United Kingdom Infra Red Telescope

Joint Astronomy Centre

Royal Observatory Edinburgh

UKIRT Data Reduction: Design

Frossie Economou, Alan Bridger, Gillian Wright, Malcolm Currie

orac008-udrd Version: 01 **Original: 11 July 1997 Modified: 2 December 1997**

This document sets out a proposed design for the UKIRT Data Reduction System, part of the UKIRT ORAC (Observatory Reduction and Acquisition Control) project.

1.0 Introduction

1.1 Purpose

This document is intended to describe and review a proposed design for the UKIRT Data Reduction System, which is illustrated by an accompanying protototype. It is intended to be read by anyone interested in the technical issues of the UKIRT Data Reduction system.

1.2 Scope

The prototyped design described here is proposed as a way of meeting the requirements made of the ORAC data reduction system. It should also highlight any issues not yet addressed by ORAC project team.

1.3 Overview

This document outlines the proposed design and its specifics as implemented in the current prototype. These are then reviewed against the detailed list of requirements set in [O3]. Finally a recommendation is made.

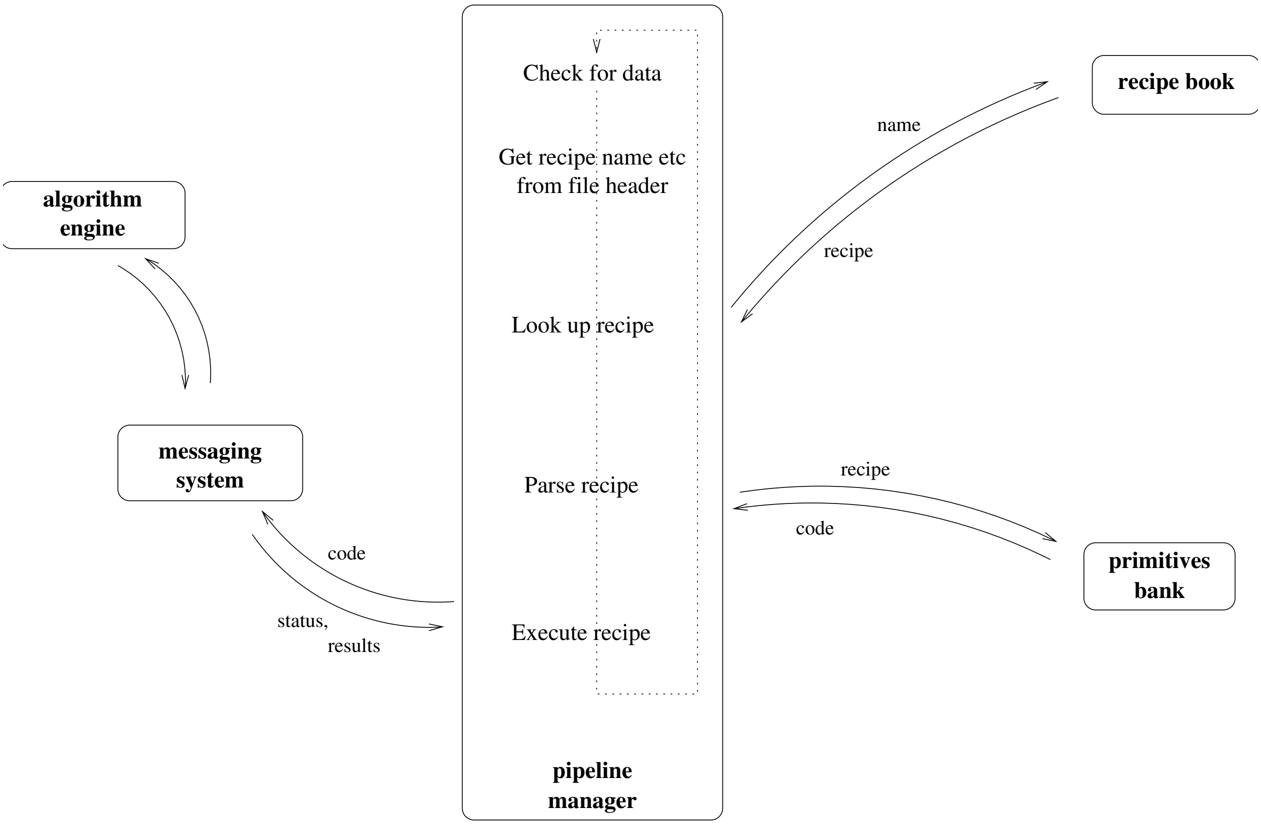
2.0 General and prototype design

Our proposed data reduction pipeline consists of five parts. These are described in the abstract and in the context of the a currently available prototype, in order to illustrate the issues involved. Note that it is considered crucial that *any* of these components can be updated or replaced without affecting the other components.

2.1 Overview

The **pipeline manager** monitors an incoming directory for the arrival of data. When it arrives, it looks in the header for the name of the data reduction recipe that was associated with the data by the preparation system. It then looks in a **recipe book** for the body of the recipe. It then parses this recipe. In some cases the recipe contains references to other recipes; eventually all references are resolved into a set of **primitives**, which contain actual code to be used in data reduction. These are then executed via a **messaging system**, which asks an **algorithm bank** to perform the data reduction commands contained in the primitives.

FIGURE 1. The ORAC-DR design



2.2 Key components

Algorithm engine . One or more generally available packages containing suitable algorithms for data reduction. This removes the burden of supporting code that is non-UKIRT specific, as well as avoiding the re-invention of the wheel. In the prototype the algorithm engine is provided by Starlink monoliths. The design of the system is such that another algorithm engine (eg IRAF) could be substituted (or simply added) transparently to the users, who are not required to be familiar with this level.

Pipeline Manager: This is the entity written in a scripting language that controls the flow of data reduction by keeping track of the arrival of data, ensuring it is reduced in a suitable manner, handling any errors that arise, keeping the user informed, bookkeeping and any other tasks that are necessary. In the prototype this is written in perl, a flexible language highly suited to the type of tasks that must or might be performed by the pipeline manager (high levels of textual processing, operating system tasks, inter-process or inter-machine communication). It is intended that the various tasks should be broken into individual modules for maximum flexibility.

2.3 Support components

The other three components of the system offer a flexible way of decoupling the algorithm engine from the pipeline control.

Messaging system: An interface between the pipeline manager and the algorithm engine, the messaging system's role is to provide a way for the pipeline manager to request and monitor the reduction of data without knowing anything about the nature of the algorithm engine involved. It is envisaged that it can by layered on top of any messaging system (eg ADAM, IRAF message bus). In the current protype the ADAM messaging system is used directly with no layering.

Recipe Book: A set of data reduction recipes that are associated with the data (eg by the data file header) and the observation sequence used to obtain it (eg by the observation preparation system). These recipes can contain references to other recipes. In the prototype design, these are text files that can easily incorporate higher features (logic, iteration etc) although it is not clear that it is desirable for them to do so. They also may have arguments, though defaults will be provided if none are specified.

Primitives Bank: A data reduction primitive is an atomic recipe, ie one that contains not other recipes but a set of actual instructions, comprehensible to the algorithm engine as to how to actually reduce data. It can map to one or more commands in the algorithm engine and/or the operating system.

3.0 Fundamental Requirements.

The review of how the proposed design fulfils the fundamental requirements set it [O-3] may take place on three levels, as to whether and how the requirements are met by :

- the overall design
- the prototype as of [Nov 25] with no more or trivial effort
- the prototype given some implementation effort

Estimates of effort are qualified as follows:

- "trivial", a few minutes to an hour or so
- "minor", a few hours to a day or so
- "modest". a few days to a week or so
- "major", a few weeks to a month or so
- "enormous", too much

and are based on FE's programming effort.

3.1 Overall Design.

As the prototype implements the proposed design, it is sufficient to examine whether the prototype fulfils the design requirements. Note that the modular nature of the design allows technologies other than the ones currently used to be substituted at a later date.

3.2 Prototyped design

To avoid excessive cross-referencing, the text of the requirements from [O-3] is reproduced here before additional comments.

FD2. [*Make extensive use of existing data reduction software (not CGS4DR for support reasons*] The prototype uses or can trivially use any Starlink package built as an ADAM monolith, including KAPPA, FIGARO, CCDPACK, CONVERT etc. These packages provide exhaustive data reduction capabilities. Other data reduction environments can be introduced with modest to major effort.

FD3. [*Extensible and flexible i.e. must separate data reduction algorithmic code from data reduction pipeline control]* The algorithm engine (Starlink software) is completely divorced from the pipeline control. The two interface only via a messaging system.

FD4. [*Must run on all Starlink supported data reduction platforms: Solaris, Linux, Digital Unix*] The prototype can in principle and by definition run on any Starlink-supported platform. It currently runs under Solaris. It runs under Linux (subject to a reported bug in the Starlink software). It has not been built or tested under Digital Unix due to lack of effort (minor).

FD5. [*Must run stand-alone*]. This requirement means that the DR system must run independently of the UKIRT OCS. Patently, it does. Minor effort is required for a semi-intelligent non-data driven front end to allow parallel-line or off-line reduction of data.

FD7. [*Must have on-line control - interruptible and configurable*] The interruptibility is a requirement that needs to be specified further - at its simplest, the pipeline can currently be terminated by normal methods (control-C, signals etc). The configurability is a requirement that the plotting can be configured in real time, and is addressed in FD17.

FD8. [*Must provide feedback ("see" scripts as they execute?)*]The prototype can provide or suppress: normal output from the pipeline control; normal output from the algorithm engine; error output from the algorithm engine.

FD9. [*Data format -tbd*] The prototype makes heavy use of the NDF format. Another major format (FITS, IRAF) could be trivially produced as a final end product or used as an initial file.

FD10. [*Capable of concurrent operation*]. The prototype can be run as many times from as many usernames (inc one) as system resources allow - but this would not be very efficient!. It is envisaged that at UKIRT it will be run once per instrument.

FD11. [*Must be robust - must handle script crashes in a sensible way.]*Primitive crashes should be handled on a case-by-case basis. Primitive crashes do not crash the pipeline control. If the pipeline control crashes (programmer error or user interrupt) it dies cleanly, it it can be run up again without problems.

FD12. [*Problems with the reduction must not affect acquisition.*] The prototype is completely decoupled from the acquisition. The only conceivable effect would be that if they were running on the same CPU they would be competing for resources.

FD13. [*Must handle both error bars and quality arrays.*] This is a primitive issue. Starlink software currently in use or probable use in the prototype do do so (error bars == variance array).

FD14. [*Must reduce data taken with different array sizes and sub-array sizes]*. This is also a primitive issue. One of the primitives in use by the prototype for example reads the size of the array from the NDF header.

FD15. [*File header of final reduced data shou ld indicate how it was reduced*]. The prototype can trivially use the NDF HISTORY mechanism for this purpose. This can record various amounts of information, but in partcicular records the commands and their parameters that have been used on a particular data file. These records can be inspected by the user via the KAPPA command *hislist.*

FD16. [*Must allow for both UKIRT observatory reduction scripts and user defined scripts*.] The current prototype uses a series of directories (observatory and user) to be searched for recipes and primitives which are in the form of simple ASCII files. If a formal database of recipes and primitives is chosen in the future, another method for fulfilling this requirement will be devised.

FD18. [*Must recognise exposure types: stare, nd-stare, chop, and handle data appropriately for bias and dark subtraction.*] Recipes and primitives can be trivially made to recognise these provided appropriate header information is available.

FD19. [*Must be extensible for new observing sequences that might be needed.*] Simply a matter for providing an appropriate recipe. In fact “new” observing sequences will be added all the time during development (the current prototype only understands one).

3.3 Current prototype plus some effort required.

FD1. *[Complete replacement of all CGS4DR and IRCAMDR on-line reduction algorithms which are needed to determine results of observing.]* This obviously requires major effort, which is available for this purpose.

FD6. [*Must run automatically i.e. must recognise and reduce an incoming sequence of observations according to a pre-selected recipe with no/minimal input from the user*]. Modest effort (minor effort from several people) is required to specify header information to be used for this. Implementing the specification requires minor effort.

FD17. [*Must auto-plot results of reduction and allow additional user specified plots].* The requirement is for a display tool that is controllable by the pipeline as well as by the user (in order that s/he can select plotting rows, control autoscaling etc). A visual display has not yet been formally adopted. One possibility is using the P4 plotting task used by CGS4DR.

FD20. [*Imaging reduction: basic functionality to be delivered with UFTI (need not be fully automated in the first instance).*] In fact this can be fully automated by UFTI delivery if the decision to build on the existing prototype is taken.

FD21. [*Spectrocopy reduction: to be delivered with Michelle.*] Assuming the full system is delivered with UFTI, Michelle DR should simply (!) be a matter of assembling the appropriate recipes and primitives.

4.0 Summary

The proposed design represented with its prototype precludes the fulfillment of no functional requirements, and particularly facilitates that of several. Many of the requirements are met, and the rest can me met with reasonable effort. It is highly modular and allows the use of other technologies to be used in its various components at a later date. The present choice of technologies can be robustly and effectively used for UFTI data reduction; therefore it is proposed that this prototype is extended to allow full testing until UFTI. After delivery of the UFTI system, a reassessment should be allowing the correcting of any problems to be made in time for the delivery of the full ORAC system for Michelle.