Tom Atnip
Jeremy Tramm
George Mammarella

Scientific Computing

What does quality mean in your selected domain?
Making sure that you accurately represent that mathematical model you are trying to simulate.

What are the important factors?
Less emphasis may be on verifying output, instead focusing on creating accurate models. This is due to the fact that often times, what output is going to be produced can be unpredictable. Though accuracy of the results is always more important than being on-time, under budget, maintainable, or being highly usable. Accuracy of the model depends on the level of understanding provided by the scientific community. Another measure of accuracy is the ability to reproduce the process, inputs, and results in multiple trials.

What processes are currently used?
1. **Metamorphic Testing** – Given an algorithm that does not have predictable output, making tests such that you can still find defects. You use properties of functions so that it is possible to predict the expected changes to the output based on the changes in the input.
2. **Science Code Manifesto** – An unofficial standard for using and publishing research in a way that allows it to be the most open to the public.
3. **Formal Methods** – Analyzing source code and formally verifying that code will run as expected.
4. **Code Inspections** – Some sort of standard process to make sure that the code being written is being developed correctly.

What processes do you suggest?
There should be a domain expert on the model that can observe the development of the computer model to ensure that it is being developed correctly. This expert should perform code inspection/review which focuses on the principle logic instead of good coding practices. There should also be a Software Engineer expert to do regular code reviews to check code quality. Also, if the model is expected to represent the interactions of several actors, testing should be done on each actor individually to ensure that each actor is acting as intended before putting everything together.

Are there any regulations/standards?
There are programing languages such as matlab that are widely used for floating point computations. Matlab follows IEEE Standard for Floating-Point Arithmetic (IEEE 754). This standard defines arithmetic formats, interchange formats, rounding rules, operations, and exception handling.

Examples of Success and Failure:
In general, success or failure is dependent upon the accuracy of the model. A computer model accurately representing the scientific one is a success. Otherwise, it is a failure.

Tom Atnip
Jeremy Tramm
George Mammarella

What are common problems associated with quality assurance in the domain?

- Scientists are protective of the model; “Developers, don’t touch my model!”
- Complex output is difficult to check and depends on many factors
- Limited oracle data (oracle data is “correct” data to check test results against)
- Scientists do not consider the QA of the software, they only care about the scientific model
- Correctness of the model results is paramount to all other factors
- Software QA needs to not impede scientific progress; time spent on software is wasted time to the scientists

What metrics are used?

According to some sources, metrics are underused, often non-existent. Other sources cited use of very simple or outdated metrics (only LOC or GOTO count). Some notable and/or current metrics are:

- LOC
- Cyclomatic Complexity
- Knot Count
  - Knots are points in the code where control flows/branches intersect
  - Can point out areas of extra complexity (for human readability and computation)
- Utilization of Resources
  - Identifies hardware bottlenecks, etc.
- System Uptime
- Individual project program scalability and efficiency
  - Metric which can show scalability of a project based on a ratio of performance on *n* cores and performance on *one* core.

What metrics do you suggest?

- Method execution time
  - Little inefficiencies in often-used methods can have big impacts for complex systems
- Bugs per LOC
- Code coverage
- MLOC
  - Should break logic into reasonably sized chunks
  - Logic should be partitioned into methods
  - Easier to test many small methods in such a large system