Object Oriented Suites

Python’s object-oriented design features allow considerable flexibility in how we design and structure our *suite definition*.

Each suite will have a different set of forces that determine how it should be designed.

Let's consider how we would design the tutorial examples in a more object-oriented manner. We start with some design criteria we must meet.

- The default variables (ECF_HOME, etc) must be configurable and independent of the suites
- New suites must enable automatic job creation checking
- We need to write out definition as a separate file
- New suites should be able to re-use the “boilerplate” code defined by the above requirements

Here is a possible design, that uses inheritance and the template design pattern:

```
import os
import ecflow

class DefaultVariables(object):
    """Provide the setup variables for each suite"""
    def add_to(self, node):
        """Adds ECF_INCLUDE,ECF_HOME to the input node"""
        node.add_variable("ECF_INCLUDE", os.path.join(os.getenv("HOME"),  "course"))
        node.add_variable("ECF_HOME", os.path.join(os.getenv("HOME"),  "course"))


class BaseSuiteBuilder(object):
    """Abstract class. Add default variables to suite and enable job creation
 checking for any derived suite
 """
    def __init__(self, default_variables):
        self.defs = ecflow.Defs()
        # use derived class name as suite name
        self.suite = self.defs.add_suite(type(self).__name__)
        default_variables.add_to(self.suite)

    def _build_definition_hook(self):
        """Derived suite should override this function to build the suite
 Should not be called explicitly. How could we enforce this ?
 """
        pass

    def setup(self):
        """Template/skeleton function.
 Provides common algorithm for *all* derivatives
 Uses Hollywood principle.
 """

        # Build the suite
        self._build_definition_hook()

        # check job creation. Could use an assert
        job_check_result = self.defs.check_job_creation()
        if len(job_check_result) != 0:
            print("Job creation failed\n" + job_check_result)

        # Check trigger expressions and limit references"
        print(self.defs.check())

        # Allows definition creation to be separated from the load
        # Use the class name as the name of the definition file
        self.defs.save_as_defs( type(self).__name__ + ".def")


class SuiteBuilder(BaseSuiteBuilder):
    """My example suite. Generates SuiteBuilder.def"""
    def __init__(self, default_variables):
        BaseSuiteBuilder.__init__(self, default_variables)

    def _build_definition_hook(self):
        f1 = self.suite.add_family("family")
        f1.add_task("task1")
        f1.add_task("task2")


class TestSuite(BaseSuiteBuilder):
    """My test suite. Generates TestSuite.def"""
    def __init__(self, default_variables):
        BaseSuiteBuilder.__init__(self, default_variables)

    def _build_definition_hook(self):
        self.suite.add_task("task1")


if __name__ == "__main__":

    my_suite = SuiteBuilder(DefaultVariables())
    my_suite.setup()

    my_test_suite = TestSuite(DefaultVariables())
    my_test_suite.setup()
```