META Test Bench Manifest Specification Documentation
Version 1, Draft 1

Adam Nagel
Institute for Software Integrated Systems (ISIS)
Vanderbilt University
adam@isis.vanderbilt.edu

Developed for the DARPA Adaptive Vehicle Make (AVM) Program

October 28, 2013

Contents

1 Overview

1.1 10,000 ft View

The Test Bench Manifest is the central part of every META test bench. It captures several key pieces of information:

- The Parameters of the Test Bench
- The Metrics to be calculated by the end of Test Bench execution
- The Artifacts generated and used during the Test Bench execution
- The steps for executing the Test Bench

The initial manifest file, along with supporting artifacts, is generated by the META tools.

A **test bench executor** uses the file to run the steps required for execution. The executor must go through each of the Steps defined, in order, one-by-one. For each, it must run the pre-processing script, execute the main Invocation statement, then run the post-processing script. Pre- and post-processing scripts may read from, and modify, the manifest file itself. This can be useful for reading Parameters, writing Metrics, and altering the Parameters of other invocation Steps.

1.2 Sample Use Case (Conceptual)

Let's say that a Test Bench is designed to support a manufacturability and costing analysis.

1. The META tools generate:
    - Design Model export
    - Component Model exports
    - Manufacturing Models for each Component
    - A Manifest file cross-indexing each component instance with its manufacturing model
    - An XML file describing how the individual Component CAD Models should be composed to form a system model
    - a **testbench_manifest.json** file indexing:
        - All above items as Artifacts, with key elements also including Tags
        - Metrics
            - Cost
            - Manufacturing Time
        - Steps (for execution)
            - Assemble the System CAD model
            - Run the manufacturability analysis
2. A **test bench executor** loads the **testbench_manifest.json** file, and iterates over the steps.
    - Step 1: Assemble the system CAD model
        - Pre-Processing: none
        - Invocation: Run the BAT file that assembles the system model
        - Post-Processing:
            - Add the "cadmetrics.xml" file to the **testbench_manifest.json** Artifacts list
            - Add the generated system CAD model as an Artifact with a tag of "System Model"
            - Add the other generated CAD models to the **testbench_manifest.json** Artifacts list
    - Step 2: Perform the manufacturability analysis
        - Pre-Processing:
            - Build an index file for the manufacturability analysis
            - Add the above index file to the **testbench_manifest.json** Artifacts list
            - Upload all artifacts to a storage system used by the manufacturability analysis
            - Use the server's response, with package ID, to alter the XXXX Parameter of the Invocation section of this step.
            - Generate the post-processing script to be used as part of this step, embedding the package ID so that it knows what results to poll for
        - Invocation: Invoke the manufacturability via a REST interface
        - Post-Processing:
            - Poll the manufacturability service, using the package ID (discovered during the preprocessing step), until a response is ready

        * Use the results to populate the Metrics of **testbench_manifest.json**
    - Step 3: Clean up the Test Bench
      - Pre-Processing: none
      - Invocation: Python script to delete all unnecessary Artifacts from disk and **testbench_manifest.json**
      - Post-Processing: none
3. The resulting package is passed to the META Dashboard to support visualization and comparison of designs

2 Classes

2.1 TestBenchManifest

This is the root object of a **testbench_manifest.json** document.

2.2 Parameter

This parameter is defined in the META test bench model. Analysis tools that run during test bench execution may use these values to parameterize execution. If the test bench is being used in a PCC or PET analysis, then this parameter may be varied for each run by the PCC or PET execution tool.

2.3 Metric

This captures a value that is exepected to be provided after test bench execution. It will be passed to the META Dashboard as the result for a specific test bench, evaluated against a specific design. Before test bench execution is completed, values must be provided for these. If the test bench is being used in a PCC or PET analysis, this is fed back to the experiment driver/optimizer.

2.4 Artifact

This allows an explicit pointer to an artifact expected by any tool in the analysis chain. The "tag" field will indicate a key artifact type, such as "Component Model Index" or "Design Model". Tools that operate on the test bench should search the Artifact list by "tag" in order to find the key artifacts needed for execution. Each artifact's "Tag" value must be unique.

3 Appendix

3.1 Sample Manifest File

```
{
  "Artifacts": [
    {
      "Location": "design1235.metadesign.xml",
      "Tag": "Design Model"
    },
    {
      "Location": "components/index.json",
      "Tag": "Component Model Index"
    }
  ],
  "Created": "2013-06-13T20:40:42.578236",
  "DesignID": "123-245-2342-23421",
  "DesignName": "ABC Analysis",
  "Metrics": [
    {
      "Description" : null,
      "ID" : null,
      "DisplayedName" : null,
      "GMEID" : null,
      "Name": "Metric1",
```

```
      "Unit": "DegreeAngle",
      "Value": null
    }
  ],
  "Parameters": [
    {
      "Description" : null,
      "ID" : null,
      "DisplayedName" : null,
      "GMEID" : null,
      "Name": "Parameter1",
      "Unit": "Millimeter",
      "Value": 1.2
    },
    {
      "Description" : null,
      "ID" : null,
      "DisplayedName" : null,
      "GMEID" : null,
      "Name": "Parameter2",
      "Unit": "Kilogram",
      "Value": 99.21
    }
  ],
  "Status": "UNEXECUTED",
  "Steps": [
    {
      "Description": "Build CAD Assembly of system. In post-processing, write calculated metrics and
          artifact locators back to this file.",
      "ExecutionCompletionTimestamp": null,
      "ExecutionStartTimestamp": null,
      "Invocation": "buildIt.bat",
      "Parameters": [],
      "PostProcess": "retrieveMetrics.py",
      "PreProcess": null,
      "Type": "CMD"
    },
    {
      "Description": "Submit artifacts to iFAB analysis. cIn post-processing, poll their service until
          results are available, then write them back to this file.",
      "ExecutionCompletionTimestamp": null,
      "ExecutionStartTimestamp": null,
      "Invocation": "https://ifab.com/analyze",
      "Parameters": [
        {
          "Name": "PackageID",
          "Unit": null,
          "Value": "1245522"
        }
      ],
      "PostProcess": "retrieveMetrics.py",
      "PreProcess": "sendArtifactsToIFAB.py",
      "Type": "REST"
    }
  ]
  "TestBench" : "BallisticsTestBench",
  "TierLevel" : 0
}
```

3.2 Sample Manifest Generation Code

```
import sys
sys.path.insert(0, '../../lib/python')
import TestBenchManifest


tbm = TestBenchManifest.TestBenchManifest()


# Set basic attributes
tbm.Name = "ABC Analysis"
tbm.DesignID = "123-245-2342-23421"
tbm.Status = "unexecuted"


# Add Parameters
p1 = TestBenchManifest.Parameter()
p1.Name = "Parameter1"
p1.Unit = "Millimeter"
p1.Value = 1.2
```

```
tbm.Parameters.append(p1)

p2 = TestBenchManifest.Parameter()
p2.Name = "Parameter2"
p2.Unit = "Kilogram"
p2.Value = 99.21
tbm.Parameters.append(p2)


# Add Metrics
m1 = TestBenchManifest.Metric()
m1.Name = "Metric1"
m1.Unit = "DegreeAngle"
m1.Value = None
tbm.Metrics.append(m1)


# Add Artifacts
a1 = TestBenchManifest.Artifact()
a1.Location = "design1235.metadesign.xml"
a1.Tag = "Design Model"
tbm.Artifacts.append(a1)

a2 = TestBenchManifest.Artifact()
a2.Location = "components/index.json"
a2.Tag = "Component Model Index"
tbm.Artifacts.append(a2)


# Add Execution Steps
s1 = TestBenchManifest.Step()
s1.Description = "Build CAD Assembly of system. " \
                 "In post-processing, write calculated " \
                 "metrics and artifact locators back to this file."
s1.Type = "CMD"
s1.PreProcess = None
s1.PostProcess = "retrieveMetrics.py"
s1.Invocation = "buildIt.bat"
tbm.Steps.append(s1)

s2 = TestBenchManifest.Step()
s2.Description = "Submit artifacts to iFAB analysis. c" \
                 "In post-processing, poll their service until " \
                 "results are available, then write them back to this file."
s2.Type = "REST"
s2.PreProcess = "sendArtifactsToIFAB.py"
s2.PostProcess = "retrieveMetrics.py"
s2.Invocation = "https://ifab.com/analyze"
s2p = TestBenchManifest.Parameter()
s2p.Value = "1245522"
s2p.Name = "PackageID"
s2.Parameters.append(s2p)
tbm.Steps.append(s2)


print tbm.SerializeToString()
```