Introduction to AMUSE

AMUSE

Francisca Concha-Ramírez

Leiden-BNU Summer School

9 July 2018

What is AMUSE?

Astronomical Multi-purpose Software Environment

> *Our aim is to provide a software framework for astrophysical simulations, in which existing codes from different domains, such as stellar dynamics, stellar evolution, hydrodynamics and radiative transfer can be easily coupled.*
>
> – *http://amusecode.org*

Origins of AMUSE

- Monolithic codes: NBODY6 (gravity), EVTwin (stellar evolution)
- Ideally: combine these two. But, difficult to achieve due to their monolithic nature…

Sverre Aarseth: *“Wouldn’t it be nice to have Peter’s stellar evolution code as a part of my beautiful N-body code?”*

Peter Eggleton: *“But, Sverre, my dear friend, how splendid would it be to have your N-body code as part of my stellar evolution code”*

Origins of AMUSE

(Future) AMUSE team: *“What about if* ***both*** *your codes could work* ***together****?”*

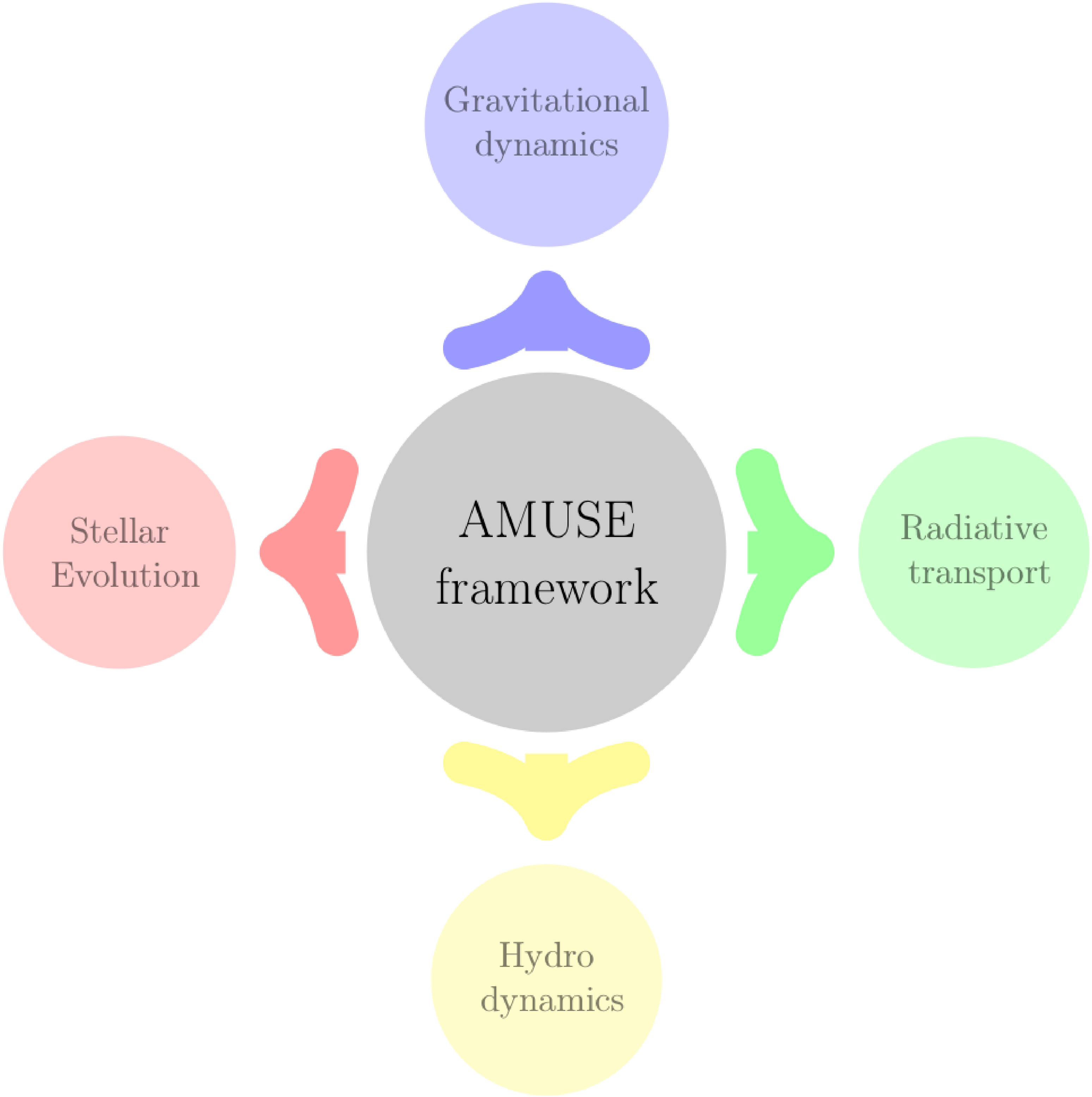
- Why redesign code, why rewrite any software if we already have the right tools at hand?
- Language-independent framework
- Codes would operate as modules that carry out specific tasks

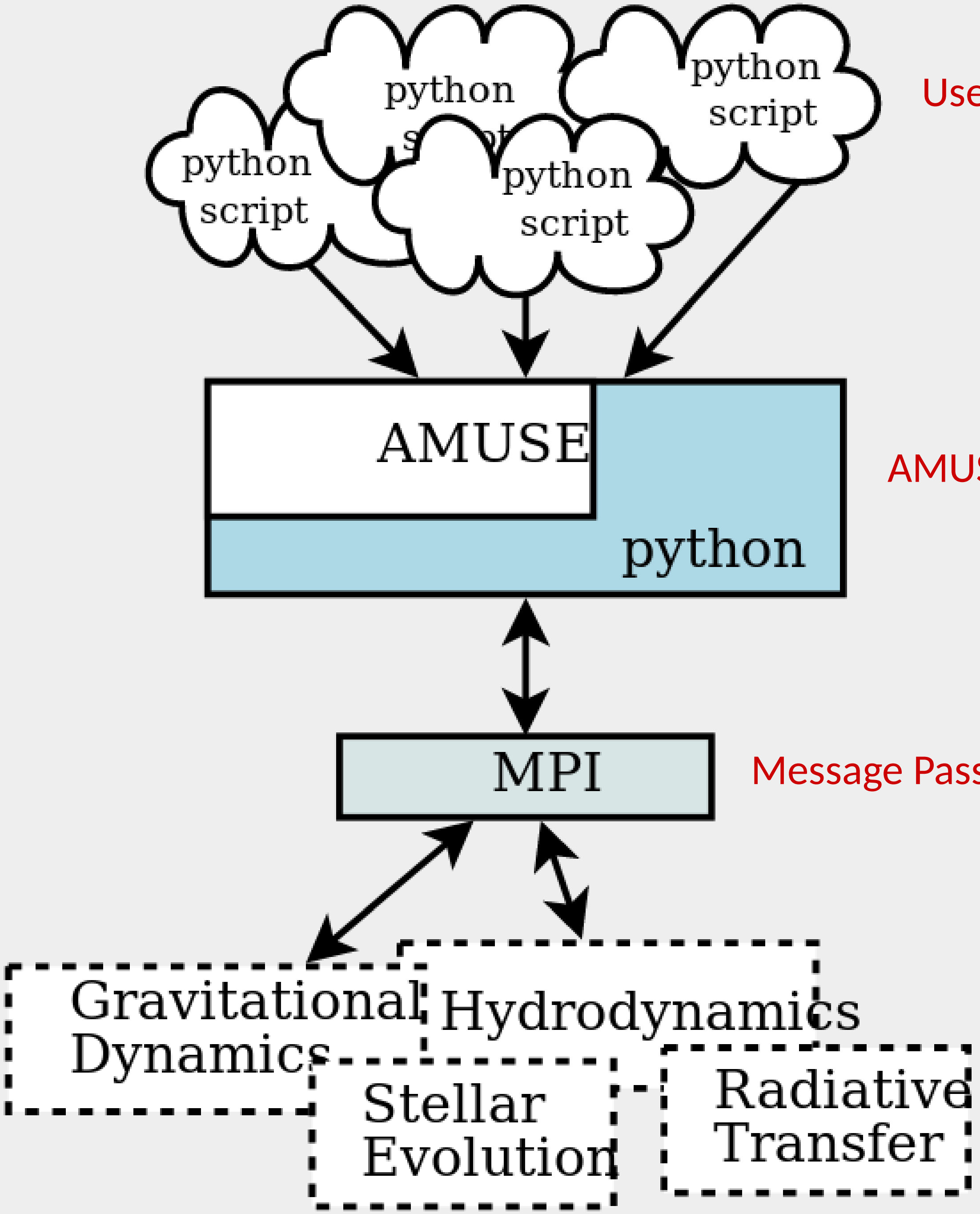
The AMUSE goals

(1) A homogeneous, physically motivated interface for existing astronomical simulation codes

(2) The incorporation of multiple community codes from four fundamental domains (stellar evolution, gravitational dynamics, hydrodynamics, and radiative transfer)

(3) The ability to design new simulation experiments by combining one or more of the community codes in various ways


Gravitational dynamics
Stellar Evolution
AMUSE framework
Radiative transport
Hydro dynamics

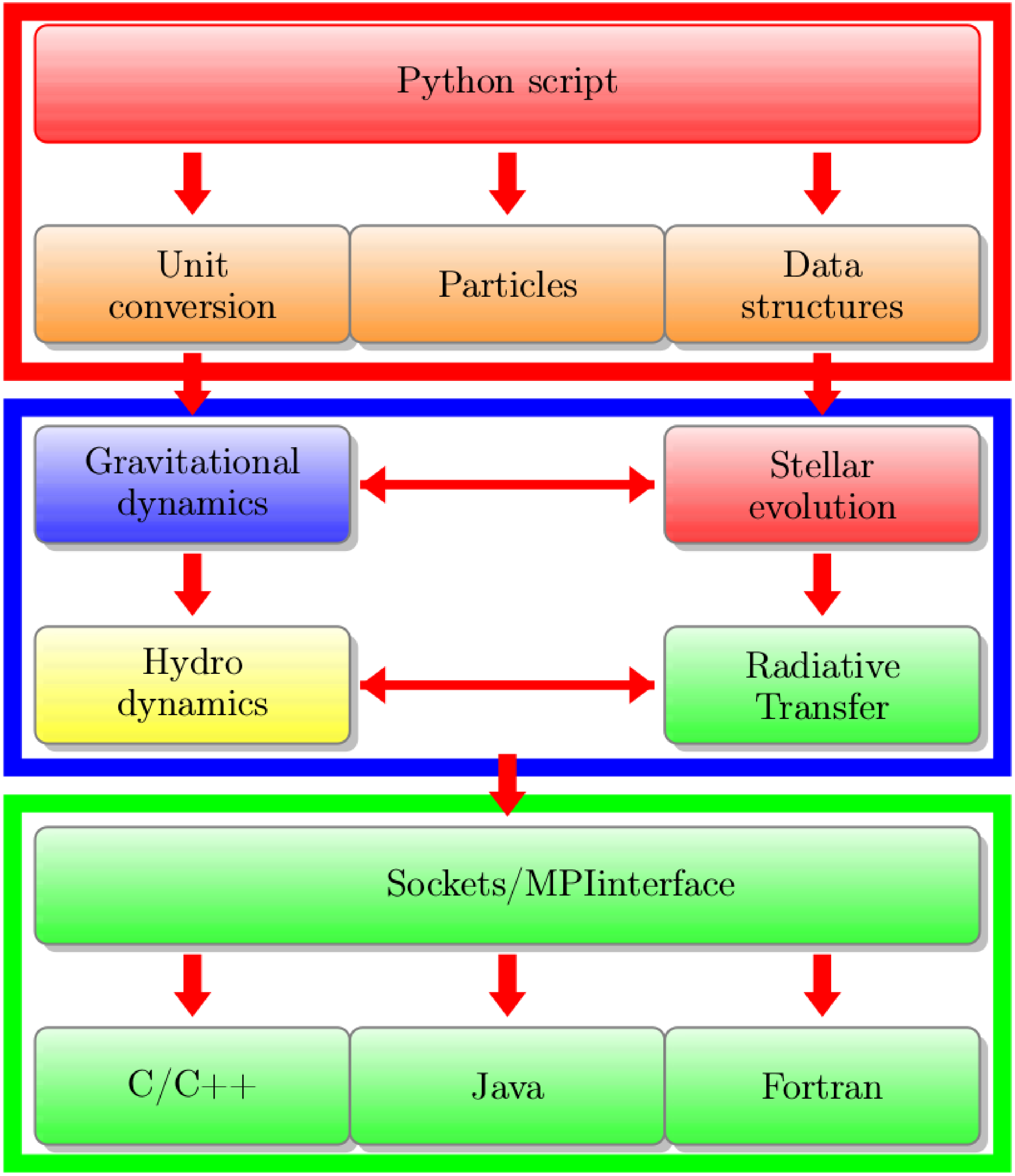


User level

AMUSE interface level

Message Passing Interface

Community codes



Python script

Unit conversion

Particles

Data structures

User level

Gravitational dynamics

Stellar evolution

Hydro dynamics

Radiative Transfer

Interfaces to community modules

Sockets/MPIinterface

C/C++

Java

Fortran

Community codes

Example: User script

```
masses = new_kroupa_mass_distribution(N, 100 | units.MSun)
converter = nbody_system.nbody_to_si(masses.sum(), Rvir)
stars = new_fractal_cluster_model(N=N, fractal_dimension=Fd,
                                    convert_nbody=converter)

stars.scale_to_standard(converter, virial_ratio=Qvir)
stars.stellar_mass = masses
stars.disk_mass = 0.01 * stars.stellar_mass
stars.mass = stars.stellar_mass + stars.disk_mass
stars.accreted_mass = 0 | units.MSun
stars.disk_radius = 400 | units.AU
stars.radius = 10 * stars.disk_radius
```

Example: User script

```
gravity = ph4(converter)
gravity.particles.add_particles(stars)
channel_from_gravity = gravity.particles.new_channel_to(stars)
channel_to_gravity = stars.new_channel_to(gravity.particles)

dt = t_end/10.
time = 0 | units.yr
while gravity.model_time < t_end:
    time += dt
    evolve_system_to(time, gravity, stars, stopping_condition,
                     channel_from_gravity, channel_to_gravity)
    write_set_to_file(stars.savepoint(gravity.model_time),
                      filename, 'hdf5')
```

More examples?

Clone repository, try playing with the examples on examples/textbook:

http://github.com/amusecode/amuse