Packing Experiments for Sharing and Publication

Fernando Chirigati
NYU-Poly
fchirigati@nyu.edu

Dennis Shasha
NYU
shasha@cs.nyu.edu

Juliana Freire
NYU-Poly
juliana.freire@nyu.edu

The Need for Computational Reproducibility

- The standard of having reproducible experiments, a long tradition in natural science, has not been adopted for computational experiments.
- Researchers often have to rely on figures, plots and tables presented in papers, which loosely describe the results. Consequently, these results are difficult to reproduce, leading to a credibility crisis in computational science [1].
- There are two main reasons why reproducibility is often not adopted:
  - Authors find it difficult to generate a compendium for their experiment,
  - Reviewers and collaborators have difficulties trying to reproduce and verify the results, even when the compendium is available.

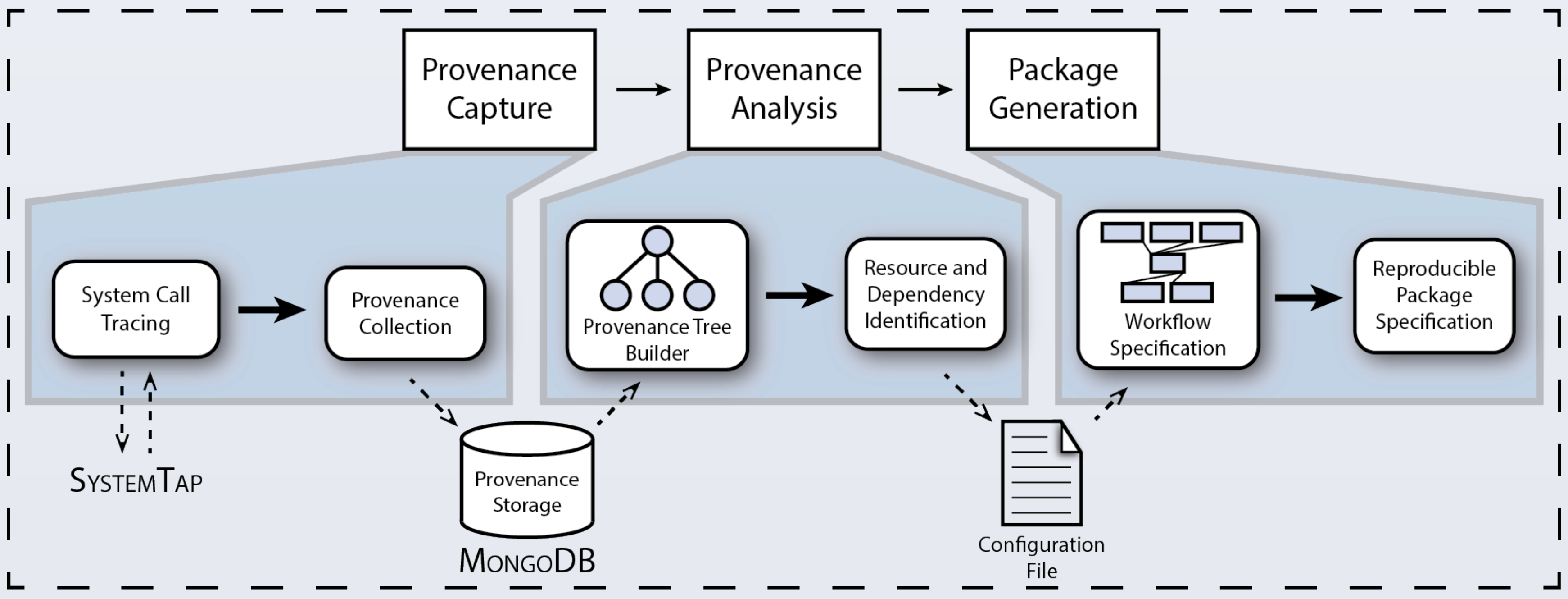
ReproZip

ReproZip is a *general* tool for packing reproducible research that

- tracks operating system calls and creates a *reproducible package* from *command-line executions*, in the author's environment *E*, with all the required files to run the experiment (***packing* step**). Authors do not need to port their experiment to a specific tool.
- generates a workflow specification to help reviewers and collaborators explore and verify the results, facilitating the review process.
- extracts files and workflow on another environment *E'* (***unpacking* step**).

Packing Experiments

Packing Step



Provenance Capture

- ReproZip transparently captures the provenance of the execution of the experiment. It uses SystemTap [2] to trace system calls and capture all the required information to correctly reproduce the experiment.
- The execution trace is stored in MongoDB [3], a NoSQL database

Provenance Analysis

- ReproZip uses the trace data to create a *provenance tree*. Each node in the tree stores information about an OS process, such as *files read*, *files written* and *command-line arguments* – if a process *p* spawns a process *p'*, an edge is inserted between their corresponding nodes.
- The provenance tree is traversed to identify *programs*, *input files*, *output files* and *dependencies*.

Package Generation

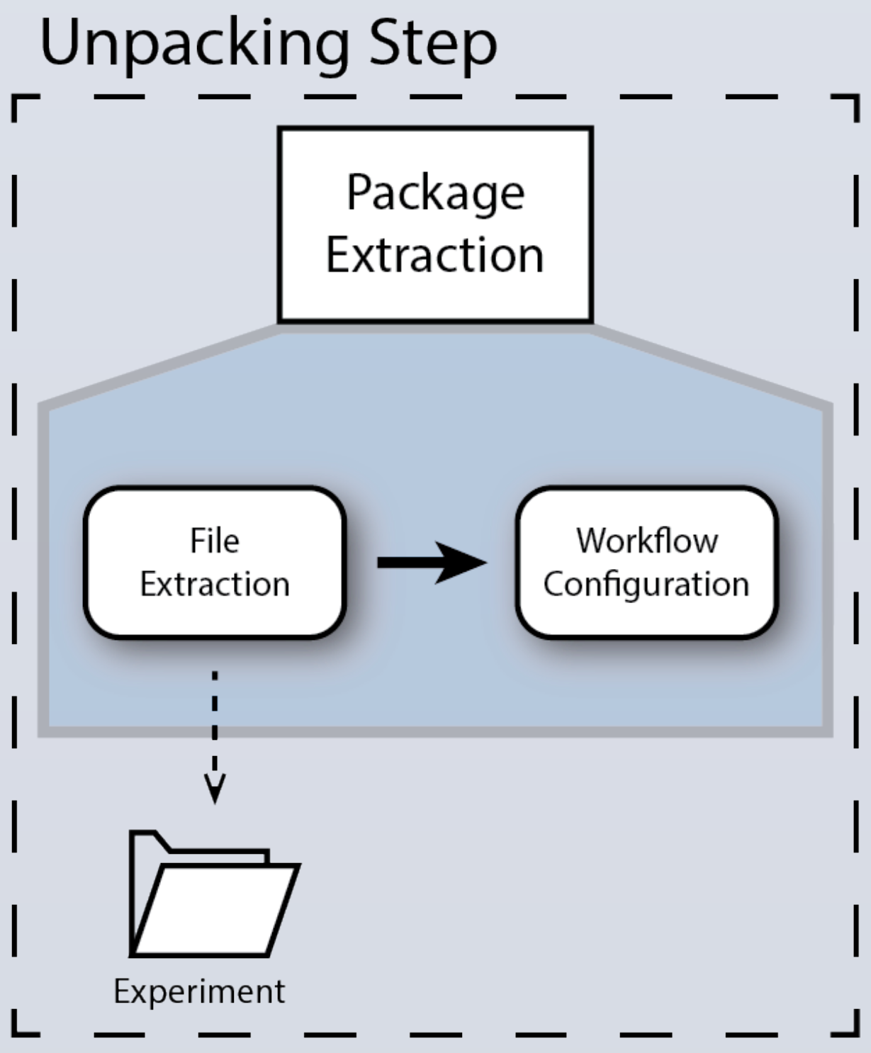
- The identified binaries and files are used to derive a workflow specification for the experiment.
- All the required files, together with the workflow, are packed on the author's environment *E* using the *original* directory structure.

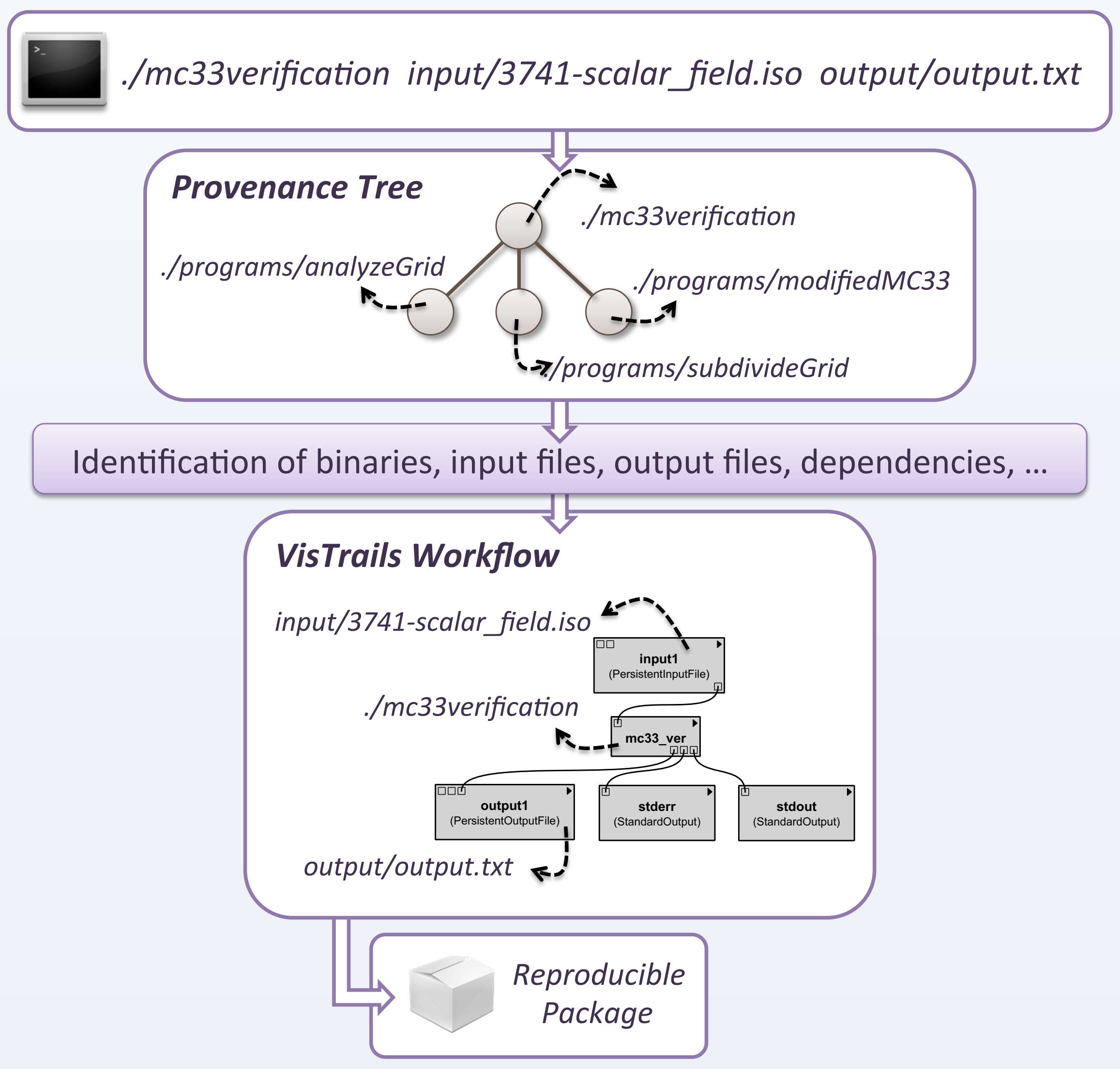
Unpacking Experiments

Package Extraction

- Given a reproducible package generated on environment *E*, ReproZip extracts all the files in a single directory on environment *E'*, without interfering with this environment.
- The workflow is automatically configured to point to the correct files inside the unpacked directory.
- Paths defined in environment variables for the experiment are adjusted to use the experiment directory in *E'*.

Unpacking Step



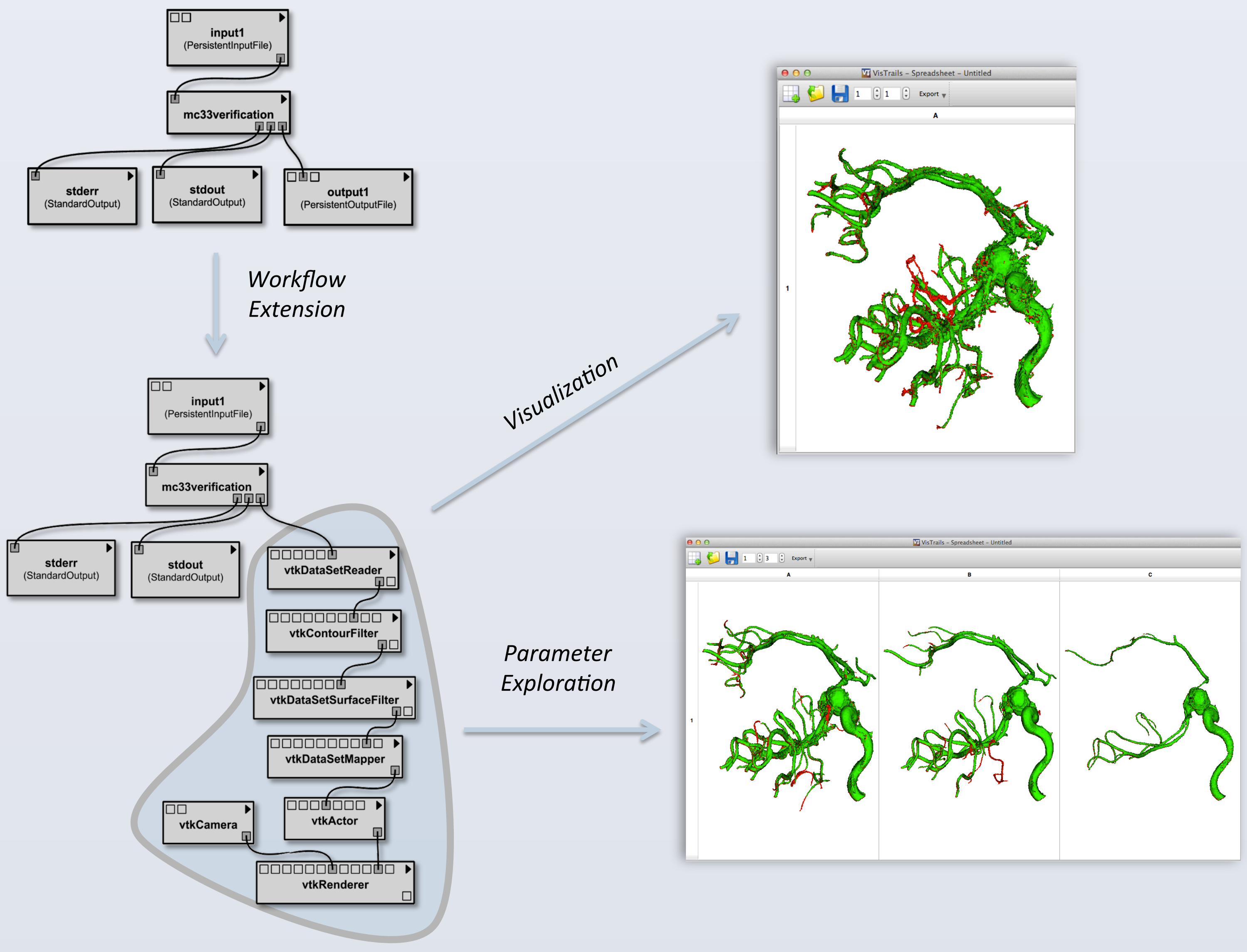


```
./mc33verification  input/3741-scalar_field.iso  output/output.txt
```

Packing an experiment on topological correctness of marching cubes [4]

Verification and Exploration

- Users may run the experiment from the command line and examine the results.
- ReproZip also creates a workflow specification that can be run within VisTrails [5]. By using this workflow: (1) experiment execution is straightforward; (2) a visual representation helps the reviewer to understand and explore the experiment; (3) users can try different parameters, as well as perform parameter sweeps and compare the results side by side; (4) users can extend the original experiment and perform additional analyses; and (5) the provenance of the verification process is automatically captured by VisTrails



Verifying the topological correctness of a marching cubes algorithm. The reviewer can extend the workflow to visualize the results; they may also verify the robustness of the algorithm by exploring different isosurface values using the parameter sweep feature of VisTrails.

Acknowledgments

We thank Jesse Lingeman for feedback on our initial ideas and for providing the molecular biology experiment we use in our demonstration. We also thank Lis Custódio and Tiago Etiene for providing their marching cubes experiment. This work was supported in part by the National Science Foundation (grants CNS-1229185, CNS-1153503, IIS-1139832, IIS-0905385, IIS-1142013, IOS-0922738, MCB-0929338 and MCB-1158273) and the National Institutes of Health (grants GM 32877-21/22 and 2R01GM032877-25A1).

References

[1] D. Donoho, A. Maleki, I. Rahman, M. Shahram, and V. Stodden. Reproducible research in computational harmonic analysis. Computing in Science & Engineering, 11(1):8{18, Jan.-Feb. 2009.

[2] SystemTap. *http://sourceware.org/systemtap/*

[3] MongoDB. *http://www.mongodb.org/*

[4] Lis Custódio, Tiago Etiene, Sinesio Pesco and Cláudio Silva. Practical Considerations on Marching Cubes 33 Topological Correctness. Computers & Graphics, 2013.

[5] J. Freire, D. Koop, E. Santos, C. Scheidegger, C. Silva, and H. T. Vo. The Architecture of Open Source Applications, chapter VisTrails. Lulu.com, 2011.