



Ariadne: a system for evaluating AMAZED's efficiency

R. C. Borges, S. Arnouts, P-Y. Chabaud, F. Fauchier, M. Gray, S. Jamal, V. Le Brun, O. Le Fèvre, A. Schmitt, C. Surace, D. Vibert & C. Vidal Aix Marseille Univ, CNRS, LAM, Laboratoire d'Astrophysique de Marseille, Marseille, France

Abstract

Here we present Ariadne, an automation subsystem for AMAZED, whose primary task is to compute AMAZED's redshift estimation efficiency. Secondarily, it serves as structured repository of processed reductions and metadata, allowing users to find existing results specifying abstract criteria (such as astronomical object parameter intervals), and remotely execute software to generate results not yet in the repository.

INTRODUCTION

LAM (Laboratoire d'Astrophysique de Marseille) is at the forefront of development of next generation spectrographic instruments, such as PFS (Prime Focus Spectrograph - see Tamura 2016) and Euclid (see Laurejis 2014). For both projects, LAM is responsible for delivering a pipeline that estimates redshift values. AMAZED (Algorithms for Massive Automatic Z Evaluation and Determination) is a project developed at LAM to deliver pipelines for PFS and Euclid (see Schmitt 2017). AMAZED's efficiency is defined for spectra which have reference redshift values.

Design

Ariadne's database stores metadata about AMAZED, ProcAOS and the samples they process. Automation is built as logic around the database.

Table_amazed_method <pre>name Table_amazed_executable amazed_cli_commit_hash conf_rodobift_commit_back</pre>		<pre>oname ofile_id oname onomical_object</pre>	0	Table_annotationtable_namefield_nameannotation	<pre>Table_feature_access feature_name full_access read_access delegate_access</pre>
<pre>°cpf_redshift_commit_hash °git_tag °file_list Table_amazed_result °aggregate_id °amazed_configuration_id °efficiency_matrix_ids °estimate °visit_id</pre>	<pre>oabsorption_velocity_dispertion oemission_velocity_dispersion odeclination oe_b_v ointer_stellar_medium magnitude name oreference_redshift oright_ascension ostar_forming_rate</pre>		<pre>Table_user oname password_salted_hash read_ndarray_list_id read_ndarray_list write_ndarray_list_id write_ndarray_list feature_access_list_id feature_access_list_id</pre>		Table_tag•name•ndarray_listTable_efficiency_matrix•amazed_result_ids•name•plot_file_id•ProcAOS_id
Table_amazed_configur <pre> convert_line_catalog_from_vacuum lambda_range_minimum lambda_range_maximum name redshift_range_minimum redshift_range_maximum redshift_range_step smoothing_width line_catalog_id method_id</pre>		Table_vis <pre> •date •instrument_config •astronomical_obje •noise_ndarray_id •aggregate_ids •signal_ndarray_id </pre> <pre> Table_aggregate <pre> •visit_ids •noise_ndarray_id</pre></pre>	uration_id ct_id	Table_ndarray <pre> array_hash file_list name tag_list Table_file_type name extension </pre>	<pre>Table_ProcAOS • hardcode • name • version • executable_file_id Table_file file_host_name file_path file_type_id ndarray_id</pre>

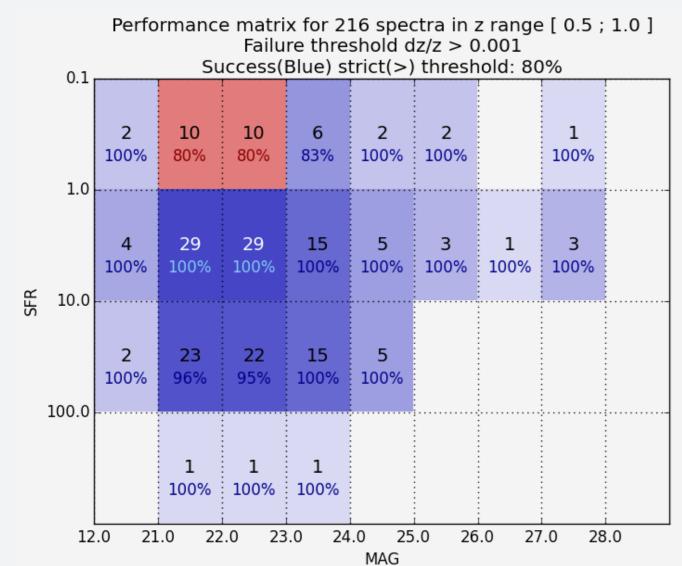
Figure: 4 - Ariadne's database tables.

 $Z_{Error} = |Z_{AMAZED} - Z_{Reference}|/(1 + Z_{Reference})$ Efficiency_{PFS} = Count(Z_{Error} < 10⁻⁴)

Efficiency depends on AMAZED's version and configuration; also on the reference spectra. These conditions change often during development, and will change during survey operations.

Ariadne is LAM's project to automate usage of AMAZED.

OBJECTIVES



The principal objective is to produce "efficiency tables" such as the one displayed on figure 2.

Each table is linked to a specific version and configuration of AMAZED, and to a set of reference spectra.

These tables illustrate the performance of AMAZED with regards to intervals of its input parameters. This allows to verify if AMAZED performs within its intended precision, and suggests the parameter intervals where the pipeline is failing.

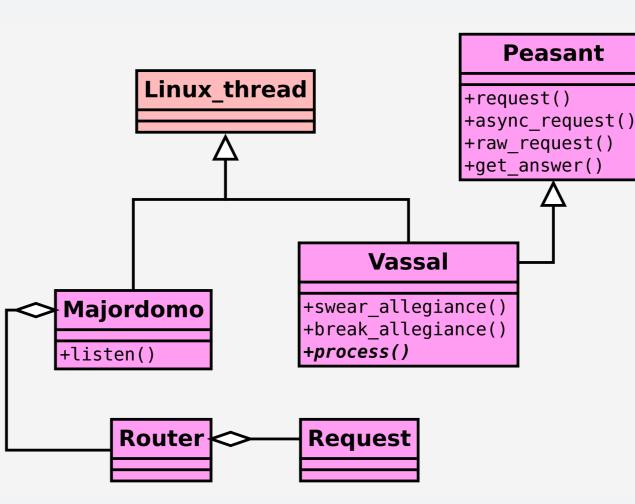


Figure: UML diagram of the classes in Pepin.API.

Conclusions

Ariadne currently allows a user to abstract many details of the complex task of producing AMAZED's efficiency tables. This is a crucial element for allowing developers to conveniently assess impacts of new code in AMAZED. The modular design of Ariadne amde easy to be refactor Pépin into an independent project.

Pépin is built on top of ZeroMQ's Majordomo pattern. It has an API subpackage with Majordomo, Vassal and Peasant classes. Vassals provide services, while Peasants consume them. Daemons inherit from Linux_process, while volatile processes are Linux_threads. As long as clients send well-formed requests, they can use the services.

15

Figure: AMAZED efficiency table.

Since each table depends on several factors, assessing AMAZED's global efficiency suffers from an excessive number of test cases. Ariadne handles the repetitive tasks and does the bookeeping, so users only deal with efficiency in an abstract manner.

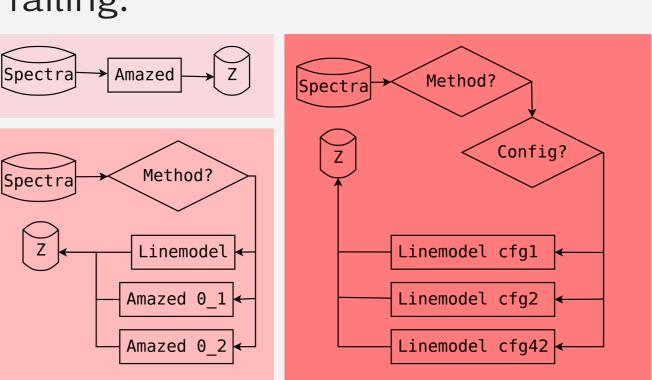


Figure: The explosive growth of AMAZED cases.

Secondarily, Ariadne was built modularly, and its components can be reused to mock telescope operations, and in this fashion we can test AMAZED's performance under conditions closer to its intended usage.

TECHNOLOGIES

Ariadne is developed through Agile practices and designed to be highly modular. It is written in Python 3. Some of Ariadne's modules use specific packages:

- Pépin is based on ZeroMQ;
- Minos database is based on SQLAlchemy (see Figure 4);

Ariadne uses PyQt.

Development is done in a GitLab instance at LAM, with continuous integration with our Jenkins instance. Unit tests use unittest and nose.

ARCHITECTURE

The administrative GUI allows to manage a large data collection with automated tools.

Selected spectra	• Amazed configuration	Remote request	Qt Add entries to database	
Add datasets Current selection Actions	Executable Configuration	Request status: Unknown	Add a directory	
Add FITS files	Amazed executable:	Service: None		
Add entries matching a certain regex	Add amazed executable	Function: None Request ID = -1	Directory: /home/nix/pfs Rem	
Add entries having a certain tag		Attempts to get a response: -1		Querying database
			File name Table_ndarray.array_hash Table_ndarray.aid Table_ndar	rray.name Table_ndarray.tag_list
	Add database record			
	convert_line_catalog_from_vacu	um to air None 🖨		
	lambda range maximum			
	lambda_range_minimum			
	line_catalog_id Nor	ne 🗘		
	method_id Nor	ne 🗢		
	name		There are 0 files to be transferred to the server, 0 fields to be changed in the d	
	redshift_range_maximum		mere are o nies to be transferred to the server, o neids to be changed in the d	Ok Apply Canc
	redshift_range_minimum			
	redshift_range_step		Qt	
	smoothing_width		Table_amazed_configuration Table_amazed_executable Table_amazed_m	ethod Table_amazed_result Table_annotation
	template_set_id Nor			
	Ok	Cancel	aid convert_line_catalog_from_vacuum_to_air lambda_range_maxim	um lambda_range_minimum line_catalog_id
1		_ _ X		
Ariadne Python Interpreter. session functions are: list, append, remove, q	net set			
Example: >>> session.list ()				
subwindows = [<ariadne.gui.amazed_window.amaze ow.amazed execution window object at 0x7f8a923</ariadne.gui.amazed_window.amaze 	ed_window object at 0x7f8a9392ad38>, < 3b0678>, <ariadne.view.logger td="" widget.l<=""><td>Ariadne.gui.amazed_execution_wind ogger widget object at 0x7f8a923b</td><td></td><td></td></ariadne.view.logger>	Ariadne.gui.amazed_execution_wind ogger widget object at 0x7f8a923b		
0868>]				
>>>				

Figure: 6 - Ariadne's administrative GUI.

FUTURE RESEARCH

The two projects we intend to attack next are Tauros and WebZ, the first being an automation module for Ariadne that will be responsible for discovering tasks that should be executed in order to produce up-to-date efficiency tables, given a template.

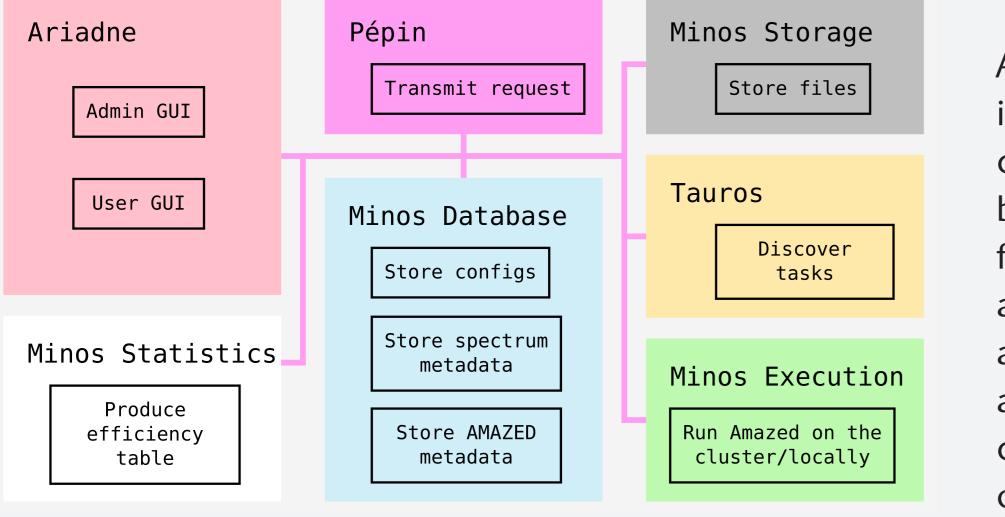


Figure: The modules that comprise Ariadne.

Ariadne's architecture is that of a set of daemons for the backend system. The frontend have Admin and User's GUIs which are multithreaded applications. We will deploy also a web client. The WebZ project is a user-friendly interface to Ariadne's services, which will eliminate the need for installing client GUI software on a user's machine.

References

Schmitt, A. et al, AMAZED: Algorithm for Massive Automated Z Evaluation and Determination, ASP Conf. Ser., 2017.

Tamura, N. et al, Prime Focus Spectrograph (PFS) for the Subaru telescope: overview, recent progress, and future perspectives, Proc. SPIE 9908, 2016.

Laureijs, R. et al, The Euclid Mission: Cosmology Data Processing and Much More, ASP Conf. Ser., vol. 485, 2014.

Acknowledgements

This project would not have been possible without grants from CNRS, connected to the PFS international collaboration.

Created with LTEXbeamerposter http://www-i6.informatik.rwth-aachen.de/~dreuw/latexbeamerposter.php