Motivation
Joy and Confidence in Development

The Harness project was conceived while reengineering a multi-faceted, mathematically complex, and computationally intensive algorithm at NOAA’s National Centers for Environmental Information.

The Pairwise Homogeneity Algorithm (PHA) is responsible for correcting faulty instrument response functions in the global-scale datasets which serve as the primary historical temperature records of the United States. The algorithm was carefully designed with the primary goal of being as scientifically defensible as possible. Its code and the results it produces must conform to published standards established through rigorous peer review.

After a time in active service, PHA was scheduled for code level reengineering by a small team of software engineers in order to improve clarity, maintainability, and technology. The team was tasked with making improvements while reproducing expected behavior exactly.

In practice, PHA spans a multitude of files, directories, modules, and scripts. Throughout the reengineering effort, established development strategies were employed to maintain quality, including registering the project with modern version control and issue tracking, creating commons libraries for logging and unit testing, deploying automated continuous integration, and regularly submitting code for review by an independent quality assurance team.

Apart from standard developmental issues, the project presented some novel challenges that had no obvious existing solution. Notably, the enormous scale (in both the time the algorithm took to run using a full dataset and the large number of input states) coupled with a high spatial resolution, along with the need for comprehensive domain coverage to ensure results were reproduced exactly, required a tool that could build and maintain a large number of varied data sets with compatible bases for comprehensive comparison using flexible, reusable methods that could be quickly defined, redefined, and shared.

A broader examination of the domain as a whole reveals PHA is an instance of a more general class. NOAA and similar agencies have a large and increasing set of projects with similar characteristics to PHA, motivating a solution that is generic enough to be quickly configured for arbitrary projects and data.

Implementation
Concept and Technology

At the beginning of development, the needs of the system were identified. The system needed to be:

- Available for deployment on various systems
- Uniform across deployments
- Shareable and useful as a collaborative tool
- Allowed to deploy on machines with tight security
- Written clearly and concisely for fast development
- Built on persistent, immutable, highly flexible storage
- Attractive, simple, fun

After a thorough investigation, Python emerged as the clear winner to serve as the primary tool in building the back end of the Harness system. The primary factors were:

- Dynamic typing and support for a functional style for fast and clear development
- Simple built in libraries for handling large amounts of file system and operating system level manipulation
- Intuitive access to document storage with PyMongo
- The ability to compile and run Fortran code with f2py
- Unobtrusive web controller options (flash and twisted)
- Python dictionaries (no custom serializers!)
- A well known language with trust from management
- An engaged and active community

The Harness front end was written as a cloud-enabled single page web application on top of the angular.js framework. With limited human resources, the initial time investment in proper setup pays off with a clearly defined framework and clear patterns of development, minimizing code debt and leading to an eventual overall efficiency improvement. Additionally, the web based UI allows for more freedom in use, including single user or shared deployments, with developer control over updates without laying hands on individual machines.

The goals for the UI were to be as simple and elegant as possible while minimizing obtrusions to the underlying data. D3.js was used to handle the majority of user interaction with configuration, setup and analytical data display, while open layers was chosen for to handle more persistent and specialized map work. Both systems use json data which translates effortlessly with the python dictionary data structures used by the back end.

Ryan Berkheimer
ryan.berkheimer@noaa.gov