Improved Software Distribution Mechanisms for SKA Software

C.Williams[1] , S.Salvini[1] , B.Mort[1] , F.Dulwich[1] , O.Smirnov[2], J.Noordam[2]

[1] Oxford e-Research Centre, Oxford University

[2] Astron, Netherlands Institute for Radio Astronomy

08/06/09

Abstract

Traditionally the installation of astrophysics software tools has been the domain of experts, requiring large chunks of their time to achieve a working system. This can prove to be a huge barrier to the introduction of new and better tools, as few people have many man hours to spare to get a tool installed in order to try it out.

This paper reports on our first steps at applying a more user orientated attitude to software deployment. In particular we show how the installation of the latest version of the MeqTrees Calibration and Simulation tool is now a matter of hitting a button, a vast improvement over the previous torturous process of tracking down, tweaking, and installing its more than 20 different dependencies.

Introduction

Rarely inside an academic environment is software deployment given the thought and attention it deserves. This is primarily because the package authors simply do not have the time, motivation or expertise to concentrate on these issues, and project managers are not aware of the importance of, and the resources required, to package software.

When there are only one or two users, the effort to package software on multiple platforms can be way out of proportion to the functionality provided by the package and so can, rightly, be ignored. As the number of users increases, more effort should be made to reduce the burden of installation, as the costs in time and effort involved in packaging can be discounted against the valuable time of the myriad of users.

The SKA Approach

In order to address these issues, we have been working on a system to aid in packaging and distribution. We aim to provide packages that mix seamlessly with the platforms native package management system so that installing our software is identical to installing any other package, involving no specialist knowledge or procedures.

This takes the form of a packaging service, and an array of repositories in which the completed packages can be published. The packaging service is designed to de-skill packaging to some extent by abstracting the packaging objective from the underlying platform idiosyncrasies. For example, an application binary should be launchable from the command line on unix platforms and from the Start Menu on Windows platforms. The packaging

service should allow you simply to state that you have a binary and the service will provide the necessary technical details to do the correct thing on each target platform.

Publishing and distributing the packages is just as important as building the packages in the first place. Many packages have long lists of dependencies that must be satisfied, and these too have to be made available for easy install. Each platform expects repositories to exist in a certain format and again, this can be a considerable burden for the non-expert. Tools to provide a level of abstraction for the publishing process are therefore essential. These allow the publisher to simply state the release level and the package name and version, with all other technicalities being taken care of behind the scenes.

With these packaging/publishing goals in mind we have developed a system composed of three main components:

1. **A collection of stock platforms**. A machine for each supported platform is required for building packages as well as testing installations. The collection consists of disk images for use in a virtual machine environment as well as real hardware.
2. **A Repository Server**. We have available a server for publishing the various repositories to the outside world, as well as a more restricted server for publishing test releases.
3. **The Multi-Platform Packaging Tool (MPP).** We have developed, in house, a software tool to aid in packaging and publication as described above. Due to limited resources, this tool is still very much a functioning prototype than a fully fledged application, but even so it has achieved some impressive results.

Current Status

Despite its prototype status, the MPP tool has been used in earnest, allowing us to create and publish packages on 4 different platforms for the OSKAR beam forming simulation package [1], and the MeqTrees Calibration and Simulation tool [2]. The latter of these has a very large number of dependencies (>20) which have also needed to be packaged and distributed where they are not available on the native platform.

Installation for either of these is a case of pointing the native package manager at our server (this has also been automated by providing a special package to configure the package manager), and then simply marking the package for installation. The package manager will then download the required package, and all the required dependencies from our server and install them immediately. The user then has no more to do than fire up the application and start working.

Currently MPP can support RPM and Debian based packaging systems. Limited support is available for the Macintosh OS-X based MacPorts system. We hope to extend this support significantly in the near future to provide a full binary distribution on the Mac.

A serious limitation of our system so far, is that it requires the target machine be kept in a pristine state – i.e. an out of the box system, with nothing extra installed on it. This enables us to fully test the installation

packages for a typical user. Such environments are easy enough to achieve with virtual machines, but systems such as the Apple Mac, which do not allow the use of virtualisation, pose a problem. A workaround solution needs to be worked out and implemented before testing becomes non-trivial on these systems as on others.

Future Directions

Our current success with MPP demonstrates the feasibility of de-skilling the packaging and publishing process significantly. We can envision offering a number of services to all SKA software projects that will provide significant benefits to both users and developers. This could be provided through a web interface.

1) A Build and Packaging Service.

   Automatically building, packaging and testing on an array of target platforms on request. By providing such a service we not only give a valuable resource to projects, but also raise awareness of software quality issues in a natural way.

2) A SKA central publishing facility. Again operated through a web interface, this could provide a single source SKA software repository for each platform.

References

[1] The Oskar station simulator: https://wiki.oerc.ox.ac.uk/Oskar

[2] MeqTrees Calibration and Simulation: http://www.astron.nl/meqwikiacceptable