Software Engineering for Economists*

1 Building Confidence in a Model

- Computational models of socio-economic phenomena are a manifestation of our perceived knowledge about the underlying processes. They key question is how much confidence should we have in a particular model?
- It turns out to be useful to structure such a discussion around three interrelated questions Council (2012).
- Software Engineering encompasses the tools and methods for defining requirements for designing, programming, testing, and managing software. It is crucial to ensure that the computational implementation is a faithful representation of the original mathematical model William L. Oberkampf (2010). Thus, it is part of the verification step.

2 Research Example

3 Running Example

- For the rest of this lecture, we will use a small examples to illustrate ideas of different software engineering tools. However, we will also have a brief look how these tools are applied in the more complex setting of my current research. The online code repository is available online.

*For further information or questions and suggestions, please contact us at info@policy-lab.org.

4 Testing

- To see these basic ideas in action, let us check out the testing harness for my current research project online.
- Using bugs to define test cases ensures that they only need to be fixed once.

5 Profiling

- Now that we have a well designed and tested version of our code, it is time address any performance issues. We will profile our program by measuring the execution time of the program.
- Profiling tools also measure the time spend in each function allowing us to target our development efforts at particularly time-consuming parts of the code.
- Studying the output directly can be rather tedious for large programs. That is when visualization tools turn out very useful. We build on SNAKEVIZ.
- For even more advanced visualization, check out pyprof2calltree. Tutorial for advanced visualization using KcacheGrind.

6 Continuous Integration Workflow

- By running the testing harness early and often, bugs are caught closer to their creation. This makes debugging much easier.
- Scalability of research team is improved as basic quality assurance is automated.
- The badges signal to your fellow researchers that we take your responsibilities as a developer of research software serious.
- Reliable work-flow increases own satisfaction.

7 Best Practices

- Iterative project development with only incremental addition of features. Testing harness ensures that old features are not broken.

References

Council, N. R., editor (2012). *Assessing the Reliability of Complex Models: Mathematical and Statistical Foundations of Verification, Validation, and Uncertainty Quantification*. The National Academies Press, Washington, D.C.

William L. Oberkampf, C. J. R. (2010). *Verification and Validation in Scientific Computing*. Cambridge University Press, Cambridge, England.