TestMethod Inheritance discussion – updated examples

```
TestMethod TestBase {
  Parameters {
    String bstr Out;   // Uninitialized, i.e., empty String
    Integer bint Out { InitialValue 1; }
  }
}

TestMethod Drv1 {
  Inherit TestBase;
  Parameters {
    String bstr Out { InitialValue "x"; }    // Override TestBase initialization
    String drv1str Out { InitialValue "y"; } // Drv1 parameter with initial value
  }
}

TestMethod Drv2 {
  Inherit Drv1;
  Parameters {
    String bstr Out { InitialValue "a"; }    // Override TestBase initialization
    String drv1str Out { InitialValue "b"; } // Override Drv1 initialization
    String drv2str Out { InitialValue "c"; } // Drv2 parameter
  }
}

TestMethod Drv3 {
  Inherit Drv1;
  // bstr initial value = x (since inheriting from Drv1, which overrides TestBase init).
  // drv1str initial value = y (from Drv1 initialization).
}

TestMethod Drv4 {
  Inherit Drv2;
  // bstr Initial value = a (from Drv2 init, which overrides TestBase and Drv1 init)
  // drv1str initial value = b (from Drv2 initialization, overriding Drv1).
  // drv2str initial value = c (from Drv2 initialization).
}
```

Figure 1: Syntax for TestMethods with Inheritance from TestBase

In Fig. 1 above, the syntax for overriding an initial value of a parameter inherited from a parent TestMethod needs to be finalized. For instance, in TestMethods Drv1 and Drv2, some parameters which are defined in the base TestMethod are assigned initial values different from the initial value(s) specified in the base TestMethod.

The syntax for overriding this initial value as shown in Fig. 1 above is the same syntax used in the initial definition of the parameter. This could lead to confusion – are we defining a new parameter in the derived TestMethod with the same type, name, and direction as was defined in the base TestMethod (which should probably be treated as an (multiply-defined?) error!!!), or are we simply changing the initial value?

A better approach would be to define the syntax for parameter definitions so that if only overriding the initial value, only the parameter name is stated – the type and direction will NOT be restated. Fig. 2 below shows 3 alternative syntaxes for changing the initial value of a parameter defined in a base TestMethod (the original one as shown in Fig. 1, and two others which do not restate the type and direction, but only the parameter name and (new) initial value.

Of the two alternatives which do not restate the parameter type and direction, the one shown in TestMethod Drv1a is preferred, since it uses similar syntax as was used to assign the initial value in the first place.

```
TestMethod TestBase {
  Parameters {
    String bstr Out;   // Uninitialized, i.e., empty String
    Integer bint Out { InitialValue 1; }
  }
}

TestMethod Drv1 {
  Inherit TestBase;
  Parameters {
    // Restating the type, name, and direction of bstr, which was defined in base
    // TestMethod TestBase in order to change the initial value.  This should
    // probably be an error!
    String bstr Out { InitialValue "x"; }    // Override TestBase initialization
    String drv1str Out { InitialValue "y"; } // Drv1 parameter with initial value
  }
}

TestMethod Drv1a {
  Inherit TestBase;
  Parameters {
    bstr { InitialValue "x"; }               // Override TestBase initialization
    String drv1str Out { InitialValue "y"; } // Drv1 parameter with initial value
  }
}

TestMethod Drv1b {
  Inherit TestBase;
  Parameters {
    bstr = "x";                              // Override TestBase initialization
    String drv1str Out { InitialValue "y"; } // Drv1 parameter with initial value
  }
}
```

Figure 2: Three syntax alternatives for specifying new initial values (in a derived TestMethod) for parameters defined in a base TestMethod

Inheritance in TestMethods is NOT required. If one chooses NOT to use inheritance, ALL TestMethods MUST explicitly include the same elements as would have been provided had we inherited from TestBase. If choosing this option, then if TestBase is updated, all those TestMethods must be also updated so that they remain in sync with TestBase. See Fig. 2 for examples of TestMethods NO_Derive1 and NO_Derive2 which are functionally equivalent to TestMethods Drv1 and Drv2 from Fig. 1. Note that this provides functionally the same behavior as results from 1-4 above, without requiring the use of inheritance.

```
TestMethod NO_Derive1 { // Same as TestMethod Drv1
  // Explicitly NOT inheriting from TestBase; therefore, MUST include all elements that
  // TestBase includes, AND initialize parameters to the same values as which Drv1 would
  // have initialized them.
  Parameters {
    String bstr Out { InitialValue "x"; }
    Out Integer bint Out { InitialValue 1; }
    // Initial values for bstr and bint are exactly what would have been provided
    // by Drv1 after it had overridden TestBase initialization.

  }
}

TestMethod No_Derive2 { // Same as TestMethod Drv2
  // Explicitly NOT inheriting from Drv1; therefore, MUST include all elements that Drv1
  // includes, both those from TestBase and those added by Drv1.  Further, variables must
  // be initialized so that they have the same values that Drv2 (with inheritance) would
  // have provided
  Parameters {
    String bstr Out { InitialValue "a"; } // Use initial value from Drv2, not from TestBase
    String drv1str Out { InitialValue "b"; } // Use init value from Drv2, not from Drv1
    String drv2str Out { InitialValue "c"; } // Drv2 parameter
  }
}
```

Figure 3: Syntax for providing "TestBase" framework WITHOUT using inheritance