New Documentation for PPAML-Eval-Tools

Philip Robinson

July 31, 2014

Goals and Requirements

It is our goal to provide a peval-toolkit that allows for us to human-efficiently compute and store quantitative metrics for team submissions to the PPAML program. In this most recent submissions iteration, there was a significant amount of time that lost to setup, configuration, and queueing executions of challenge problem solutions.

We have been asked by the teams to provide a more rigid acceptance criteria

Workflow

By better discussing the intended worakflow we can outline the changes that need to be applied to the present peval-toolkit.

Terminal

Present

The ini files beeing addressed/created do not correctly allow for seperation of PPS and CPS. Many of the submissions were delivered such that seperation of the PPS from CPS is not possible. This prevents us from performing progress comparisons.

There were also no tools for quereying the database for valid input variables, so teams must know their id values without help (or by using sql queries).

The present model could be extended to have a cps.ini, pps.ini, and eval.ini files. but this becomes unmanagable quickly, it may be better to have conf.ini that is updated as states are accomplished.

We need to include reporting tools, database updating/merging, and sample data in the released versions of the peval-toolkit.

```
# create ini file to manually populate only
#   cooresponds to cp-solutions
$ ppaml init <team-id> <challenge-id>

# run solution as defined by ini file
$ ppaml run <cps.ini>

# evaluate output from run execution
$ ppaml eval <cps.ini>

# reports are externally executed and directly
#   querey the database
$ ./generate_reports.py
```

Goal

Instead of 'team' owning 'pps-type' owning 'pps-instances' it makes sense to just make 'team' representative of pps-type which owns engines[1]

```
# provide list operations
$ ppaml list teams
$ ppaml list problems

# engines listed by date and population
$ ppaml list engines <team-id>
```

We frequently didn't know the status of the teams and challenge problem wrt evaluations and runtime. It would be nice to have a simple tool to report run_id date eval(#t/#f) runtime information.

```
# provide simple table
$ ppaml status teams
$ ppaml status problems
```

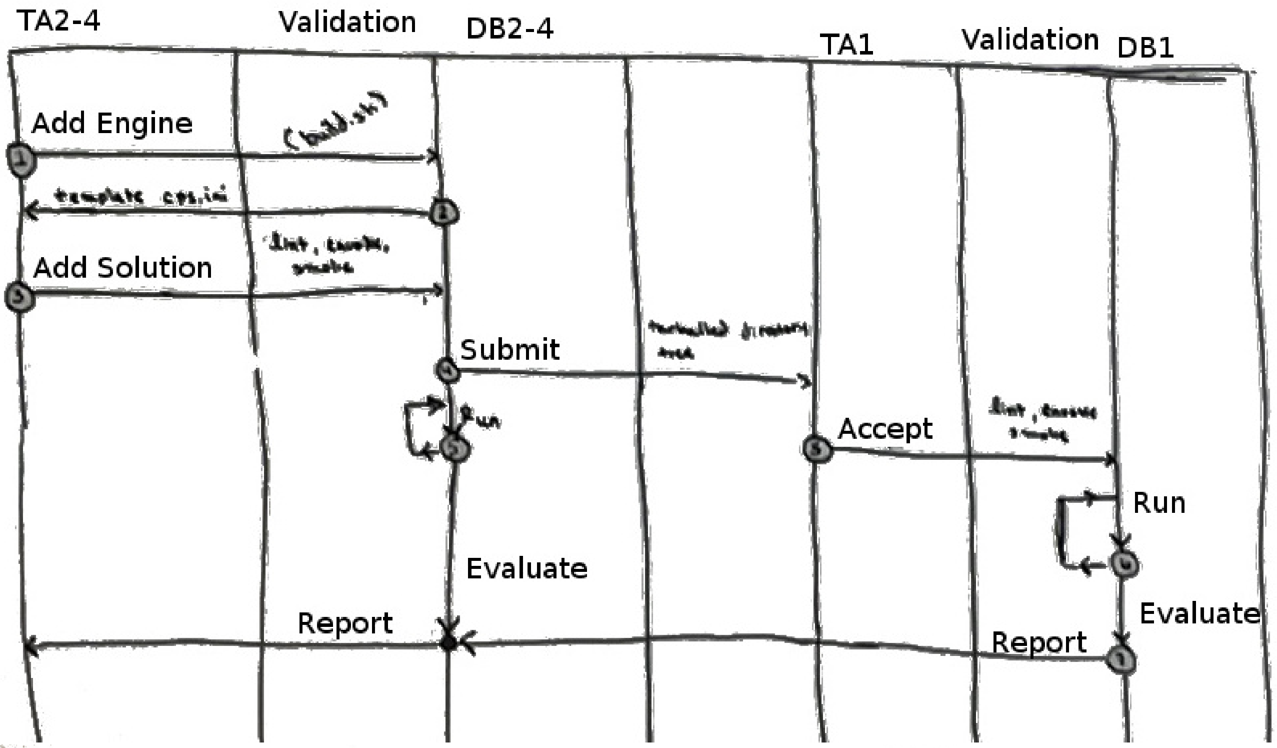
We will now treat the ini file as a descriptor to a user state machine. This will require that any artifact also contains a baseline ini file.

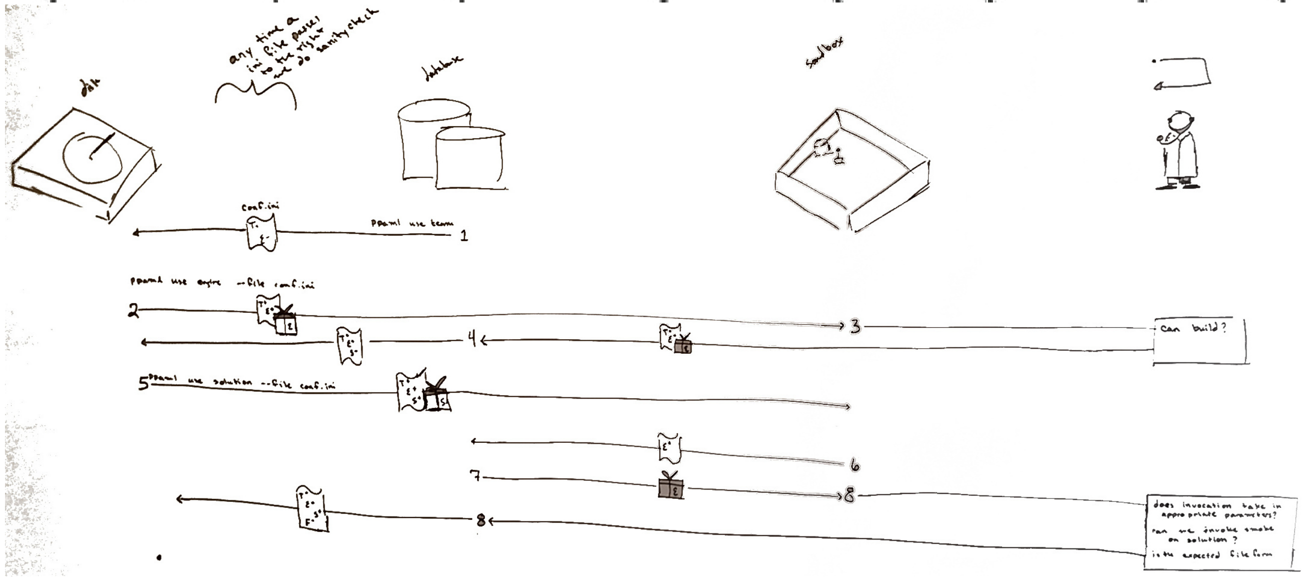
```
# to create a baseline ini file we should only
#   need team-id (pps-id seems redundtant)
$ ppaml use team <team-id> -output <conf.ini>

# manually populate with pps information
#   testing by build or check-install script
#   conf.ini will be stored in the artifact

# update and extend conf.ini to include
#   template challenge problem solution fields
$ ppaml use engine   --file <conf.ini> # OR
$ ppaml use engine   --id <engine-id>

# manually populate cps information including
#   challenge problem id
$ ppaml use solution --file <conf.ini> # OR
$ ppaml use solution --id <solution-id>

# nothing to manually change
$ ppaml run solution --file <conf.ini> # OR
$ ppaml run solution --id <artifact-id> 5

# execute latest from each team
$ ppaml run <challenge-id>
```

Sequence Diagram

The sequence diagram below designates what the intedned workflow of a submition from a team.

[1] To increase clarity, I am decoupling the term 'pps' and 'engine'.


TA2-4
Validation
DB2-4
TA1
Validation
DB1
Add Engine
(build.sh)
template cfg.ini
Add Solution
lint, tarball, smoke
Submit
tarballed directory area
Run
Accept
lint, tarball, smoke
Run
Evaluate
Report
Evaluate
Report


database
any time a ini file passes to the right we do sanity check
database
sandbox
conf.ini
ppaml use team
1
ppaml use engine --file conf.ini
2
3
can build?
4
5 ppaml use solution --file conf.ini
6
7
8
8
does invocation take in appropriate parameters?
can we invoke smoke on solution?
is the expected file form