Building modules in SpaDES

Alex M. Chubaty
Natural Resources Canada, Pacific Forestry Centre
email: achubaty@nrcan.gc.ca

Eliot McIntire
Natural Resources Canada, Pacific Forestry Centre
email: emcintir@nrcan.gc.ca

July 29, 2014

Contents

1 Introduction **2**
1.1 Module overview . . . 2
1.2 Events . . . 2
1.3 Objects . . . 2

2 Creating a new module **3**
2.1 Module design considerations . . . 3
2.2 Using the module template . . . 3

3 A simple landscape module **4**
3.1 Modelling forest succession . . . 4

4 A simple spread module **5**
4.1 Modelling fire spread . . . 5

5 A simple agent module **6**
5.1 Modelling caribou movement . . . 6

1 Introduction

1.1 Module overview

`SpaDES` modules are event-based, meaning that different actions (calculations) are performed on data objects based on the order of scheduled events. Basically, a module consists of a collection of events which are scehduled depending on the rules of your simulation. Each event may evaluate or modify a simulation data object.

1.2 Events

Simulation event list Lorem ipsum ...

Module events Lorem ipsum ...

Module event dependencies Typically, each module schedules its own events (e.g., a "fire" module may schedule "burn" events) and only uses its own data objects. Modules that behave in this way are indepedent of each other, and this is generally the prefered way to design and implement modules. A module that schedules events from another module is said to depend on that module. Module event dependencies compilcate the construction of simulation models, and hinder the ability to develop and deploy models with modularity. If two modules are actually depedent on each others' events, then you should consider whether they really are separate modules or should be merged into a single module.

1.3 Objects

As you build your module / simulation, you can use any of R's data types to store your objects / data. In particular, matrices (including vectors) and lists work well for this purpose because they are pass-by-reference, which reduces your model's memory footprint and speeds up your codes execution. Other useful datatypes include Raster* and SpatialPoints* objects.

Global objects Use ■- to assign global objects to reduce copying large objects (such as maps), which slows model execution.

Module object dependencies As noted above, modules can depend on one another for event scheduling. Modules can also be design to rely on outputs (data objects) from other modules. A module that relies on a global simulation data object that previously used by another module is said to be dependent on that other module. It is often useful to develop collections of modules that interact indirectly and are dependent in this way. Note that modules need not be inter-dependent on one another: module B may depend on module A (for example to initialize a data object), without module A depending on module B.

2 Creating a new module

2.1 Module design considerations

perhaps a bit about design phlosophy, relating back to dependencies and how to carefully build modules that actually retain modularity.

2.2 Using the module template

...

3 A simple landscape module

3.1 Modelling forest succession

...

4 A simple spread module

4.1 Modelling fire spread

...

5 A simple agent module

5.1 Modelling caribou movement

...