Software for autonomous astronomical observatories: challenges and opportunities in the age of big data

Piotr W. Sybilski[a,b], Rafał Pawłaszek[a,b], Stanisław K. Kozłowski[a], Maciej Konacki[a,c], Milena Ratajczak[a] and Krzysztof G. Hełminiak[a,d]

[a]Nicolaus Copernicus Astronomical Center, Polish Academy of Sciences, Rabiańska 8, 87-100 Torun, Poland;
[b]Sybilla Technologies, Toruńska 59, 85-023 Bydgoszcz, Poland;
[c]Astronomical Observatory, Adam Mickiewicz University, Słoneczna 36, 60-186 Poznań, Poland;
[d]Subaru Telescope, National Astronomical Observatory of Japan, 650 North Aohoku Place, Hilo, HI 96720, USA

ABSTRACT

We present the software solution developed for a network of autonomous telescopes, deployed and tested in Solaris Project. The software aims to fulfil the contemporary needs of distributed autonomous observatories housing medium sized telescopes: ergonomics, availability, security and reusability. The datafication of such facilities seems inevitable and we give a preliminary study of the challenges and opportunities waiting for software developers. Project Solaris is a global network of four 0.5 m autonomous telescopes conducting a survey of eclipsing binaries in the Southern Hemisphere. The Project's goal is to detect and characterise circumbinary planets using the eclipse timing method. The observatories are located on three continents, and the headquarters coordinating and monitoring the network is in Poland. All four are operational as of December 2013.

Keywords: autonomous astronomical observatories, Soalris Project, observatory control systems, automation of optical observations, availability, astronomical software

1. INTRODUCTION

Autonomous observatories are the next step in automation of astronomical observations. They will replace remote observatories in efficient generation of high quality observations with minimal human intervention. Automatic or rather autonomous telescopes of small and medium size will provide unprecedented rate of new data that will be comparable to the largest surveys from a single large telescope.

BlackGEM Array[1] compared to the Large Synoptic Survey Telescope (LSST)[2] may be an example of this trend. The first project will generate data from twenty 0.6 m telescopes equipped with 81 Mpix cameras (total of 1.6 Gpix) and second project will generate 3.2 Gpix from a single camera. It will mean half of the data generation rate in the case of BlackGEM when compared to LSST, but much earlier (2017 vs 2022), for a fraction of the cost and with inbuilt failover and redundancy that will assure high availability and efficiency (i.e. the stream of the data will not stop due to the single telescope failure). Of course these two projects have different goals and will work with different science objectives. But with higher reliability, automation, lower prices and good data quality the small and medium sized autonomous observatories will be a major part of the astronomical data generated in the future.

In section 2 we present a set of definitions and a reference to a more detailed description of the best patterns and practices for fault tolerant systems with high availability that we used in designing software for Project Solaris. Section 3 describes shortly Project Solaris and summarises lessons learnt during all the project phases from design to normal operation. This is where we diagnose and select most important issues that we think are

Further author information: (Send correspondence to P.W.S.)
P.W.S.: E-mail: sybilski@ncac.torun.pl

Software and Cyberinfrastructure for Astronomy III, edited by Gianluca Chiozzi, Nicole M. Radziwill,
Proc. of SPIE Vol. 9152, 91521C · © 2014 SPIE · CCC code: 0277-786X/14/$18
doi: 10.1117/12.2055836

important for the future of medium-sized autonomous observatories and we would like to address in the future design and implementation of "Project Solaris 2.0". Section 4 discusses four important subjects in assuring the good quality of autonomous observatory (AO). Chapter 5 concentrates on crucial handicap of many observatories which is a close software and hardware coupling. In chapter 6 we present a short summary and a few recommendations.

2. DEFINITIONS

In our software design we adapt the vocabulary and definitions used by Robert S. Hanmer in "Patterns for Fault Tolerant Software"[3] to the context of astronomical observatories. For example a bug in the device driver occurs only when the USB communication is heavily utilised and the delay between consecutive USB messages reaches some high value. The bug or the implementation of the driver which is not taking into account the USB performance is a latent fault that in some conditions activates and causes an error. If it is an error of a weather sensor driver reporting the current humidity it may be handled by a fault tolerant system and the failure of the system may be avoided. We define the failure of the observatory as a state in which the observatory is unable to conduct its work according to the requirements. Requirements on the observatory depend on the type of the observations (for example photometric, spectroscopic or polarimetric observations), instruments involved, observing program and many more. However one may establish basic requirements for the observatory in two modes:

- Night mode - observatory conducts the observations in the good weather conditions in timely manner.
- Day mode - observatory conducts the maintenance procedures and additional actions required for the normal operation as specified in the system specification (for example it may involve scientific and system data backups, data transfer and reduction, system updates, cleaning or calibrating the equipment or preparing it for observations).

When the observatory is unable to fulfil its requirements due to the error it experiences a failure. Fault tolerant observatory is able to proceed in the case of one or many errors. In our example faulty driver causes loss of the weather data and fail-silent failure if the data does not arrive to the control system. If the data arrives but has a well-known and consistent failure characteristic (for example the checksum of the data transferred over the USB implies corrupted data or NaN value) we encounter a consistent failure. If there is a secondary or tertiary source of the humidity data the system may continue in case of fail-silent and consistent failures respectively. In the third scenario where we have malicious failure characterized by different errors for the same fault not easily recognised (i.e. inconsistent) we need $3n+1$ devices where n denotes the number of simultaneous failures.

2.1 Definition of autonomous observatory

We define three levels of automation and independence for astronomical observatory: typical, remote and autonomous. Discrimination between levels is based on the human presence and intervention required for the full functionality of the observatory:

- Conducting the observations.
- Taking actions for the system safety in case of normal conditions.
- Taking actions for the system safety in case of errors.
- Taking independent decision in regard of planned tasks based on the data available and logic allowing for the interpretation of complex observing plans.
- Maintenance.

We present three examples for the three levels described:

Typical observatory (TO) - observer is required for the observation's plan preparation, execution and assuring the safety of the process. Local presence is required as some buttons and functionality of the observatory are not fully exposed to the control system available over the Internet. Most of the optical observatories before year 2000 were working like that. Nowadays the trend is reversed especially in the segment of small (<30 cm aperture) and medium sized (30 cm - 1.5 m) telescopes. Large telescopes (>1.5 m aperture) tend to be still manually operated. This may be due to the system complexity, safety and the low cost of human work when compared to the cost of overall structure. Big telescope are also constantly upgraded to maximize the efficiency of the observations and instruments used, which requires prototype solutions and human presence.

Remote observatory (RO) - all instruments and functionality required for the observatory are available over the Internet. The observer may conduct and supervise observations from any location. Usually it is achieved via VPN (Virtual Private Network) or RD (Remote Desktop). Some observing procedures, like focusing, switching from target to target or closing the dome in case of bad weather may be automated. Still in case of network malfunction or failures human presence is required.

Autonomous observatory (AO) - it is the highest level of automation where the system is capable of fault tolerant functionality, independent planning and prioritisation of targets. Human presence is required only for optimized in terms of time maintenance of the observatory's hardware and software. With redundancy of sensors and hardware components it is possible to keep high availability of the system.

2.2 Relative Efficiency

We define relative efficiency Eff_{rel} of one observatory as a ratio between the sum of exposure times of all good scientific frames Δt_1 to the total time Δt_2 of good scientific frames in exactly the same conditions in the second observatory (weather and observing program):

$$\text{Eff}_{rel} = \frac{\Delta t_1}{\Delta t_2} \tag{1}$$

2.3 Absolute Efficiency

Assuring exactly the same weather and observing program conditions is a privilege of few observatories and projects. To give a chance to compare many observatories in real-world conditions we propose to measure the absolute efficiency Eff_{abs} of the observatory as a ratio between the total time when the observations were conducted (exposure time $\Delta t_{\text{exposure time}}$ plus CCD camera's readout time $\Delta t_{\text{CCD readout time}}$) and the total time $\Delta t_{\text{good observing conditions}}$ when they could have been conducted. To simplify the algorithm we may measure the total night time in one year with good weather conditions (if the weather conditions in the night were not measured due to the observatory failure we count this time as a valid for the denominator time) to measure the denominator and add all valid scientific frames exposure times to total number of frames multiplied by readout time to get the nominator. We decided to count CCD camera readout time to the observing time to avoid the situation where the readout frequency of the camera may be manipulated to increase the absolute efficiency. On the other hand such a change would degrade the quality of the data increasing the readout noise and by this entangling the equation with the quality of the data, which is also a complex parameter. N is a total number of good scientific observations. This results in following equation:

$$\text{Eff}_{abs} = \frac{\sum_{n=1}^{N} \Delta t_{n\text{th exposure time}} + N \Delta t_{\text{CCD readout time}}}{\Delta t_{\text{good observing conditions}}} \tag{2}$$

2.4 Availability

The availability of an astronomical observatory may be measured by the percentage of all nights within a year in which the observatory's hardware and software could conduct observations. The number is given in percent and is decreased from 100% value only by software and hardware failures and/or malfunctions that make observing unsafe and/or impossible. High availability is often expressed in a metric of nines, corresponding to the number of nines following the decimal point in availability value, as indicated in Table 1. Operators of typical astronomical

Table 1. Availability as a number of nines.

Expression			Minutes per year of downtime during the night
Name	Availability	Unavailability	
One 9	90%	10%	19710 (328.5 hours)
Two 9s	99%	1%	1971 (32.85 hours)
Three 9s	99.9%	0.1%	197.1
Four 9s	99.99%	0.01%	19.71
Five 9s	99.999%	0.001%	1.971
Six 9s	99.9999%	0.0001%	0.1971
	100%	0%	0

observatories are in a lucky position as the system has to perform well and be available only during the night (which is on average around nine hours per day). The rest of the day may be spent on maintenance, updates and repairs, which don't count to the unavailability time. Unless the day cycle specification encompass additional calibration, performance testing, data reduction, analysis and data transfer.

2.5 Metrics of a system's fault tolerance

In case of small and medium sized, optical, ground based, astronomical observatories it is uncommon to publish[4,5] or even measure in a systematic way the system's fault tolerance, even in the more instrumental-oriented publications. It seems however that the trend reverses and with new projects and publications the importance of metrics becomes visible. We want to emphasise the importance of common metrics allowing for objective estimation of system's performance for use by astronomers, technicians, system builders and software developers. We propose to adopt and use the following metrics:

- **Reliability** - probability that the system will perform without deviation from agreed-upon behavior for a specific period of time.
- **Availability** - the percentage of time that the system is able to perform its designed function (more detailed definition in context of autonomous observatories is given here: 2.4).
- **Absolute Efficiency** - ratio between the total time when the observations were conducted and the total time when they could have been conducted.

To calculate the metric of the system we should measure Mean Time To Failure (MTTF), Mean Time To Repair (MTTR), Mean Time Between Failures (MTBF) or at least record failure and recovery times. Then we can easily calculate:

$$\text{reliability} = e^{-\frac{1}{\text{MTTF}}} \tag{3}$$

$$\text{availability} = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}} \tag{4}$$

In Project Solaris we started the systematic measurement of these values in April 2014 for first three telescopes. The results between April 4th 2014 and May 20th 2014 are:

- Availability: 99.6%
- Reliability: 97.8%
- Absolute efficiency: 78.53%-98.35% (depending on the observing program)

These numbers are very good but based on a short time statistics. Absolute efficiency range depends on the number of telescope slews during the night, filter wheel changes, dome opening and closing time and similar actions that require time and are not a part of the absolute efficiency equation. We can estimate on the base of the data from period of 46 days and with experience gathered during the commissioning phase that these numbers should be around:

- Availability (estimate based on 2012-2013): 90
- Reliability (estimate based on 2012-2013): 98
- Absolute efficiency (estimate based on 2012-2013): 70

Estimated availability is lower due to the length of the hardware repair procedures (specialised and expensive equipment with long repair time between a few days and a few weeks, and high cost being a barrier for spare parts available quickly). Reliability doesn't change much as we are already within the part of the reliability curve with high values (MTTF between 90 and 100 days). Absolute efficiency will be lower due to failures and high MTTR.

3. SOFTWARE USED IN PROJECT SOLARIS

The primary goal of Project Solaris[6] is to detect and characterise circumbinary planets using the eclipse timing method.[7] As shown in numerical simulations, it is possible to detect exoplanets orbiting eclipsing binary systems using a network of 0.5 m telescopes by achieving 0.1-1.0 s precision in timing. A global network of identical 0.5-m telescopes has been proposed to accomplish this task. Apart from photometry aimed at eclipse timing, the network can be used for other stellar astrophysics projects, including spectroscopy, near earth objects monitoring or tracking, etc.

We decided to use Microsoft .NET environment in Project Solaris as a base for our development. Small team, the need for rapid software development, existence of operating system specific drivers, professional support for technologies and tools were primary factors for the decision. On top of .NET works Microsoft Robotics environment with stateless, open standard protocol called Distributed Software Services Protocol. Implementation of the fastest binary version of this protocol is a proprietary code written by Microsoft but it can be easily replaced by custom endpoints writing and reading to different protocols over different than default TCP transports. With the binary serialisation we achieve latency between sending and receiving messages of 0.15 ms and throughput of 6713 messages per second per thread. These numbers were achieved on a single computer (Intel i7-3820 3,6 GHz, 24 GB DDR3 RAM) across the separate software nodes communicating via a loopback network interface. With two nodes on the same local network but on two different computers we noticed latency of 0.16 ms and the same throughput. No visible performance loses were noted. Microsoft Robotics allows also for XML serialisation which is on average 5-6 times slower but is easily readable by human and other web service oriented protocols. Both DSSP TCP and XML TCP versions are wrapped by SOAP (Simple Object Access Protocol) envelope.

Two main components from Microsoft Robotics are heavily used: CCR (Concurrency and Coordination Runtime) and DSS (Decentralized Software Services). They are designed specifically for implementing distributed software, ideally with the state machine concept in mind. Overall features and design principles of our software architecture are:

- robustness,
- composability,
- observability,
- asynchrony,
- concurrency,
- event driven state machine,
- redundancy of security.

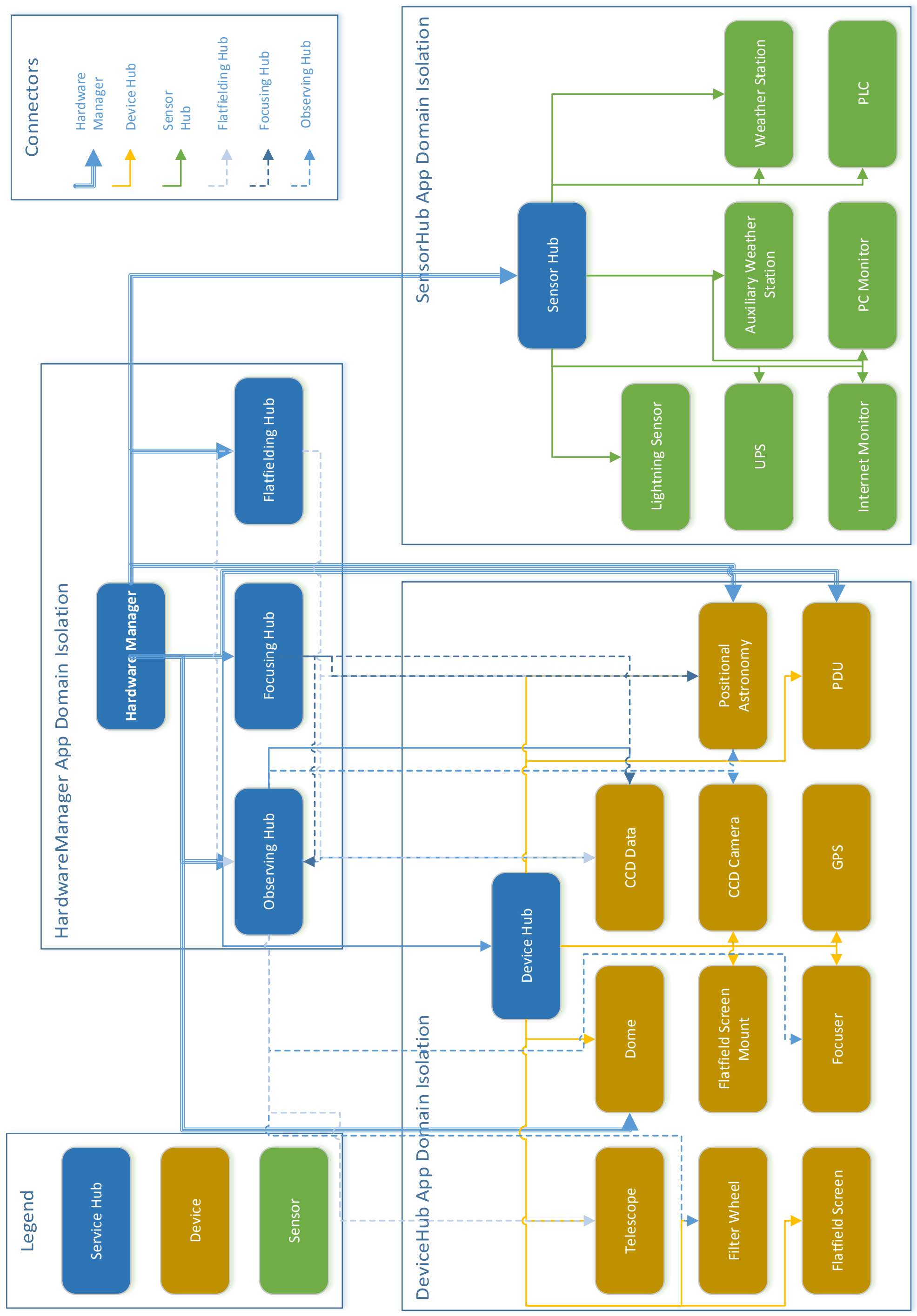


Figure 1. Current software architecture of Abot^{TM}(Astronomical Robot) used in autonomous observatory control system in Project Solaris. Software components are grouped by the type and role they fulfil in autonomous observatory control system. Arrows represent communication links between components. Grouping boxes shows the software isolation between components achieved with isolation of executable in separate DSS nodes (separate processes and separate memory).

The software paradigm is illustrated in Figure 1. We isolate the low level hardware management behind the Device Hub and Sensor Hub, for actuators and sensors respectively, from the database and general observatory management. The isolation is provided by separate processes and different working memory for each application domain. We separated the components in this way for easy updates, performance and security reasons.

We are very satisfied by current design and solution. Average CPU use (Intel i7-950, 3 GHz, with 12GB RAM) by the software is around 3% and peaks during the CCD data processing up to 15%. The average RAM usage is around 500 MB and can be dampened to 150 MB but it would impact the performance as the garbage collector would switch to single thread mode instead of working in server mode. The short burst of CPU usage for a few seconds does not limit the systems performance as only CCD Data Service is busy and the rest of the system is able to perform its tasks as quickly as possible (which is usually well below 1 ms in our event driven architecture). The most important result of choosing the asynchronous and parallel approach with remote procedure calling (RPC) based on CCR and DSS is clearly visible when compared to a different approach: synchronous, sequential and local procedure calling. It is the execution time that makes theoretical and practical difference. In such a setup one loses roughly 1600 seconds per observing night per observatory. It is slightly more than 6% of the observing time. This number may seem small but in the global perspective of the Project Solaris we would lose 1300 observing hours.[8]

3.1 Datafication

In Project Solaris we can observe a global trend of slow "datafication" of small and medium-sized astronomical observatories. We participate in this global phenomenon by increasing the number of variables monitored that were not usually recorded. And we increase this number constantly. Right now we monitor and record 98 physical variables (for example, external temperature, wind speed, rain presence, fan speed) from sensors available for one observatory. We also monitor 148 physical variables derived from physical variables (for example, average wind speed or wind direction, rain accumulation or number of lightning strikes within radius of ten kilometres in the last hour). We also log and watch numerous software components for performance statistics and data correctness. Fault observer in our system receives messages from 24 software services and takes recovery action in case of strong deviations from predicted behavior. We also constantly improve the system's efficiency and fine tune its parameters and number of faults it can handle autonomously. For example we are currently testing a focusing system based on a neural network that requires only 2-3 exposures to find a proper focuser position in comparison to the system that fits a curve to 8 measurements and calculates its minimum. The amount of data we record as a telemetry from single telescope each day ranges from 10 MBs to 100MBs, depending on verbosity level and constantly grows as we add more variables we want to log.

This approach resulted in two situations where gathered data allowed for fault prevention. For the first time it happened when the motherboard of the computer in Australia failed. Data recorded prior to failure had a distinct signature that was observed later on in another unit in Republic of South Africa. Mainly a few sensors (not critical for the system) failed within the month of permanent motherboard failure. This pattern allowed for a preparation for a failure and optimised maintenance. The other case of anticipated failure was based on S.M.A.R.T (Self-Monitoring, Analysis and Reporting Technology) technology implemented in hard drives.

3.2 Most viable issues

In our work with autonomous observatories we encountered not only nice surprises like Microsoft Robotics but also a few outstanding problems with integration, hardware and software. Structured overview of current design is shown in Figure 2. Most viable issues we want to address in this study are:

- Hardware fails and point to point connections like USB, RS232, Bluetooth, direct radio or infrared link are preventing system from being fault tolerant.
- Single PC can handle 5 observatories but without a standby unit and update procedures, single unit failure or malfunctions cause system downtime.
- Lack of standards in terms of hardware and software security standards, drivers, fault tolerance metrics, communication, procedures.

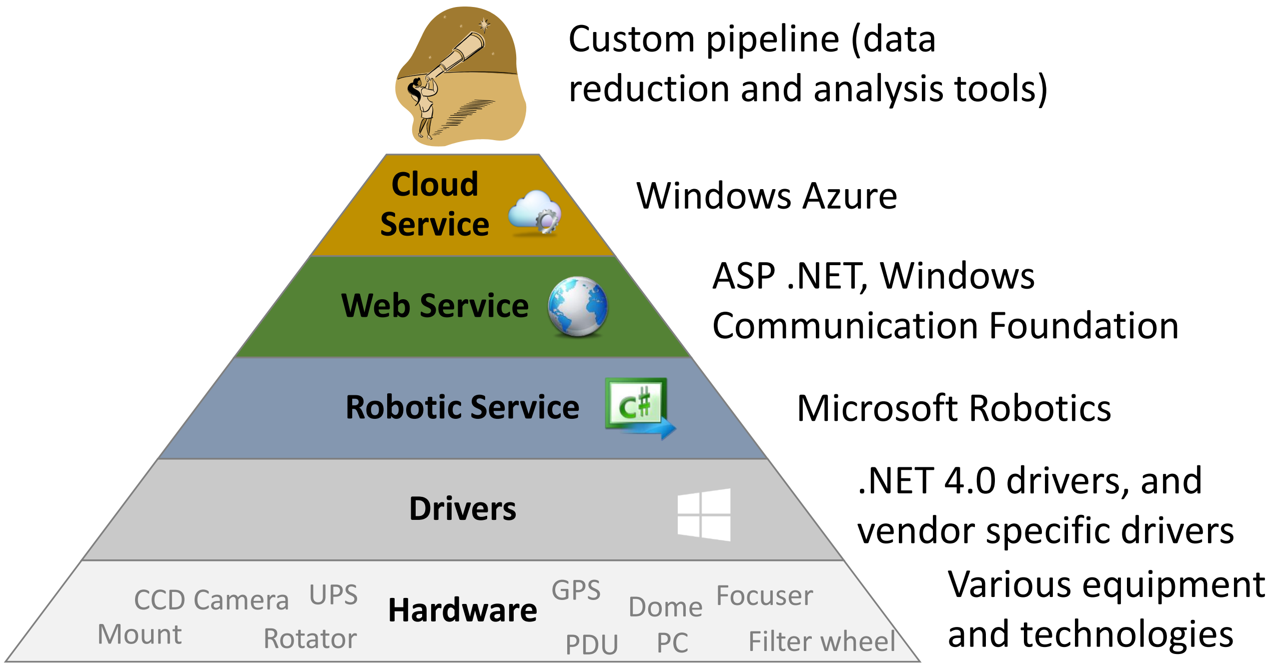

Custom pipeline (data reduction and analysis tools)
Cloud Service
Windows Azure
Web Service
ASP .NET, Windows Communication Foundation
Robotic Service
Microsoft Robotics
Drivers
.NET 4.0 drivers, and vendor specific drivers
Hardware
CCD Camera
UPS
Mount
Rotator
GPS
Dome
Focuser
PDU
PC
Filter wheel
Various equipment and technologies

Figure 2. Current design of Abot command and control system for autonomous observatories employed in Project Solaris. The pyramid shows layers of subsystems and on the right the technologies used in every layer. In Project Solaris we are happy with Robotic Services, Web Services and Cloud Services layers. Drivers written by third parties are sometimes proprietary and of low quality and additional wrapper drivers have to be written to compensate and isolate the faults of drivers available. The same issue is present in hardware layer where the hardware design is faulty and causes hard to isolate failures. We try to correct it in higher layers but the possibilities are limited and resulting system is a patchwork of workarounds.

- Lack of standards in data reduction and analysis makes comparison and simultaneous use of data from different observatories a hard task.
- Low quality and high failure rate of a single device has a much bigger impact on the system stability than the impact judged by the device's role in the system. Hardware and drivers faults tend to have the highest impact on the autonomous observatory efficiency and availability.

4. QUALITY OF AUTONOMOUS OBSERVATORY SERVICE

Quality of service in case of autonomous observatory has two different aspects. One is the quality of the data generated by AO and is a part of AO's metrics and the second is the performance of the hardware and software. There is no established set of definitions and performance counter that are widely used in the sector of autonomous observatories of small and medium size. Of course the satellite missions and the biggest projects have well written specifications and thoroughly discussed metrics but they have much bigger budgets and more restricted design process. In case of smaller projects the full-scale design would be disproportionate to the size of investment. However, with advances in software and hardware more and more observatories can benefit from exactly the same design and scale effect.[4,5,8] It is within the grasp of community to benefit from the situation and implement some well-known industry standards in small and medium sized observatories.

We propose to use system availability and efficiency as parameters measured in every AO. Reliability and unavailability[3] may be also used but they have much lower significance in case of AOs as they have small variance (reliability) in typical cases and are harder to estimate (unavailability).

4.1 Layered security of the autonomous observatory

We propose to systematise the description of security and fault tolerance system in AOs in the "Autonomous Observatory Security (AOS) 5 Layer Model". For details see Figure 3 and 4. One additional term that we use is the environment risk level (ERL). We distinguish five levels of ERL:

- Low - weather conditions are good, network traffic is normal, no malicious actions originating from the network are present (all channels: Internet, LAN, GSM, Wi-Fi are within the limit of normal traffic and the network security of the system has not been compromised). Good weather is defined as follows: all major sensors readings (humidity, rain intensity) minor (temperature, wind speed, cloud level, outside light level) are within the limits for the AO.
- Moderate: weather conditions are changing but all major weather readings are good, one or two minor reading may crossed their limits. Network traffic is normal, no malicious actions.
- Substantial: One major sensor reading is outside the limits or network traffic overloads the system or three or more minor sensor limits are out of secure range.
- Severe: Two major sensor readings are out of range, in which one is non-zero rain intensity or the malicious actions like DDoS attack (Distributed Denial of Service) is underway.
- Critical: Extreme weather conditions due to the passing heavy thunderstorm, close fire (high temperature), strong wind (over or close to the structural limits), strong earthquake or similar. The password for the administration access is lost or the control over the system is lost and third party is using it.

The autonomous observatory capability to deal with certain threats requires all layers of security to be operational and executed when necessary. There is no security on level 5 when the system malfunctions on lower levels. It is why the coherence of communication, logic and performance on lowest levels are so important for the overall security of the system. For the illustration see Figure 5.

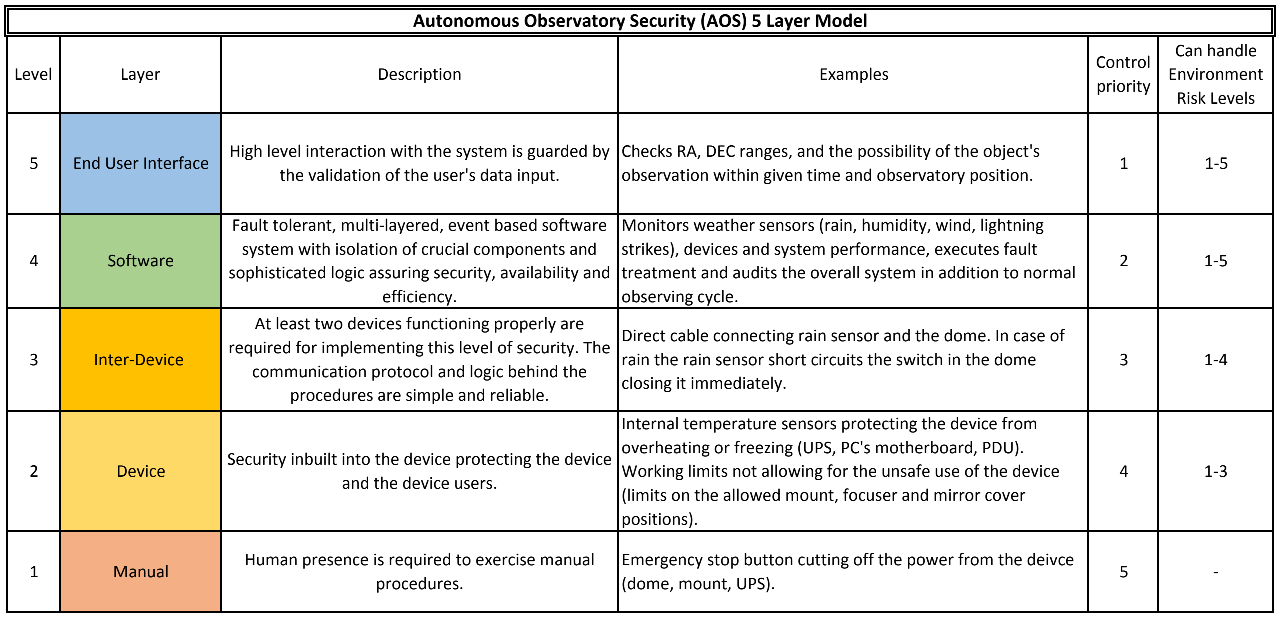
4.2 Know your time

One of the most important characteristics of the astronomical observation is the start and end time of the observation and the photometric centre of the observation. The photometric centre is especially important in case of objects moving with different celestial rates. In such a case background stars may be recorded as distorted line-shaped objects or the object moving with different than celestial speed (roughly 360 degrees over 24 hours) may form a line. The centre of the line may be distorted by asymmetric opening and closing of the shutter. This effect is important for quickly moving objects and short time exposures when the opening and closing time are not negligible when compared to the exposure time. We can estimate that the effect is negligible when compared to the photon noise once the exposure time is ten thousand times longer than the shutter opening and closing times.

To fully understand the timing uncertainty of our system we have to understand and describe the following elements:

- Software.
- Control computer (here abbreviated as PC, but it also can be an embedded device, server-grade device or PLC[*].
- Electronics (it can be the special device registering time, intermediate boards transferring data and control commands or the CCD camera electronics).
- Clock (every device registering time uses a reference clock it's stability over short and long periods of time should be known as well as the synchronization algorithms and overall precision achieved).

[*]Programmable Logic Controller



Autonomous Observatory Security (AOS) 5 Layer Model					
Level	Layer	Description	Examples	Control priority	Can handle Environment Risk Levels
5	End User Interface	High level interaction with the system is guarded by the validation of the user's data input.	Checks RA, DEC ranges, and the possibility of the object's observation within given time and observatory position.	1	1-5
4	Software	Fault tolerant, multi-layered, event based software system with isolation of crucial components and sophisticated logic assuring security, availability and efficiency.	Monitors weather sensors (rain, humidity, wind, lightning strikes), devices and system performance, executes fault treatment and audits the overall system in addition to normal observing cycle.	2	1-5
3	Inter-Device	At least two devices functioning properly are required for implementing this level of security. The communication protocol and logic behind the procedures are simple and reliable.	Direct cable connecting rain sensor and the dome. In case of rain the rain sensor short circuits the switch in the dome closing it immediately.	3	1-4
2	Device	Security inbuilt into the device protecting the device and the device users.	Internal temperature sensors protecting the device from overheating or freezing (UPS, PC's motherboard, PDU). Working limits not allowing for the unsafe use of the device (limits on the allowed mount, focuser and mirror cover positions).	4	1-3
1	Manual	Human presence is required to exercise manual procedures.	Emergency stop button cutting off the power from the deivce (dome, mount, UPS).	5	-

Figure 3. Layered security of the autonomous observatories. Description of each layer is complemented by some examples of implementation. Higher control priority level means that the command with higher priority will be executed or will override the logic of layer with lower priority. When two contrary commands reach the system the one with higher control priority is executed and the one with lower priority is aborted. Last column describes which environment risk levels can be handled by which layers. It is important to note that in this model higher layers functionality depends on proper work done by lower layers. Software from layer 4 cannot assure the security of the system if the dome doesn't close due to the power grid failure or due to the telescope blocking the shutter because of faulty driver.

- Time source (different time sources are used in astronomical observatories, for example a GPS synchronised clocks, Network Time Protocol, radio synchronised clocks or atomic clocks).

In Project Solaris we utilise a connection between a high precision GPS card with its own very stable battery powered clock (Meinberg 170 PEX) and camera's shutter (Andor iKon-L) directly connected to the GPS card. In this design we are able to investigate latency and variation of different components. For details see Figure 6 and Table 2. We can clearly see that without the direct connection and relaying only on PC and CCD Camera, we can achieve only a precision of few milliseconds. This precision (estimated on standard deviation of registered by PC end of exposure time) is achievable only in case when the systematic differences were measured and in case when they do not change over longer periods of time. Without these reference measurements our precision drops significantly and is worse than a few seconds (5 seconds in our case if we would take the maximum recorded deviation from start and end time). It is interesting to see that the end of exposure time has much smaller variations than the start of exposure, as well as much smaller latency. Asymmetry may have its source in the CCD camera's software and electronics that are responsible for shutter opening and closing. Investigation of Figure 6 yields another interesting observation. Top-left panel assures us that the requested exposure time and real exposure time are close to each other and the difference on average is smaller than 0.05 ms. But there are a few points with a much worse precision. These points correspond to longer exposure times (around 90 s in comparison to only a few seconds for the rest of the points). It suggests that the CCD camera's clock on longer exposures starts to show deviation from a linear rhythm.

5. HARDWARE-SOFTWARE COUPLING

One of the most urgent issues we diagnosed in Project Solaris' network of telescope is the hardware-software coupling and downtime caused by this coupling.

Autonomous Observatory Security (AOS) 5 Layer Model - Continued

Level	Layer	Access via	Protocols within the layer	Protocols between layers	Example protocols
5	End User Interface	End user interfaces (web pages, desktop and mobile applications) required by observatory's specification.	Communication between system and end user (examples: HTTP, FTP, HTTPS).	End User	HTTP, HTTPS
				Software - End User	HTTP, HTTPS
4	Software	Software maintenance interfaces (web pages, desktop and mobile applications) with advanced options available.	Communication between software components (examples: HTTP, HTTPS, DSSP, TCP).	Inter-Device - Software	DDS
3	Inter-Device	Maintenance and communication ports.	Cross-device communication (examples: DDS, custom RS485 protocol or custom protocol over the wire).	Device - Inter-Device	DDS
2	Device	Maintenance and communication ports.	Internal protocols (examples: CAN, custom protocol on I2C).	Technician-Device	DDS
1	Manual	Buttons, handles and switches accessible for humans.	Physical manipulators (examples: buttons, switches, handles, screws).	Technician	Physical manipulators

Figure 4. Continuation of Figure 3. In this table we describe typical canals via the layers can be accessed and examples of protocols used within the layer. We also show some examples of communication across the layers. For layers 1-4 the ultimate answer for across layer communication may be given by Distributed Data Services (DDS) and on levels 4-5 by SOAP protocols implemented by Web Services and Cloud Services.

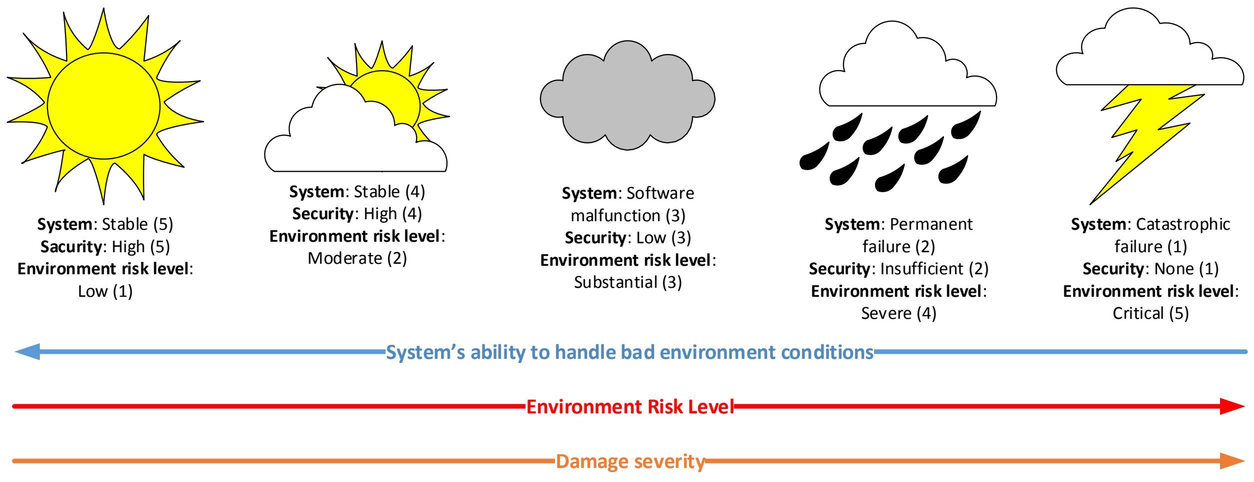

System: Stable (5)
Sacurity: High (5)
Environment risk level: Low (1)
System: Stable (4)
Security: High (4)
Environment risk level: Moderate (2)
System: Software malfunction (3)
Security: Low (3)
Environment risk level: Substantial (3)
System: Permanent failure (2)
Security: Insufficient (2)
Environment risk level: Severe (4)
System: Catastrophic failure (1)
Security: None (1)
Environment risk level: Critical (5)
System's ability to handle bad environment conditions
Environment Risk Level
Damage severity

Figure 5. Simple diagram illustrating the security status of autonomous observatory in changing environment conditions and with different layers of system security operational. With increasing ERL the system's capability to handle the threat is tested. If all components are working temporary or permanent damage risk is very low and its potential severity reduced. If the system lacks one or more higher levels of security the potential damage is higher. If due to the system or one of its components design lower level security (1-3) is not present then the probability of catastrophic failure is high.

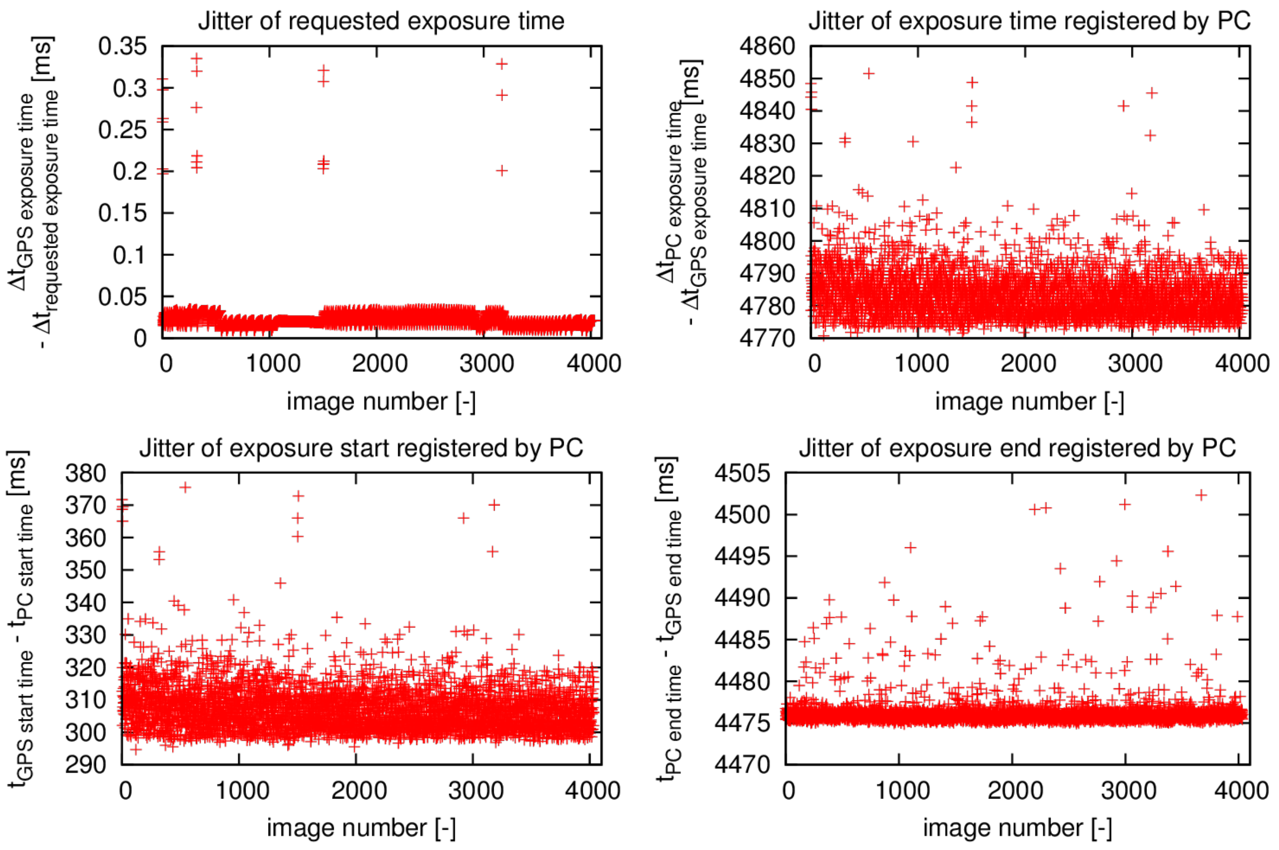


Figure 6. The result of timing precision of Solaris system's subcomponents measured directly by GPS card Meinberg PEX 170 with direct link to CCD camera's shutter Andor iKon-L.

Table 2. Statistical analysis of values presented in Figure 6.

Variable	Min (in ms)	Max (in ms)	Average (in ms)	Median (in ms)	Standard deviation of average (in ms)
$\Delta t_{\text{GPS exposure time}} - \Delta t_{\text{requested exposure time}}$	0.01	0.34	0.02	0.02	0.02
$\Delta t_{\text{PC exposure time}} - \Delta t_{\text{GPS exposure time}}$	4770.72	4851.58	4783.28	4782.49	7.29
$t_{\text{GPS start time}} - t_{\text{PC start time}}$	294.65	375.50	307.13	306.12	7.06
$t_{\text{PC end time}} - t_{\text{GPS END time}}$	4474.88	4502.32	4476.15	4475.89	1.67

5.1 Issue description

The hardware-software coupling is the primary issue limiting the efficiency and automation of many recovery processes and by this availability in Project Solaris' network of autonomous telescopes. For example, point to point connection between the CCD camera or the telescope's mount via USB causes all latent faults of electronics or drivers participating in the communication to produce failures with long recovery time. Basic procedures require restart and recalibration of the involved devices. As USB is widely used and on the PC's motherboard single chip is responsible for handling all the USB requests the failure easily propagates on the hardware level disabling more and more devices. Isolation on the software level assures the overall system security but can't help with prolonged downtime.

Moreover, in case of failure of one of many components that build the PC unit the whole observatory is unable to fulfil its role. With correctly implemented security layers 1-3 the security of the system is still assured but the downtime and impact on availability is tremendous, especially in remote locations where observatories are usually located and human help is delayed. According to the best patterns and practices we should minimise the human interaction with the system and maximise human participation in rare cases when the system cannot take care of itself. How to achieve that? The simplest change would be to use transport that can easily reroute communication when the one node is down to another one assuring the continuity of work. It may be achieved with TCP or UDP transport. It definitely cannot be done with USB, direct radio link or RS232, where the human intervention is required to reroute communication. If the UDP and TCP is chosen the next question is about the protocol. The protocol should provide following features:

- Supported by major organization (long term development and standardisation).
- Open source and commercial implementation available.
- Community and commercial support available.
- Well established (more than a decade of development).
- Proven with successful stories in industry.
- Secure when required.
- Able to work on devices with limited memory, processing power and on constrained power consumption.
- Platform independent (Linux, Windows, iOS).
- Language independent (many languages supported, mainly: C, C++, C#, Java, Python).
- Easy to fix or bypass architecture in case of failures (bus or P2P).
- Support for request-response and subscribe-publish.
- Low latency communication between nodes.
- Small overhead for messages.

5.2 Proposed solution

After careful investigation (CoAP, Continua HDP, DDS, DPWS, HTTP/REST, MQTT, SNMP, UPnP, XMPP, ZeroMQ) we came into the conclusion that only Distributed Data Service protocol provides all features and even more. It is a standard maintained by Object Management Group with open source implementation and commercial support provided by three companies. Moreover the DDS is a protocol based on bus with a high interoperability and other features helpful in autonomous observatory operation. DDS has already found its place in astronomical observatories in case of automation and upgrade of VLT (Very Large Telescope) operated by ESO (European Southern Observatory) or is a protocol of choice for LSST middleware.[9]

6. SUMMARY

Small and medium sized autonomous observatories may take the example of large ground-based observatories and even some good techniques and solutions of space observatories. Best patterns and practices of fault tolerant system in design and implementation are required for smaller telescopes to allow them to participate in the era of astronomical Big Data. We want to emphasise especially two patterns: minimise human intervention and maximise human participation.[3] On the other hand networks of smaller observatories have the advantage of allowing for redundancy and standardisation which is practically impossible for larger observatories. We would like to give a few recommendations based on our experience gained during work for Project Solaris and on the subjects described in this document:

- Metrics of a system's fault tolerance should be introduced in every autonomous observatory. Availability should be systematically monitored and optimized as well as absolute efficiency and reliability.
- Layered security of the system should be implemented with special care for layers 1-3. Layers 4-5 are important in ergonomics and efficiency of the system, but without levels 1-3 properly implemented the risk of a catastrophic failure is high.
- Time of observation has to be carefully measured, preferably with a use of a dedicated GPS card.
- Decoupling of hardware and software should be assured by a well-established industrial standard like DDS.
- Promotion of existing standards and creation of new standards is in the best interest of all parties involved in design, building and operating of autonomous observatories.

ACKNOWLEDGMENTS

This work is supported by the European Research Council through a Starting Grant and by the Polish National Science Center grants 2011/03/N/ST9/03192, 5813/B/H03/2011/40 and 2011/03/N/ST9/01819, ESA's contract 4000110300/14/NL/CBi, the Ministry of Science and Higher Education through grant W103/ERC/2011 and the Foundation for Polish Science through grant "Ideas for Poland". We are grateful to the technical and administration staff in SAAO, SSO and CASLEO for their help during the construction and commissioning of the telescopes.

REFERENCES

[1] BlackGEM Project, "The blackgem array for gravitational wave counterparts." `https://www.astro.ru.nl/wiki/research/blackgemarray` (2014).

[2] Large Synoptic Survey Telescope, "Large synoptic survey telescope." `http://www.lsst.org` (2014).

[3] Hanmer, R., [*Patterns for fault tolerant software*], John Wiley, Chichester, England Hoboken, NJ (2007).

[4] Bakos, G. Á., Csubry, Z., Penev, K., Bayliss, D., Jordán, A., Afonso, C., Hartman, J. D., Henning, T., Kovács, G., Noyes, R. W., Béky, B., Suc, V., Csák, B., Rabus, M., Lázár, J., Papp, I., Sári, P., Conroy, P., Zhou, G., Sackett, P. D., Schmidt, B., Mancini, L., Sasselov, D. D., and Ueltzhoeffer, K., "HATSouth: A Global Network of Fully Automated Identical Wide-Field Telescopes," *Publications of the Astronomical Society of the Pacific* **125**, 154–182 (Feb. 2013).

[5] Lister, T. A., Anderson, D. R., and West, R. G., "The Status of SuperWASP-South," in [*Transiting Extrapolar Planets Workshop*], Afonso, C., Weldrake, D., and Henning, T., eds., *Astronomical Society of the Pacific Conference Series* **366**, 108 (July 2007).

[6] Sybilski, P. and Kozlowski, S. K., "Project Solaris - a Southern Hemisphere robotic telescope network," *Monthly Notes of the Astronomical Society of South Africa* **70**, 131–135 (Aug. 2011).

[7] Sybilski, P., Konacki, M., and Kozłowski, S., "Detecting circumbinary planets using eclipse timing of binary stars - numerical simulations," *Monthly Notices of the Royal Astronomical Society* **405**, 657–665 (June 2010).

[8] Kozlowski, S., Sybilski, P., Konacki, M., Ratajczak, M., Pawlaszek, R., and Helminiak, K., "Solaris: A network of autonomous observatories in the southern hemisphere," in [*Robot Motion and Control (RoMoCo), 2013 9th Workshop on*], 66–73 (July 2013).

[9] Mills, D. and Schumacher, G., "Middleware design and implementation for lsst," *Proc. SPIE* **7740**, 77402C–77402C–14 (2010).