Gemini Program Platform

Operational Concept Document

Version 1.1 - Last updated: 2019 December 4

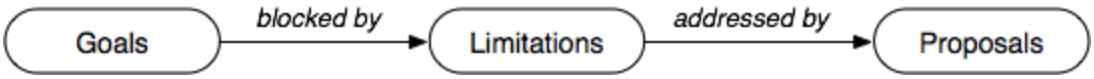
1 Introduction

This document presents operational concepts for the Gemini Program Platform (GPP) project, aimed at improving the Gemini user experience and the quality of its high-level software.

The GPP is a complete overhaul of Gemini's existing high-level software and is intended to form the foundation on which future improvements will be built.

> Gemini high-level software comprises a suite of applications for end-users of observatory facilities. The major functions considered here are proposal preparation, program elaboration, facility planning, telescope scheduling, and program execution. It does not include data reduction nor archiving.

In this document we introduce the institutional goals that Gemini is trying to achieve, and identify limitations in the current system that prevent us from achieving these goals. From there we suggest a set of technical and policy proposals that address the identified limitations and allow us to reach the goals.



We focus here on significant changes and improvements and do not exhaustively list features that one could reasonably assume would be present (proposal submission, learning resources, and so on).

2 Institutional Goals

Our primary goal is to ensure that Gemini remains relevant and popular as larger telescopes become operational. The following sub-goals are not being adequately addressed by the existing system, and will contribute to keeping Gemini relevant:

- **Improve Usability**. External PIs and staff alike experience Gemini through its proposal and program preparation tools, which should be simple to use and easy to learn. If the experience is unpleasant or cumbersome, users will be left with the same impression of Gemini as a whole.

- **Improve Maintainability**. Requirements (user-based and technology-driven) change continuously, therefore the software must be able to adapt in a reasonable timeframe with minimal disruption to end users.
- **Improve Efficiency**. More efficient operations yield more and better science. Time spent performing tasks that could be handled automatically by a computer detracts from time available for science and introduces opportunity for errors.
- **Support TDA** (Time Domain Astronomy). A key use of Gemini in the coming decade will be following up on transient object events detected by deep, wide-area, synoptic sky surveys. Reacting to these events efficiently while maintaining Gemini's traditional queue-based observing role for its partners is crucial to success.
- **Support New Instruments**. Next generation, multi-detector instruments like GHOST and Scorpio, as well as more frequent visitor instruments, need to be easier to accommodate.
- **Avoid Obsolescence**. Interfaces for external PIs must support hardware that is not under our control, so it is important to avoid technologies that risk obsolescence (such as downloadable Java applications, for which support is becoming increasingly difficult).

3 Current Observing Limitations

A summary of Gemini's existing high level software is provided in [1]. Particular areas of concern that obstruct our institutional goals are highlighted here, grouped according to their impact on external PIs, Gemini operations staff, or software maintenance. Each limitation references the particular goal(s) from the previous section that it hinders in square brackets (e.g., [**Efficiency**]).

3.1 PI experience

- **L-1** Complexity of proposal construction. [**Usability**]
    - PIs have to search multiple Gemini web pages to determine instruments and configurations that address their science goals. If they cannot quickly find what they are looking for they may start the process frustrated or decide not to use Gemini.
    - The proposal tool does not offer visualization of the science field or support for calculating integration time, so multiple, disconnected tools must be used. This significantly slows down the proposal preparation process and can lead to errors.

- **L-2** Complexity of program construction. [**Usability**, **TDA**]
    - Observations are defined primarily in terms of instrument configurations and other system details, rather than in terms of scientific objectives. It can be difficult to be confident that these technical definitions are consistent with the original scientific intent.
    - Gemini provides a template observation feature to help PIs design observations, but they can be difficult to understand and are underutilized.
    - The tool offers minimal support for undoing changes.
    - Sequence editing requires thinking in terms of programming language loops and a deep understanding of instrument features. The sequence is both a log of what happened and a forward-looking list of steps but there are no safeguards against changing what appears to have happened. It is also very easy to inadvertently change the intent of the original.
    - Help is available in the form of web pages and tutorial videos, but these are external to the application itself and it can be difficult to locate the relevant information.
- **L-3** Complexity of managing acquisitions and calibrations. [**Usability**]
    - Acquisition sequences are logically associated with their corresponding science observations but the current implementation requires that they be kept separate. This leads to more observations to manage, and additional checks to ensure that the appropriate configurations remain in sync.
    - Nothing links a science observation with associated calibrations such as telluric standards. Relevant changes to the instrument configuration in the science observation must be manually duplicated and checked in the calibration.
- **L-4** Awkwardness of native installations. [**Maintainability**, **Obsolescence**]
    - Two separate desktop applications are downloaded each semester, one for the proposal process and one for phase 2 details. Repeat users must manage multiple versions of this software on their personal computers.
    - Migrating a program to a new version requires a complicated export/import cycle with the danger of losing uncommitted changes.
- **L-5** Complexity of managing/automating time-domain observations. [**Usability**, **TDA**]

    - Template “target of opportunity” (ToO) observations have to be defined and checked in advance before they can be used to trigger new observations once the targets are known. This adds work and reduces flexibility.
    - Without a way to automate configuration updates, *n* potential configurations require *n* ToO template observations, making template preparation cumbersome and restrictive.
    - There isn’t a programmatic way to determine the open status of a Gemini telescope and whether it is accepting ToO triggers.
- **L-6** Complexity of managing inter-observation dependencies, including timing constraints. [**Usability**, **TDA**]
    - The program model cannot express logical constraints amongst observations (ordering, optionality). These are handled via free-form notes that have to be taken into account by queue coordinators and observers. These notes cannot be interpreted by an automatic scheduler and understanding them slows reaction times to ToOs.
    - Setting up timing constraints between observations requires specifying timing windows in each individual observation *after* the prior observation has finished, which makes monitoring programs clumsy.
- **L-7** Artificial site affinity. [**Usability**, **TDA**]
    - Gemini is logically a single observatory with two telescopes, but science programs are tied to a specific site.
    - Proposals that require targets or resources at both sites have to be split into two programs, making it more difficult to manage the observations.
    - Often an observation would work equally well at either site, but one site or the other must be chosen. This reduces the opportunities for scheduling the observation or requires two observations to be submitted, only one of which is needed.
- **L-8** Long Phase 2 process. [**Usability**, **TDA**]
    - Once an observation is defined by a PI there are one or two layers of manual checking to ensure there are no errors. This process can take days or weeks, making it hard for PIs to get observations in the queue for relatively urgent or changing events.
- **L-9** Managing programs with many observations. [**Usability**, **TDA**]

    - Making common updates to many observations has been a historical problem due to the time required to tediously click into every observation with the OT GUI. The template application capabilities have improved the situation but they are underutilized.

    - Due to the complexity of validating observations, a policy has been put in place that limits PIs' ability to update and manage observations once they have been checked. They must work with Gemini staff to do so, which is very time consuming for PIs and staff.

3.2 QC/Science/Ops experience

- **L-10** Complexity of queue construction. [**Efficiency**, **TDA**]

    - Queue Coordinators spend a significant portion of each day manually constructing static observing plans for a wide range of conditions for the coming night.

    - When conditions change the nighttime observer must transition between the pre-prepared static plans, which may result in missing the highest priority observations, and/or leave unused time at the end of the night.

    - Faults and interrupting rapid "target of opportunity" observations are by their nature unplanned and require on-the fly adjustments to the nightly plan. Without an appropriate tool changes can be time consuming, result in time loss, and be suboptimal.

- **L-11** Complexity of sequence manipulation. [**Efficiency**, **TDA**]

    - Once a sequence is partially executed, updating it (e.g., to repeat problematic steps or adjust exposure times) can lead to unintended changes to both the executed and remaining steps. Editing a sequence to avoid these problems is difficult and time consuming and users tend to avoid these issues by duplicating observations.

- **L-12** Complexity of time accounting. [**Efficiency**]

    - Acquisition and science sequences are unlinked and yet what happens in one sequence has an impact on how time is charged in the other. This results in inaccurate time accounting that has to be either accepted or else manually (and tediously) corrected by queue coordinators.

- **L-13** Complexity of event logging. [**Efficiency**]

    - Two separate logging systems, the nightlog and the obslog, with very different interfaces are in use. Frequently switching between the interfaces is inefficient.
    - Information manually entered into the nightlog tends to duplicate information already in the ODB and the obslog.
    - Obslog comments are tied to FITS files so it is not possible to enter comments in the obslog if no files have been generated.
- **L-14** Architecture of current system makes searching, reporting, and automation difficult. [**Efficiency**]
    - OCS stores data in binary form in an object database. Some standard reports are provided, but ad-hoc searching/reporting can only be done via custom code by the software department (with considerable difficulty due to the cumbersome data model).
    - Science staff cannot write scripts to automate OCS operations because the system was not designed with this possibility in mind.
- **L-15** Difficulties caused by site affinity. [**Efficiency**, **Usability**]
    - Moving an observation from one site to the other (possible in some cases due to instrument similarities) is cumbersome, requiring that an independent program be created at the other site.
- **L-16** Effort for Phase 2 checking. [**Efficiency**]
    - NGO and Gemini staff have to spend considerable time and effort to check submitted observations before they can be scheduled.
    - Even with Phase 2 checks and AGS some Phase 2 problems are not caught before the observations are executed, wasting telescope time.

3.3 Software Maintenance Experience

- **L-17** Complexity of modifications. [**Instruments**, **Maintainability**]
    - Codebase is monolithic, requiring a lengthy rebuild even for minor changes.
    - Large portions of the codebase are written in a style that makes verifying correctness difficult.
    - Data is stored in a custom binary format which is easy to break, cumbersome to verify, and in general unusable outside a specific build of OCS.

    - The core data model used by the database and Observing Tool (OT) is overly complex and treacherous to work with. The Phase 1 Tool (PIT) has a distinct data model mandated by an obsolete requirement[1]. The Queue Planning Tool , Laser Clearing House and Queue Visualization have distinct data models created to avoid the complexity of the core data model. Translating among these models and keeping them in sync is difficult.
    - End-user tools including the PIT and OT are built with an obsolete technology[2] that has no modern replacement.
- **L-18** Complexity of testing and deployment. [**Efficiency**, **Maintainability**]
    - Releases tend to be monolithic. In general a change to the program model requires that all of the OCS be redeployed, and that users download new versions of desktop applications.
    - Building a test release is a cumbersome and lengthy process, so test releases tend to be infrequent and contain many accumulated changes. Science staff are often under pressure to verify many changes at once, in time for a planned release.
    - Changes are not subject to automated regression testing. Before end-of-semester test releases they are usually not hand-tested by anyone other than the developer performing the change due to the difficulty of assembling a test version of OCS on one's local machine. Users typically do not see such changes until a subsequent test release, which makes the feedback cycle very slow.

4 Proposed Software Improvements

We propose the following improvements, which we feel would address the limitations described above. In each case specific limitations are referenced in trailing square brackets (e.g., [**L-1**]). These are grouped as before according to the primary beneficiaries.

4.1 PI Experience

We envision a PI experience that is more focused on science goals, allowing users to specify desired outcomes rather than requiring fully-specified instrument configurations and observing sequences. The system would encourage experimentation, allowing users to revert changes

[1] Originally the NGO / Gemini Phase 1 interface was defined to be an XML document conforming to a particular schema designed for describing observing proposals. Gemini did not host a centralized proposal database, did not dictate how proposals were created and did not facilitate NTAC proposal evaluation. It simply received NTAC-evaluated proposals in the form of XML files from the NGOs.
[2] The Java Swing desktop application user interface toolkit.

easily. API support would meet the needs of advanced users who require more automation or flexibility than end-user tools provide.

We suggest the following specific improvements:

- Replace desktop applications with a web-based application. [**L-1**, **L-2**, **L-4**]
    - **P-1** The existing desktop PI tools (i.e., the PIT and OT) would be replaced with a single web-based application that would run in any compatible web browser.
    - **P-2** User interfaces would be designed using modern conventions and follow a common design language, to promote familiarity and consistency.
- Make changes easier to revert. [**L-2**]
    - **P-3** User interfaces would have affordances that encourage experimentation, allowing the user to undo changes or otherwise revert to a previous state.
    - **P-4** All significant program edits would be logged and associated with the user, providing a history of changes.
- Add an integrated help system. [**L-1, L-2**]
    - **P-5** Context sensitive help would be available within the application to obtain relevant information quickly without searching through web pages.
- “Phase 0” search for relevant observing modes. [**L-1**, **L-2**, **L-7**]
    - **P-6** The user would specify target details, physical constraints on-target, and other relevant parameters such as desired resolution and S/N.
    - **P-7** The system would provide a list of recommended observing modes that meet the specified constraints, along with estimated execution time. The user could then select the desired modes and continue to Phase 1 if desired.
- Automated sequence generation for supported modes. [**L-2**, **L-3**, **L-8**, **L-11**, **L-12**, **L-16**]
    - **P-8** The user would specify a supported observing mode, target details, physical constraints on-target, and other relevant parameters such as desired resolution and S/N.
    - **P-9** The integrated ITC would be used to calculate exposure times, numbers of exposures and coadds.
    - **P-10** The system would generate a valid sequence that the user could further refine if desired. The system would ensure that sequences remain valid wherever possible. Warnings and errors would only be flagged, and only be necessary,

when a sequence is manually edited in a way that renders it potentially problematic.

    - **P-11** Acquisition sequences would appear in the corresponding science observation rather than in a separate observation.

- Improved program model that handles relationships among observations. [**L-3**, **L-6**]
    - **P-12** The model would permit timing constraints between observations to support cadence or monitoring programs.
    - **P-13** The user could specify that all observations/subgroups in a group be executed in a particular order.
    - **P-14** The user could indicate that only a subset of observations/subgroups in a group are required and may be executed in any order.
- Support cross-site programs and observations. [**L-7**, **L-15**]
    - **P-15** Programs could include observations destined for either or both Gemini sites throughout proposal, phase 2 preparation, and execution.
    - **P-16** Site-independent observations would be supported where similar instrument configurations exist at both sites.
- Automated calibrations. [**L-3**]
    - **P-17** Sequences would continue to support automatic calibration unit configurations based on instrument configuration.
    - **P-18** In supported configurations the system would automatically generate necessary observations for photometric, Telluric, spectral response, and other baseline calibrations picking optimal targets as necessary from catalogs.
- Open API for advanced users. [**L-5**, **L-9**]
    - **P-19** A secure web API would be provided, to allow end users to manipulate their science programs programmatically, with the intent of opening more general and advanced usage scenarios than are currently practical. It would also enable automation of time domain observations and direct access to services like the ITC and AGS.

4.2 QC/Science/Ops Experience

We envision an experience for science staff that is both more automated and less constrained, allowing observations to be executed at the right place and time to achieve better utilization of

observatory facilities. Event recording systems will be more integrated, giving better visibility and allowing better analysis of nighttime operations.

We suggest the following specific improvements.

- Automated scheduling. [**L-5**, **L-7**, **L-10, L-15**]
    - **P-20** An automated scheduler would evaluate the available observations, the current and predicted site conditions, availability of facility resources, laser clearances, etc. and propose an optimal schedule based on metrics to be provided by science staff.
    - **P-21** Observers would typically execute observations in the recommended order, but would be able to deviate if desired (if a manual plan is being followed, for example), and the scheduler would work around these deviations.
    - **P-22** Some observations might be executable at either site, so programs would no longer have site affinity. The scheduler would schedule both telescopes in tandem.
    - **P-23** The existing Laser Clearing House (LCH) application will be adapted to work with the new model.
- Single event log / timeline application for nighttime operations. [**L-13**]
    - **P-24** Nightlog and ObsLog would be combined into a single interface eliminating the need to copy/paste between applications.
    - **P-25** The system would incorporate observation execution events and links to datasets as well as weather conditions.
- Open API for operations staff. [**L-14**, **L-16**]
    - **P-26** A secure web API would be provided to facilitate database searching, usage reporting, automation, and generating alerts about observations that may need special attention.

4.3 Software Maintenance Experience

We envision a developer experience that allows changes to be integrated continuously, for a tighter feedback loop with end users; and that allows users (rather than software staff) to query the system and write their own specialized tooling via standard software interfaces. In addition, we expect to combine several disparate software representations, making it easier to introduce updates such as new instruments.

We suggest the following specific improvements.

- **P-27** Use modern technology, replacing custom-built in-memory database for standard relational database technology. Opens up query support to staff needs. [**L-17**, **L-18**]
- **P-28** Cloud-based, continuous integration and deployment. [**L-18**]
- **P-29** Programmatic interfaces to support TDA, reporting, and other ad-hoc tooling, freeing software effort. [**L-5**, **L-9**, **L-14**]

5 System Verification Process

The changes proposed in Section 4 (as well as the focus on end-user experience) will necessitate a process of continuous verification. A system of this scale must be up and available for end users (including PIs) as quickly as possible, to support continuous testing and user feedback.

We propose to run an equivalent of System Verification for instruments. Calls for test proposals would be released periodically as new modes become available or observations are needed. PIs get access to shared-risk time in exchange for exercising the system and providing feedback. They would submit proposals using the new system. Those would then be evaluated by a process to be determined. We propose that the accepted programs be given an "XT" program type to distinguish them from regular SV and make it clear that these are for validating the new system. The amount of time dedicated to this would have to balance the need of having enough observations to make the tests meaningful with not having too large an impact on normal operations.

Once we are satisfied with the design and implementation of the GPP, and it has been fully verified on the telescope, the special XT mode can be retired along with the old system. By that point proposal evaluation system processes will have been provided via a separate project that is dependent on the APIs provided by the GPP.

6 Reference Documents

1. Legacy OCS and Operations Concepts
2. OCS Upgrades Program Top-Level Vision and Concepts
3. List of Acronyms and Terms