ENGAGE TESTING

YAX Team

Evan Bolyen
Mike Deberg
Andrew Hodel
Hayden Westbrook

Contents

Introduction

The engage step is responsible for actually running the functionality embedded in the modules. All checking of variables and states has been accomplished in the init or prep steps and engage will go forward with the assumption that all input parameters, features, and dependencies are present.

The runid and artifacts have already been registered to the database. Engage will begin by evaluating the presence of absence of artifacts its final output is dependent on. Engage will then begin completing artifacts it finds necessary until the final output can be achieved.

Testing environment and purpose

- Testing pipeline
  - A “dirty DAG” has been created that will test multiple relationships that are expected in the final YAX pipeline.
  - These include but are not limited to:
    - Branching paths
    - Merging paths
    - Multiple independent modules
    - Multiple dependencies on one artifact
    - Multiple dependencies on one parameter
- Four test modules
  - Modules have minimal functionality and function only to show evidence of the state systems successful use of them
  - Each module creates at least one artifact
  - Each module creates a file in the working directory as evidence of its operation
- Six test artifacts
  - Artifacts will be used to show there completion is successfully registered to the database
  - .complete lock files are generated to show successful identification of a complete artifact
- SQLite
  - Artifacts are registered complete to the database to show successful update of table

Expected outcome and relevance

Two tests will be run to show that engage runs successfully in both situations. These include the case of a new run where all artifacts must be completed and the case of a run that is dependent on existing artifacts.

New Run:

- All artifacts are generated
  - Artifacts are registered to database
  - Artifacts are identifiable as complete based on .complete file existence

Divergent Run:

- Only necessary artifacts are generated
  - New artifacts are registered to database
  - New artifacts are identifiable as complete based on .complete file existence
  - Artifacts which can be reused are reused
    - Lack of database entries for new versions
    - No artifact evidence of new versions

YAX Testing Pipeline

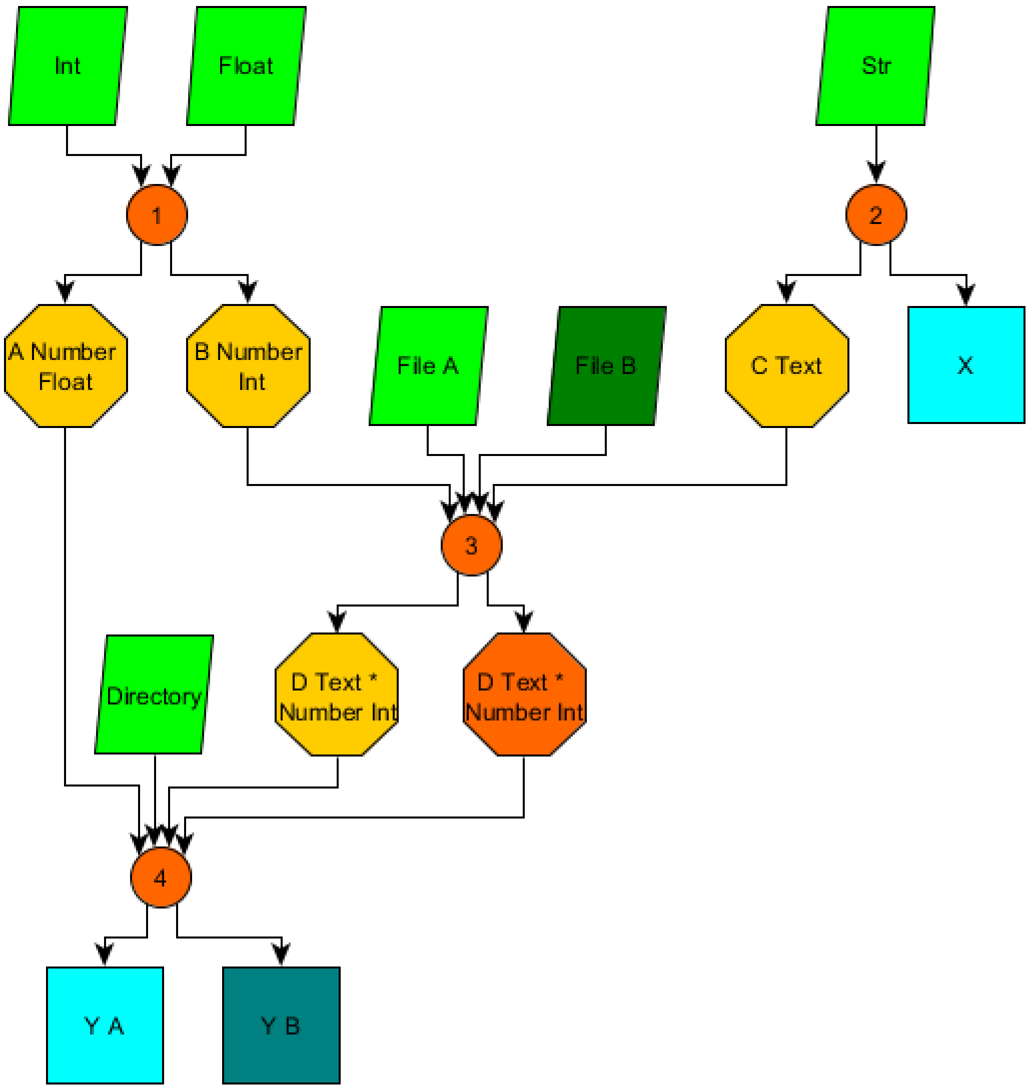

FIGURE 1 YAX TEST PIPELINE DESCRIPTION

In Figure 1 all five possible input parameters are utilized (represented in green). The four modules, 1, 2, 3, and 4, representing possible paths used in the YAX pipeline. There are two final output files, x and y, which are created at two different levels of the test pipeline (represented in blue). There are four internal artifacts, A, B, C, and D, which are produced and consumed by the various modules (represented in yellow).