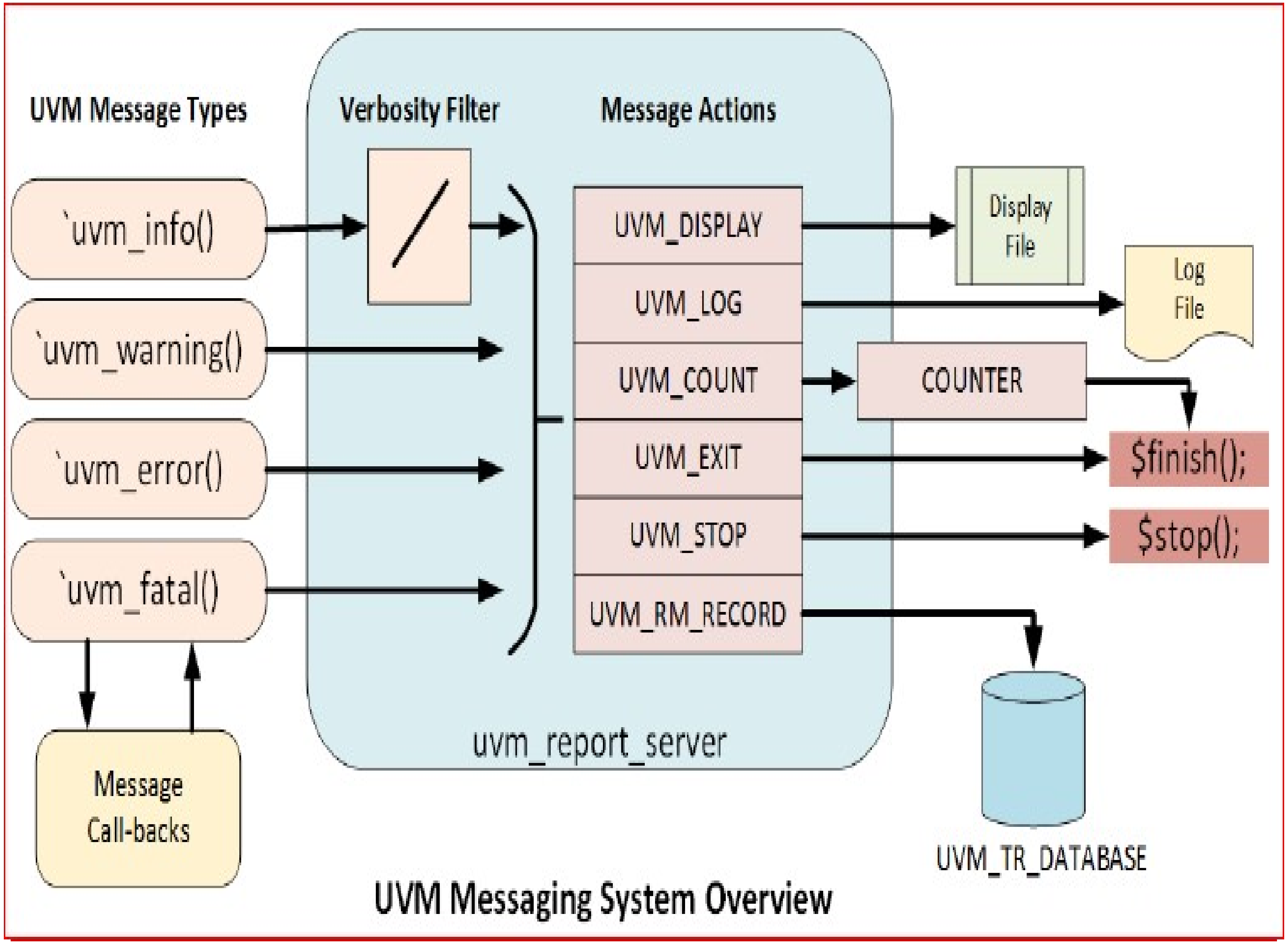
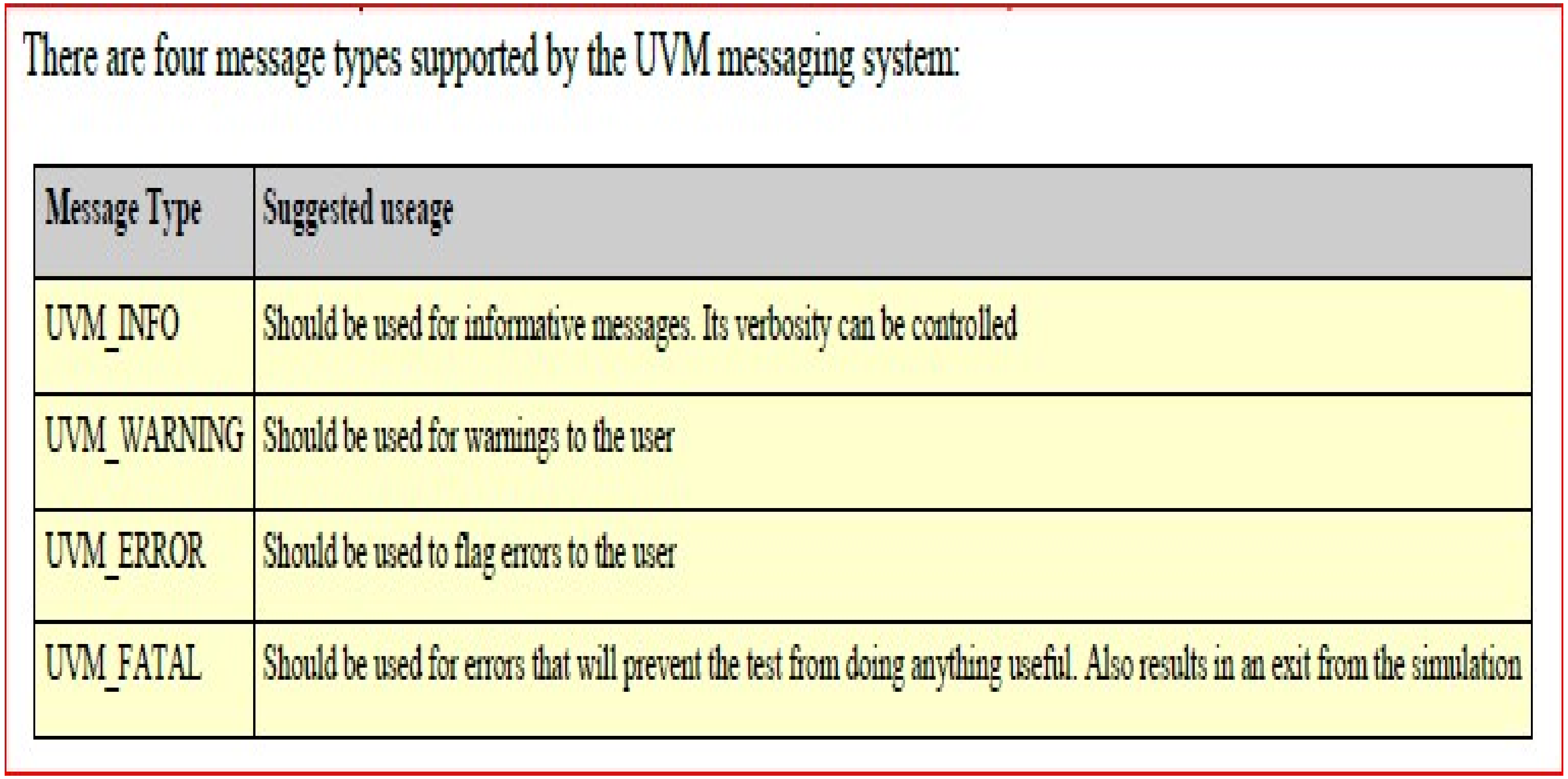
Universal Verification Methodology Lecture3

Pankaj Badhe



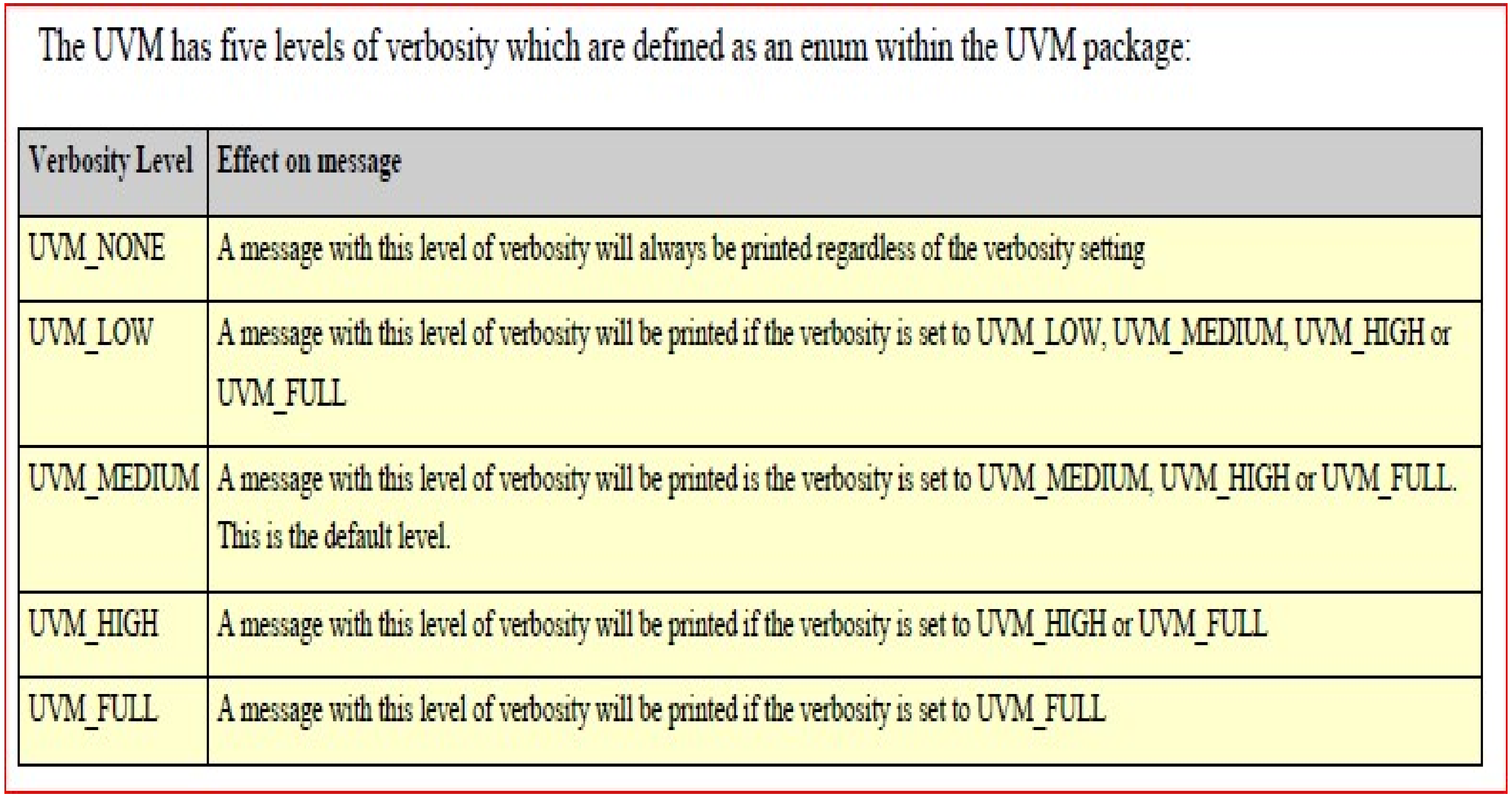
UVM Messaging System Overview

UVM Messaging



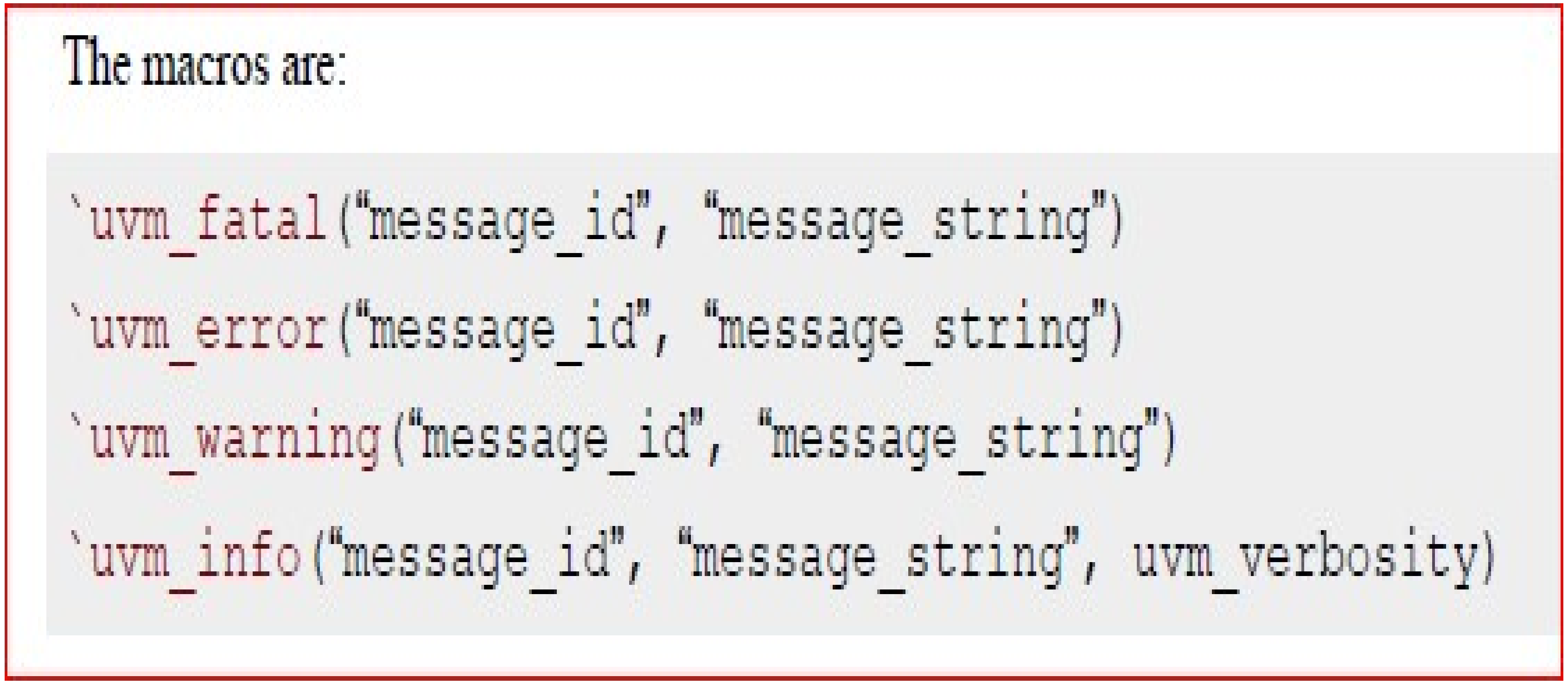
There are four message types supported by the UVM messaging system:

Message Type	Suggested useage
UVM_INFO	Should be used for informative messages. Its verbosity can be controlled
UVM_WARNING	Should be used for warnings to the user
UVM_ERROR	Should be used to flag errors to the user
UVM_FATAL	Should be used for errors that will prevent the test from doing anything useful. Also results in an exit from the simulation

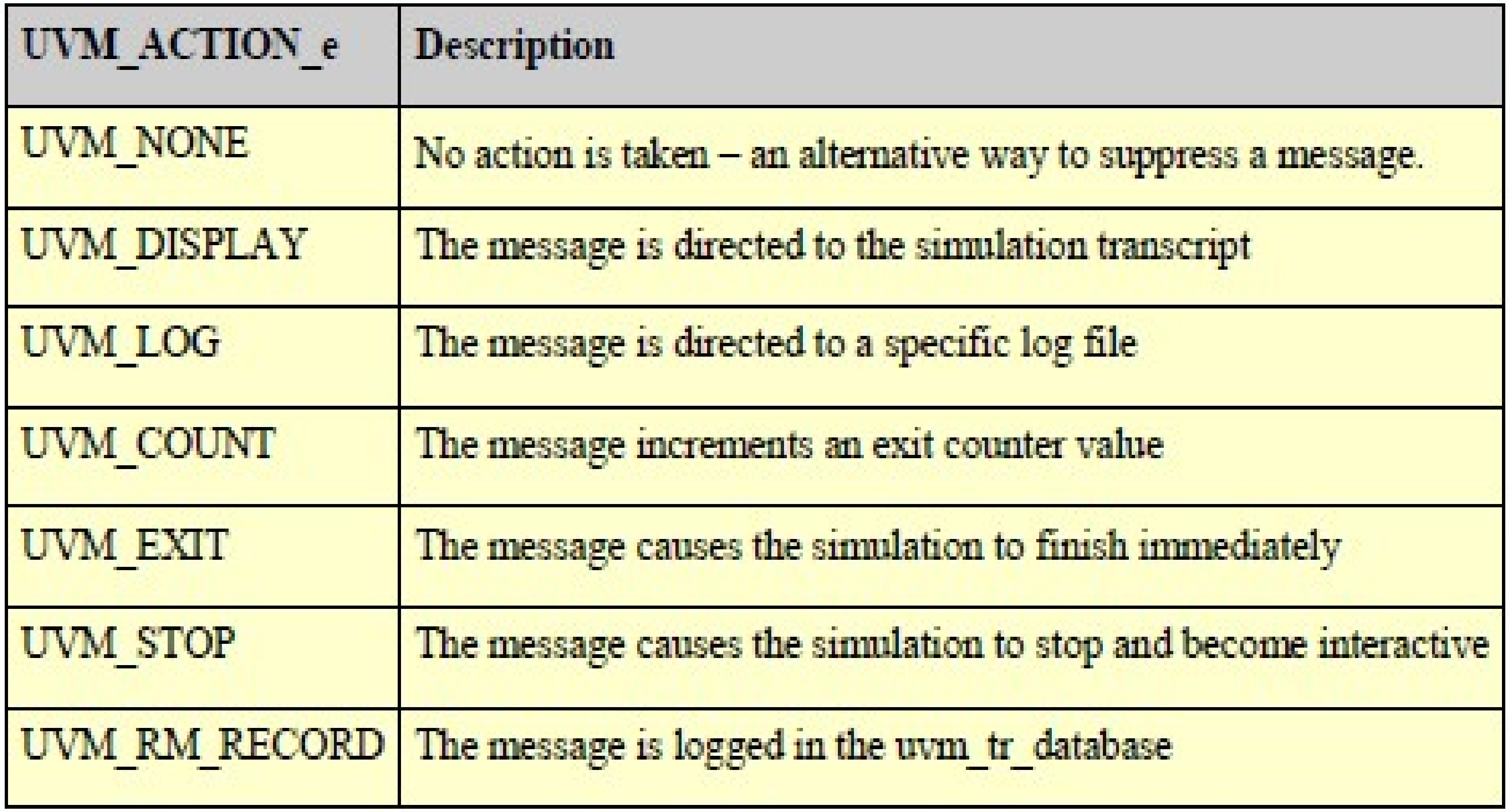


The UVM has five levels of verbosity which are defined as an enum within the UVM package:

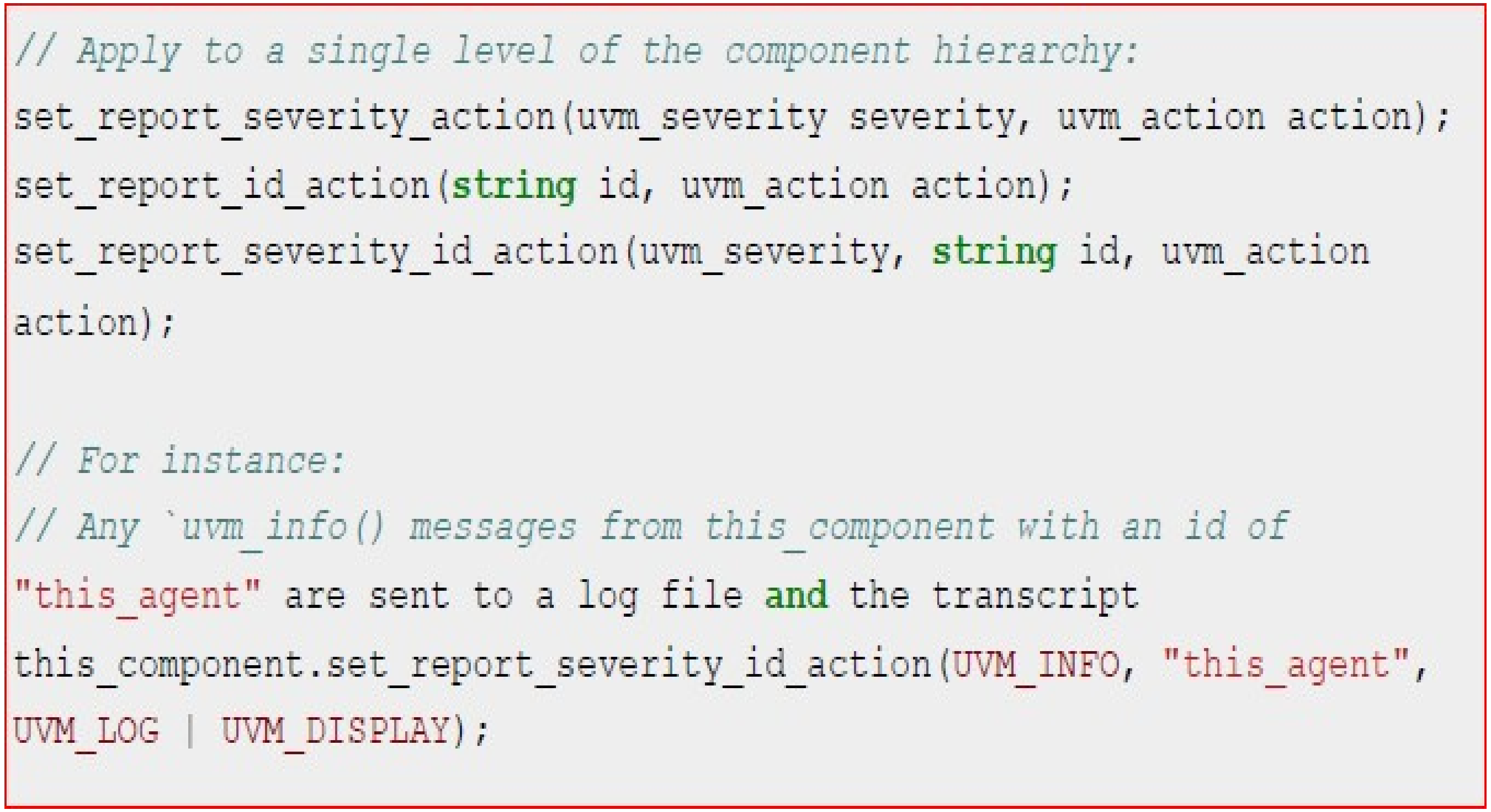
Verbosity Level	Effect on message
UVM_NONE	A message with this level of verbosity will always be printed regardless of the verbosity setting
UVM_LOW	A message with this level of verbosity will be printed if the verbosity is set to UVM_LOW, UVM_MEDIUM, UVM_HIGH or UVM_FULL
UVM_MEDIUM	A message with this level of verbosity will be printed is the verbosity is set to UVM_MEDIUM, UVM_HIGH or UVM_FULL. This is the default level.
UVM_HIGH	A message with this level of verbosity will be printed if the verbosity is set to UVM_HIGH or UVM_FULL
UVM_FULL	A message with this level of verbosity will be printed if the verbosity is set to UVM_FULL


The macros are:

```
`uvm_fatal("message_id", "message_string")
`uvm_error("message_id", "message_string")
`uvm_warning("message_id", "message_string")
`uvm_info("message_id", "message_string", uvm_verbosity)
```



UVM_ACTION_e	Description
UVM_NONE	No action is taken – an alternative way to suppress a message.
UVM_DISPLAY	The message is directed to the simulation transcript
UVM_LOG	The message is directed to a specific log file
UVM_COUNT	The message increments an exit counter value
UVM_EXIT	The message causes the simulation to finish immediately
UVM_STOP	The message causes the simulation to stop and become interactive
UVM_RM_RECORD	The message is logged in the uvm_tr_database

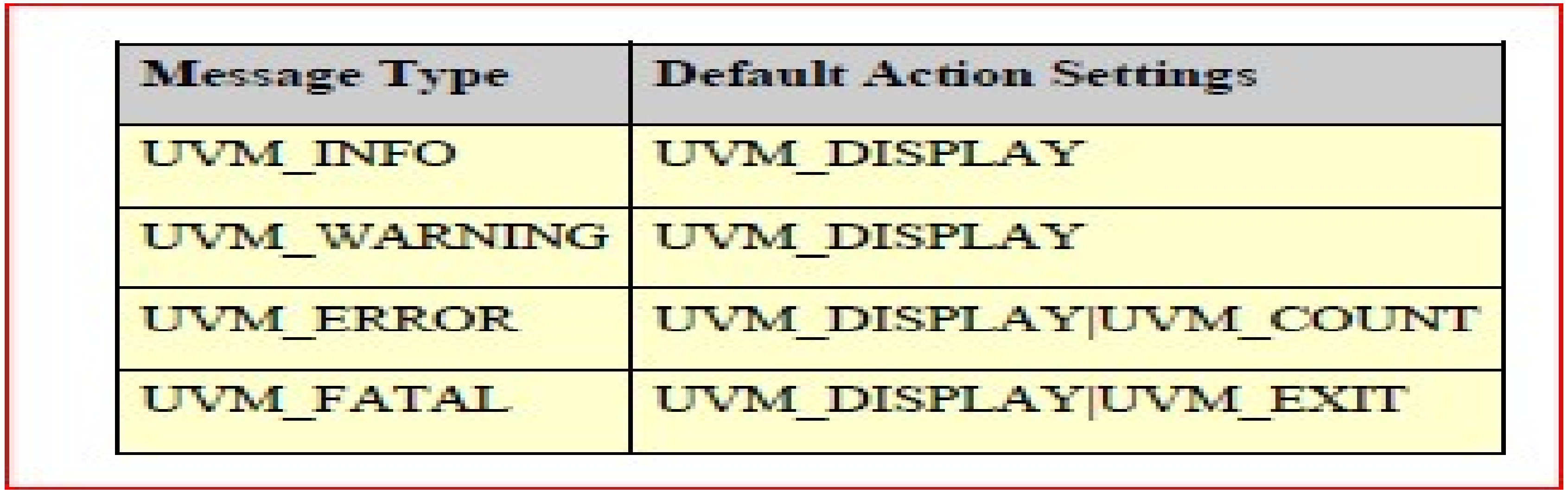


```
// Apply to a single level of the component hierarchy:
set_report_severity_action(uvm_severity severity, uvm_action action);
set_report_id_action(string id, uvm_action action);
set_report_severity_id_action(uvm_severity, string id, uvm_action
action);

// For instance:
// Any `uvm_info() messages from this_component with an id of
"this_agent" are sent to a log file and the transcript
this_component.set_report_severity_id_action(UVM_INFO, "this_agent",
UVM_LOG | UVM_DISPLAY);
```

Log file handling

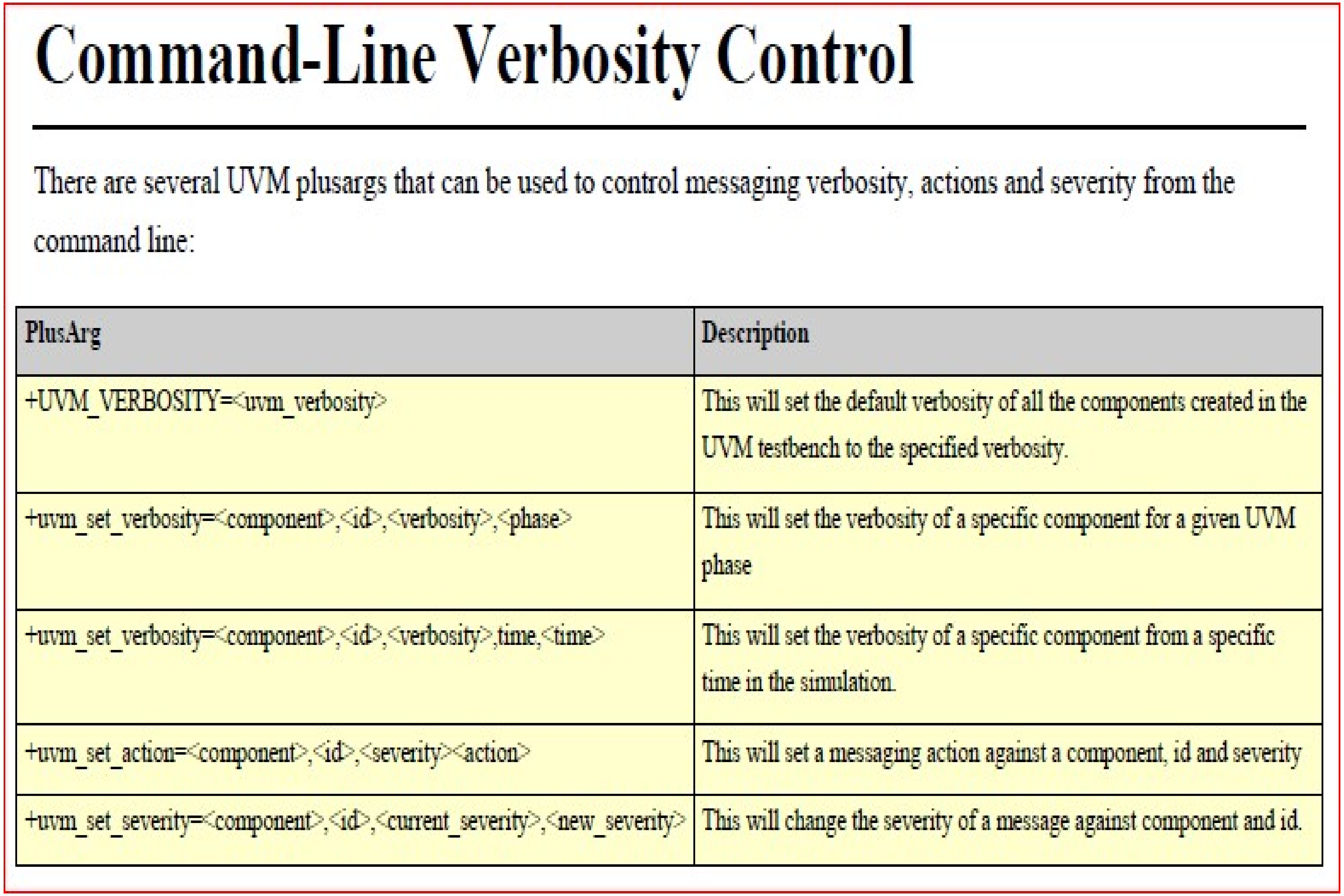
```
task run_phase(uvm_phase phase);

  // Create a file handle by opening a file:
  UVM_FILE green_log_fh = $fopen("green_messages.log");

  // Adding the log action for the "green_id":
  env.green.set_report_id_action("green_id", (UVM_DISPLAY | UVM_LOG));
  // Setting the file handle for the log action
  env.green.set_report_id_file("green_id", green_log_fh);


  phase.raise_objection(this);
  #1us;
  // Test code goes here
  phase.drop_objection(this);

  // Closing the log file at the end of test:
  $fclose(green_log_fh);
endtask
```

Default actions



Message Type	Default Action Settings
UVM_INFO	UVM_DISPLAY
UVM_WARNING	UVM_DISPLAY
UVM_ERROR	UVM_DISPLAY|UVM_COUNT
UVM_FATAL	UVM_DISPLAY|UVM_EXIT

- Report-catcher

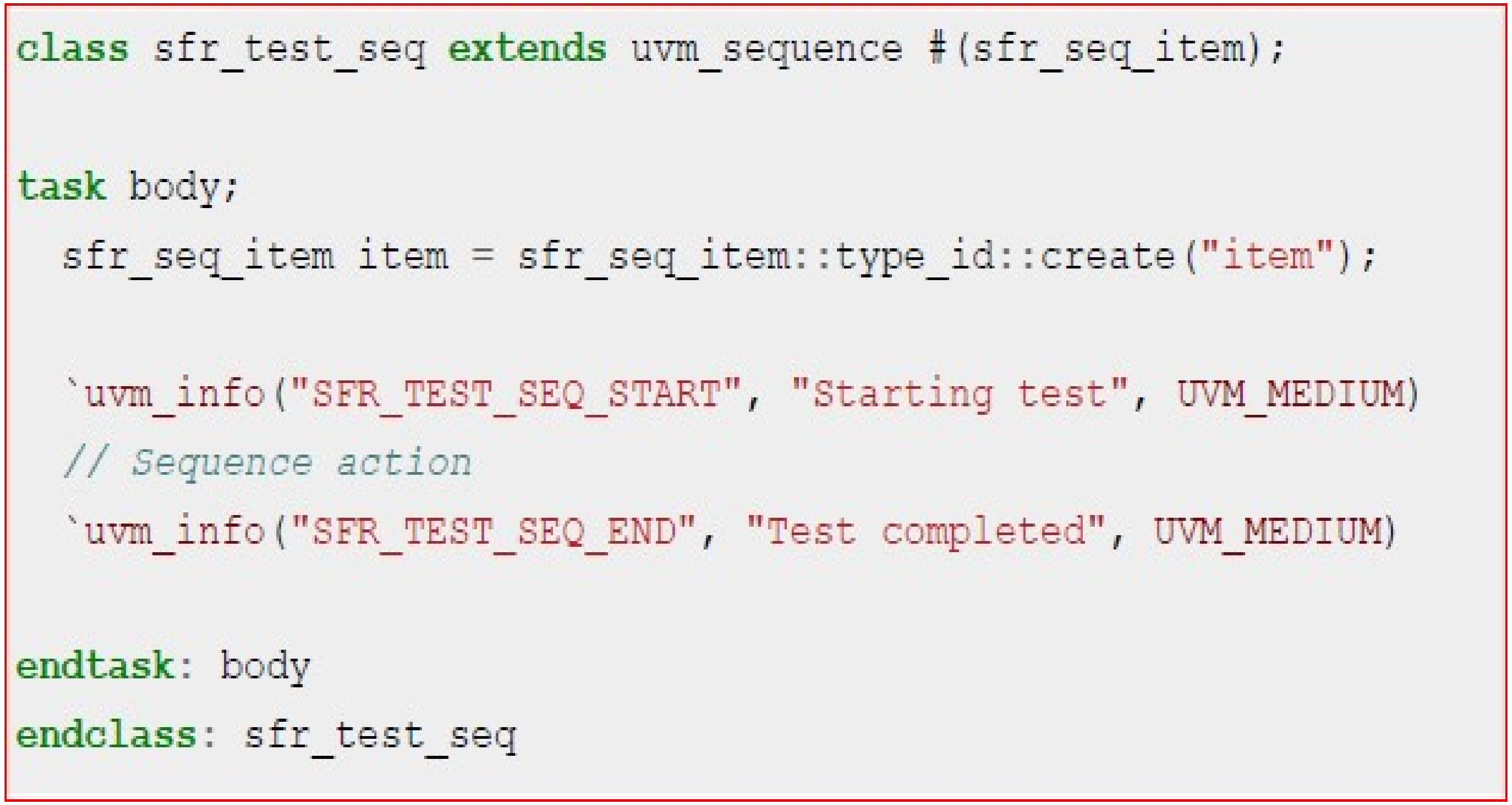


Command-Line Verbosity Control

There are several UVM plusargs that can be used to control messaging verbosity, actions and severity from the command line:

PlusArg	Description
+UVM_VERBOSITY=<uvm_verbosity>	This will set the default verbosity of all the components created in the UVM testbench to the specified verbosity.
+uvm_set_verbosity=<component>,<id>,<verbosity>,<phase>	This will set the verbosity of a specific component for a given UVM phase
+uvm_set_verbosity=<component>,<id>,<verbosity>,time,<time>	This will set the verbosity of a specific component from a specific time in the simulation.
+uvm_set_action=<component>,<id>,<severity><action>	This will set a messaging action against a component, id and severity
+uvm_set_severity=<component>,<id>,<current_severity>,<new_severity>	This will change the severity of a message against component and id.

Examples of messages



```
class sfr_test_seq extends uvm_sequence #(sfr_seq_item);


task body;
  sfr_seq_item item = sfr_seq_item::type_id::create("item");


  `uvm_info("SFR_TEST_SEQ_START", "Starting test", UVM_MEDIUM)
  // Sequence action
  `uvm_info("SFR_TEST_SEQ_END", "Test completed", UVM_MEDIUM)


endtask: body
endclass: sfr_test_seq
```

RAL

- uvm_ral