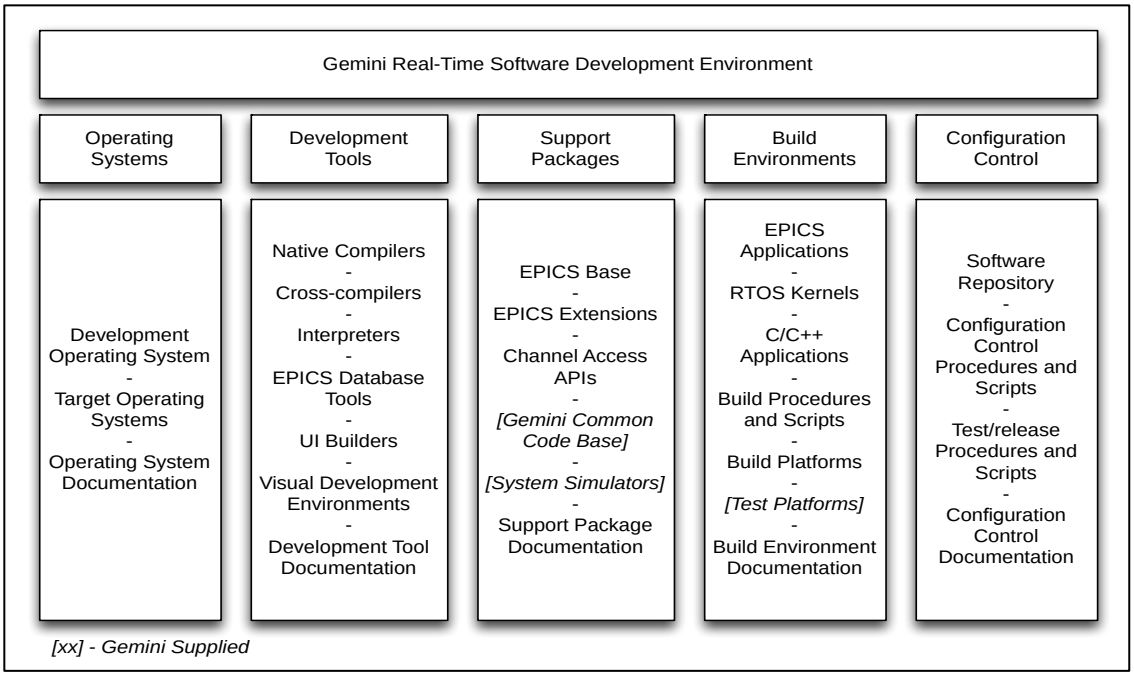

Figure 3 – Real-time software development environment.

3.1.2 *Component selection*

The feasibility study identified a number of existing components that need to be re-evaluated to create an effective environment. To assist in this process a series of trade studies were completed by OSL. This section describes the options studied and components selected for inclusion as components in the development environment, as summarized in Table 4.

Table 4 – Development Environment Component Selection.

Software Component	Current Configuration	Feasible Alternatives	Selected Option
Real-Time Operating System	vxWorks 5.4.2	vxWorks 5.5.1; vxWorks 7.0; RTEMS; QNX; Linux	RTEMS
EPICS Version	EPICS 3.13	EPICS 3.14 ; EPICS 3.15; EPICS 4	EPICS 3.14
EPICS database tool	Capfast	VDCT; TDCT; Custom	TDCT
Display Tool	DM and EDM	CSS/BOY; epicsQt; caQtDm	epicsQt
Configuration Management tool	CVS	SVN; GIT	SVN

Although many Gemini real-time systems do not have hard real-time constraints on their performance and could, in principle, be run as standard Linux EPICS Soft IOCs[10] there are still systems that do require this performance and must keep the existing VME hardware. Two alternatives to VxWorks, Real-Time Linux (including Linux with real-time patches, RedHat MRG kernel, etc) and the Real Time Executive for Multiprocessor Systems (RTEMS)[11], are directly supported by EPICS. Real Time Linux options were discounted due to the limited support available from the EPICS community for this operating system on Single Board Computers. RTEMS was chosen as the replacement for VxWorks for upgraded and future systems based on its open source model, comparable real-time performance[12] [13] and level of EPICS community support.

EPICS was originally designed to run on VME-based systems running VxWorks as the target operating system. As a result, extensive use was made of VxWorks specific features, tying the applications to this operating system. An Operating System Independent (OSI) layer has been introduced into EPICS[14], encapsulating operating system dependent functions and resources such as semaphores, threads, sockets, timers, and a symbol table to contain function and variable names. This made it possible to run EPICS on any operating system for which an OSI layer has been provided. Since there will be long-term support for the current stable EPICS version (3.14), and more recent versions (3.15 and 4) are not yet ready for operational use, it was decided to adopt version 3.14 as the baseline for Gemini systems. In the process of upgrading the EPICS version applications will be modified to use the OSI interface since the additional investment in time is justified by making them more portable in the future.

Gemini systems currently include a large number of EPICS database schematics, including extensive use of hierarchical schematics. A replacement was required for the outdated Capfast schematic editor tool used to create these schematics. The TRIUMF Database Configuration Tool (TDCT)[15], developed in Java, has been chosen to replace Capfast since it is fully EPICS aware and supports the whole development process from schematic drawing to runtime database generation.

There is no directly compatible replacement for the DM and EDM display manager tools currently in use at Gemini. Although EDM is still supported a more modern tool is required for the longer term. CSS/BOY is widely supported however the simple nature of these engineering level displays does not justify the overhead and additional complexity of the Eclipse and Java environment. Of the Qt based options studied the epicsQt tool[16] was chosen since it provides the capability of building simple interfaces without custom coding as well as the tools needed to create more complex displays.

Most real-time systems at Gemini still use CVS for version control. SVN and Git (or another similar Distributed Version Control System) present viable alternatives and provides some attractive features. The advantages of Git are only significant when used on highly distributed software projects with many simultaneous developers. Since this is not the case for real-time software development at Gemini SVN was chosen as the tool for configuration control.

3.1.3 *Build Environment*

The feasibility study identified that the lack of a standard software development framework was a major contribution to maintenance overheads. Although there are a number of configuration control and build/deployment systems in use for different systems a generic framework and set of processes could be identified as shown in Figure 4.

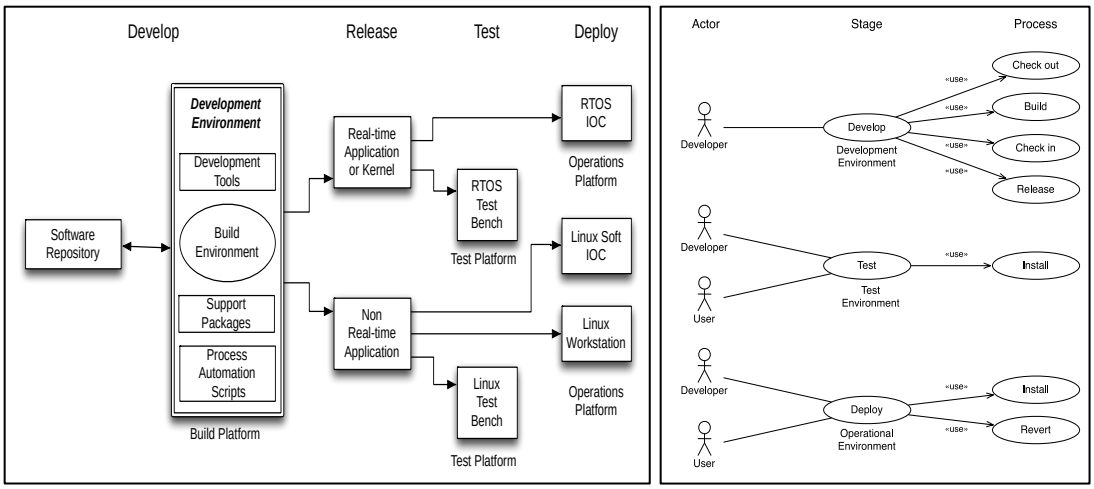

Figure 4 – Gemini software development framework and processes.

After reviewing the software development frameworks and processes at various institutions the SVN based Diamond Application Development Environment was found to be the best match to the desired Gemini software development process. Diamond's features include a comprehensive, standard directory structure, which defines the areas where different types of software modules can be accessed and built. It also provides standard locations, which reflect the software's functionality and maturity. Python scripts standardize the processes in the software development cycle. An automated build server ensures that a comprehensive set of built software is kept up-to-date. The standard EPICS Build conventions are adopted (using GNU Make), with enhancements including new rules, additional templates, macros, configuration files and consistency checking features.

3.2 Common Code Base

To create a common code base for use with all real-time systems it will be necessary to analyze the existing system software on a module by module basis to identify which have diverged, and the reasons for any divergence. Identical copies of modules will be removed from each system and put into a common library. Modules that have evolved over time because of upgrades and bug fixes will be brought up to the most up to date version and put into the library. For drivers or libraries that were customized to accommodate special system needs we propose three different strategies depending on the specific case: modify the telescope system software to remove the need of this customization; move the customization to a separate and smaller module and move the non-customized to the common library; redesign the software to make the module configurable so customization will be achieved by loading a configuration at run time.

It will also be necessary to create test cases for each module to validate the final product. A system simulator complete with at least one of each VME hardware module used in the telescope systems is currently under construction. This will be used as a test bench for the common code base work and the final system upgrades. The Common Code Base work will start in August 2014. We expect the process of defining and generating a common code base to take at least 9-months to complete.

3.3 System Upgrades

Once the new development environment; common code base and system simulator are complete the work of upgrading the telescope facility systems will begin. This effort is expected to take approximately 18-months when allowances are made for the difficulty of gaining access to the operational systems for integration and testing. The least complex systems will be upgraded and put into routine operation first, followed by the rest in ascending order of complexity. It is

expected that the process of upgrading the first few systems will be time consuming due to the learning curve involved but that the pace will pick up with experience.

A complete upgrade will consist of replacing direct vxWorks system calls with their OSI equivalents, converting the real time operating system from vxWorks to RTEMS, confirming that the CAPFAST schematics can be maintained with TDCT, and converting any remaining DM user interfaces to EDM. Any system with minimal hardware connections (such as the telescope control system) will be considered for conversion to a Linux based Soft IOC. To help stabilize the telescope control infrastructure channel access all science instruments and facility AO systems will be connected to networks isolated by Channel Access Gateways[17]. This will reduce the impact of channel access communication problems between the remaining EPICS 3.12/3.13 systems and the upgraded 3.14 systems.

3.4 Future Control Software and Hardware Standards

An additional component of the upgrade program is to bring the Gemini real-time hardware and software standards developed in the 1990s up to date. The goal of this activity will be to reduce the number of different systems to be supported by providing external developers with a set of preferred software and hardware control architectures for instrument systems. External institutions and instrument builders will be consulted in the process of defining these standards to ensure that the results reflect community best practice. In the process a prototype Linux based controller with selected Ethernet based distributed control hardware will be developed to act as a template for future systems.

4. CONCLUSIONS

The lack of a coherent program to keep Gemini real-time software current and aligned has resulted in the telescope systems becoming increasingly obsolete as well as time consuming and costly to maintain. This situation is being addressed by a systematic upgrade plan based on a thorough analysis of both the system software and development practices used to maintain it. The result of this work will be a unified software development environment, updated standards for future system development, and a collection of telescope systems that can be maintained and kept current by a smaller group of software developers.

REFERENCES

[1] Gillies, K. and Walker, S., "The Design of the Gemini Observatory Control System," ADASS V, 1996.
[2] Experimental Physics and Industrial Control System, http://www.aps.anl.gov/epics/index.php
[3] Goodrich, B., Johnson, A. and Boyer, C., "Standard Controller," Document ICD-13, Gemini 8M Telescope Project.
[4] Johnson, J., Tsubota, K. and Mader, J., "KECK Telescope Control System Upgrade Project Status," ICALEPCS 2013, San Fransisco, CA, USA.
[5] Heron, M.T., Cobb, T., Mercado, R., Rees, N., Uzun, I. and Wilkinson, K., "Evolution of Control Systems Standards on the Diamond Synchrotron Light Source," ICALEPCS 2013, San Fransisco, CA, USA.
[6] Matias, E., Berg, R., Johnson, T., Tanner, R., Wilson, T., Wright, G. and Zhang, H., "CLS: A Fully Open Source Control System," ICALEPCS 2007, Knoxville, TN, USA.
[7] Clausen, M., Kraimer, M., Hill, J., Kasemir, K-U., Korhonen, T. and Dalesio, L., "Epics – Future Plans," ICALEPCS 2007, Knoxville, TN, USA.
[8] Wind River VxWorks RTOS, http://www.windriver.com/products/vxworks/
[9] Johnson, A.N., "The Gemini Standard Controller Hardware Documentation," Document SPE-C-G023, Gemini 8m Telescopes Project (1995).
[10] Marty Kraimer et al. "EPICS Application Developer's Guide," http://www.aps.anl.gov/epics/base/R3-14/12-docs/AppDevGuide.pdf
[11] RTEMS Real Time Operating System, www.rtems.org
[12] Straumann, T., "Open Source Real Time Operating Systems Overview," ICALEPCS 2001, San Jose, CA, USA.
[13] Feng, S., Siddons, D., Straumann, T., Heron, M. and Singleton, S., "EPICS/RTEMS/MVME55000 for Real-Time Controls at NSLS," ICALEPCS 2005, Geneva, Switzerland.
[14] Kramer, M., "EPICS: Porting iocCore to Multiple Operating Systems," ICALEPCS 1999, Trieste, Italy.
[15] Keitel, R., "TDCT – A Configuration Tool for EPICS Runtime Databases," ICALEPCS 2009, Kobe, Japan.
[16] http://www.epicsqt.org/
[17] Evans, K., "The EPICS Process Variable Gateway – Version 2," ICALEPCS 2005, Geneva, Switzerland.