

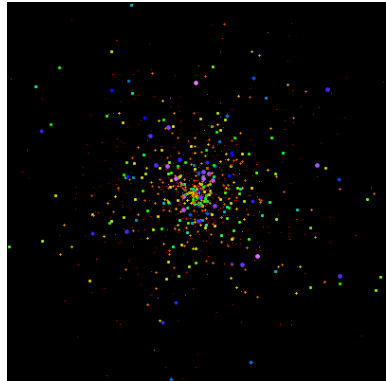


Gabriel Schwartz
Tim McJilton
Andrew Sherman
Rajkumar Jayachandran
Daniel Bagnell
Jason Economou

Advisor: Prof. Jeremy Johnson
Stakeholders: Prof. Steve McMillan
Alfred Whitehead
The Leiden Observatory

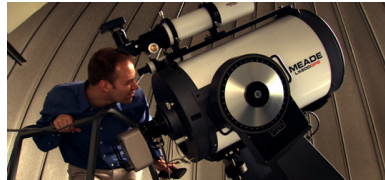
Motivation

- ▶ Makes physics simulations accessible.
- ▶ Targeted at astrophysicists and students.
- ▶ Simulations are large and complex.
 - ▶ 1K - 1M particles
 - ▶ Simulation must be run for many steps.
- ▶ Complex software (AMUSE) exists to perform these computations efficiently.
 - ▶ Hardware-specific algorithms
 - ▶ More/less accurate algorithms



Target Audiences

- ▶ Physics Students
- ▶ Observational Astrophysicists
- ▶ Theoretical Astrophysicists



$$\begin{aligned}
 \tilde{x} &= \frac{C}{M} \tilde{x} + \omega \tilde{y} \left[1 - \frac{f(\phi)}{2} (\Delta k_1 + \Delta k_2 \cos 2\phi) \right] \tilde{x} \\
 &\quad - \frac{\omega \tilde{f}(\phi) \Delta k_2 \sin 2\phi}{2} \left(\tilde{y}_m - \frac{p}{K} \right) \\
 &= e(\Omega + \phi)^2 \cos(\phi + \delta) + e\phi \sin(\phi + \delta), \\
 \tilde{y}_m &+ \frac{C}{M} \tilde{y}_m - \frac{\omega \tilde{f}(\phi) \Delta k_2 \sin 2\phi}{2} \tilde{x} \\
 &\quad - \omega \tilde{y} \left[1 - \frac{f(\phi)}{2} (\Delta k_1 - \Delta k_2 \cos 2\phi) \right] \tilde{y}_m \\
 &= e(\Omega + \phi)^2 \sin(\phi + \delta) - e\phi \cos(\phi + \delta) \\
 &\quad - \frac{p f(\phi)}{2M} (\Delta k_1 - \Delta k_2 \cos 2\phi), \\
 \dot{\theta} &+ \frac{K_t + K_c}{I_0} \dot{\theta} - \frac{K_t}{I_0} \dot{\phi} = -\frac{C_t + C_c}{I_0} \dot{\phi} + \frac{C_t}{I_0} \dot{\phi}, \\
 \dot{\phi} &+ \frac{C_t}{I_1} \dot{\phi} - \frac{C_t}{I_1} \dot{\theta} + \frac{K_t}{I_1} \dot{\phi} - \frac{K_t}{I_1} \dot{\theta} \\
 &= \frac{p}{2I} \frac{e f(\phi)}{2} (\Delta k_1 \cos(\phi + \delta) - \Delta k_2 \cos(\phi - \delta)) \\
 &\quad + \frac{p^2}{2KI} \frac{1}{2} \frac{e f(\phi)}{2} (\Delta k_1 - \Delta k_2 \cos 2\phi) \\
 &\quad - f(\phi) \Delta k_2 \sin 2\phi \} = \tau_{\phi}^*
 \end{aligned}$$

Overview

Introduction

Target Audiences

Purpose

Purpose of GPUTest

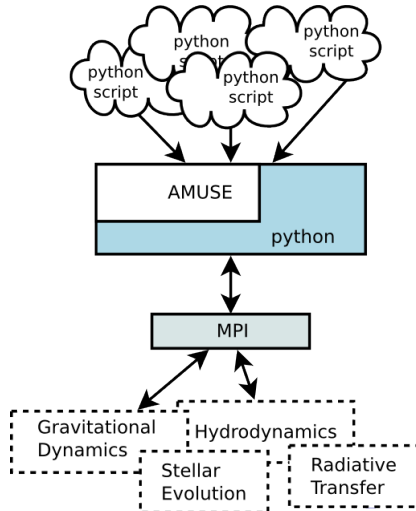
Features and Design

Software Engineering

Impact

Demo

Astrophysical Multipurpose Software Environment (AMUSE)



State of AMUSE

- ▶ Currently used by researchers to run large-scale simulations.
- ▶ Scripts, diagnostics, logging are all written by hand.
- ▶ AMUSE API/programming knowledge is required to create experiments.
- ▶ Still better than separated and opaque FORTRAN codes.

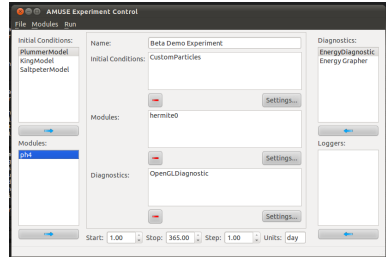
```

first_try = True
model_number = get_model_number(AMUSE_id, ierr)
if (evolve_failed('get_model_number', ierr, evolve, -3)) return
step_loop: do ! may need to repeat this loop for retry or backup
result = star_evolve_step(AMUSE_id, first_try)
if (result == keep_going) result = check_model(s, AMUSE_id, 0)
if (result == keep_going) result = star_pick_next_timestep(AMUSE_id)
if (result == keep_going) exit step_loop
model_number = get_model_number(AMUSE_id, ierr)
if (evolve_failed('get_model_number', ierr, evolve, -3)) return
result_reason = get_result_reason(AMUSE_id, ierr)
if (result == retry) then
! trying to spark interest... Why should I care and not fail as
if (evolve_failed('get_result_reason', ierr, evolve, -4)) return
if (report_retries) & summary of what the project is
write(*, '(i6,3x,a,/)') model_number, &
'retry reason', trim(result_reason_str(result_reason))
else if (result == backup) then
if (evolve_failed('get_result_reason', ierr, evolve, -4)) return
if (report_backups) & summary of what the project is
write(*, '(i6,3x,a,/)') model_number, &
'backup reason', trim(result_reason_str(result_reason))
end if
if (result == retry) result = star_prepare_for_retry(AMUSE_id)
if (result == backup) result = star_do1_backup(AMUSE_id)
if (result == terminate) then
evolve = -11 ! Unspecified stop condition reached, or:
check if ($% number of backups in a row > $% max backups in a row ) then
! terminate
evolve = -14 ! max backups reached
endif
! Show ugly FORTRAN code, transition to simple Python API
if ($% max model number > 0 .and. $% model number >= 6) code is X lines
evolve = $% max model number ! evolve = -13 ! max iterations reached

```

Purpose of GPUit

- ▶ Ease the use of AMUSE
- ▶ Create/Design/Modify experiments
- ▶ Select, configure, swap out modules and initial conditions
- ▶ Store and restore progress of running experiments.

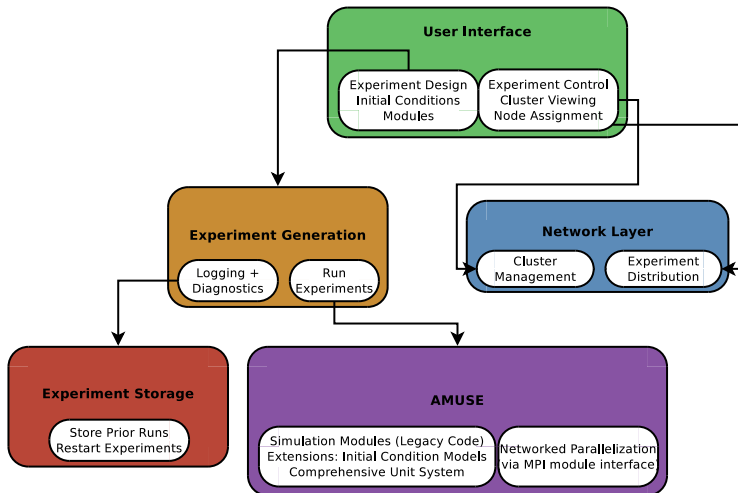


Features

- ▶ Configurable experiments that can be saved and shared.
- ▶ Diagnostic tools that compute and display useful measurements.
- ▶ Storage of experiment state in case of crashes.
- ▶ Custom diagnostics and code generation.
- ▶ Provides a display of cluster usage to aid in scheduling.



Architecture



Design

- ▶ AMUSE is the only integrated simulation environment available.
- ▶ AMUSE is written in Python, streamlines interaction.
- ▶ C++ was considered as it supports Qt as well.
 - ▶ Communication w/AMUSE would be cumbersome.
 - ▶ AMUSE would be in a separate process.
- ▶ Designed APIs for diagnostics, logging and experiment persistence.
 - ▶ Users can create new diagnostics easily.
 - ▶ Experiments can be stored in a file structure, a remote DB etc...

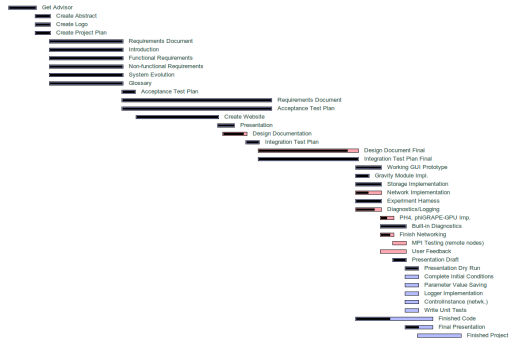


Tests

- ▶ GUI / Integration tests were performed manually.
 - ▶ Created and ran a simple experiment from scratch to ensure functionality.
- ▶ Unit testing performed using Python's unittest module (PyUnit).
- ▶ Tests:
 - ▶ networking
 - ▶ built-in diagnostics
 - ▶ object serialization
 - ▶ experiment storage
 - ▶ experiment running

Project Plan

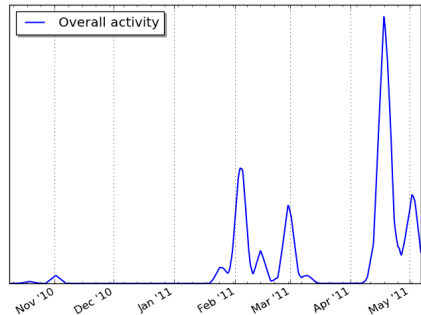
- ▶ Mostly waterfall design process.
- ▶ Initial phases were spent learning the domain (Physics/AMUSE).
- ▶ Roles
 - ▶ Tim: Physics reference, test subject
 - ▶ Andrew/Jason: Experiment and Module design.
 - ▶ Dan: Diagnostics
 - ▶ Raj: Logging
 - ▶ Gabe: Network, GUI.



Team Management

- ▶ Used Mercurial as our version control system.
 - ▶ Distributed, allows off-line commits.
- ▶ Team met weekly.
 - ▶ Once to plan work, once to code.
- ▶ Bi-weekly advisor meetings.

GPUUnit Commit History



Project Impact

- ▶ Gives students and physicists easy access to state-of-the-art tools.
- ▶ Simple experiment creation → faster turnaround on experiments.
- ▶ Faster experiments → more time to study them.
- ▶ Current state:
 - ▶ Software is usable to create simple experiments.
 - ▶ Comes with useful diagnostics, from real experimental setups.
 - ▶ Ready to get feedback from more advanced users.
 - ▶ Capability/APIs already exist to provide more advanced features.

Demo

- ▶ Demonstration of a simulation.

Questions

- ▶ Questions?