# RAMICES III

$(JF)^2G$

April 20, 2025

## Contents

**1 Design Thesis** **2**

1.1 Oh no, here we go again . . . . . 2

1.2 Needs, Nice To Haves & Leave-Alones . . . . . 2

**2 Diagrams** **5**

# 1 Design Thesis

## 1.1 Oh no, here we go again

We're refactoring RAMICES. Again.

Why? Why would we do this to ourselves? What on God's Green Earth would drive us to such extreme lengths, when we could be doing so much else with our lives?

The long story short is that more than 3 years have passed since it was written: we've both learned a lot. There's also been a bit of space between its original design and the idea that it would have a lifetime outside of my thesis[1]

In particular, there was one big black mark that has hung over RAMICES II (`R2`) which was the stellar catalogue synthesis: Jenny has since found a whole bunch of stuff that's basically held together by hope and good intentions - or simply not held together at all.

### 1.1.1 Don't Be So Dramatic

I'm being deliberately hyperbolic here. It's worth noting in writing that I don't expect this to be anywhere near as painful as the original Great Refactoring. `R2` is leagues ahead of its predecessor in terms of being able to be read by a mortal mind without causing it to fragment into a trillion tiny pieces.

If all goes well, this will be a nice quick project that results in a vastly superior end product. Make a wish, children!

Some of that means doing some up front work, however, which is what this document is designed for.

## 1.2 Needs, Nice To Haves & Leave-Alones

This section is therefore designed to lay out some basic things we're trying to accomplish here - rather than just taking things apart for the sake of taking them apart[2].

To that end, I'm setting forth 3 categories:

### 1.2.1 The Needs

A Need is some aspect of the refactor, be it functionality, property, upgrade or anything else on the 'to-do list', which we consider essential, and which the project is not considered complete until it is implemented. Needs fix fundamental flaws in the previous iteration.

1. **Temporal Binning:** Next section details a major overhaul of the current mass-binning of stars. This is the main driver of the refactor, and is therefore a Need
2. **Stellar Catalogue:** I have never been happy with the catalogue synthesis. It was always considered a temporary solution. If we're Doing Things Properly, I would consider it necessary to redesign this section.
3. **Better modularisation:** A key design principle of `R2` was modularisation. This has been successful to an extent, but a lot of stuff was still hardcoded in at the last second.

[1] My thesis made some grand claims about it being a general purpose code. Lies.
[2] Ask me about 'The Pig Torch incident' some time.

   (a) **NSD:** This is more of a 'don't lose it', because Jenny has already put a lot of work into this bit!
   (b) **More Flexibility:** Less hardcoding of core functions (like the IMF) so that we can have them as toggles.
4. **Better Output:** This encompasses a lot of things, but a better way to detect what's going on inside is important.
   (a) **Logging Framework:** Should be easy to replace cout calls with a LOG entity similar to that used in GenCHORD
   (b) **Archiver:** No more dozens of loose files! The GenCHORD archiver should make packaging up our output much nicer
5. **Better makefile:** That makefile is an abomination wrought of lack of knowledge, and only persists because it still works (somehow). I have a better makefile I use now, I will retrofit it in.
6. **Documentation:** I think I already have a doxygen auto-setup in place, should expand on that & actually have some documentation

#### 1.2.2 The Nice To Haves

Nice-To-Have is something which is not considered critical, and which we could 'ship' (i.e. write & publish) without, but which we can add if we have the time. Nice To Haves are things you distract yourself with when you don't want to do a Need, but which can be ignored if needs be.

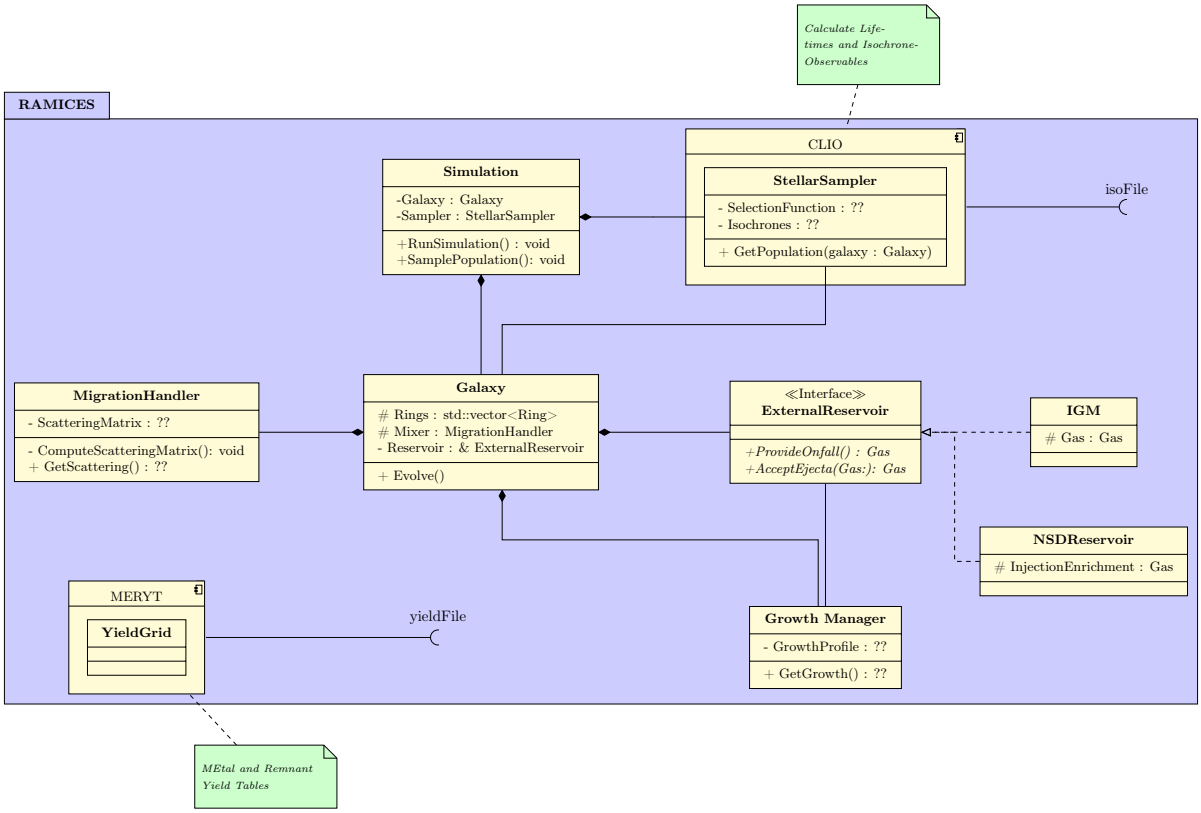
1. **Remove Streams:** One-upon-a-time it seemed useful to differentiate chemicals by stream-origin (i.e. stellar, accreted etc). However this just adds overhead to any Gas Operations, and I don't think it really adds anything. Strip it out.
   (a) **Tracer pseudo-Elements:** We could think about adding in a 'tracer element' option? This is speculative at this point
2. **Config Interface:** As `R2` has grown, the config files have sprawled. A nice GUI would make configuring much easier
   (a) **JSL::Argument Rework:** JSL::Argument currently only accepts configs OR command-line arguments, which means a unique config file for each launch – problematic if we launch an iterative bunch. Rework it so it uses config *first*, then overwrites it with CLA.
   (b) **Write the GUI:** Use the framework to actually write the GUI – should be only a few lines of code per-parameter, but we have a lot of parameters!
   (a) ~~**GUI Framework:** Write a generic GUI framework for writing JSL-compliant config files~~
3. **Unit Testing:** Or some other kind of testing framework so that we can run tests & check that updates haven't broken anything
   (a) **Interface/Layer Design:** This is a fairly major shift in design principles (see below). But would make the code much easier to test.
4. **Variable Timestep:** Smaller timesteps are often important in the early galaxy, whilst at late times you're just wasting CPU cycles. Being able to 'speed up' the simulation at later times would be quite pleasing.
5. **Variable Ringwidth:** Variable width is presently supported by the code, but there is no actual method for a user to propose a non-uniform ring width
6. **Isochrone Warning Flags:** Check if the isochrone density/lifetime grid is poor around a proposed simulation resolution, and produce a warning that this can alter the expected CCSN resolution by several timesteps.

### 1.2.3 Leave-Alones

This is stuff which – upfront – we decide we are going to leave alone, and not attempt. `R2` is perfectly functional in many ways, and there's no need to reinvent the wheel. There might have to be some minor refactoring in terms of putting the jigsaw pieces back into place, but the fundamental algorithms and large-scale functionality should remain exactly as it was. Leave-Alones are things which – where possible – are simply copied-and-pasted from `R2`.

1. **Chemical Yields:** I think this section is probably fine. It reads in external data and creates a yield grid. I don't think we need to touch that bit.

# 2 Diagrams




(Note: + is public, # is protected and - is private)