A Reusability Combat in UVM
Callbacks vs Factory

Deepak Kumar EV, Vikas Billa, Sathish Dadi, Ranganath Kempanahally (R K)
dkumar@eliteplustech.com, vikas.billa@eliteplustech.com, sathish.dadi@eliteplustech.com, ranganath@eliteplustech.com

INTRODUCTION

UVM Factory and UVM Callbacks are two amongst many features, which are predominately used in making/creating the UVM test bench to make it reusable and more efficient. Many resources available online are offering contradicting guidelines about their usage

This paper looks at these two features in depth and makes an attempt to suggest better ways of using them.

The below are the suggestions from two major EDA vendors:

EDA Vendor1	EDA Vendor2
Suggests not to use callbacks as there are many convoluted steps involved that are required to register and enable the callbacks.	Suggests not to use factory as OOP hierarchy becomes unstable due to multiple class definitions for the same component.

UVM CALLBACKS & UVM FACTORY

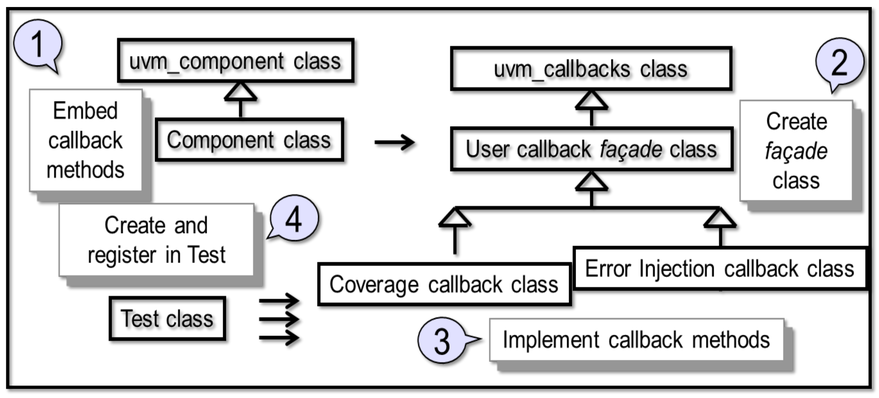


Figure 1: UVM Callback Steps

Source : Synopsys

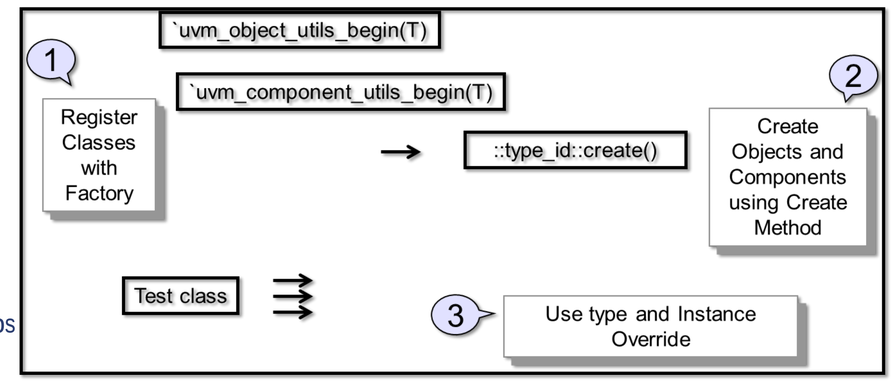


Figure 2: UVM Factory Steps

A CASE STUDY

Scenario 1: Adding additional functionality to existing logic.

```
class elite_control_trans_c extends uvm_sequence_item;
rand wr_rd_t wr_rd;
rand bit[7:0] rddata;
rand bit[7:0] wrdata;
rand bit[2:0] addr;
endclass : elite_control_trans_c
```

Only Write or Read

```
typedef enum bit { READ, WRITE} wr_rd_t;
```

```
class elite_control_driver_c extends uvm_driver #(elite_control_trans_c);
`uvm_register_cb(elite_control_driver_c , elite_pkt_control_driver_cb)
task run_phase(uvm_phase phase);
forever begin
  seq_item_port.get_next_item(req);
  pre_send();
  drive_data();
  post_send();
endtask : run_phase
endclass : elite_control_driver_c
```

Callback Function

Actual Drive Logic

Callback Function

A CASE STUDY (CONTD ...)

Scenario 2: Injection of error in Packet field.

```
class elite_data_trans_c extends uvm_sequence_item;
  rand bit parity_enable_disable;
endclass : elite_data_trans_c
```

Random Parity error insertion

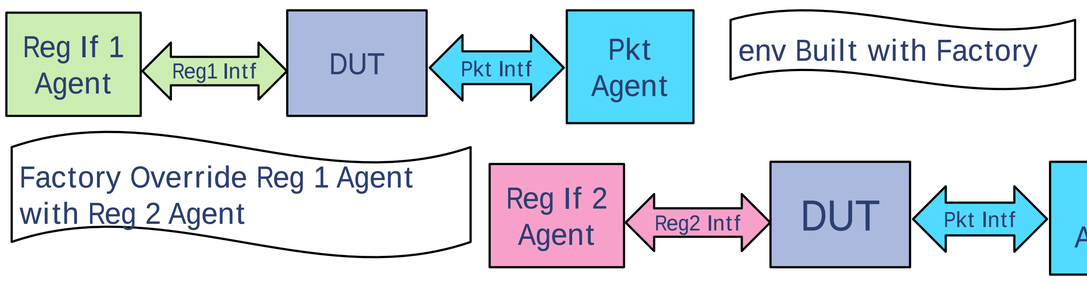
In Driver

pre_partiy

Fine Grain Control

post_partiy

Scenario 3: Interface driving logic is completely changed.



The Register Interface has changed , a common enhancement in recent IP's.

UVM CALLBACKS vs UVM FACTORY

S.No	UVM Callbacks	UVM Factory
1	Suitable for minimal feature addition.	Suitable when feature enhancement is a major one.
2	Can add or remove functionality after the object has been built.	Cannot change the behavior once the object is built.
3	Callbacks have to be planned and placed at strategic locations in the code.	Not much planning is required if the modularity is maintained factory can be used easily.
4	Easy to Maintain.	Multiple copies are required.
5	The callbacks are more popular with VIPs.	Not Suitable for VIP Components.

CONCLUSION

- Callbacks and factory address different areas of reusability.
  - ❑ Callback : Add functionality to existing logic.
  - ❑ Factory : Change the existing component before build , keeps environment same.
- While Architecting and coding the testbench, proper care should to be taken in introducing callback points. A thumb rule we should follow is, if a testbench requires more than 3 callbacks(to be passed) to achieve a given functionality, the component should be overridden by factory.
- All the components need to be created using factory, which will be helpful in factory override later.
- Although the callback and factory can be interchangeably used to address the same problem. Depending on the need and demand, as recommended above, a wise decision should be made while adopting either of the techniques as they have their own merits and demerits.

elitePLUS Semiconductor Technologies Pvt Ltd.
2nd Floor, #991 & 992, 5th Main, 7th Sector, HSR Layout,
Bengaluru, Karnataka - India 560102.