# Standard Model of Scientific Computing

**All users must do these things...**

### Define the Problem

Write an input file in a format reminiscent of a dead language

### Run the Simulator

Manually launch jobs on impressively terrifying machines

### Analyze Output

```
01100010
01101001
01101110
01100001
01110010
01111001
```

Analyze simulation output in its most raw and unlimited form

### Archive Output

Store data... somewhere!

***Super-users think these are easy tasks, but most users are overwhelmed!***

# A cooler model of Scientific Computing

**Define the Problem**

Write an input file in a format reminiscent of a dead language

**Run the Simulator**


Manually launch jobs on impressively terrifying machines

**Analyze Output**

01100010
01101001
01101110
01100001
01110010
01111001

Analyze simulation output in its most raw and unlimited form

**Archive Output**

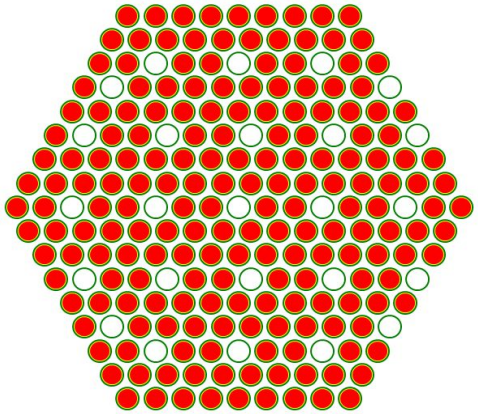
Store data... somewhere!

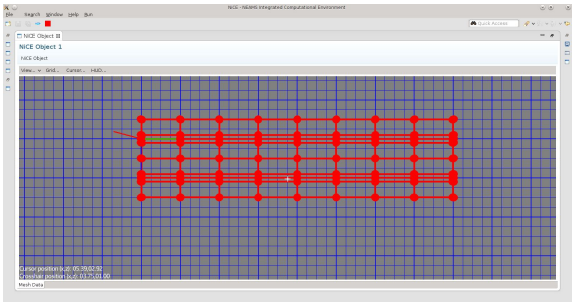
It would be better to have a computer program handle all of that...

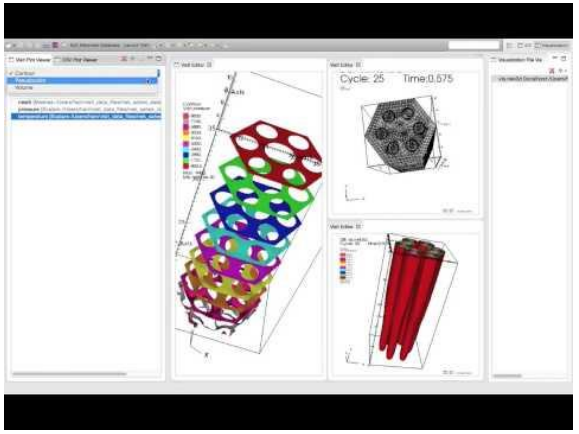
ice

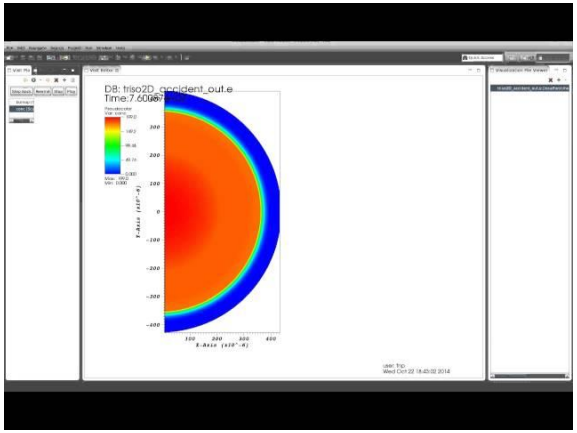
***Most of the stuff we need to do can be encapsulated for ease of use and/or automated entirely with improvements.***

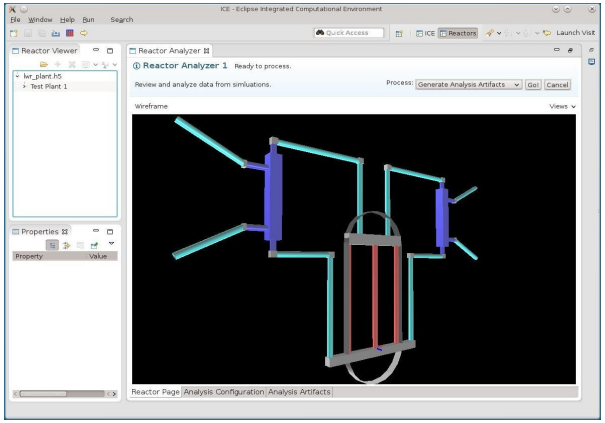
# What can it do in 9 pictures or less?

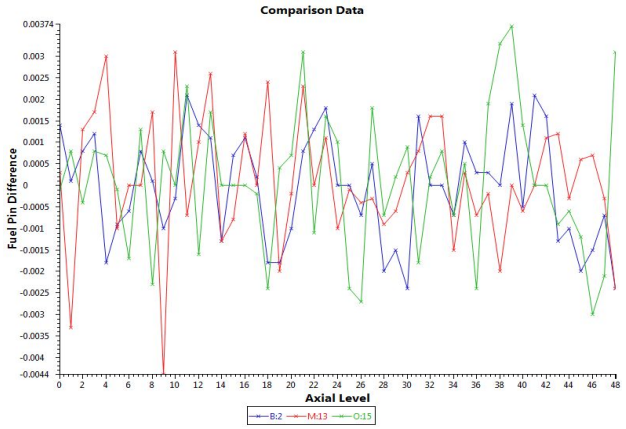


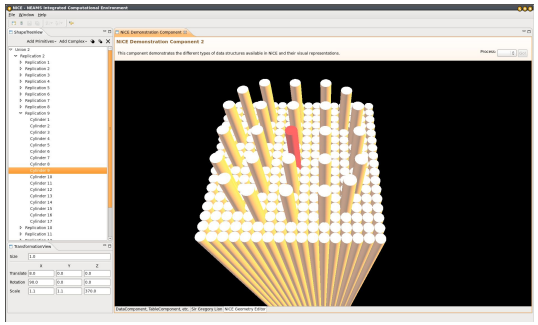


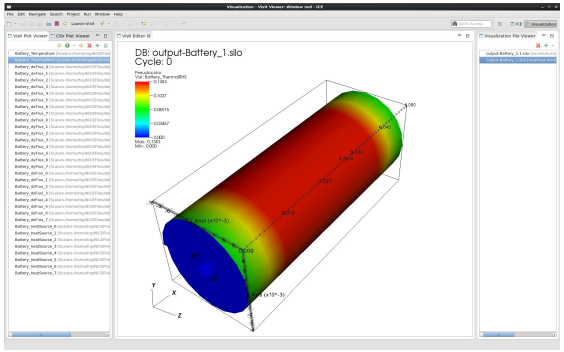


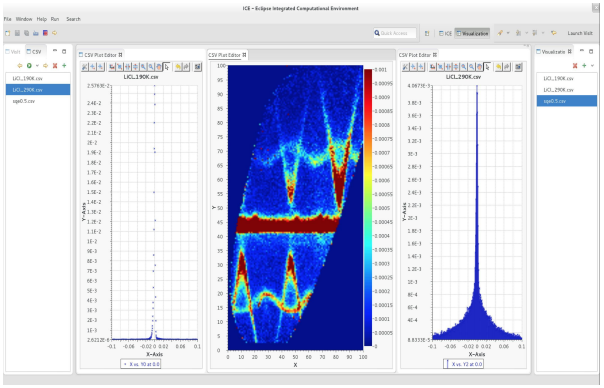





# Where does it work?

Nuclear Energy

Data Analysis

Batteries


Advanced Manufacturing

More 3rd Party Tools

*Coming in FY16!*

Advanced Materials

Quantum Computing

Astrophysics

Basic 3D Geometry and 2D Mesh Editing

# Usability in Modeling and Simulation

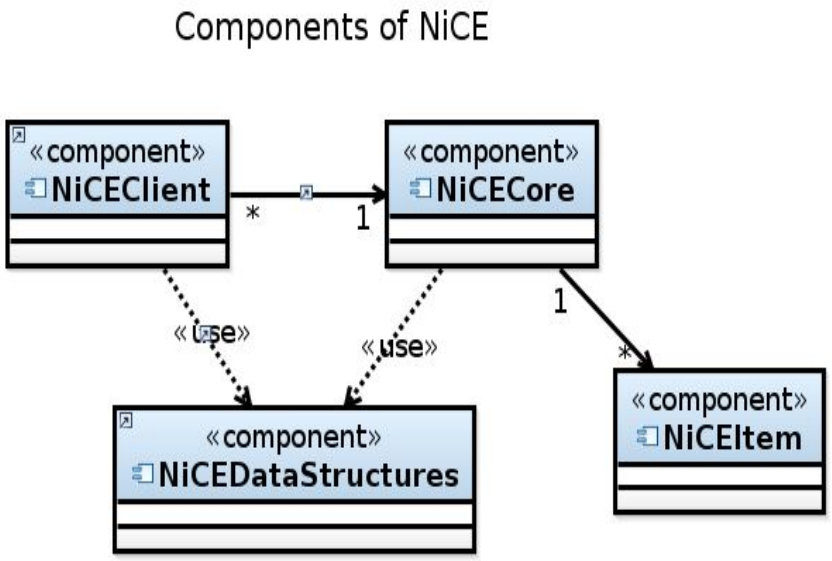
How I send messages around the World:


How I send messages to my code:

*./xolotl ../benchmarks/he-W_2067.txt --handlers dummy --petsc -ts_final_time 1000 -ts_final_time 1000 -ts_adapt_dt_max 10 -ts_max_snes_failures 200 -pc_type fieldsplit -pc_fieldsplit_detect_coupling -fieldsplit_0_pc_type redundant -fieldsplit_1_pc_type sor -ksp_monitor -ts_max_steps 3*

***Really?!***

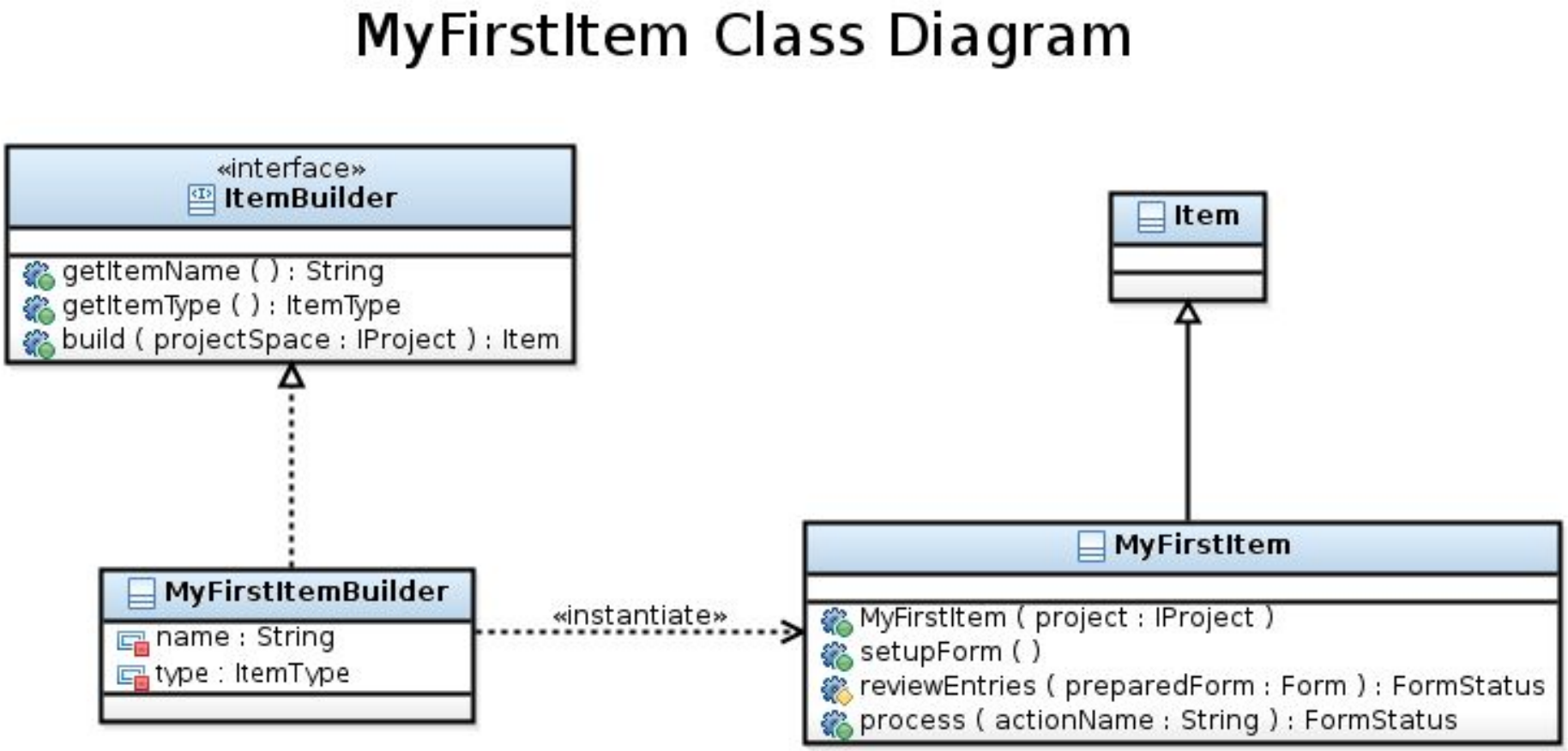
# How does it work? Plugins!



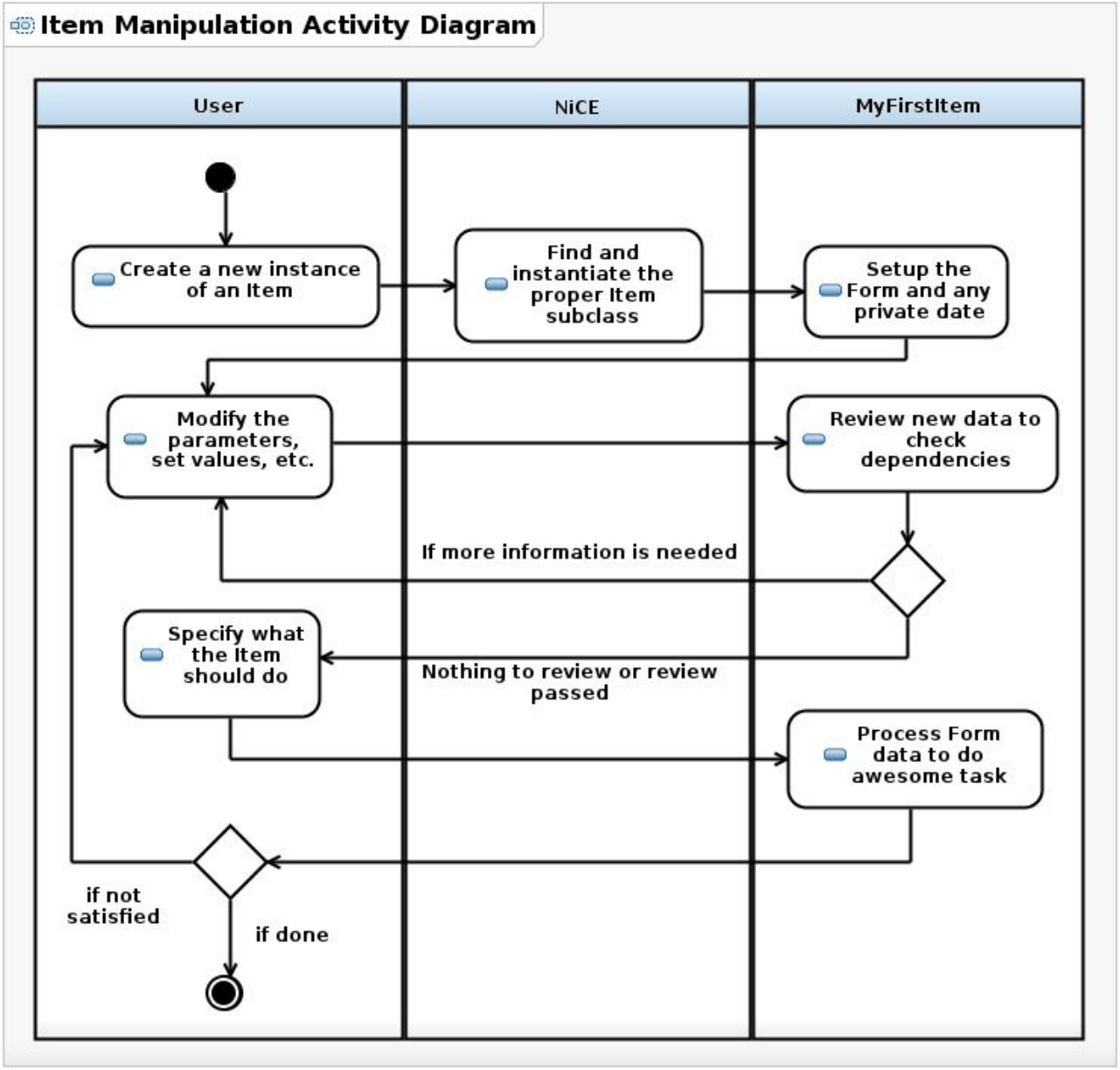

Plugins are:

- Dynamic Services - Completely reusable components!
- “Item” Subclasses - Most of the work is already done by the platform
- Self-contained, business logic - **ONLY** your code, not UI, etc.
- Tools - Reusable components, tools, or things other

# How does it work? Part 3



# How does it work? Part 4



# How does it work? Part 5

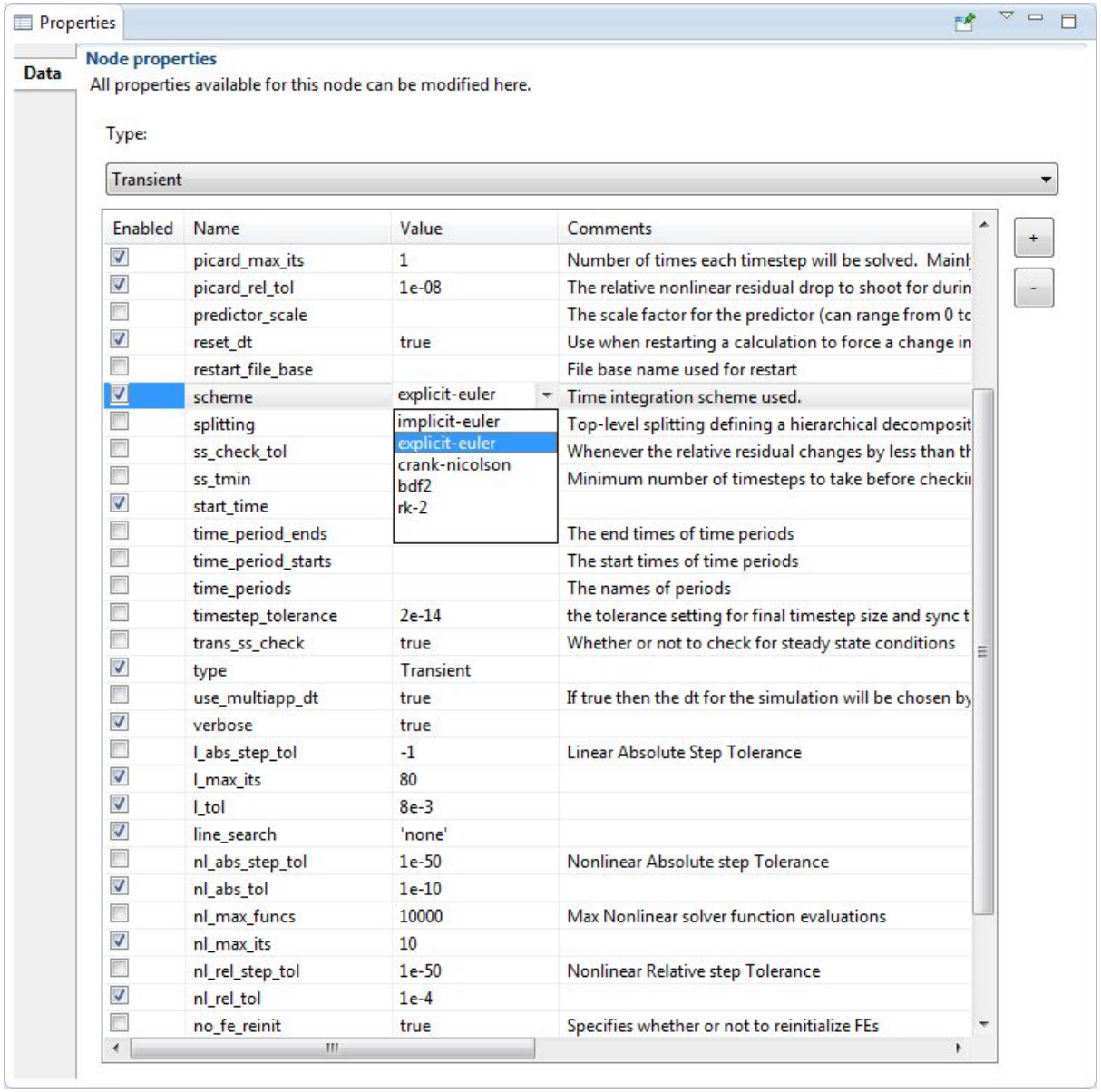
Things to keep in mind:

- You only write business code
- UI and marshalling are provided by the platform (unless you want to extend it)
- Codify only what is needed; reuse what you already have (preprocessors, etc.)

# How does it work? Part 6

All of the data structures are backed by sophisticated tools so you deal with your domain.

***Standardization for the win!***



# Different views of the same data

```
<Entry defaultValue="1.7899" ready="true" changeState="false">
    <AllowedValueType>Undefined</AllowedValueType>
</Entry>
```

```
entry1 = new Entry() {
        protected void setup() {
            allowedValues = new ArrayList<String>();
            allowedValues.add("0");
            allowedValues.add("50");
            defaultValue = "1";
            allowedValueType = AllowedValueType.Continuous;
        }
};
entry1.setName("Generic 1");
```

*All of these are logically equivalent because of the standardization!*