Extending UVM components Functionality by using the **Visitor** design pattern

Darko M. Tomušilović
Vtool LTD

accellera SYSTEMS INITIATIVE

2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE


Can you put a rabbit in a box...
...without opening it?

Goal

Add a new operation to each class in an existing class hierarchy

Goal

Add a new operation to each class in an existing class hierarchy

Common Solutions

Goal

Add a new operation to each class in an existing class hierarchy

Common Solutions

Add code that will perform each operation
into each class in the environment

Goal

Add a new operation to each class in an existing class hierarchy

Common Solutions

Add code that will perform each operation into each class in the environment

Goal

Add a new operation to each class in an existing class hierarchy

Common Solutions

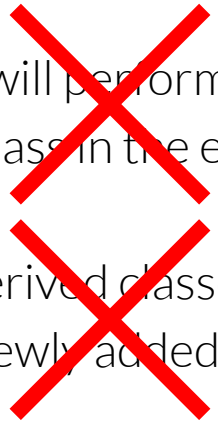
Add code that will perform each operation into each class in the environment

Create derived classes that will perform newly added operations

Goal

Add a new operation to each class in an existing class hierarchy

Common Solutions



Add code that will perform each operation into each class in the environment

Create derived classes that will perform newly added operations

Goal

Add a new operation to each class in an existing class hierarchy

The alternative to these approaches has been well established in the software development world

Goal

Add a new operation to each class in an existing class hierarchy

The Visitor design pattern

Goal

Add a new operation to each class in an existing class hierarchy

The Visitor design pattern

Benefits

Decreases code complexity
Facilitates maintenance
Improves code stability
Simple to use

Drawbacks

Requires advanced planning

Goal

Add a new operation to each class in an existing class hierarchy

The Visitor design pattern

Benefits

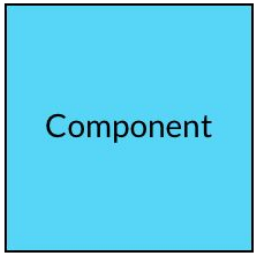
Decreases code complexity
Facilitates maintenance
Improves code stability
Simple to use

Drawbacks

~~Requires advanced planning~~

Visitor design pattern infrastructure

Block diagram

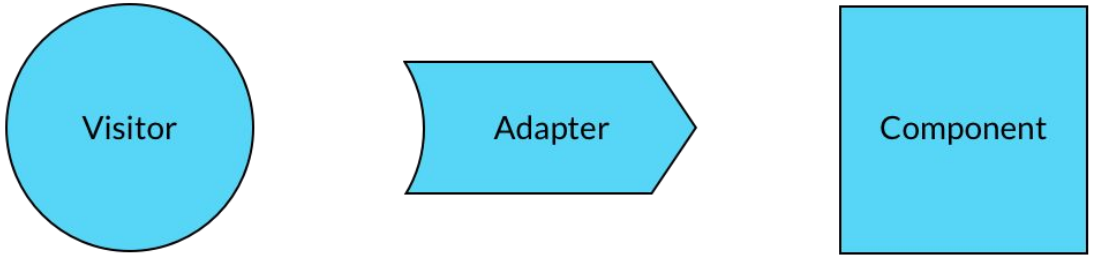


Visitor design pattern infrastructure

Block diagram

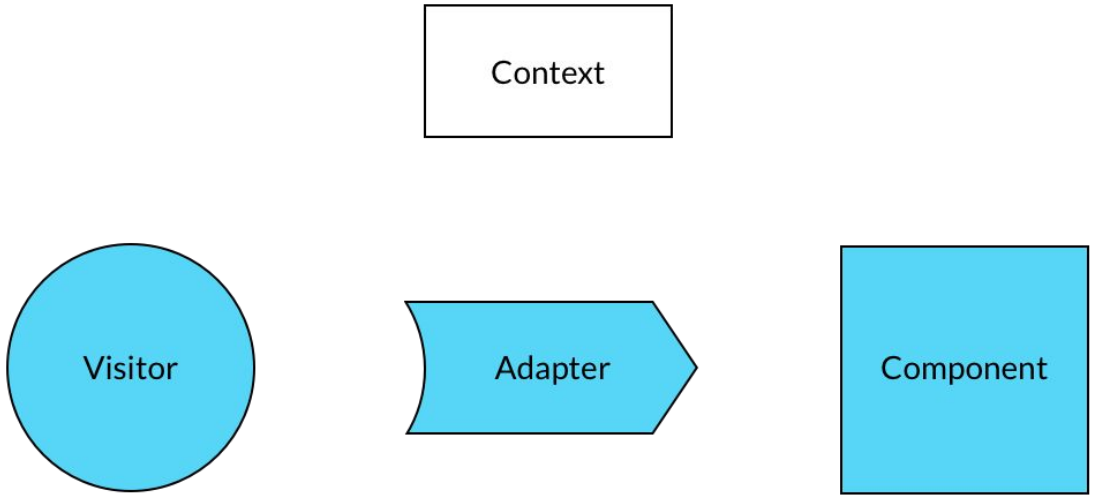
Visitor design pattern infrastructure

Block diagram



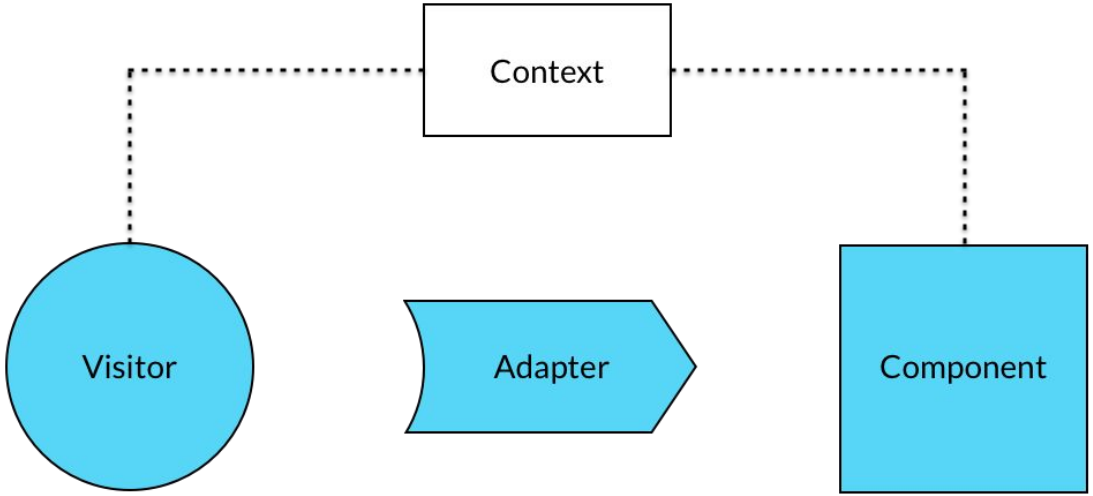
Visitor design pattern infrastructure

Block diagram



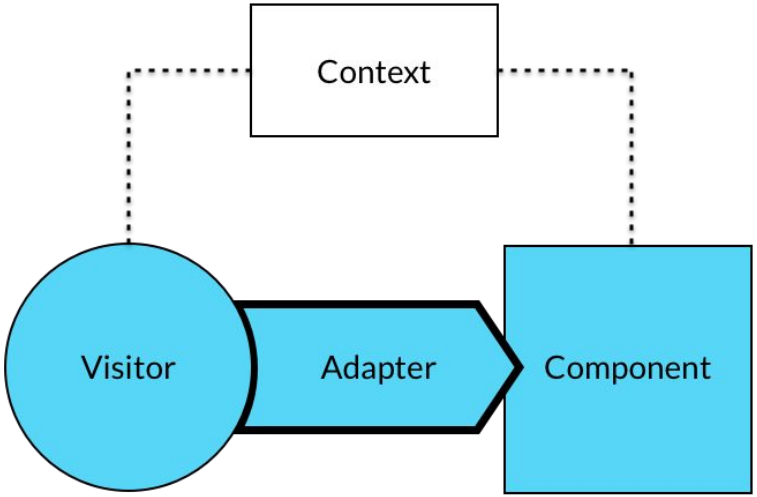
Visitor design pattern infrastructure

Block diagram



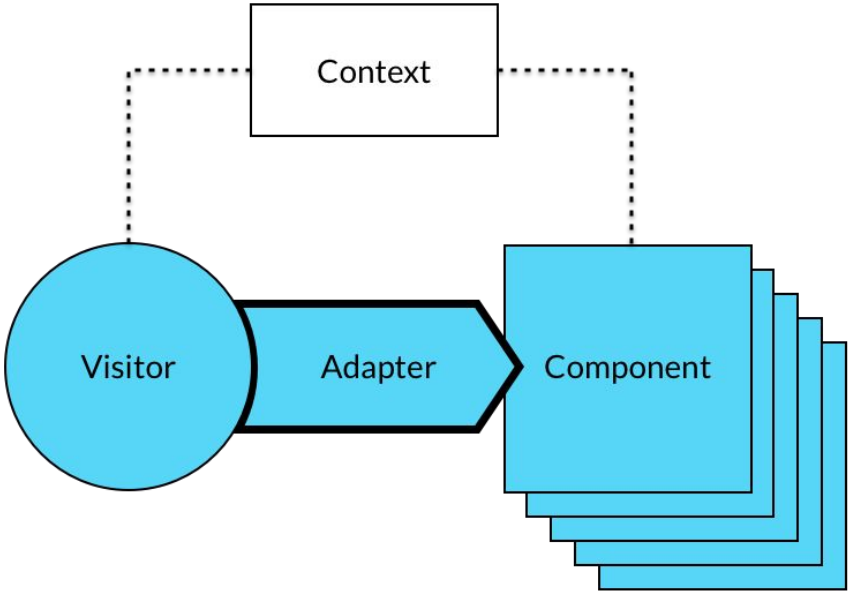
Visitor design pattern infrastructure

Block diagram



Visitor design pattern infrastructure

Block diagram



Visitor design pattern infrastructure

uvm_visitor

- An abstract class defines a general **visit operation** on a node.
- A **concrete visitor** gives an implementation to a visit operation according to the action the visitor needs to accomplish.
- Pre-processing and post-processing **hooks.**

Visitor design pattern infrastructure

uvm_visitor

```
class name_display_visitor extends uvm_visitor;

  virtual function void visit(uvm_component node);
      `uvm_info("NAME DISPLAY VISITOR",node.get_full_name(),UVM_LOW)
  endfunction

  function new (string name = "");
    super.new(name);
  endfunction
endclass
```

Visitor design pattern infrastructure

uvm_adapter

- An abstract class defines a general **accept operation** that in turn applies the corresponding visitor on every element of the structure that the adapter wraps.

- The following adapter wraps a single component:

Visitor design pattern infrastructure

uvm_adapter

```
class basic_adapter extends uvm_visitor_adapter;

  virtual function void accept(uvm_component s, uvm_visitor v, uvm_structure_proxy#(uvm_component) p,
                               bit invoke_begin_end=1);
    if(invoke_begin_end)
      v.begin_v();

    v.visit(s);

    if(invoke_begin_end)
      v.end_v();
  endfunction

  function new (string name = "");
    super.new(name);
  endfunction
endclass
```

Visitor design pattern infrastructure
Context

- Invokes the **accept method** of an object of an adapter class.
- Provides the component to be visited as an argument.
- Provides the visitor object as an argument.

Visitor design pattern infrastructure

Context

```
task visitor_env::run_phase(uvm_phase phase);
    name_display_visitor  name_display_v;
    basic_adapter         adapter;

    name_display_v = new("name_display_v");
    adapter        = new("adapter");

    adapter.accept(this, name_display_v, null);
endtask
```

```
UVM_INFO ../src/visitor_env.sv(7) @ 0: reporter [NAME DISPLAY VISITOR] uvm_test_top.env
```

UVM library predefined adapters

- Traverses elements in a complex composite structure in a specific way.

- Applies a visitor operation upon each of the elements in a defined order:
  - uvm_top_down_visitor_adapter
  - uvm_bottom_up_visitor_adapter
  - uvm_by_level_visitor_adapter

UVM library predefined adapters

- An abstract uvm_structure_proxy class provides all children sub-elements of a certain element in a structure, facilitating **traversal**.

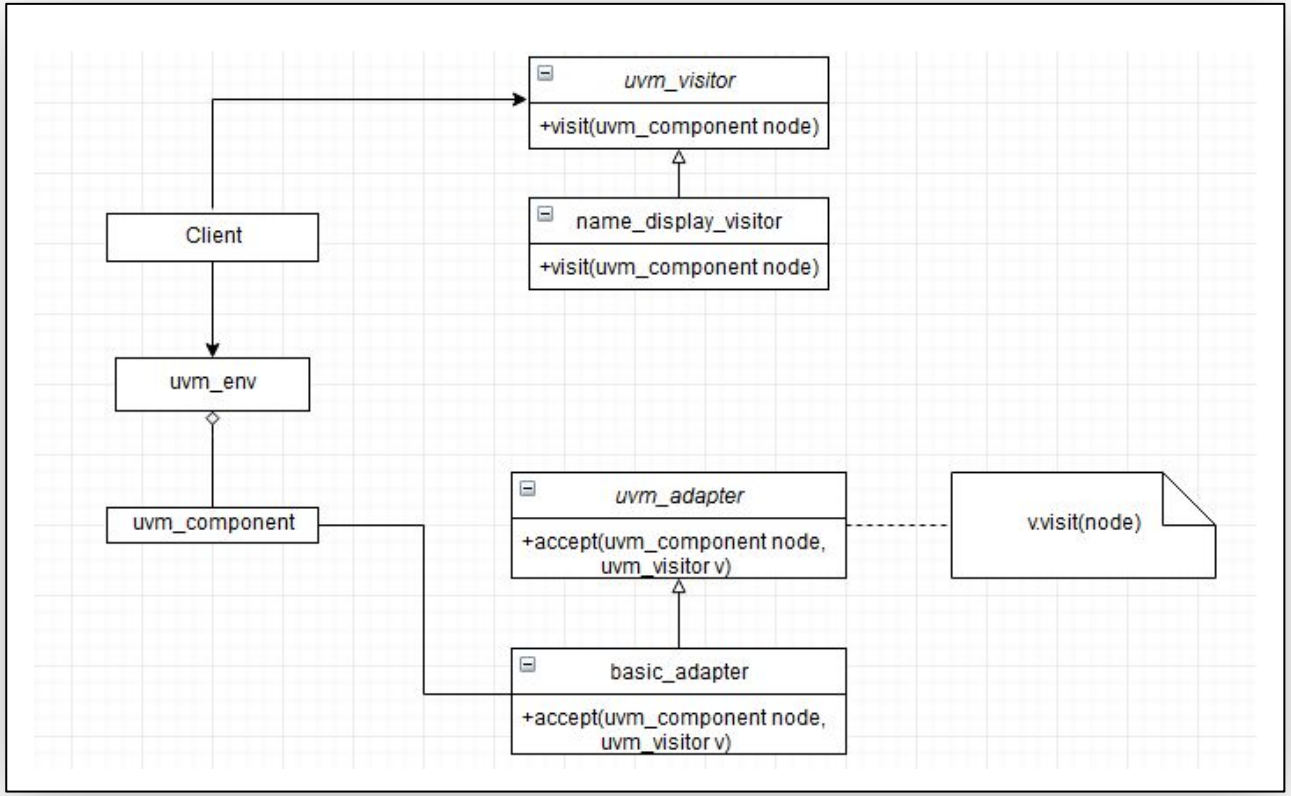
- The specialization class **uvm_components_proxy** provides all subcomponents for a given UVM component.

Visitor Traversal

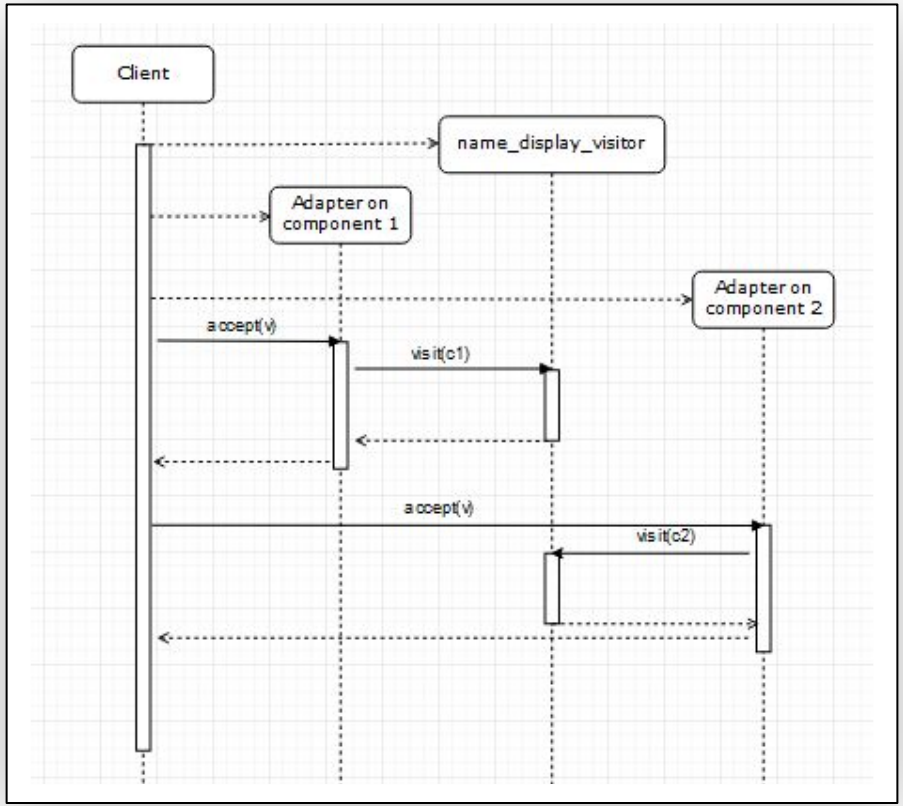
```
task visitor_env::run_phase(uvm_phase phase);
        name_display_visitor          name_display_v;
        uvm_top_down_visitor_adapter  adapter;
        uvm_component_proxy           proxy;

        name_display_v = new("name_display_v");
        adapter        = new("adapter");
        proxy          = new("proxy");

        adapter.accept(this, name_display_v, proxy);
endtask
```

```
UVM_INFO ../src/visitor_env.sv(7) @ 0: reporter [NAME DISPLAY VISITOR] uvm_test_top.env
UVM_INFO ../src/visitor_env.sv(7) @ 0: reporter [NAME DISPLAY VISITOR] uvm_test_top.env.master_agent
UVM_INFO ../src/visitor_env.sv(7) @ 0: reporter [NAME DISPLAY VISITOR] uvm_test_top.env.master_agent.drv
UVM_INFO ../src/visitor_env.sv(7) @ 0: reporter [NAME DISPLAY VISITOR] uvm_test_top.env.master_agent.drv.rsp_port
UVM_INFO ../src/visitor_env.sv(7) @ 0: reporter [NAME DISPLAY VISITOR] uvm_test_top.env.master_agent.drv.seq_item_port
UVM_INFO ../src/visitor_env.sv(7) @ 0: reporter [NAME DISPLAY VISITOR] uvm_test_top.env.master_agent.mon
UVM_INFO ../src/visitor_env.sv(7) @ 0: reporter [NAME DISPLAY VISITOR] uvm_test_top.env.master_agent.mon.analysis_port
UVM_INFO ../src/visitor_env.sv(7) @ 0: reporter [NAME DISPLAY VISITOR] uvm_test_top.env.master_agent.mon.peek_imp
UVM_INFO ../src/visitor_env.sv(7) @ 0: reporter [NAME DISPLAY VISITOR] uvm_test_top.env.master_agent.seqr
```

UML class diagram



UML sequence diagram



Verification use-case examples

- Component configuration check visitor.
- Reset and clock generation check visitor.
- Add messages and improve the reporting system.

Verification use-case examples

- Component configuration check visitor.
  - **Check that every component in the environment is properly configured.**
- Reset and clock generation check visitor.
- Add messages and improve the reporting system.

Verification use-case examples

Component configuration check visitor

```
class component_check_visitor extends uvm_visitor;

  virtual function void visit(uvm_component node);
          if (node.get_object_type() == visitor_master_driver::type_id::get()) begin
            visit_driver(node);
          end
  endfunction

        virtual function void visit_driver(uvm_component node);
          visitor_master_driver drv;
          $cast(drv, node);

          if (drv.visitor_if == null)
            `uvm_error("COMPONENT CHECK VISITOR", $sformatf("%s: Interface not set", drv.get_full_name()))
          else
            `uvm_info("COMPONENT CHECK VISITOR",$sformatf("%s: Interface set", drv.get_full_name()),UVM_LOW)
          if (drv.cfg == null)
            `uvm_error("COMPONENT CHECK VISITOR",$sformatf("%s: CFG not set", drv.get_full_name()))
          else
            `uvm_info("COMPONENT CHECK VISITOR",$sformatf("%s: CFG set", drv.get_full_name()),UVM_LOW)
        endfunction
  function new (string name = "");
    super.new(name);
  endfunction
endclass
```

Verification use-case examples

- Component configuration check visitor.
- Reset and clock generation check visitor.
- Add messages and improve the reporting system.

Verification use-case examples

- Component configuration check visitor.
- Reset and clock generation check visitor.
  - **Check the components in the environment are provided with a proper clock and reset.**
- Add messages and improve the reporting system.

Verification use-case examples

Reset check visitor

```
class reset_check_visitor extends uvm_visitor;

  virtual function void visit(uvm_component node);
        if (node.get_object_type() == visitor_master_driver::type_id::get()) begin
          visit_driver(node);
        end
  endfunction

      virtual function void visit_driver(uvm_component node);
        visitor_master_driver drv;
        $cast(drv, node);

        if (drv.visitor_if.reset_n === 1'b1)
          `uvm_info("RESET CHECK VISITOR",$sformatf("%s: reset deasserted", drv.get_full_name()),UVM_LOW)
        else
          `uvm_info("RESET CHECK VISITOR",$sformatf("%s: reset asserted", drv.get_full_name()),UVM_LOW)
      endfunction

  function new (string name = "");
    super.new(name);
  endfunction
endclass
```

Verification use-case examples

- Component configuration check visitor.
- Reset and clock generation check visitor.
- Add messages and improve the **reporting system**.

The Reporting System

What prevents us from having a good messaging system?

- People are too lazy to add messages.
- Hard to anticipate places where to add messages.
- Having too many messages reduces code readability.
- Working in big teams (other people's code, vendor code).


Host_TX_FIFO_eth_aux_rnd_small_10_pkt_test.1.log
Timeline
686104950
42153966
203392885
364631804
687109642
All Messages
AUX POP END
words_in_packet:
pkt_size_words:
pkt_size
ETH AUX matched !
AUX
ETH
Log Messages
Bookmarks (0)
Unparsed Lines (36)
Machine Learning
verbosity
filename
timepoint
emitter
message
UVM_INFO Host_TX_FIFO_env_sb.sv(369) 741602000 uvm_test_top.env.sb [Host_TX_FIFO_env_sb] cnt_valid_data: 586
UVM_INFO Host_TX_FIFO_env_sb.sv(366) 741602000 uvm_test_top.env.sb [Host_TX_FIFO_env_sb] pop_aux_pkt.data[j]: 33, exp_mas
UVM_INFO Host_TX_FIFO_env_sb.sv(369) 741602000 uvm_test_top.env.sb [Host_TX_FIFO_env_sb] cnt_valid_data: 587
UVM_INFO Host_TX_FIFO_env_sb.sv(366) 741602000 uvm_test_top.env.sb [Host_TX_FIFO_env_sb] pop_aux_pkt.data[j]: 139, exp_mas
UVM_INFO Host_TX_FIFO_env_sb.sv(369) 741602000 uvm_test_top.env.sb [Host_TX_FIFO_env_sb] cnt_valid_data: 588
UVM_INFO Host_TX_FIFO_env_sb.sv(366) 741602000 uvm_test_top.env.sb [Host_TX_FIFO_env_sb] pop_aux_pkt.data[j]: 184, exp_mas
UVM_INFO Host_TX_FIFO_env_sb.sv(369) 741602000 uvm_test_top.env.sb [Host_TX_FIFO_env_sb] cnt_valid_data: 589
UVM_INFO Host_TX_FIFO_env_sb.sv(380) 741602000 uvm_test_top.env.sb [Host_TX_FIFO_env_sb] ETH PUSH packet matched successfu
UVM_INFO uvm_objection.svh(1270) 741660000 reporter [TEST_DONE] 'run' phase is ready to proceed to the 'ext
UVM_ERROR Host_TX_FIFO_env_sb.sv(259) 741660000 uvm_test_top.env.sb [Host_TX_FIFO_env_sb] There are 1 predicted E
UVM_INFO uvm_report_catcher.svh(705) 741660000 reporter [UVM/REPORT/CATCHER]

Verification use-case examples

- Component configuration check visitor.
- Reset and clock generation check visitor.
- Add messages and improve the **reporting system.**

Verification use-case examples

- Component configuration check visitor.
- Reset and clock generation check visitor
- Add messages and improve the **reporting system.**
  - **Attach a visitor to certain environment events.**
  - **Perform proper reporting on trigger event.**
    For example, attach a visitor to a queue in the scoreboard.

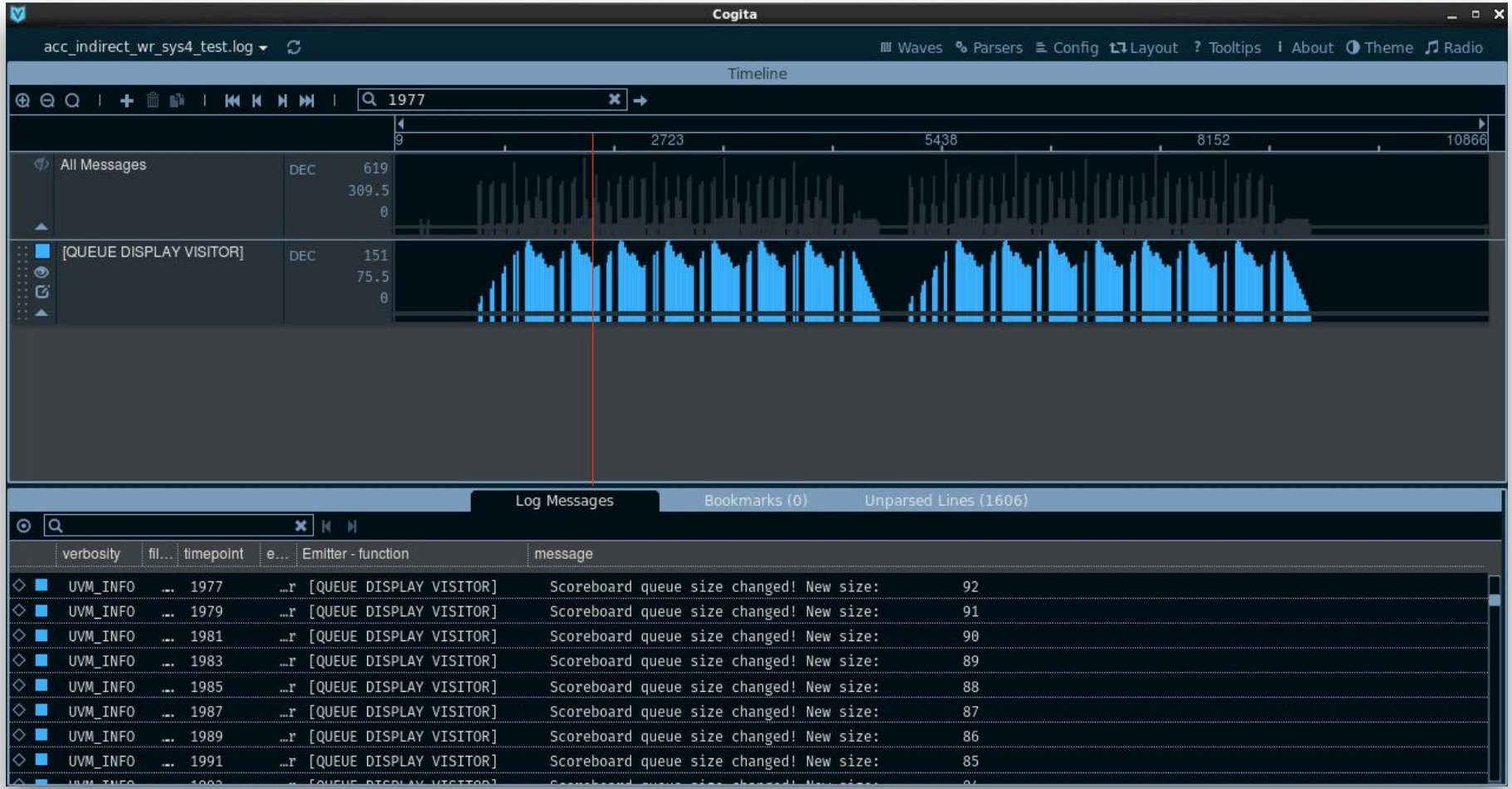
Verification use-case examples

Queue display visitor

```
class queue_display_visitor extends uvm_visitor;

  virtual function void visit(uvm_component node);
        if (node.get_object_type() == visitor_sb::type_id::get()) begin
          fork
            visit_sb_tcm(node);
          join_none
        end
  endfunction

      virtual task visit_sb_tcm(uvm_component node);
        visitor_sb sb;
        $cast(sb, node);

        `uvm_info("QUEUE DISPLAY VISITOR",$sformatf("Start monitoring scoreboard queue"),UVM_LOW)

        forever begin
          @(sb.data_q.size());
          `uvm_info("QUEUE DISPLAY VISITOR",$sformatf("Queue size changed. New size: %d", sb.data_q.size()),UVM_LOW)
          `uvm_info("QUEUE DISPLAY VISITOR",$sformatf("Queue content: %p", sb.data_q),UVM_LOW)
        end
      endtask

  function new (string name = "");
    super.new(name);
  endfunction
endclass
```

Output log - grep

```
UVM_INFO ../../src/acc_env.sv(33) @ 7975 ns: reporter [QUEUE DISPLAY VISITOR] Scoreboard queue size changed! New size:         133
UVM_INFO ../../src/acc_env.sv(33) @ 7977 ns: reporter [QUEUE DISPLAY VISITOR] Scoreboard queue size changed! New size:         132
UVM_INFO ../../src/acc_env.sv(33) @ 7979 ns: reporter [QUEUE DISPLAY VISITOR] Scoreboard queue size changed! New size:         131
UVM_INFO ../../src/acc_env.sv(33) @ 7981 ns: reporter [QUEUE DISPLAY VISITOR] Scoreboard queue size changed! New size:         130
UVM_INFO ../../src/acc_env.sv(33) @ 7983 ns: reporter [QUEUE DISPLAY VISITOR] Scoreboard queue size changed! New size:         129
UVM_INFO ../../src/acc_env.sv(33) @ 7985 ns: reporter [QUEUE DISPLAY VISITOR] Scoreboard queue size changed! New size:         128
UVM_INFO ../../src/acc_env.sv(33) @ 7987 ns: reporter [QUEUE DISPLAY VISITOR] Scoreboard queue size changed! New size:         127
UVM_INFO ../../src/acc_env.sv(33) @ 7989 ns: reporter [QUEUE DISPLAY VISITOR] Scoreboard queue size changed! New size:         126
UVM_INFO ../../src/acc_env.sv(33) @ 7991 ns: reporter [QUEUE DISPLAY VISITOR] Scoreboard queue size changed! New size:         125
UVM_INFO ../../src/acc_env.sv(33) @ 7993 ns: reporter [QUEUE DISPLAY VISITOR] Scoreboard queue size changed! New size:         124
UVM_INFO ../../src/acc_env.sv(33) @ 7995 ns: reporter [QUEUE DISPLAY VISITOR] Scoreboard queue size changed! New size:         123
UVM_INFO ../../src/acc_env.sv(33) @ 7997 ns: reporter [QUEUE DISPLAY VISITOR] Scoreboard queue size changed! New size:         122
UVM_INFO ../../src/acc_env.sv(33) @ 7999 ns: reporter [QUEUE DISPLAY VISITOR] Scoreboard queue size changed! New size:         121
UVM_INFO ../../src/acc_env.sv(33) @ 8001 ns: reporter [QUEUE DISPLAY VISITOR] Scoreboard queue size changed! New size:         120
UVM_INFO ../../src/acc_env.sv(33) @ 8003 ns: reporter [QUEUE DISPLAY VISITOR] Scoreboard queue size changed! New size:         119
UVM_INFO ../../src/acc_env.sv(33) @ 8005 ns: reporter [QUEUE DISPLAY VISITOR] Scoreboard queue size changed! New size:         118
UVM_INFO ../../src/acc_env.sv(33) @ 8007 ns: reporter [QUEUE DISPLAY VISITOR] Scoreboard queue size changed! New size:         117
UVM_INFO ../../src/acc_env.sv(33) @ 8009 ns: reporter [QUEUE DISPLAY VISITOR] Scoreboard queue size changed! New size:         116
UVM_INFO ../../src/acc_env.sv(33) @ 8011 ns: reporter [QUEUE DISPLAY VISITOR] Scoreboard queue size changed! New size:         115
UVM_INFO ../../src/acc_env.sv(33) @ 8013 ns: reporter [QUEUE DISPLAY VISITOR] Scoreboard queue size changed! New size:         114
UVM_INFO ../../src/acc_env.sv(33) @ 8015 ns: reporter [QUEUE DISPLAY VISITOR] Scoreboard queue size changed! New size:         113
UVM_INFO ../../src/acc_env.sv(33) @ 8017 ns: reporter [QUEUE DISPLAY VISITOR] Scoreboard queue size changed! New size:         112
```


Cogita
acc_indirect_wr_sys4_test.log
Waves
Parsers
Config
Layout
Tooltips
About
Theme
Radio
Timeline
1977
9
2723
5438
8152
10866
All Messages
DEC
619
309.5
0
[QUEUE DISPLAY VISITOR]
DEC
151
75.5
0
Log Messages
Bookmarks (0)
Unparsed Lines (1606)
verbosity
fil...
timepoint
e...
Emitter - function
message
UVM_INFO ... 1977 ...r [QUEUE DISPLAY VISITOR] Scoreboard queue size changed! New size: 92
UVM_INFO ... 1979 ...r [QUEUE DISPLAY VISITOR] Scoreboard queue size changed! New size: 91
UVM_INFO ... 1981 ...r [QUEUE DISPLAY VISITOR] Scoreboard queue size changed! New size: 90
UVM_INFO ... 1983 ...r [QUEUE DISPLAY VISITOR] Scoreboard queue size changed! New size: 89
UVM_INFO ... 1985 ...r [QUEUE DISPLAY VISITOR] Scoreboard queue size changed! New size: 88
UVM_INFO ... 1987 ...r [QUEUE DISPLAY VISITOR] Scoreboard queue size changed! New size: 87
UVM_INFO ... 1989 ...r [QUEUE DISPLAY VISITOR] Scoreboard queue size changed! New size: 86
UVM_INFO ... 1991 ...r [QUEUE DISPLAY VISITOR] Scoreboard queue size changed! New size: 85


Cogita
acc_indirect_wr_sys4_test.log
Waves
Parsers
Config
Layout
Tooltips
About
Theme
Radio
Timeline
2881
1
2715
5430
8144
10858
All Messages
[REPORTING VISITOR]
[REPORTING VISITOR] - driver: get_next_item
[REPORTING VISITOR] - signal change
[REPORTING VISITOR - event
[REPORTING VISITOR] - queue size
DEC
3
1.5
0
152
76
Log Messages
Bookmarks (0)
Unparsed Lines (1606)
verbosity
fil...
timepoint
e...
Emitter - function
message
UVM_INFO
2881
[REPORTING VISITOR]
Scoreboard queue size changed!. New size:
104
UVM_INFO
2881
[REPORTING VISITOR]

Summary

- Visitors are an ideal way to externally and retroactively add functionality to UVM testbenches
- Reporting system using a dedicated tool such as Cogita, making the concept even better.
- Particularly beneficial in large and complex SoCs, with large teams and many 3rd party IPs and VIPs


TSUTAYA


TSUTAYA


Questions?