Astronomical Data Analysis Software and Systems VIII
ASP Conference Series, Vol. 172, 1999
D. M. Mehringer, R. L. Plante, and D. A. Roberts, eds.

ORAC-DR: Pipelining With Other People's Code

Frossie Economou

Joint Astronomy Centre, 660 N. A'ohōkū Place, Hilo, HI 96720

Alan Bridger, Gillian S. Wright

UKATC, Royal Observatory, Blackford Hill, Edinburgh, UK

Tim Jenness

Joint Astronomy Centre, 660 N. A'ohōkū Place, Hilo, HI 96720

Malcolm J. Currie

UKATC, Royal Observatory, Blackford Hill, Edinburgh, UK

Andy Adamson

Joint Astronomy Centre, 660 N. A'ohōkū Place, Hilo, HI 96720

Abstract. As part of the UKIRT ORAC project, we have developed a pipeline (ORAC-DR) for driving on-line data reduction using existing astronomical packages as algorithm engines and display tools. The design is modular and extensible on several levels, allowing it to be easily adapted to a wide variety of instruments. Here we briefly review the design, discuss the robustness and speed of execution issues inherent in such pipelines, and address what constitutes a desirable (in terms of "buy-in" effort) engine or tool.

1. Throwing Away Software

UKIRT is a 4-meter class infrared telescope that has recently celebrated its 20th anniversary. ORAC (Observatory Reduction and Acquisition Control, Bridger, Wright, & Economou 1998) is a project to re-implement all software with which an observer interacts, from observation preparation through data acquisition to data reduction.

One might well ask why it is necessary to throw away perfectly good software. The answer lies in a variety of reasons, including the following:

- Most of the existing software is tied in to a particular (VAX/VMS) platform and is non-portable. We wish to have software that runs under a Unix platform but is also portable to other operating systems.
- We would like to capitalize on modern technologies and techniques.

- Some of the existing software is not only arcane, but has grown in such a fashion as to be difficult to support and extend.

More specifically, let's consider data reduction. In the past, UKIRT has provided on-line data reduction for its near-IR spectrometer (the instrument CGS4) using the purpose built programme CGS4DR. CGS4DR provided in many cases publication quality data while running automatically and on-line – a remarkable feat when one considers the complexity of infrared spectroscopic data analysis.

However, because UKIRT is on the brink of receiving two new instruments (the mid-IR spectrometer MICHELLE and the imaging spectrometer UIST), we are forced to contemplate dedicated instrument-specific software with some misgivings:

- Software effort to cater for what are increasingly more (and more complex) instruments on an individual basis is lacking.
- Existing software re-invent many wheels (e.g., image display) that greatly increase the body of code that must be supported by observatory staff.
- Instrument-specific packages often require great effort to adapt to new instruments, even if these have fundamental scientific similarities with existing ones.

2. ORAC-DR: A Framework for Software Re-use

We have developed a data reduction pipeline (ORAC-DR) with the particular aim of addressing the shortcomings of instrument specific software outlined above. ORAC-DR is a pipeline that drives external packages (algorithm engines) via a messaging system using data reduction recipes. This has the following advantages:

- The use of external algorithm engines for data analysis and display uses software supported by the community, thus restricting observatory effort and support to instrument-specific aspects of data reduction.
- The use of recipes (a series of instructions) and primitives (instructions for a single reduction step) to direct the pipeline provides a modular system of components, many of which can be shared amongst different instruments.
- The software effort to cater to new instruments is restricted to the scientific aspects of data reduction rather than the mechanics of it (file manipulation, display, etc)

The detailed implementation of the software design and its implementation in Perl is partly described elsewhere in these proceedings (Jenness & Economou 1999). An outline is presented in Fig. 1.

3. Re-using Other People's Software

In last year's proceedings, Mandel & Murray (1998) address the issue of software buy-in (the effort required to use somebody else's software), and discuss the concept of minimal buy-in – software that's too easy not to try out. Trying out software, however, is only a beginning. When one is prepared to put other people's code on their telescope's mission-critical machines (not to mention in front

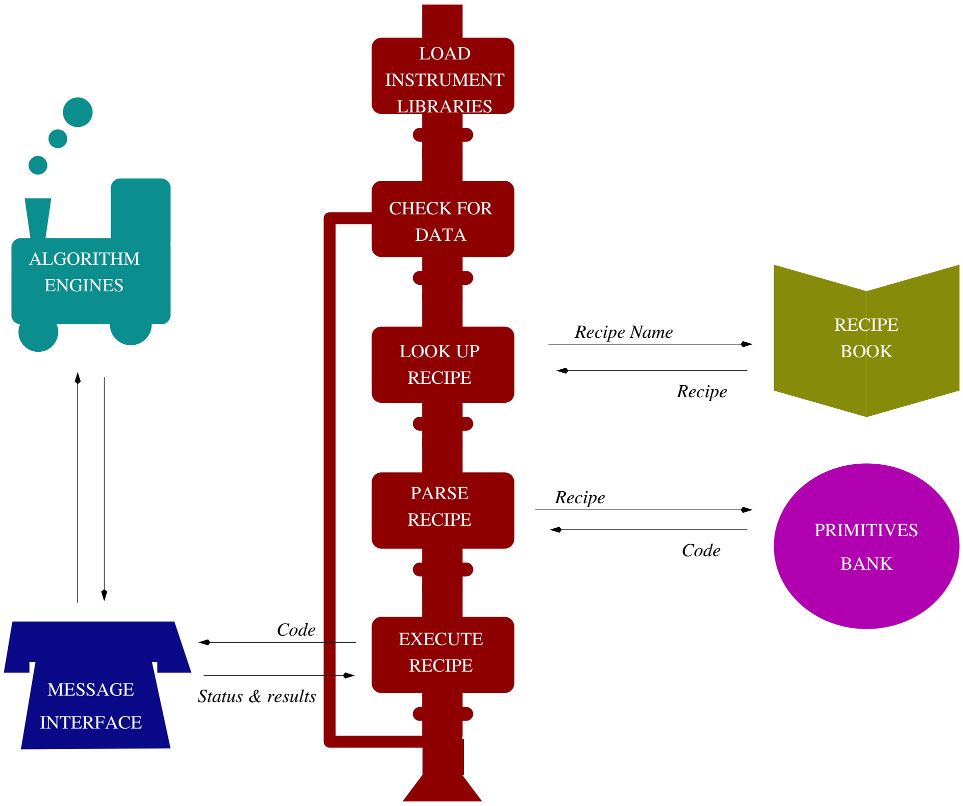

Figure 1. The components of the ORAC-DR pipeline. In our operational model, the pipeline itself is supported by the ORAC-DR development team, the recipes by an instrument scientist, the primitives by an instrument software specialist, and the messaging and algorithm engines by the community.

of funding-critical observers) a variety of issues, technical and non-technical, achieve paramount importance.

Here we discuss each in turn, providing by way of illustration some of the choices we made for the ORAC-DR pipeline. Note that we are specifically referring to software use by software developers in, or in lieu of, in-house software development, rather than end-user software consumption.

3.1. Technical Issues

By far the most important issue is robustness. When using external packages and tools, one has to not only issue a command, but also receive back parameters and a meaningful status that can differentiate between predictable errors (such as data being too noisy for an algorithm to provide a reliable result) to catastrophic errors (missing files, crashes, etc.).

The key to this is the ability to use an interface that allows reliable two-way communication between the pipeline and its external components. In practice this means:

- Algorithm engines that can be addressed via a messaging system, e.g., we use the ADAM messaging systems to control a wide variety of Starlink monoliths,
- Tools that can be addressed via a messaging system or a socket interface. ORAC-DR uses for its display the Starlink monolith KAPPA as an example of the former, and the tool GAIA (based on RTD aka SKYCAT) as an example of the latter.
- Libraries that have C-bindings. For example, we use the NDF library to directly access files for header information.

3.2. Non-Technical Issues

Mandel & Murray suggest that software re-use, or rather the lack of it, may well come down to to "money, power and control", and that "dealing with economic, social and psychological factors is well beyond the scope of this short paper" and we can only echo their sentiments. Nevertheless, there are non-technical issues that should be addressed by those willing to adopt an existing body of code. Here is some advice from our experience:

- Design: Spec your product as if you were going to write it yourself. This enables you to understand what you want (so that you can evaluate existing software against it), what you have to give up (probably not as much as you may fear) and what do you get "for free" (bells and whistles that you would not have wanted to expend the effort to put in, but that your users would like).
- Product support: When considering a certain package, evaluate the level of support – and remember that support for you the developer (who might require code changes to support greater flexibility) is a different matter than end-user support (help and bug fixing). Also beware of single points of failure – a tool supported by a single person, no matter how expert and responsive, may be left unsupported if that person changes jobs. Also beware of commercial packages and support who may raise their price tag above the level you can afford.
- Consumer savvy: Also consider local expertise. It is preferable to have a local expert in the technology that you are adopting. Furthermore, it is preferable to adopt a technology that is not arcane so that your local expert may be replaced if they leave. Using technologies that are in use outside the immediate astronomical community is one way to ensure this.

References

Bridger, A. B., Wright, G. S., & Economou, F. 1998, in ASP Conf. Ser., Vol. 145, Astronomical Data Analysis Software and Systems VII, ed. R. Albrecht, R. N. Hook, & H. A. Bushouse (San Francisco: ASP), 292

Jenness, T. & Economou, F. 1999, this volume, 171

Mandel, E. & Murray, S. S. 1998, in ASP Conf. Ser., Vol. 145, Astronomical Data Analysis Software and Systems VII, ed. R. Albrecht, R. N. Hook, & H. A. Bushouse (San Francisco: ASP), 142