Sharing code and support between heterogeneous telescopes: the UKIRT and JCMT joint software projects

Frossie Economou[a], Tim Jenness[a] and Nick P. Rees[a]

[a]Joint Astronomy Centre, 660 N. A'ohōkū Place, Hilo, Hawaii, USA

ABSTRACT

UKIRT and JCMT, two highly heterogeneous telescopes, have been embarking on several joint software projects covering all areas of observatory operations such as observation preparation and scheduling, telescope control and data reduction. In this paper we briefly explain the processes by which we have arrived at such a large body of shared code and discuss our experience with developing telescope-portable software and code re-use.

Keywords: JCMT, UKIRT, software development

1. INTRODUCTION: TWO (VERY) HETEROGENEOUS TELESCOPES

The Joint Astronomy Centre was created to provide support for two telescopes on Mauna Kea, the UK Infrared Telescope (UKIRT) and the James Clerk Maxwell Telescope (JCMT). However the joint support was in the engineering and administrative areas, with each telescope retaining separate science and software groups. Despite this, in the last few years, the two software groups embarked on a number of joint software projects and in 2000 formally merged the two groups into a single JAC Software Group providing development and support to both telescopes.

We think our experience may be of general interest given the increasing number of groups offering multi-telescope software development. It also has a place in the great software re-use debate that is increasingly talked about but unfortunately, rarely acted upon. We believe that using the same software on very different telescopes is the ultimate example of software re-use.

It is worth noting that UKIRT and JCMT present, in many ways, the worst case scenario for common software platforms. UKIRT is an equatorial-mount infrared telescope whose 4-metre mirror lights imagers and spectrographs; JCMT is an Alt-Az sub-millimetre telescope whose 15-metre antenna feeds receivers and a bolometer array. UKIRT is currently scheduled classically by a single Time Allocation committee; JCMT is flexibly scheduled [1–3] from four international TAGs. Moreover each of these telescopes has operated independently for a long time - UKIRT is 22 years old and JCMT has been operating for 12.

Technically and even operationally these telescopes are about as different as they can be while remaining single-dish ground-based telescopes. Yet almost every software system outside low-level instrument control is now using a common code-base. This includes observation preparation, scheduling and observation selection, data-reduction and even the telescope control system.

Why we did this, how we did it and how come we got away with it are the questions this paper intends to answer.

Further author information: (Send correspondence to F.E.)
F.E.: E-mail: f.economou@jach.hawaii.edu, Telephone: 1 808 961 3756
T.J.: E-mail: t.jenness@jach.hawaii.edu
N.P.R.: E-mail: n.rees@jach.hawaii.edu

Observatory Operations to Optimize Scientific Return III, Peter J. Quinn, Editor,
Proceedings of SPIE Vol.4844 (2002) © 2002 SPIE · 0277-786X/02/$15.00

2. BACKGROUND: NOBLE (AND NOT-SO-NOBLE) MOTIVATIONS

It is very easy to convince ourselves that our telescope is different. It was built before or after the other telescopes so it is different. It has different instruments built by different people from different countries. It has different operational models and different users and different funding agencies. And in the case of the new telescope on the block – well it's clearly better than the previous ones, so it must *obviously* be different. No wonder the software has to be different!

It takes a small act of faith to see a few similarities, but like in any good paradigm shift one is rewarded by suddenly seeing them everywhere. We'll discuss these in detail in the sections dealing with the individual software projects, but suffice it to say for now that although technically and operationally telescopes do differ, the astronomical process of observing is essentially the same. By focussing on what astronomers and observatories *really* want to do, as opposed to everything one could *possibly* do in principle given the capabilities of the hardware, one can abstract the process well enough to create a single suitable software design.

For us, the rewards of moving to a single code-base for most of our software packages were huge. The quality of software is significantly raised, as generic design and clean interfaces become absolutely necessary. The level of support we can provide increases as the number of code-bases decreases. The maintainability and upgradability of the software is assured given the flexibility of the design and implementation required by catering for two heterogeneous telescope. Last, but not least, the intellectual challenge of the design compensates our software engineers for the reduced amount of coding needed.

There are also real benefits for our user community. The age of the single-wavelength astronomers are over. Today's researchers apply for time in a wide variety of international facilities and are therefore increasingly less adept with the intricacies of each one. The more abstracted the general observing process becomes the easier it is to attract and retain new users who, despite their inexperience, can then use the telescope productively.

Despite all the obvious advantages we have alluded to and of which we are now completely convinced, honesty compells us to reveal that we embarked on this path partly because we did not have enough qualified effort to continue developing separate applications for each instrument and/or telescope. Necessity, it seems, is still the mother of invention. Our logistical problems helped secure one of the crucial elements in our joint enterprise: management backing. More on this in our conclusions.

3. OBSERVATION PREPARATION: THE OBSERVING TOOL (OT)

A common preparation tool was one of those areas that seemed the most daunting at first inspection due to the fundamental differences between UKIRT and JCMT. Nevertheless, the process of completely specifying a block of observations to be carried out comprises of a small number of communal elements: defining configuration information for the telescope, the instrument, the scheduler and the data reduction systems. Additionally, a sequence of events that causes data to be taken is required.

An important issue here is that the software's interface to the observer needs to be couched in scientific terms. Technical information should be derived from the scientific information as much as possible. For example, it is much preferable to ask for the magnitude's of one's source and derive from that the exposure time, rather than just ask for an exposure time. It is better to ask for a wavelength range that a filter name. An expert user should be able to override the software's derived values where that is sensible, but the naive user should be able to define a sensible observing program with only a basic understanding of the mechanics of observing. Libraries of pre-prepared observations that match the recommended observing modes can go a long way toward reducing the apparent complexity of the preparation task. Ideally the researcher will start by selecting one of those standard observing scenarios and only have to change the target information and integration information. On the other hand, the full complexity that allows an expert user to specify a novel observing mode is available for those able to use it.

The UKIRT-OT and JCMT-OT[4] consist of a single codebase that is the successor to the ORAC-OT[5–8] which in turn was based on the Gemini Phase II Observation Preparation Tool.[9] Unfortunately, our codebase and the Gemini one have diverged for non-technical reasons. UKIRT uses the OT for all observing preparation; JCMT has recently introduced the OT for SCUBA[10] observing and will extend its use to the heterodyne receivers in 2003 with the arrival of ACSIS.[11]

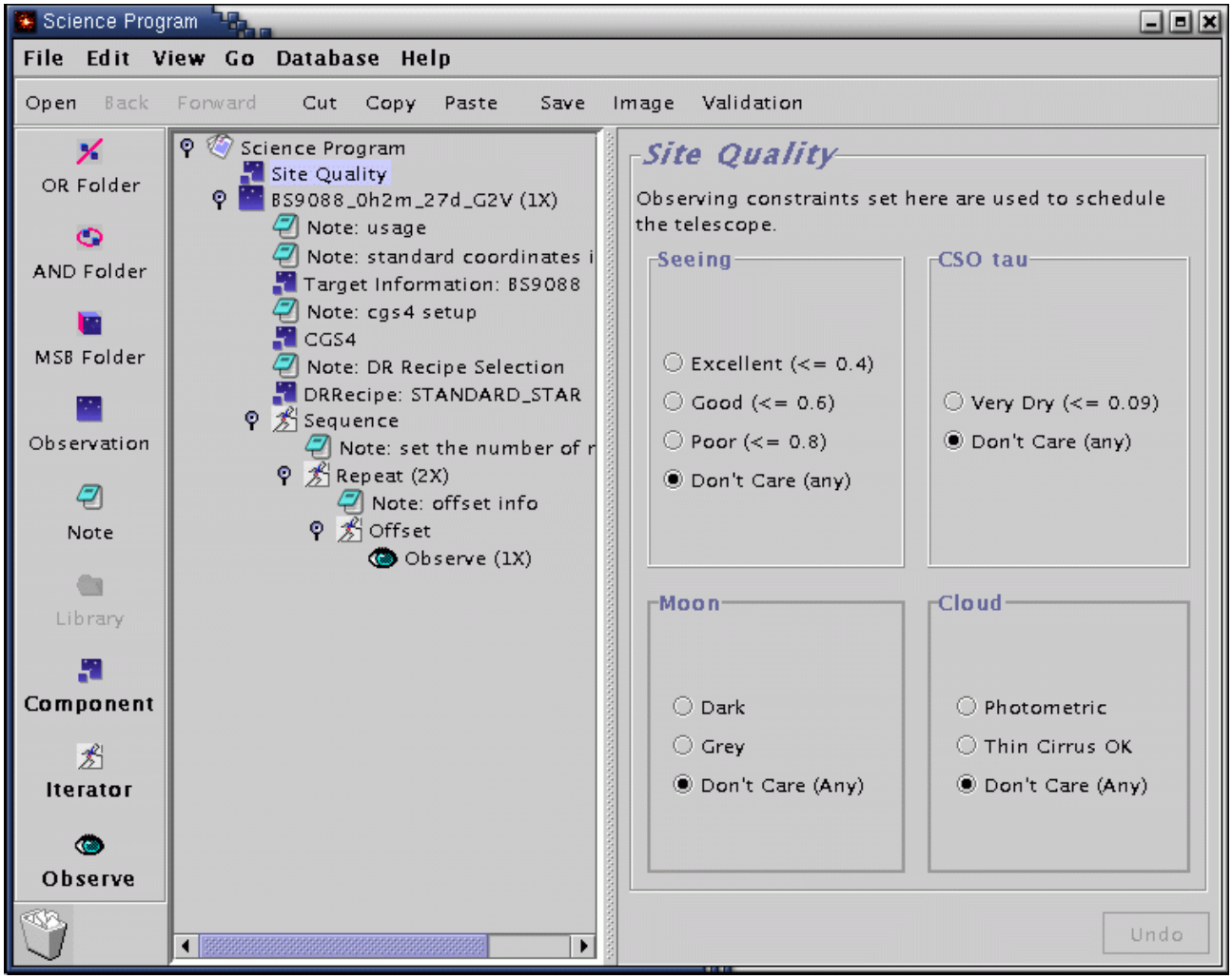


Figure 1. A UKIRT science program in the OT

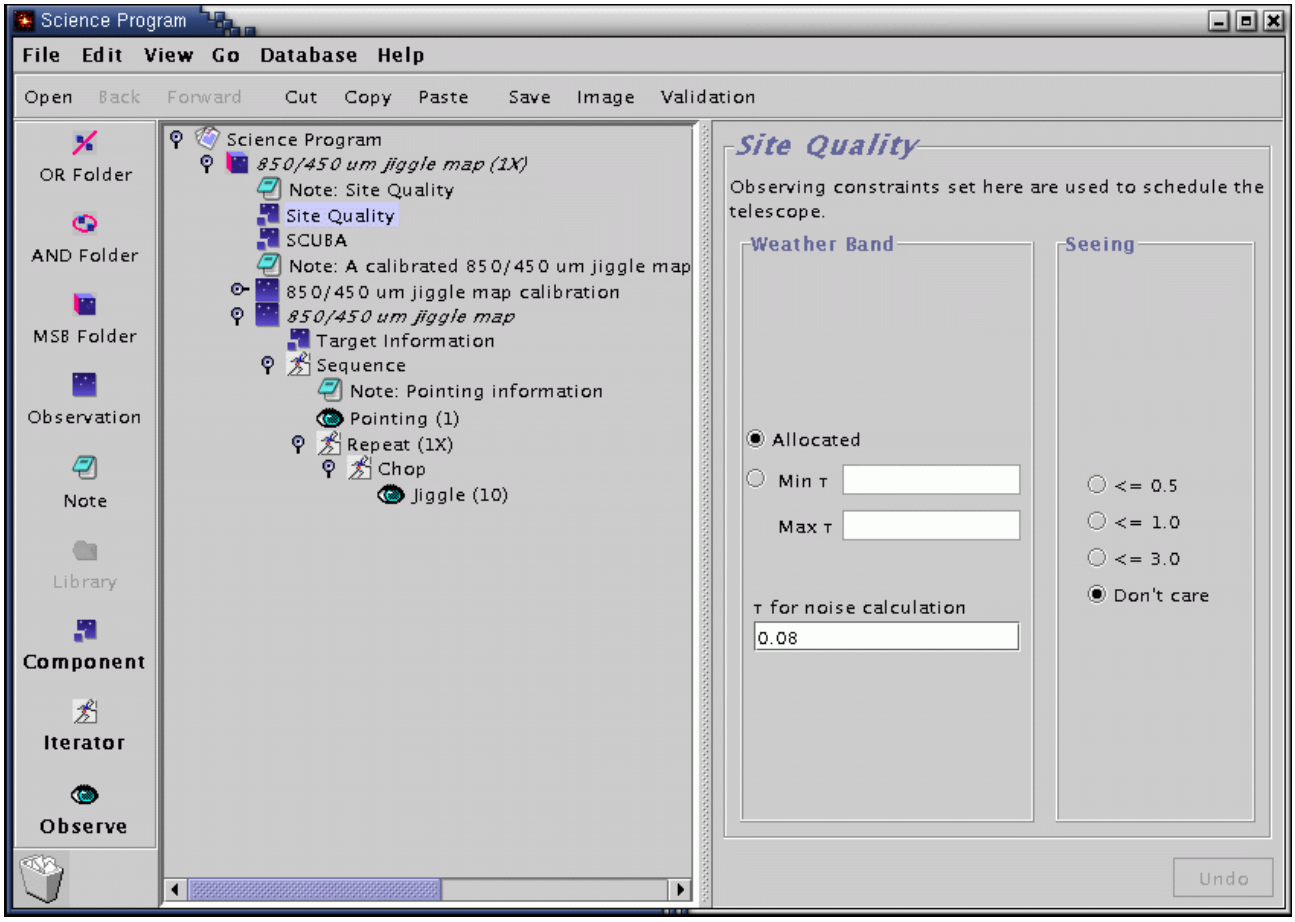


Figure 2. A JCMT science program in the OT

4. OBSERVATION STORAGE: THE MSB SERVER

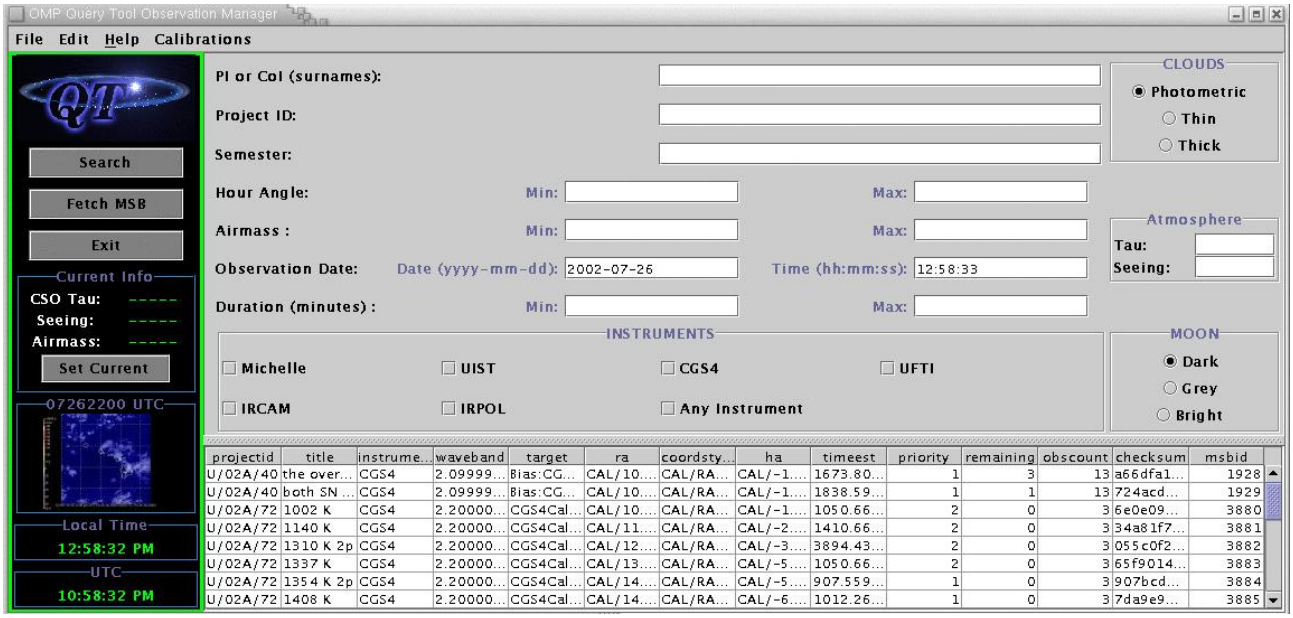
In a flexibly (aka queue) scheduled facility, there is an obvious need to turn project specifications into smaller blocks of observations that can be scheduled at the telescope. We call these Minimum Schedulable Blocks (MSB) and they consist of observations on the science target as well as any calibrations required to fully reduce the data. Of course the detailed contents of these MSBs are instrument-specific. However for the purpose of scheduling the telescope, only certain items of information are important, such as the position of the target, required instrument and condition-related requirements such as water vapour, seeing etcetera.

When a researcher uses the OT to submit a prepared program into our database it is received by our MSB server whose task it is to split it into individual MSBs, extract schedulable information which is then entered in a relational database for easy searching and store the MSB for later retrieval.[12] It can also perform astrometric calculations based on the schedulable information, such as whether a source is visible.

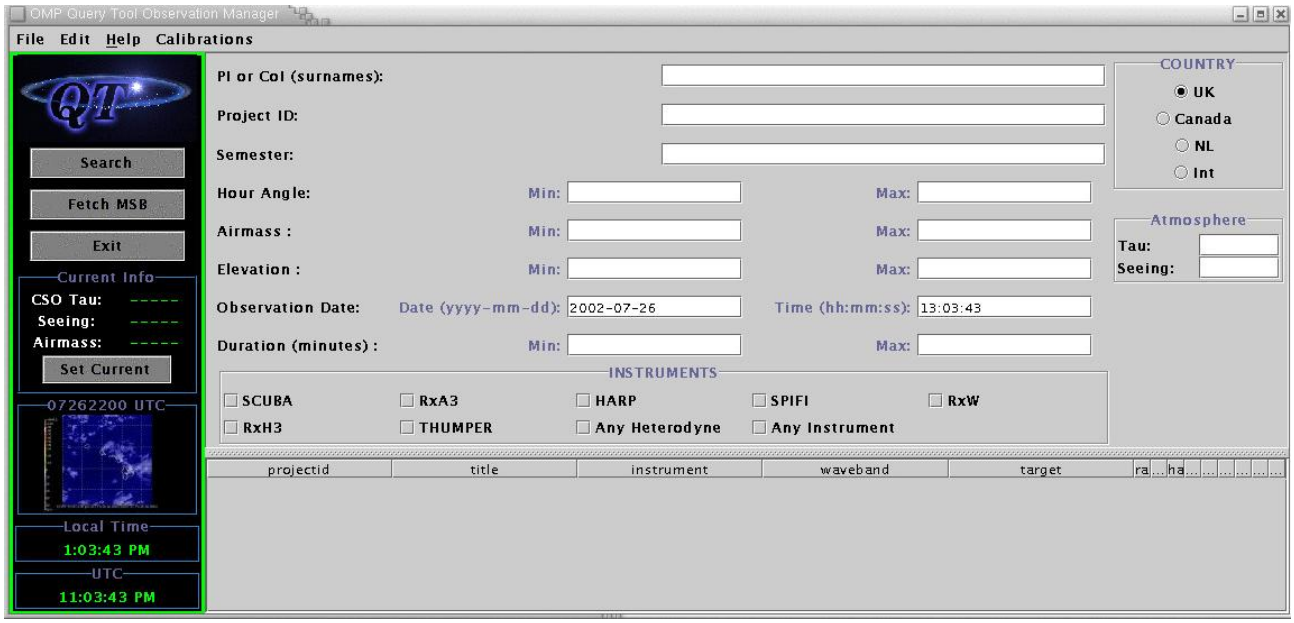
There is a single MSB server for both UKIRT and JCMT which is used for all observations prepared with the OT.

5. OBSERVATION RETRIEVAL AND PLANNING: THE QUERY TOOL (QT)

Having prepared and stored one's observing blocks, the observer at the telescope needs some way of retrieving them. The QT[13] is essentially a simple one-at-a-time scheduling tool, using information entered by the observer and/or derived automatically (such as current water vapour conditions and sidereal time) to query the MSB server for a list of suitable



(a)



(b)

Figure 3. The Query Tool for both (a) UKIRT and (b) JCMT

observations to be carried out at that time. The MSB returns such a list ranked by observatory specified criteria, such as scientific priority and project completion status.

Additionally, once an MSB has been selected for observation, the QT provides a telescope-agnostic screen that allows the observer to remove or stack calibration observations. When the observer is satisfied the MSB is then submitted to the telescope specific data acquisition system.[14]

The QT consists of a single codebase for UKIRT and JCMT, though the search GUI screen obviously looks different for each telescope. The QT is used to retrieve all observations submitted to the MSB server.

6. GETTING THERE: THE TELESCOPE CONTROL SYSTEM(TCS)

The merged JCMT and UKIRT telescope control system is presented elsewhere in this meeting.[15] In many ways the TCS is an area where we see software duplication in the extreme. The core of a TCS is to the astrometric calculations to get from celestial coordinates to mount coordinates, and that really doesn't change much from telescope to telescope. However, there are nearly as many different TCS's as there are telescopes – and the JAC was no better until recently.* However, both telescopes now run largely the same control system, with the core of the system being the Portable Telescope Control System (PTCS) written by Jeremy Bailey of the Anglo Australian Observatory.[16] Underneath this is a telescope-dependent hardware interface layer, but that still uses many shared components within the EPICS real-time database system.[17]

Once again, the more we focus on true astronomical requirements, the more obvious the solutions become. The system utilizes the Virtual Telescope concept developed by Pat Wallace for the Keck Telescope.[18] Therefore, the autoguider simply becomes another telescope with a slightly different set of pointing transformations. Also, the main difference between Alt-Az and equatorial telescopes from an astronomers perspective is the limitations imposed by cable wind-up and de-rotation. Hence, the approach we have adopted is to append an expected track duration to the slew command. This information is known because of scheduling requirements, and the telescope control system can then work out the shortest slew distance that satisfies the track time requirements.

7. PI ACCESS: THE OMP FEEDBACK

Another suite of applications fall under the umbrella of researcher feedback. Once again, irrespective of the actual specific details of their observing programme, there is a large amount of book-keeping associated with flexed or service schedule queues. These involve allowing the PI to monitor the status of their submitted MSBs, give them access to the eavesdropping facility,[19,20] retrieving the data, and keeping them informed of any discussion or faults relating to the project as well as providing them access to data quality information calculated by the data reduction pipeline.

Observatory staff, especially queue managers and support scientists, also can use these integrated tools to monitor the status and activity of their projects or queues. This integrated feedback system is also of great interest to archive users, who in the future will also be able to access the information stored in our feedback system relating to the project which generated the data they are interested in retrieving. This addresses the very serious concern about the unknown quality of data retrieved in data mining research.[21]

There is a common feedback system which is used both by UKIRT and JCMT staff and researchers.

8. DATA REDUCTION: ORAC-DR

A near-real-time data reduction pipeline is a requirement for every modern astronomical facility. It allows immediate quality assessment, flexible observing strategies and frequently results in publication quality data.

There are in fact two layers of communality in data reduction pipelines. On the outer layer there are the mechanics of pipelining: detecting the arrival of data, deciphering data processing instructions encoded in the header, making header information available to the data processing code, dealing with file-naming conventions, keeping track of suitable calibrations, displaying data and statistics to the user etc. On the inner layer are standard astronomical algorithms: adding and

*In our defence, the difference between the requirements of the JAC telescopes is about as large as you can get for two ground based single-mirror telescopes – the JCMT being an Alt-Az sub-millimetre system needing tight synchronisation for raster scanning, and UKIRT being a polar mounted optical/IR system needing all the paraphernalia associated with autoguiding.

subtracting images, detecting sources, mosaicking frames and so on. So the instrument specific code merely occupies a relatively thin middle layer.

Our design reflects this by implementing an infrastructure that takes care of all outer layer functions. Any instrument-specific requirements are handled by inheriting sub-classes (for naming conventions) or simple configuration files (such as what constitutes a suitable calibration).

In the middle layer, we have packaged up instrument-specific instructions in data reduction recipes containing primitives, i.e. code packages that perform a single astronomically-meaningful step (such as flat-fielding).

In the inner layer, we practice what we preach by driving existing astronomical applications (such as CCDPACK [22] and SURF[23, 24]) to carry out individual mathematical steps. In practice any package with a calling scheme can be used as the ORAC-DR algorithm engine, but for reasons of robustness we only allow those with good error handling facilities.

ORAC-DR[25, 26] is currently in use at UKIRT[27, 28] for the reduction of data from the mid-IR imaging spectroscope MICHELLE; the IR imagers UFTI and IRCAM; the near-IR spectrograph CGS4; and the imaging spectrometer UIST, which also contains an integral field unit. In the future it will be used for the wide-field IR survey instrument WFCAM. ORAC-DR is also used on JCMT for the sub-millimetre bolometer array instrument SCUBA.[29, 30] It has also been adopted by the Anglo-Australian Telescope for use with their imaging spectrometer IRIS2.

9. CONCLUSIONS: HOW WE GOT AWAY WITH IT

It's not hard to write software. Writing good software however, especially when trying to serve multiple masters with limited resources, is a lot more difficult. It is vital to give one's users what they need to do their jobs; they will not be impressed by impassioned lectures on software re-use if they feel the software is limiting their legitimate activities.

These are the issues we now feel are critical when developing software portable to multiple telescopes. Note that we do not claim to have always gotten these right!

Project Architect(s): Every project needs at least one person who can carry the overall system in their head, negotiate issues between developers and users, enforce the project philosophy and allow design goals to evolve in light of experience while preventing developers from straying away from those goals. It is a commonly held misconception that this is the role of the project manager and indeed in some of our projects it may be that the architect and the manager are the same person. But the roles are very distinct, chiefly in that the architect has a real technical role and should have at least an academic understanding of the coding issues, if not actually program on the particular project themselves. Ironically, given the current insistence on project management in astronomical circles, it is the architect who critically determines the success of a project. Anyone who remains unconvinced need only look as far as the Open Source development community for an extreme demonstration of the point: Linux is one of many examples of successfull projects that have strong architects but no real project management. A good project manager is necessary to bring in the software within time and budget, but a good architect will ensure that the software is worth having in the first place.

Constant feedback with users: Software projects need at least one "friend", that is a non-software person whose role is to participate in the design and rule on questions of usage. It does not matter whether that person is formally recognised as an internal or external project scientist, as long as they are willing and active participants. It is certainly not sufficient for the project friend to receive project reports once in a while. They need to be involved in the design process not only because they need to understand it well enough to contribute effectively throughout the development cycle, but also because it is they who are best placed to communicate the software philosophy to their colleagues and community. We estimate that it takes at least 20% of a person's time (and 100patience sometimes) to be a good software project friend so they need to be explicitly budgeted into any plans.

Real understanding of observatory processes. One really needs an understanding of the processes in an observatory not only in order to fulfil requirements but perhaps more importantly, in order to know which corners to cut. When developing software for heterogeneous telescopes, especially with limited staff effort, one really has to be rather brutal about requirements for exotic features. A useful guide for software requirements is that if something happens 90% of the time it should be easy to do; if something happens 9% of the time it should be possible to do; and if

it happens 0.9% of the time it is acceptable to make people get up and go poke at something to make it happen. At least one person on the project needs the ability to determine where a feature falls in that spectrum. Now this may be counter-intuitive, but such a policy promotes *more* user-pleasing "bell-and-whistle" features since more programming effort is spent on a constrained number of usage scenarios rather than on obscure and little-used functionality.

Squeaky clean interfaces. It is vital to have clean, language independent interfaces between components. This is so that telescopes can interface their own specific sub-systems to the shared applications and also so that components can be replaced as required. Another advantage that should not be overlooked is that it allows software engineers to write collaborative applications in their programming language of choice. While the choice of interfaces does not really matter, we mention as an example that we made use of XML, SOAP, DRAMA and HTTP to communicate between our various components which are written in perl, Java or C.

Distributed development is evil though unfortunately sometimes a necessary one. Good communication is difficult in any project with physically scattered programmers, but more so in writing software with multiple end-users in mind that requires a constant cross-check of implementation details. Despite extended use of the modern collaborative tools (mailing lists, telecons and so on), it has been in face-to-face visits that we made the most progress in distributed projects. One hour in the pub is worth a thousand e-mailed words. A generous allowance of time and money for the travel overhead should always be made in such projects.

Senior management backing: It is true that more than one software team has wished, often not without cause, that their entire senior management would disappear to the outer moons of Saturn until delivery is complete. Nevertheless, in joint software projects, senior management backing is essential through the critical period in a project's life where the users' fear of losing control (which frequently accompanies discussions of software reuse) outweighs the seemingly obscure technical advantages. User education by the software team is also crucial in this phase.

Battling the lack of coolness factor: There is possibly no less glamorous expression in software engineering than "Software Reuse".[25, 31] Like data-mining an archive instead of obtaining new data, it is a process that is thought worthy but dull. Software engineers enjoy (and are funded) to write software, and may regard software re-use as threatening. Nothing could be further from the truth; despite our committment to re-using existing software whenever it is appropriate, our own development activities are as vigorous, if not more so, than they have been in the past and in comparison with similarly-sized groups. By being released from the need to re-invent the wheel, we have instead been able to spend programming effort in areas traditionally outside the scope of a small telescope support group that have been a lot more interesting to engineer than re-implementing routine operational software (not to mention more useful to our community). We also feel that a lot of the fear of other people's code originates in a lack of understanding of its internal structure and that technical workshops where programs are discussed in terms of their design rather than their look-and-feel would go a long way toward remedying this. When we have held such workshops for interested parties, we have sparked off fruitful collaborations as people see that the existing software provides a good infrastructure on which to build on, rather than a substitute for any coding on their part.

One could argue that the above list represents good practice in any type software development and that is certainly true. We would argue, however, that in developing effective software for heterogeneous telescopes or other examples of disparate user bases, good practice is obligatory.

ACKNOWLEDGMENTS

Due to the nature of this paper, the author list consists of the architects of the UKIRT-JCMT joint software projects. Many more people contributed code and useful discussions without which the projects would not have been complete or successfull. Their names can be found in the References under entries for the individual projects. Some of the components discussed in this paper re-used components of the UKIRT ORAC project, [5] whose architects were Alan Bridger and Gillian Wright, now at the ATC. The joint systems discussed in this paper were developed and delivered as part of the JCMT OCS project[32, 33] and the joint UKIRT-JCMT OMP project.[12]

REFERENCES

1. R. P. J. Tilanus, "Queue Scheduling at the JCMT: the Real-life Experience," in *ASP Conf. Ser.: Astronomical Data Analysis Software and Systems IX*, **216**, p. 101, 2000.
2. G. D. Watt, "Submillimeter flexible scheduling with the JCMT," in *Proc. SPIE, Observatory Operations to Optimize Scientific Return, Peter J. Quinn; Ed.*, **3349**, pp. 126–134, July 1998.
3. E. I. Robson, "Lessons learned from four years of queue flexible scheduled observing with the James Clerk Maxwell Telescope," in *Proc. SPIE, Observatory Operations to Optimise Scientific Return III, Peter J. Quinn; Ed.*, **4844**, Aug. 2002.
4. M. Folger, A. Bridger, W. R. F. Dent, K. B. D., A. J. Adamson, F. Economou, P. Hirst, and T. Jenness, "A new Observing Tool for the James Clerk Maxwell Telescope," in *ASP Conf. Ser.: Astronomical Data Analysis Software and Systems XI*, **in press**, 2002.
5. G. S. Wright, A. B. Bridger, D. A. Pickup, M. Tan, M. Folger, F. Economou, A. J. Adamson, M. J. Currie, N. P. Rees, M. Purves, and R. D. Kackley, "An Observer's View of the ORAC System at UKIRT," in *ASP Conf. Ser.: Astronomical Data Analysis Software and Systems X*, **238**, p. 137, 2001.
6. A. Bridger, G. S. Wright, M. Tan, D. A. Pickup, F. Economou, M. J. Currie, A. J. Adamson, N. P. Rees, and M. H. Purves, "ORAC: 21st Century Observing at UKIRT," in *ASP Conf. Ser.: Astronomical Data Analysis Software and Systems IX*, **216**, p. 467, 2000.
7. A. Bridger, G. S. Wright, F. Economou, M. Tan, M. J. Currie, D. A. Pickup, A. J. Adamson, N. P. Rees, M. Purves, and R. D. Kackley, "ORAC: a modern observing system for UKIRT," in *Proc. SPIE, Advanced Telescope and Instrumentation Control Software, Hilton Lewis; Ed.*, **4009**, pp. 227–238, June 2000.
8. A. B. Bridger, F. Economou, G. S. Wright, and M. J. Currie, "Observing Control and Data Reduction at UKIRT," in *Observatory Operations to Optimize Scientific Return*, P. J. Quinn, ed., *Proc. SPIE* **3349**, pp. 184–194, 1998.
9. K. K. Gillies and S. Walker, "Infrastructure of the Gemini Observatory control system," in *Proc. SPIE, Observatory Operations to Optimize Scientific Return, Peter J. Quinn; Ed.*, **3349**, pp. 105–114, July 1998.
10. W. S. Holland, E. I. Robson, W. K. Gear, C. R. Cunningham, J. F. Lightfoot, T. Jenness, R. J. Ivison, J. A. Stevens, P. A. R. Ade, M. J. Griffin, W. D. Duncan, J. A. Murphy, and D. A. Naylor, "SCUBA: a common-user submillimetre camera operating on the James Clerk Maxwell Telescope," *Monthly Notices of the Royal Astronomical Society* **303**, pp. 659–672, 1999.
11. G. J. Hovey, T. A. Burgess, R. V. Casorso, W. R. Dent, P. E. Dewdney, B. Force, J. F. Lightfoot, A. G. Willis, and K. K. Yeung, "New spectral line multibeam correlator system for the James Clerk Maxwell Telescope," in *Proc. SPIE, Radio Telescopes, Harvey R. Butcher; Ed.*, **4015**, pp. 114–125, July 2000.
12. F. Economou, T. Jenness, R. P. T. Tilanus, P. Hirst, A. J. Adamson, M. Rippa, K. DeLorey, and K. Isaak, "Flexible software for flexible scheduling," in *ASP Conf. Ser.: Astronomical Data Analysis Software and Systems XI*, **in press**, 2002.
13. M. Rippa, F. Economou, T. Jenness, P. Hirst, R. P. T. Tilanus, K. Delorey, and A. J. Adamson, "The Infinitely Configurable Observation Query Tool," in *ASP Conf. Ser.: Astronomical Data Analysis Software and Systems XI*, **in press**, 2002.
14. M. Tan, A. Bridger, G. S. Wright, A. J. Adamson, M. J. Currie, and F. Economou, "Graphical User Interface for an Observing Control System for the UK Infrared Telescope," in *ASP Conf. Ser.: Astronomical Data Analysis Software and Systems IX*, **216**, p. 471, 2000.
15. R. D. Kackley and N. P. Rees, "JCMT/UKIRT telescope control system," in *Proc. SPIE, Advanced Telescope and Instrumentation Control Software II, Hilton Lewis; Ed.*, **4848**, Aug. 2002.
16. J. A. Bailey and R. Prestage, "Portable telescope control system project," in *Proc. SPIE, Telescope Control Systems II, Hilton Lewis; Ed.*, **3112**, pp. 124–131, Sept. 1997.
17. Dalesio, L. R. and Kraimer, M. R. and Kozubal, A. J., "EPICS Architecture," in *Proceedings of International Conference on Accelerator and Large Experimental Physics Control Systems, C. O. Pac, S. Kurokowa and T. Katoh, Eds.*, pp. 278–282, 1991.
18. P. Wallace, "Pointing and Tracking Algorithms for the Keck 10-METER Telescope," in *Instrumentation for Ground-Based Optical Astronomy, Present and Future. The Ninth Santa Cruz Summer Workshop in Astronomy and Astrophysics, July 13- 24, 1987, Lick Observatory. Editor, L.B. Robinson; Publisher, Springer-Verlag, New York*, p. 691, 1988.

19. F. Economou, A. Bridger, P. N. Daly, and G. S. Wright, "Remote Eavesdropping via the World Wide Web," in *ASP Conf. Ser.: Astronomical Data Analysis Software and Systems V*, **101**, p. 384, 1996.
20. T. Jenness, F. Economou, and R. P. J. Tilanus, "Remote Eavesdropping at the JCMT via the World Wide Web," in *ASP Conf. Ser.: Astronomical Data Analysis Software and Systems VI*, **125**, p. 401, 1997.
21. T. Jenness, D. Bohlender, S. Gaudet, F. Economou, R. P. T. Tilanus, D. Durand, and N. Hill, "Reducing and Calibrating SCUBA Data on Demand," in *ASP Conf. Ser.: Astronomical Data Analysis Software and Systems XI*, **in press**, 2002.
22. P. W. Draper, M. Taylor, and A. Allan, *CCDPACK – CCD data reduction package*. Starlink User Note 139, Starlink Project, CLRC, 2001.
23. T. Jenness and J. F. Lightfoot, "Reducing SCUBA Data at the James Clerk Maxwell Telescope," in *ASP Conf. Ser.: Astronomical Data Analysis Software and Systems VII*, **145**, p. 216, 1998.
24. T. Jenness and J. F. Lightfoot, *SURF – SCUBA User Reduction Facility*. Starlink User Note 216, Starlink Project, CLRC, 2000.
25. F. Economou, A. Bridger, G. S. Wright, T. Jenness, M. J. Currie, and A. Adamson, "ORAC-DR: Pipelining With Other People's Code," in *ASP Conf. Ser.: Astronomical Data Analysis Software and Systems VIII*, **172**, p. 11, 1999.
26. A. Allan, T. Jenness, F. Economou, M. J. Currie, and M. Bly, "Generic data pipelining using ORAC-DR," in *ASP Conf. Ser.: Astronomical Data Analysis Software and Systems XI*, **in press**, 2002.
27. F. Economou, T. Jenness, B. Cavanagh, G. S. Wright, A. B. Bridger, T. H. Kerr, P. Hirst, and A. J. Adamson, "Infrared Spectroscopy Data Reduction with ORAC-DR," in *ASP Conf. Ser.: Astronomical Data Analysis Software and Systems X*, **238**, p. 314, 2001.
28. M. Currie, G. Wright, A. Bridger, and F. Economou, "Data Reduction of Jittered Infrared Images Using the ORAC Pipeline," in *ASP Conf. Ser.: Astronomical Data Analysis Software and Systems VIII*, **172**, p. 175, 1999.
29. T. Jenness, J. F. Lightfoot, W. S. Holland, J. S. Greaves, and F. Economou, "SCUBA Observing Techniques and Data Reduction Pipeline," in *ASP Conf. Ser.: Imaging at Radio through Submillimeter Wavelengths*, **217**, p. 205, 2000.
30. T. Jenness and F. Economou, "The SCUBA Data Reduction Pipeline: ORAC-DR at the JCMT," in *ASP Conf. Ser.: Astronomical Data Analysis Software and Systems VIII*, **172**, p. 171, 1999.
31. E. Mandel and S. S. Murray, "Other People's Software," in *ASP Conf. Ser.: Astronomical Data Analysis Software and Systems VII*, **145**, p. 142, 1998.
32. N. P. Rees, F. Economou, T. Jenness, R. D. Kackley, C. A. Walther, W. R. F. Dent, M. Folger, X. Gao, B. D. Kelly, J. F. Lightfoot, I. Pain, G. J. Hovey, A. G. Willis, and R. O. Redman, "The JCMT observatory control system," in *ASP Conf. Ser.: Astronomical Data Analysis Software and Systems XI*, **in press**, 2002.
33. N. P. Rees, F. Economou, T. Jenness, R. D. Kackley, C. A. Walther, W. R. F. Dent, M. Folger, X. Gao, B. D. Kelly, J. F. Lightfoot, I. Pain, G. J. Hovey, A. G. Willis, and R. O. Redman, "JCMT observatory control system," in *Proc. SPIE, Advanced Telescope and Instrumentation Control Software II, Hilton Lewis; Ed.*, **4848**, Aug. 2002.