The cost of developing and maintain the monitoring and control software of large ground-based telescopes

Juan Carlos Guzman*[a], Gianluca Chiozzi[b], Alan Bridger[c], Jorge Ibsen[d]
[a]CSIRO Astronomy and Space Science, PO Box 76, Epping, NSW Australia 1710; [b] European Organisation for Astronomical Research in the Southern Hemisphere, Karl-Schwarzschild-Strasse 2, D-85428 Garching, Germany;; [c]UK Astronomy Technology Centre, Royal Observatory, Blackford Hill, Edinburgh, EH9 3HJ, Scotland, UK; [d]Joint ALMA Observatory, Alonso de Cordova 3107, Vitacura, Santiago de Chile, Chile;

ABSTRACT

There are several large ground-based telescopes currently under development such as the SKA and CCAT in radio as well as several 30m-class in the optical. A common challenge that all of these telescopes are facing is estimating the cost of design, construction and maintenance of their required software. This paper will present a cost breakdown of the monitoring and control software packages implemented for ASKAP including the effort spent to develop and maintain the in-house code and the effort saved by using third-party software, such as EPICS. The costing for ASKAP will be compared to those for the monitoring and control software of other large ground-based telescopes such as ALMA and VLT. This comparison will highlight trends and commonalities in the costing and provides a useful guide for costing future telescope control software builds or upgrades and the ongoing maintenance cost.

Keywords: Software effort estimate, Software costs, Software Engineering, Software Project Management

1. MOTIVATION

There are not many papers available publicly that describe in detail existing software effort and costing for astronomical applications. Kemball and Cornwell's paper [1] makes an attempt to an early estimation of the software costs for the SKA based on some preliminary ALMA costing data.

Hence, there are two main motivations to write this paper: sharing the software cost data collected for ASKAP, ALMA and VLT for use as a reference for other projects; and, collecting and using the productivity data to calibrate future estimates for similar projects within the same organization.

This paper describes the detailed software development and maintenance effort of the Australian SKA Pathfinder (ASKAP) Telescope Operating System (TOS). This sub-system is responsible of the monitoring and control software, including observation preparation functionality. The ASKAP TOS architecture and its components are described in more detailed in [2] and [3]. This paper does not report on software effort for the data processing or archiving of ASKAP.

2. THE ASKAP EXPERIENCE

2.1 Overview

Figure 1 shows the level of effort per financial year since 2008 for the development and maintenance of the ASKAP Telescope Operating System (TOS) software package. Currently the ASKAP TOS development is in the later construction stages with several modules already in maintenance mode. It is expected that construction should be completed by mid 2015. Three major stages are also highlighted in the figure: pre-construction, construction and maintenance.

The Pre-Construction stage consisted in all activities up to the Critical Design Review milestone. In addition to the documentation produced, there was significant effort in developing prototype software for a single 12m antenna at

Software and Cyberinfrastructure for Astronomy III, edited by Gianluca Chiozzi, Nicole M. Radziwill,
Proc. of SPIE Vol. 9152, 91521P · © 2014 SPIE · CCC code: 0277-786X/14/$18
doi: 10.1117/12.2055921

Parkes. This prototype is still in use today. This stage started in January 2008 and was formally completed in March 2010. The pre-construction cost is breakdown into four main activities and/or packages:

- Management and Software Engineering, comprises activities such as software project management, setting up and support software engineering tools and support for the software development environment, including development computers/servers.
- Analysis and Design. Including requirements analysis, software architecture and design, preparing all the design documentation packages for the Design Review.
- Prototype implementation using key technologies identified during the Preliminary Design Review such as EPICS [4] and ICE [5].
- Infrastructure (Common Software) software package includes: a recursive build tool built in collaboration with SKA South Africa supporting multiple compilers and build tools for C++, Python and Java applications; integration of 3rd Party software: EPICS base, EPICS support modules and extensions, ICE, casacore; and the common software (logging, common libraries) for writing High Level Components.

The Construction and Maintenance activities and software packages are listed below:

- Management and Software Engineering, and Infrastructure are the same as before.
- Observation Preparation. This only includes the software development effort for the Observation Management Portal (OMP) components, which is responsible for presenting a user interface for the creation, modification and monitoring of Scheduling Blocks by user of Guest programs or members of existing Science Survey teams. For management of observing proposal, ASKAP will (re)use existing system named OPAL. For cost consideration only the preparation part is accounted in this report.
- High Level Components (excluding OMP). These components comprises of Operator Displays, Monitoring Archiver, Log Archiver, Executive, Facility Configuration Manager, Scheduling Block Data Service, Observation Procedure Library (OPL) and Observation Script base libraries (Python).
- Local Monitoring and Control (LMC) Applications, implemented using the EPICS Input/Output Controller (IOC) software, including LMC/Engineering GUIs developed using a panel editor tool. These software components are mapped to ASKAP hardware sub-systems. By the time of writing all the LMC for the Boolardy Engineering Test Array (BETA) have been implemented. The ASKAP Design Enhancement (ADE) or Mark-II hardware subsystems LMC are still under development.

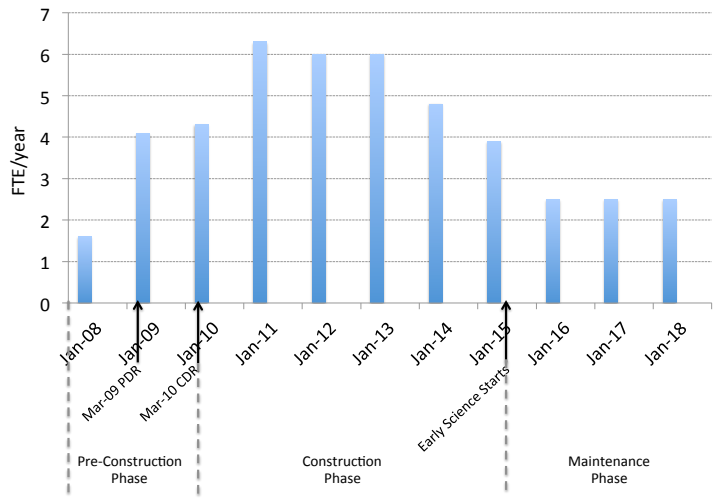

Figure 1: Staffing levels for ASKAP TOS development from Jan 2008 onwards

For the purpose of documenting the costing as simple as possible it is assumed the project phases are discrete. In reality there was some overlap between the three major stages but does not affect the total effort estimates.

2.2 Pre-Construction Stage (Jan 2008 – Mar 2010)

Table 1: ASKAP TOS pre-construction effort

Activity	FTE
Management and Software Engineering	**1.2**
Analysis and Design	**2**
Prototype (Parkes 12-m)	**4.1**
Infrastructure (Common Software)	**1.5**
Total Pre-Construction	**8.8**

During pre-construction it was decided and planned the implementation of a software prototype for use in the 12-m antenna at Parkes, equipped with a Phase Array Feed (PAF) and electronics prototypes. This prototype system served as a platform to test several key technologies such as EPICS and ICE. The system was used primarily for testing and performance measurement of the first PAF.

Implementing the prototype system included a 25% cost in learning curve, dominated by learning EPICS technologies (3 months/developer) for a total of approx. 1 FTE.

These effort measurements are based on planned effort instead of "real" effort since during this stage the project did not track real effort (including extra time). Therefore the values expressed here are approximate to the real ones.

2.3 Construction Phase (Apr 2010 – Jun 2015)

The software development life cycle used during this stage was iterative and incremental, starting with the first official release of the Telescope Operating System (TOS) software version 0.1 in Dec 2010. Since then, 21 versions have been released and deployed in the production system (latest is TOS version 1.0). Each iteration consists of ***planning, coding, testing, verification and release***, and usually last between 6 to 9 weeks depending on the user needs. The amount of effort listed in **Table 2** corresponds to the total effort from planning to release of all software components. The construction stage is not fully completed yet with several components still under development.

During the first 6 – 8 months of the construction phase, a new technology was introduced to replace the Motif-based EPICS GUI builder called EDM. The new technology called Control System Studio (CSS) [6] is based in Eclipse, which is relatively easy to use but more complex to develop and customize. The estimated CSS development learning curve is approximately 0.5 FTE per developer. Only two developers were required to learn the CSS development part for a total learning cost of about 1 FTE.

Table 2: Construction effort

Activity	FTE up to Jun 2014	FTE remaining (up to Jun 2015)
Management and Software Engineering	1.4	**0.4**
Observation Preparation (excluding proposal handling)	2	**1**
High Level Components (ICE/EPICS/CSS)	5.6	**1**
Local Monitoring and Control (LMC, EPICS/CSS)	10	**1.3**
Infrastructure (Common Software)	5.3	**0.2**
Total Construction	**24.3**	**3.9**

As of the time of writing this paper, the total effort spent so far is 33.1 FTE since April 2010 (50 months). The effort measurements expressed here are a combination of nominal and real effort. The author estimates an error of approx. +/- 5%, mainly due to the fact that during a substantial amount of time, there was no tool or process to track "real" effort (including overtime).

The total estimated pre-construction and construction is 37 FTE for the whole ASKAP TOS software. The total ASKAP project budget is AU$188M as of June 2014. The average labor cost for a software developer at CSIRO is approximately AU$200k (wages + overheads). The additional computing hardware and operating expenses is approximately AU$ 600k. Therefore the approximate cost of the ASKAP TOS development as a percentage of the total budget of approximately 4.3 %. This value is consistent with other recent large ground-based telescopes projects as described later in this paper.

2.4 Estimation of Productivity (Jan 2008 – present)

Productivity is simply calculated by software size divided by effort. The metrics for software size used in this report is Logical Source of Line of Codes (LSLOC) because is relatively easy to obtain via automated tools. For this report, the ***sloccount*** implemented by David Wheeler tool [7] has been used to determine Physical Source Lines of Code (SLOC) of the TOS software. The cost estimate handbook [8] describes a conversion table between physical and logical source lines of code depending of the programming language. The measure of effort is described as FTE-year.

Table 3: Productivity estimates for ASKAP TOS development (2008 - present)

Activity	3rd Gen Language	OO Language	Total LSLOC	Spent FTE	LSLOC/FTE year
Prototype development (including system design)	10791	33255	31372	8.8	**3565**
Observation Preparation (excluding proposal handling)		5629	3940	2.3	**1713**
High Level Components new code		45231	31661	5.9	**5366**
High Level Components reused code (Alarm System and Monitoring Archiver)		*119856*	*83899*	*5.9*	***14220***
Local Monitoring and Control (LMC, EPICS/CSS)	37344		28008	10.4	**2693**
Infrastructure (Common Software)		42190	29533	5.7	**5181**
Average Productivity (only new code)			**124514**	**33.1**	**3762**

Industry data [9] suggests that for command and control type applications a typical productivity range is 1140 to 4200 (LSLOC/FTE-year), with an average of 3000. The data collected and calculated for ASKAP TOS development falls within the average industrial productivity range.

There is great variability in the calculated productivity for different components. This is due to the fact that LSLOC does not necessarily represents the size of specific components. Moreover, and in particular for EPICS IOC development the size of the application is not directly represented by the size of the code rather it is better represented by the amount and type of records (monitoring/control points) of the sub-system. The author is currently investigating alternatives methods to better represent the size of EPICS IOC applications.

2.5 Software Maintenance (Jul 2015 – onwards)

Software maintenance typically consumes about 40 to 80% (60% average) of the total lifecycle costs of the software systems as reported in many articles ([10], [11]) and books. During maintenance, 60% of the costs on average relate to user-generated enhancements (changing requirements), 23% to migration activities, and 17% to bug fixes. The 60% of lifecycle costs related to maintenance coupled with the fact that 60% of maintenance activities relate to enhancements gives us the so-called ***60/60 rule***, one of the few proposed "laws" of software maintenance [12].

Applying the 60/60 rule to a 20-year lifetime of software products (not unheard in the astronomical community) lead to approx. 10% of the development effort that should be allocated per year to maintenance activities (if the costs of maintenance versus construction were exactly 2:1 and the maintenance lifetime is 20 years, then 10% is the exact result).

Table 4 shows the estimated yearly maintenance effort for the ASKAP TOS software during the steady-state operations of the instrument (expected post July 2015). The result is approx. 8% of the total development cost of ASKAP TOS software. The main reason of being slightly lower than the 60/60 rule is due to the fact that the infrastructure software support probably will not require major enhancements as it relies heavily on third-party software (relatively "thin") and thus it represents less than 10% of the total cost of the infrastructure/common software.

Table 4: Estimated yearly software maintenance effort for the ASKAP TOS software (July 2015+)

Activity	FTE/year
Management and Software Engineering	**0.2**
Observation Preparation (excluding proposal handling)	**0.3**
High Level Components (ICE/EPICS/CSS)	**0.8**
Local Monitoring and Control (LMC, EPICS/CSS)	**1.2**
Infrastructure	**0.2**
Total Maintenance	**2.7**

3. COMPARISON WITH OTHER LARGE GROUND BASED TELESCOPES

Table 5 shows the software construction costs as a percentage of the total construction budget for the ALMA, VLT and ASKAP telescopes. These figures were taken from [1] and [13]. For the ALMA and VLT case, the costs were calculated based on 2006 year figures and using the ESO equivalent cost for an FTE in 2006 US$. The ASKAP case as described in this paper is based on CSIRO equivalent FTE in 2013 AU$.

ALMA software maintenance effort is estimated of the order of 58 FTE per year for the whole ALMA software (including data reduction and archiving components) from 2015 onwards. Out of the 58 FTE, 10 FTE will be allocated to "pure" software development effort (dominated by development in the data reduction software package), leaving approx. 48 FTE/year for "true" software maintenance. This estimate corresponds to approximately 10% of the total development cost per year, i.e. ALMA software maintenance is roughly following the 60/60 rule over a 20 years expected lifetime.

Table 5: Comparison of software cost as a % of the total project cost for VLT, ALMA and ASKAP

Project	Total Construction Cost	Construction Time	Subsystems	FTE	% of Total Project Cost	
VLT	US$850M	1992 - 2000	VLT SW Framework and TCS	216	3.4	**7.1**
			Instrumentation and AO	234	3.7	
ALMA	US$1000M	2002 – 2012	ALMA Common Software (ACS)	36	0.5	**5.7**
			All other ALMA software	386	5.2	
ASKAP	AU$188M	2008 – 2015(*)	Infrastructure (Common Software)	7	1	**4.3**
			ASKAP TOS and 12m Prototype	30	3.3	

4. CONCLUSIONS

- Despite having only "three" points, there is some evidence that suggests that the cost of developing the control software for medium to large ground-based telescopes is between 4 – 5% of the total project budget. This "ball-park" figure seems to be decreasing over time. One of the reasons is the increased use of third-party software, including control software frameworks and/or middleware. New upcoming projects such as SKA, E-ELT, TMT,

GMT and others could provide more data to demonstrate if the "ball-park" figure has stabilized towards the 4% figure.

- Despite the simplicity to capture productivity measures based on SLOC, it presents several constrains that make it very difficult to compare with other applications. However they can be very useful to calibrate future estimate of the same type of application within an organization. The advice: start collecting and measuring your data for improving your estimates.
- Software maintenance costs represents a significant cost over the lifetime of a software product in use for astronomical applications (lifetime usually over 10 – 20 years). Use the 60/60 rule for estimate software maintenance costs, which yields approximately 10% of the total development cost per year. Measure continuously to calibrate your estimates.
- The software framework cost (ACS for ALMA, EPICS/ICE for ASKAP) represents a very small fraction of the total cost of development. It is impossible to estimate accurately how much the project saved by using a control software framework. Bear in mind that adopting a control software framework comes with a cost in learning curve. For ASKAP the learning curve effort on EPICS and CSS was around 1 to 2 FTE. Certainly there were some savings by not having to develop the provided functionality from scratch. The use of a control software framework to homogenize the local monitoring and control has substantial benefits in terms of minimizing risk during integration and long-term maintenance cost.
- Using a control framework also makes sense for medium to large projects where the number of sub-systems to be controlled and integrated is sufficiently large that justify investing in the learning curve to offset the inherent risk of system integration of complex systems.

REFERENCES

[1] Kemball, A.J. and Cornwell, T.J., "A simple model of software costs for the Square Kilometre Array"
[2] Guzman, J.C., "Status of the ASKAP Monitoring and Control System", Proc. ICALEPCS 2011, Grenoble, France.
[3] Guzman, J.C. and Humphreys, B., "The Australian SKA Pathfinder (ASKAP) Software Architecture", Proc. SPIE 2012
[4] EPICS: http://www.aps.anl.gov/epics
[5] ICE from ZeroC: http://www.zeroc.com
[6] Control System Studio (CSS): http://controlsystemstudio.org
[7] http://www.dwheeler.com/sloccount
[8] Lum, K. et. al., "Handbook for Software Cost Estimation", JPL D-26303, Rev.0, Jet Propulsion Laboratory, Pasadena, California, 2003-05-30.
[9] Reifer, D., "Industry Software Cost, Quality and Productivity Benchmarks", Reifer Consultants Inc. Report, April 2004.
[10] Glass, R., "Frequently Forgotten Fundamental Facts about Software Engineering", IEEE Software, May/June 2001.
[11] Lechner, D., "Software Recapitalization Economics", The Journal of Defence Software Engineering, November 2006.
[12] Davis, B., [97 Things Every Project Manager Should Know], O'Reilly Media, Inc. 14 August 2009.
[13] Taylor, P., "Evaluation of the ALMA Common Software as a Software Framework for the E-ELT TCS", E-ELT Programme, E-TRE-OSL-439-0003 Issue 1.0, 2007-11-27 (restricted access)