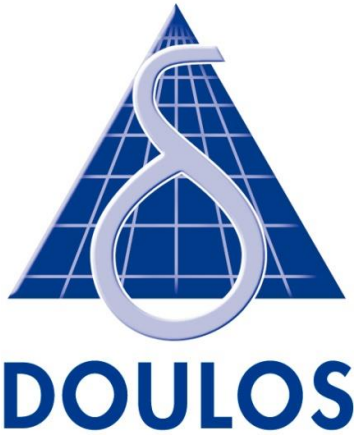
Easier UVM: Learning and Using UVM with a Code Generator

John Aynsley, Doulos



accellera SYSTEMS INITIATIVE

2015 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

Easier UVM:
Learning and Using UVM with a Code Generator

- Introduction to UVM
- Easier UVM?
- The Easier UVM Code Generator
- Reporting
- Phases and Configuration
- TLM Connections
- The Factory
- Sequences and Tests

accellera SYSTEMS INITIATIVE

2015 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

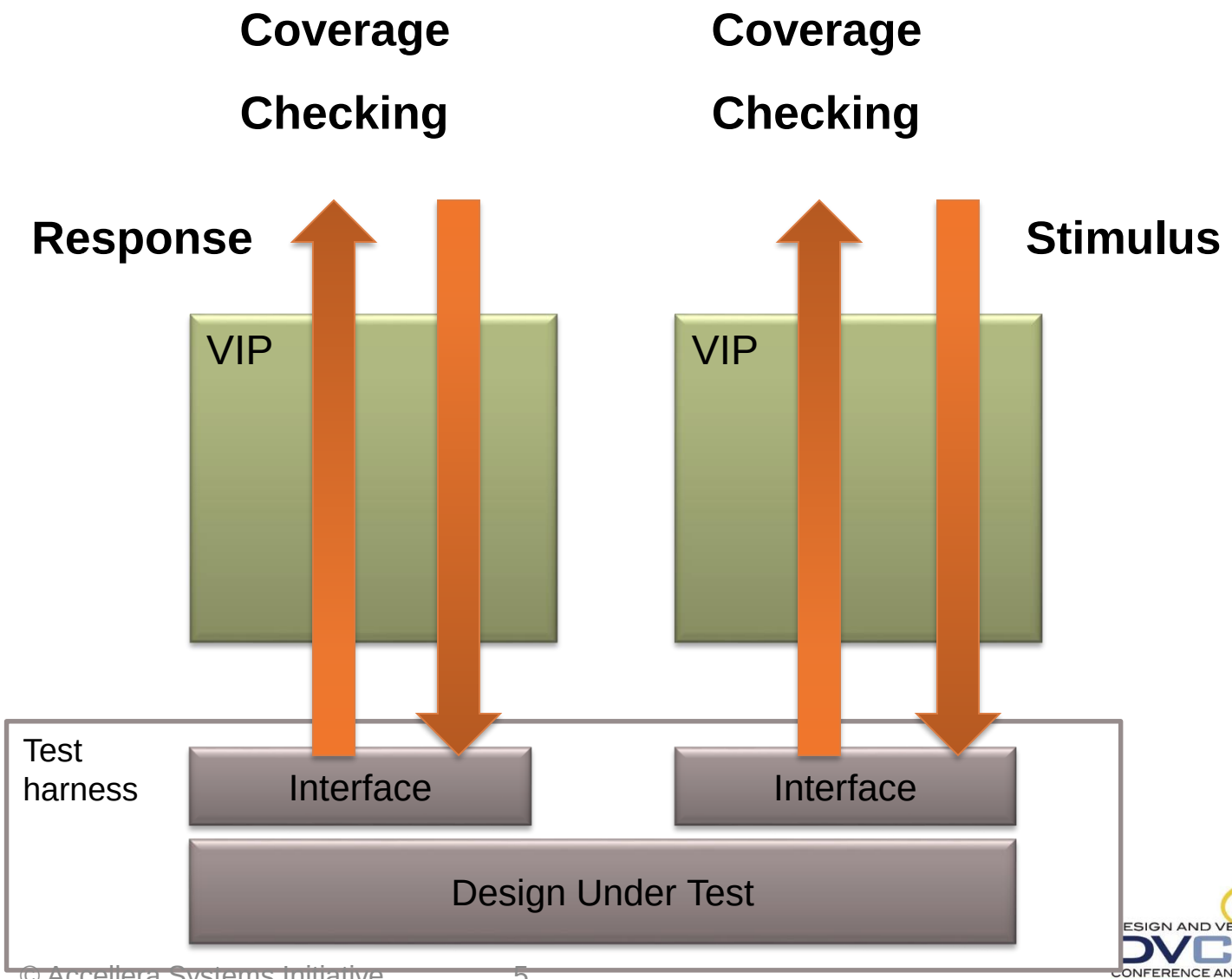
What is UVM?

- The Universal Verification Methodology for SystemVerilog
- Supports constrained random, coverage-driven verification
- An open-source base class library
- An Accellera standard
- Supported by all major simulator vendors
- Very strong adoption!

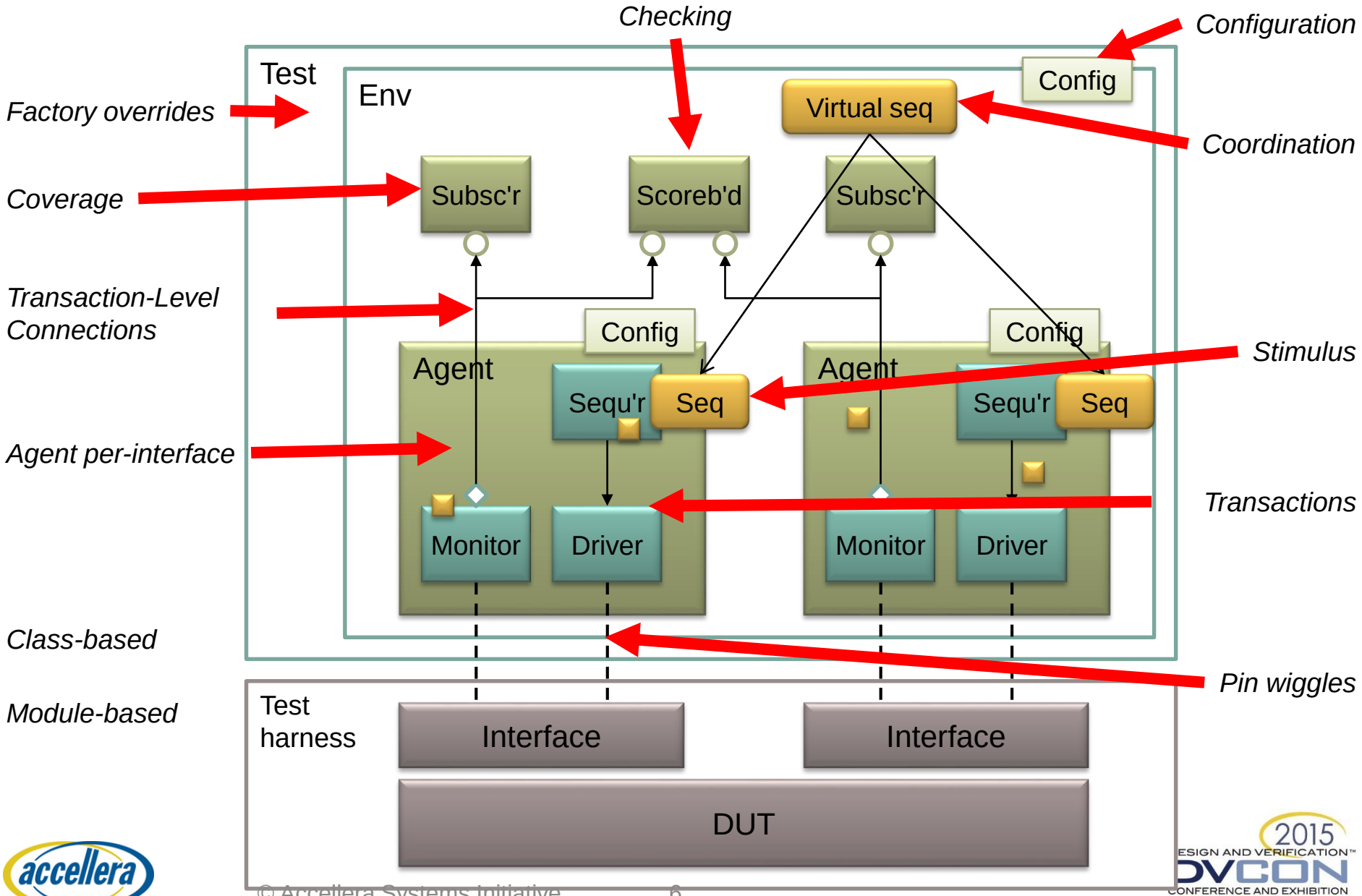
Why UVM?

- **Best practice**
  - Consistency, uniformity, don't reinvent the wheel, avoid pitfalls
- **Reuse**
  - Verification IP, verification environments, tests, people, knowhow

Constrained Random Verification



UVM Verification Environment



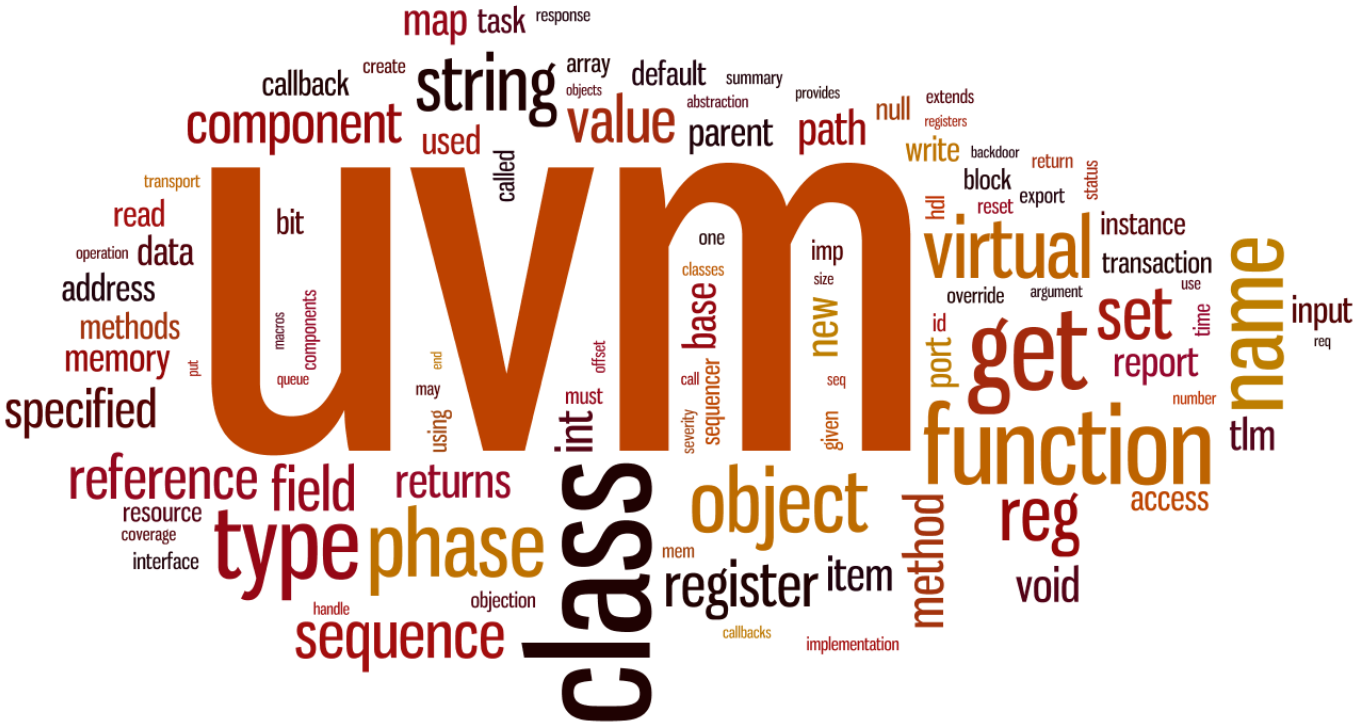
Easier UVM:
Learning and Using UVM with a Code Generator

- Introduction to UVM
- Easier UVM?
- The Easier UVM Code Generator
- Reporting
- Phases and Configuration
- TLM Connections
- The Factory
- Sequences and Tests

accellera SYSTEMS INITIATIVE

2015 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

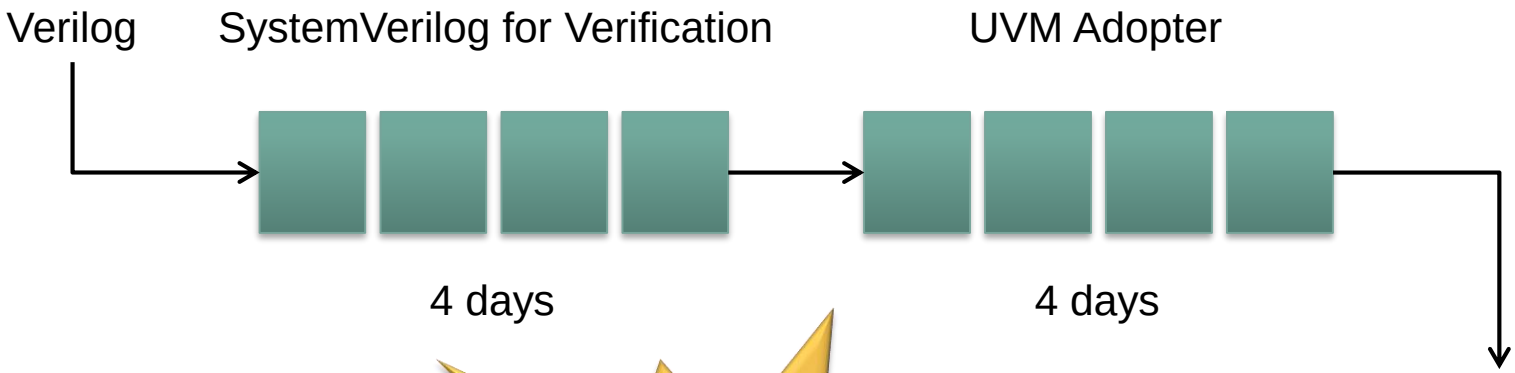
UVM Itself is Challenging



- ~ 300 classes in UVM Base Class Library
- The UVM documentation does not answer all the questions
- There are many ways to do the same thing - need a methodology

The Learning Curve

- Doulos training



What next?

Coaching

UVM-aware tools

Coding guidelines

Doulos – Easier UVM

- Coding guidelines – "One way to do it"
- Automatic code generator

Free and open

Apache 2.0 license

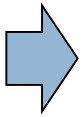
- Help individuals and project teams
  - learn UVM and avoid pitfalls
  - become productive with UVM (saves ~ 6 weeks)
  - use UVM consistently
- Reduce the burden of supporting UVM code

Easier UVM Coding Guidelines

- 180 detailed guidelines with explanations and examples
- Consistent with code generator
- Common sense

Consistency
Avoiding pitfalls
Reusability
High quality verification

Easier UVM:
Learning and Using UVM with a Code Generator



- Introduction to UVM
- Easier UVM?
- The Easier UVM Code Generator
- Reporting
- Phases and Configuration
- TLM Connections
- The Factory
- Sequences and Tests

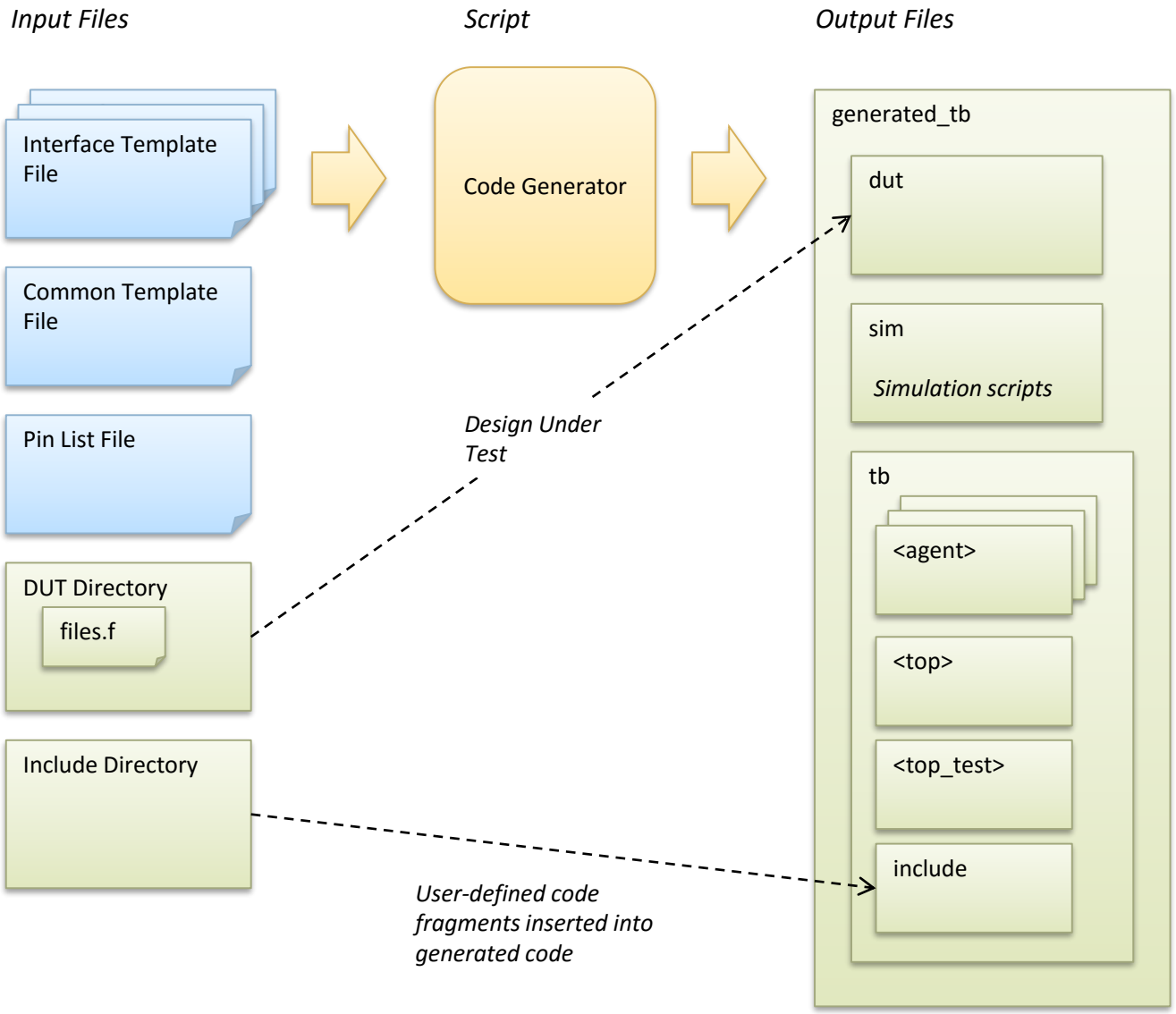
accellera SYSTEMS INITIATIVE

2015 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

Ways to Use the Code Generator

- Generate examples simply as a learning aid
- Create an initial framework for production code
- Continue to regenerate the code throughout the project

Code Generator Inputs and Outputs



Download Easier UVM

http://www.doulos.com/easier

```
easier_uvm_gen.pl
release_notes
examples
    example_templates
    minimal
    minimal_plus
    minimal_reg
    multi_if
```

From the Command Line

```
unzip easier_uvm_gen-2015-06-29.zip
cd examples/minimal_plus
```

Run the code generator

```
perl ../../easier_uvm.pl clkndata.tpl

cd generated_tb/sim
```

Run your simulator

```
compile_ius.sh
```

or

```
compile_vcs.sh
```

or

```
vsim -c -do "do compile_questa.do; run -all"
```

or

```
cd ../dut
vsimsa -do ../sim/compile_riviera.do
```

Generated Files

```
tb/clkndata/sv/
  clkndata_data_tx.sv
  clkndata_if.sv
  clkndata_config.sv
  clkndata_driver.sv
  clkndata_monitor.sv
  clkndata_sequencer.sv
  clkndata_agent.sv
  clkndata_coverage.sv
  clkndata_seq_lib.sv
  clkndata_pkg.sv
```

Per-agent

Transaction

Interface

Agent

Sequences

Package

```
tb/top/sv/
  top_config.sv
  top_env.sv
  top_seq_lib.sv
  top_pkg.sv

tb/top_test/sv/
  top_test.sv
  top_test_pkg.sv

tb/top_tb/sv/
  top_tb.sv
  top_th/sv

tb/include
```

Env

Test

Modules

User-defined

Through Code Generation to the Simulator

Interface template: 1 agent + 1 interface

```
agent_name = clkndata
trans_item = data_tx
trans_var  = rand byte data;

driver_inc      = clkndata_do
monitor_inc     = clkndata_do
agent_cover_inc = clkndata_co
agent_seq_inc   = my_clkndata
agent_factory_set = clkndata_

if_port    = logic clk;
if_port    = byte data;
if_clock   = clk
```

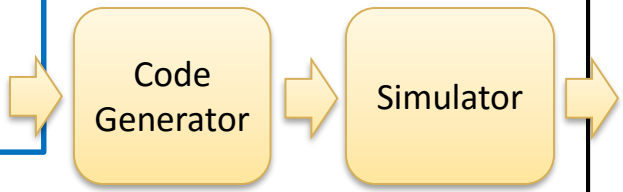
Common template

```
dut_source_path = mydut
dut_top         = mydut
dut_pfile       = pinlist
inc_path        = include
```

Pin list

```
!clkndata_if
clk  clk
data data
```

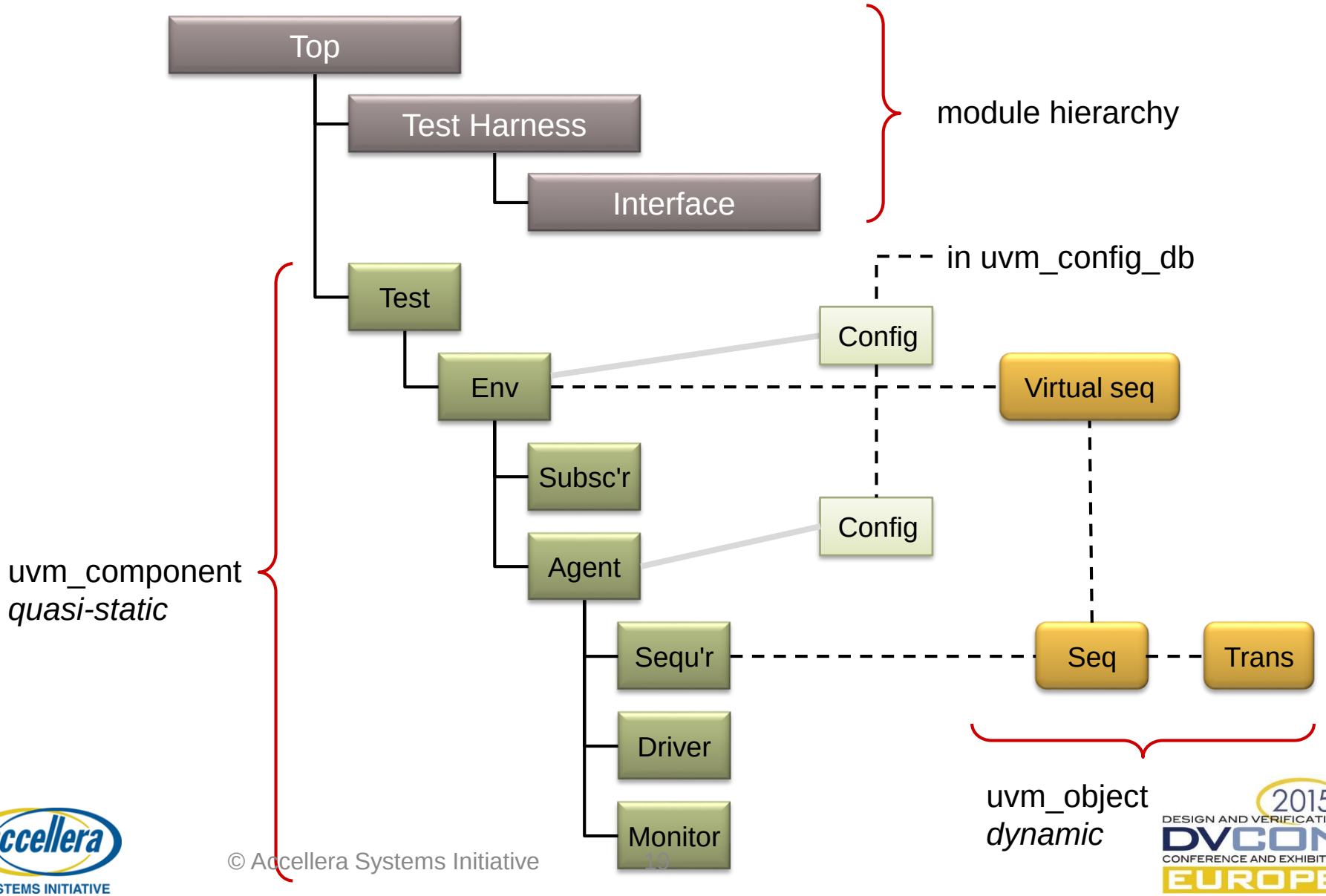
```
perl ../../easier_uvm.pl clkndata.tpl
```



UVM component hierarchy

- uvm_test_top
  - m_env
    - m_clkndata_coverage
    - m_clkndata_agent
      - m_sequencer
      - m_monitor
      - m_driver

Modules and UVM Objects



Easier UVM:
Learning and Using UVM with a Code Generator

- Introduction to UVM
- Easier UVM?
- The Easier UVM Code Generator
- Reporting
- Phases and Configuration
- TLM Connections
- The Factory
- Sequences and Tests

accellera SYSTEMS INITIATIVE

2015 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

Transaction Class

```
`ifndef CLKNDATA_SEQ_ITEM_SV
`define CLKNDATA_SEQ_ITEM_SV

class data_tx extends uvm_sequence_item;
  `uvm_object_utils(data_tx)

  rand byte data;

  extern function new(string name="");
  extern function void do_copy(uvm_object rhs);
  extern function bit  do_compare(uvm_object rhs, uvm_comparer comparer);
  extern function void do_print(uvm_printer printer);
  extern function void do_record(uvm_recorder recorder);
  extern function string convert2string();

endclass : data_tx
```

User-defined in template file

```
function void data_tx::do_copy(uvm_object rhs);
  data_tx rhs_;
  if (!$cast(rhs_, rhs))
    `uvm_fatal(get_type_name(), "cast of rhs object failed")
  super.do_copy(rhs);
  data = rhs_.data;
endfunction : do_copy
```

UVM reporting

Reporting

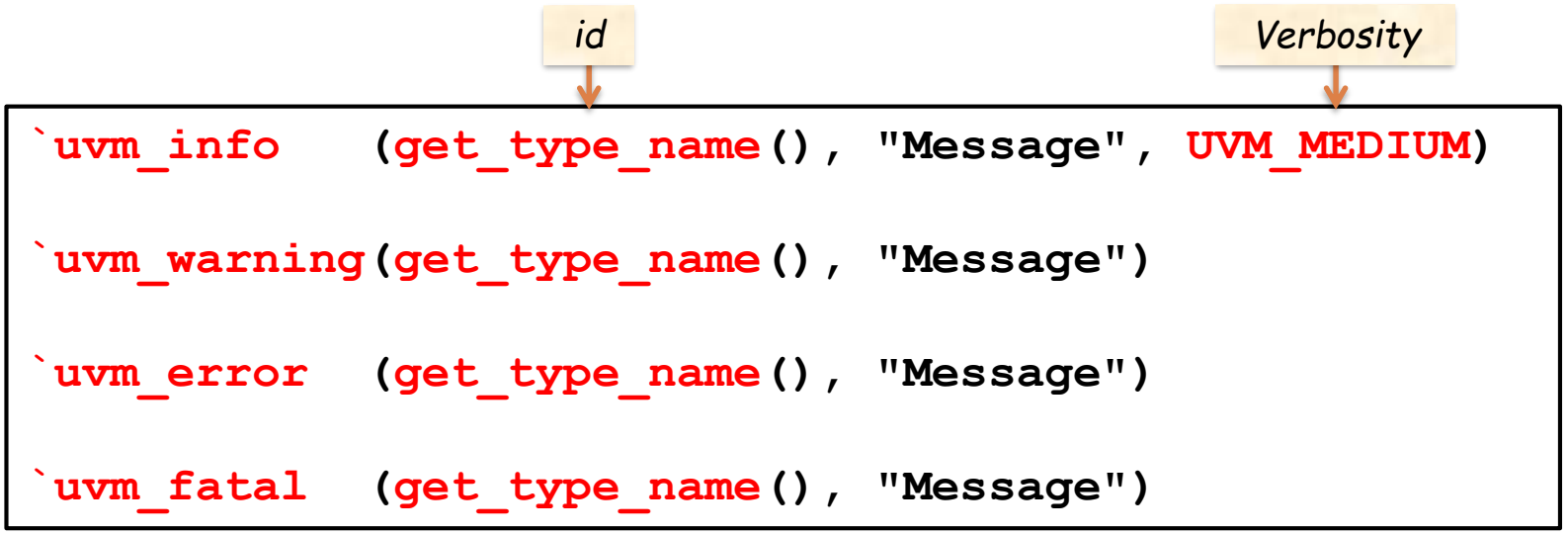
id *Verbosity*

```
`uvm_info    (get_type_name(), "Message", UVM_MEDIUM)

`uvm_warning(get_type_name(), "Message")

`uvm_error   (get_type_name(), "Message")

`uvm_fatal   (get_type_name(), "Message")
```

```
UVM_WARNING foo.sv(320) @ 105: uvm_test_top.env.path [data_tx] Bad!
```

Severity *File* *Line* *Time* *Instance* *id* *Message*

Verbosity




```
simulator ... +UVM_VERBOSITY=UVM_FULL
simulator ... +UVM_VERBOSITY=UVM_NONE
```

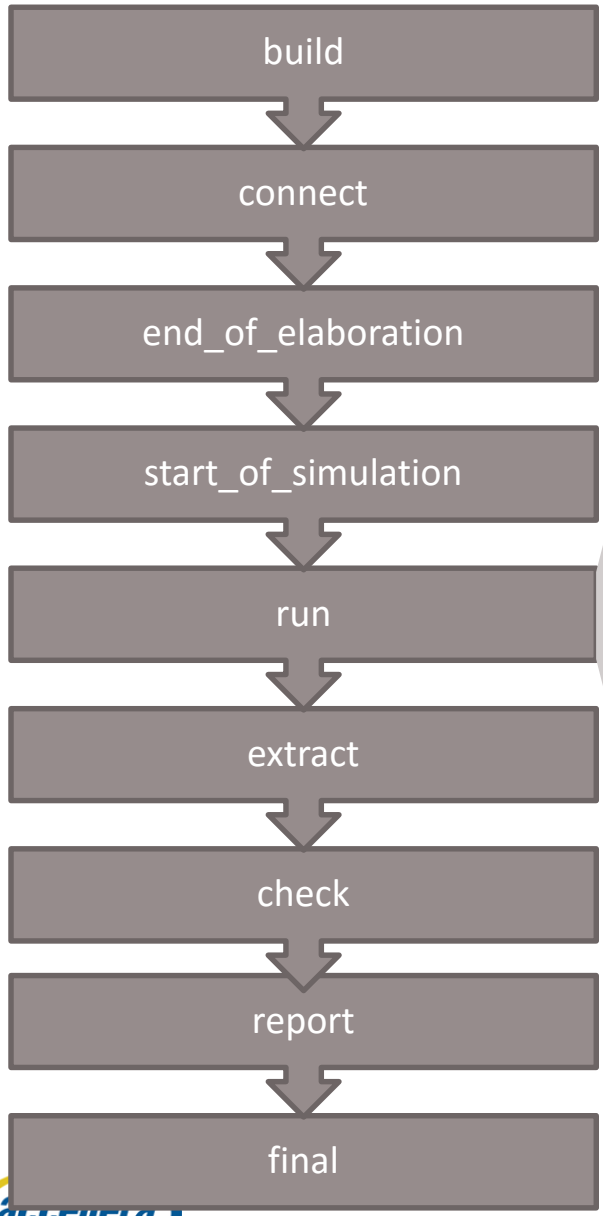
Report all info messages

Report only UVM_NONE messages

Easier UVM:
Learning and Using UVM with a Code Generator

- Introduction to UVM
- Easier UVM?
- The Easier UVM Code Generator
- Reporting
- Phases and Configuration
- TLM Connections
- The Factory
- Sequences and Tests

Execution Phases

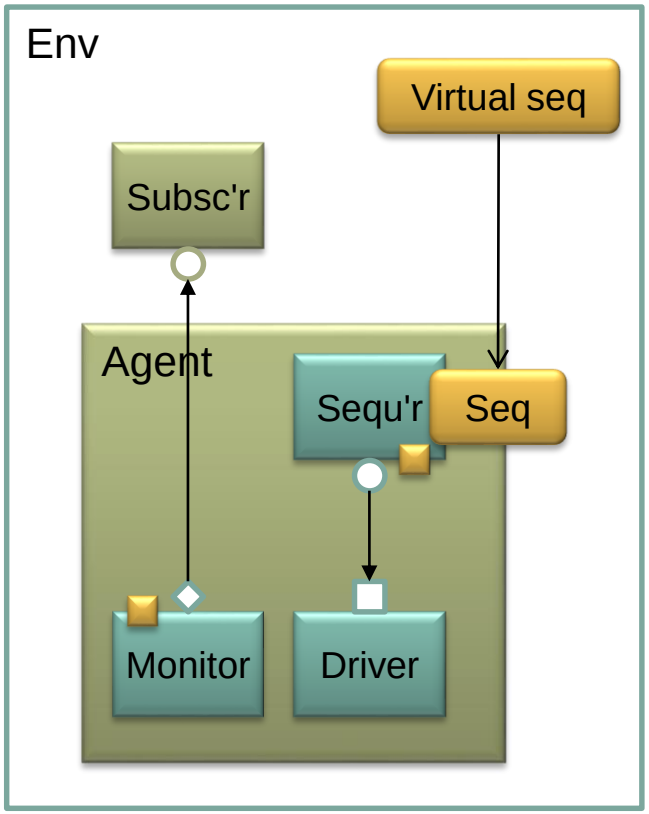


pre_reset
reset
post_reset

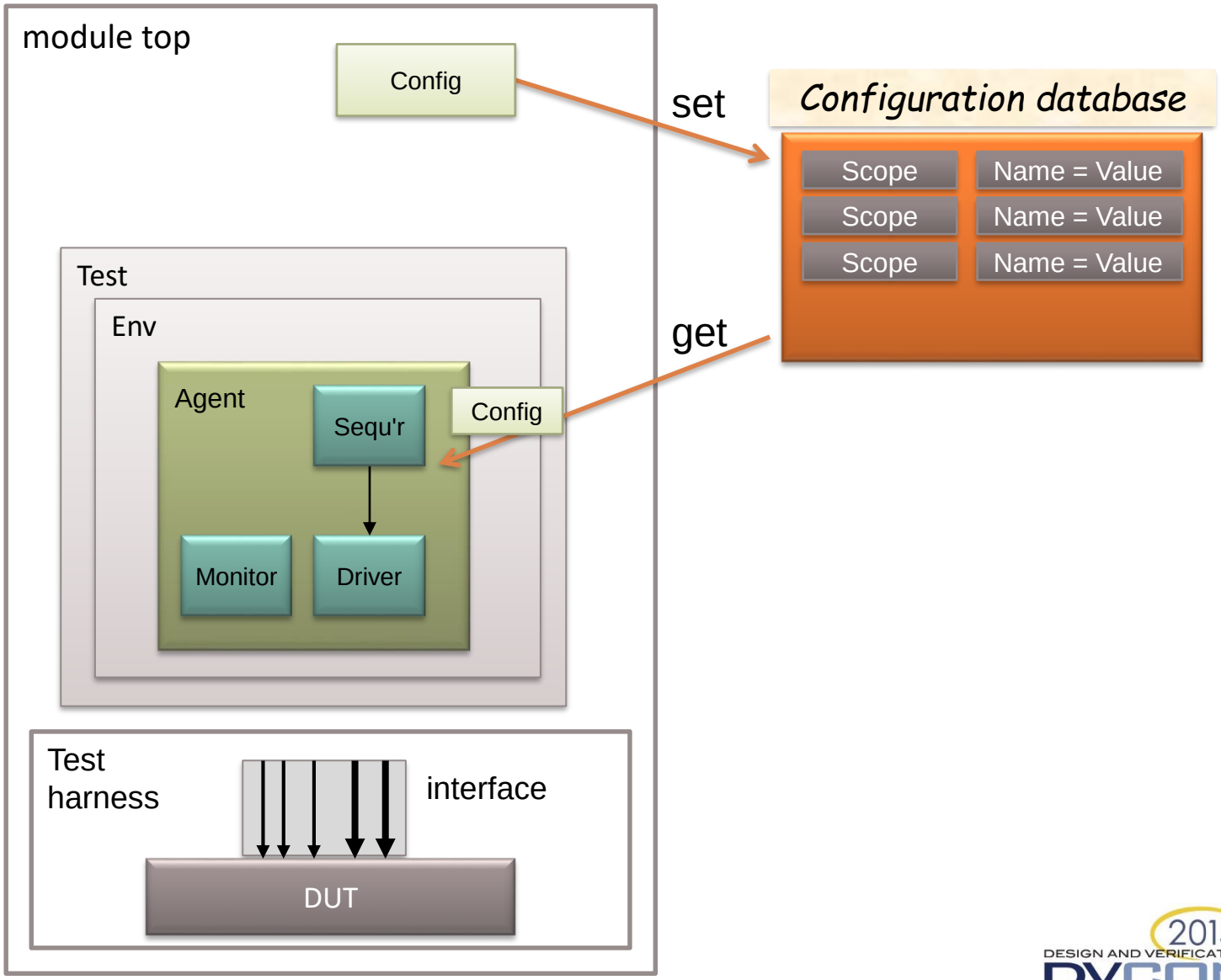
pre_configure
configure
post_configure

pre_main
main
post_main

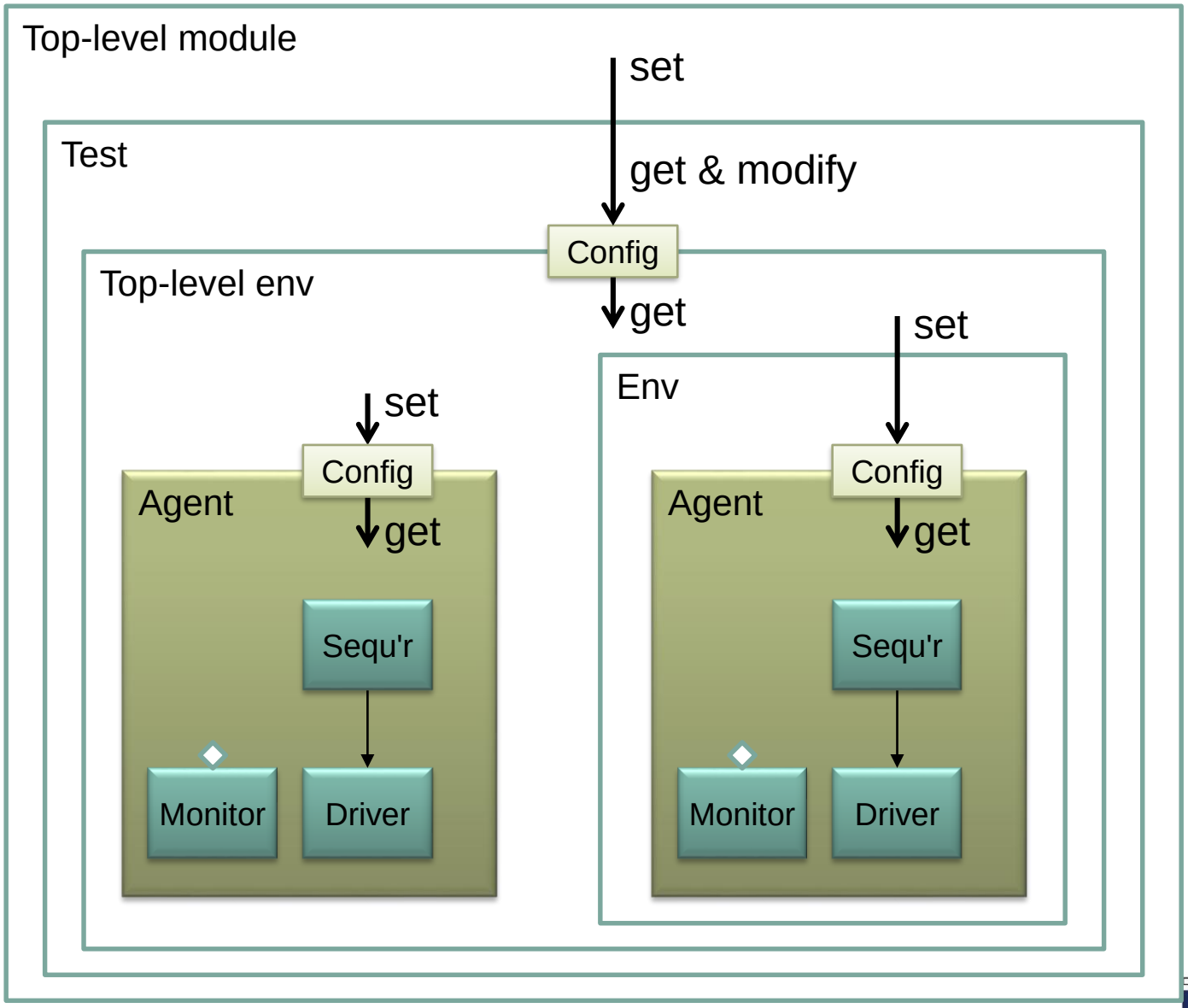
pre_shutdown
shutdown
post_shutdown



Configuration Database



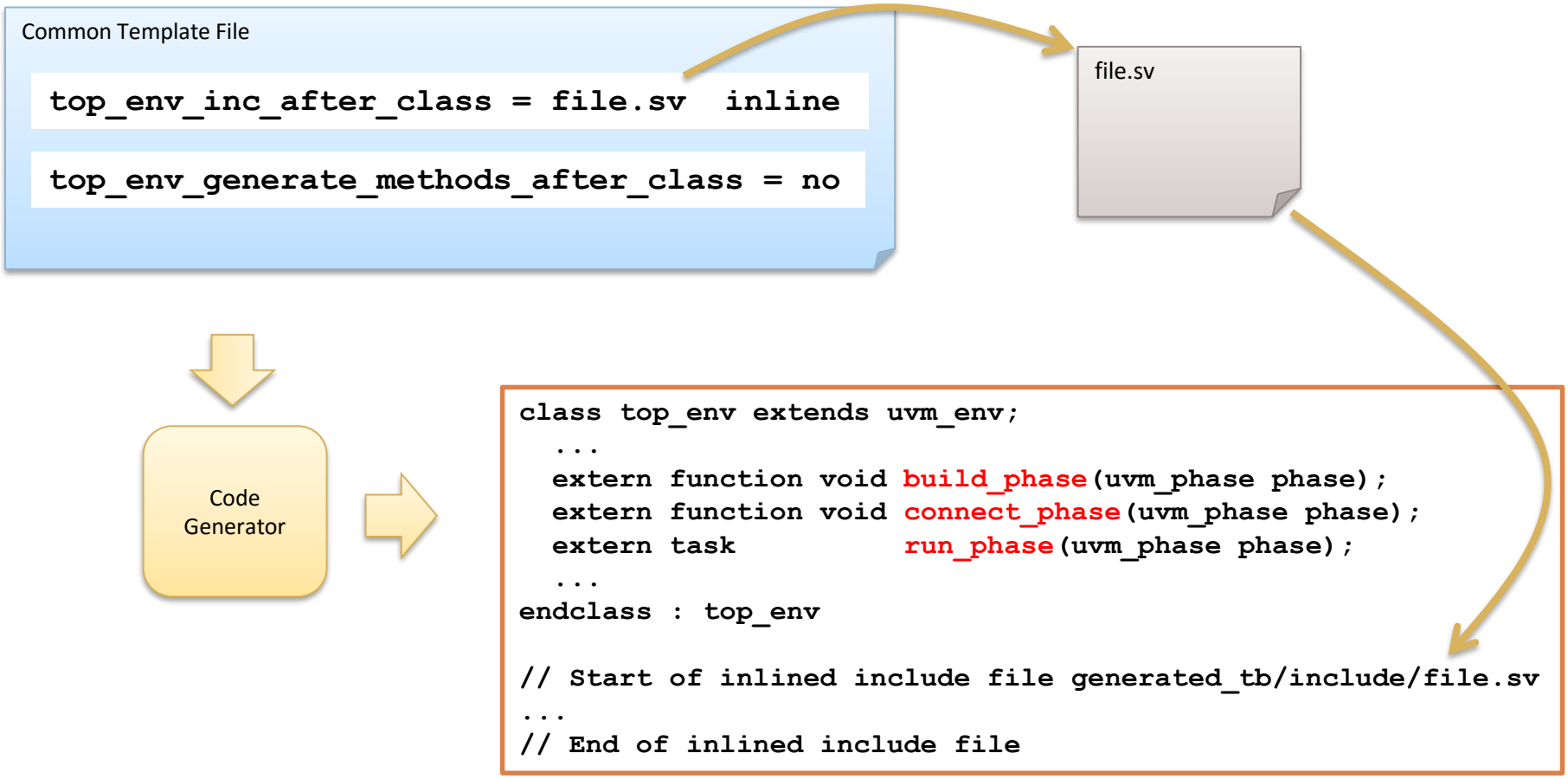
Env and Agent Configuration Objects



Configuration Class

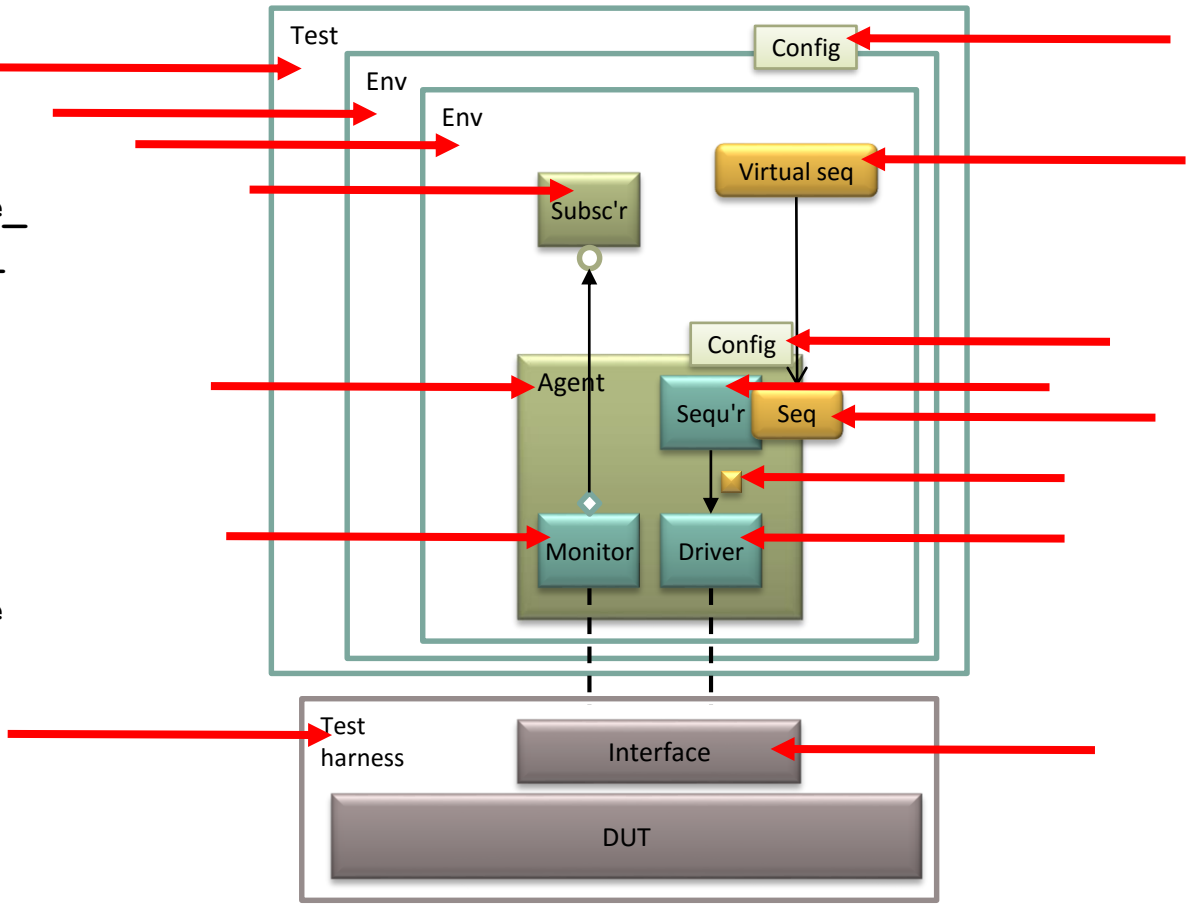
```
// You can insert code here by setting agent_config_inc_before_class

class top_config extends uvm_object;

  virtual clkndata_if     clkndata_vif;

  uvm_active_passive_enum is_active_clkndata = UVM_ACTIVE;
  bit                     checks_enable_clkndata;
  bit                     coverage_enable_clkndata;

  // You can insert variables here by setting config_var in clkndata.tpl

  extern function new(string name = "");

  // You can insert code here by setting agent_config_inc_inside_class

endclass : top_config

function top_config::new(string name = "");
  super.new(name);
endfunction : new

// You can insert code here by setting agent_config_inc_after_class
```

Inserting User-Defined Code Fragments



Template Settings

```
_generate_methods_inside_
_generate_methods_after_

_inc_before_
_inc_inside_
_inc_after_

_prepend_to_build_phase
_append_to_build_phase
_append_to_connect_phase
```



Top-Level Module

```
module top_tb;

  ...

  top_th th();

  top_config env_config;

  initial
  begin
    env_config = new("env_config");
    if ( !env_config.randomize() )
      `uvm_fatal("top_tb", "Failed to randomize config object")

    env_config.clkndata_vif              = th.clkndata_if_0;
    env_config.is_active_clkndata        = UVM_ACTIVE;
    env_config.checks_enable_clkndata    = 1;
    env_config.coverage_enable_clkndata = 1;

    uvm_config_db #(top_config)::set(
      null, "uvm_test_top.m_env", "config", env_config);

    run_test();
  end
endmodule
```

Test harness

config_db

Top-Level Env

```
function void top_env::build_phase(uvm_phase phase);
  `uvm_info(get_type_name(), "In build_phase", UVM_HIGH)

  // You can insert code here by setting top_env_prepend_to_build_phase

  if (!uvm_config_db #(top_config)::get(this, "", "config", m_config))
     `uvm_error(get_type_name(), "Unable to get top_config")

  m_clkndata_config                  = new("m_clkndata_config");
  m_clkndata_config.vif              = m_config.clkndata_vif;
  m_clkndata_config.is_active        = m_config.is_active_clkndata;
  m_clkndata_config.checks_enable    = m_config.checks_enable_clkndata;
  m_clkndata_config.coverage_enable = m_config.coverage_enable_clkndata;

  // You can insert code here by setting agent_copy_config_vars

  uvm_config_db #(clkndata_config)::set(
    this, "m_clkndata_agent", "config", m_clkndata_config);

  if (m_config.is_active_clkndata == UVM_ACTIVE )
    uvm_config_db #(clkndata_config)::set(
      this, "m_clkndata_agent.m_sequencer", "config", m_clkndata_config);
  ...
```

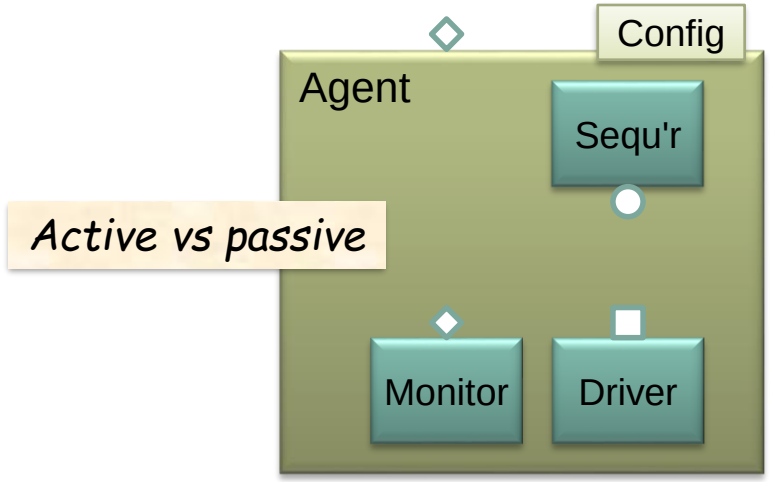
config_db

Agent Class – Build Phase

```
function void clkndata_agent::build_phase(uvm_phase phase);

  if (!uvm_config_db #(clkndata_config)::get(this, "", "config", m_config))
    `uvm_error(get_type_name(), "clkndata config not found")

  m_monitor     = clkndata_monitor    ::type_id::create("m_monitor", this);

  if (get_is_active() == UVM_ACTIVE)
  begin
    m_driver    = clkndata_driver     ::type_id::create("m_driver", this);
    m_sequencer = clkndata_sequencer_t::type_id::create("m_sequencer", this);
  end

endfunction : build_phase
```

config_db

Factory methods

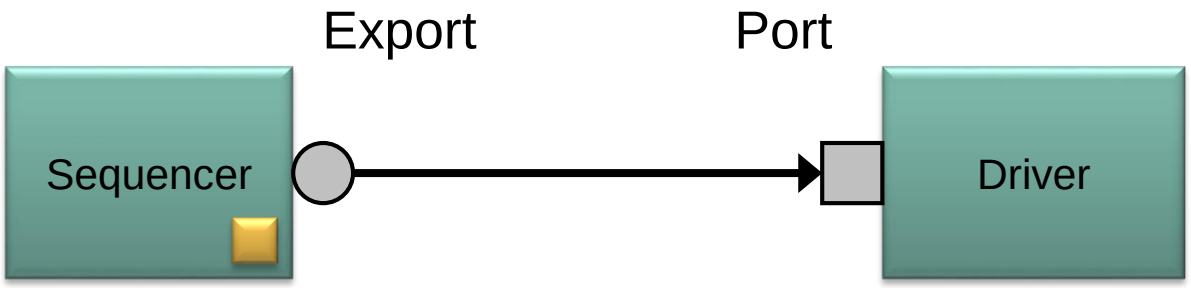


Active vs passive

Easier UVM:
Learning and Using UVM with a Code Generator

- Introduction to UVM
- Easier UVM?
- The Easier UVM Code Generator
- Reporting
- Phases and Configuration
- TLM Connections
- The Factory
- Sequences and Tests

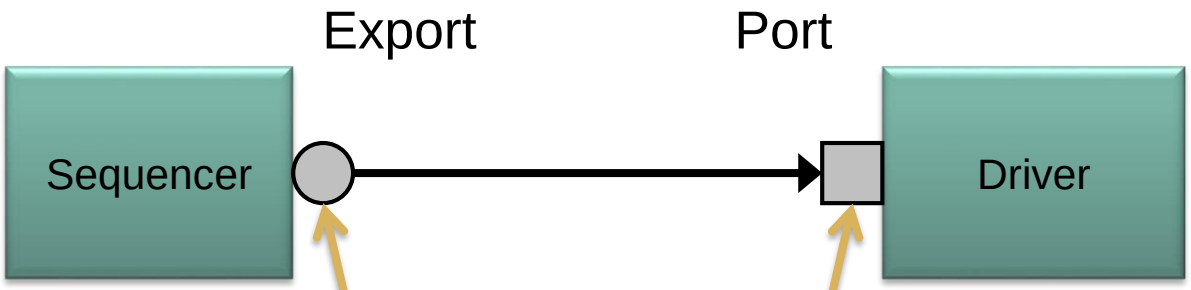
TLM, UVM-Style




```
task get(REQ t);
```

```
seq_item_port.get(req);
```

TLM, UVM-Style



Provides the get()

```
uvm_seq_item_pull_imp #(REQ) seq_item_export;
```

```
uvm_seq_item_pull_port #(REQ) seq_item_port;
```

Requires a get()

```
m_driver.seq_item_port.connect(
                              m_sequencer.seq_item_export);
```

In agent

Driver Class – Run Phase

```
task clkndata_driver::run_phase(uvm_phase phase);
  `uvm_info(get_type_name(), "run_phase", UVM_HIGH)

  forever
  begin
    seq_item_port.get_next_item(req);
    `uvm_info(get_type_name(), {"req item\n",req.sprint}, UVM_HIGH)
    do_drive();
    seq_item_port.item_done();
  end
endtask : run_phase
```

Get transaction from sequencer

Interface template file

```
driver_inc = clkndata_do_drive.sv inline
```

```
// Start of inlined include file generated_tb/tb/include/clkndata_do_drive.sv
task clkndata_driver::do_drive();
  vif.data <= req.data;
  @(posedge vif.clk);
endtask
// End of inlined include file
```

Wiggle pins of DUT

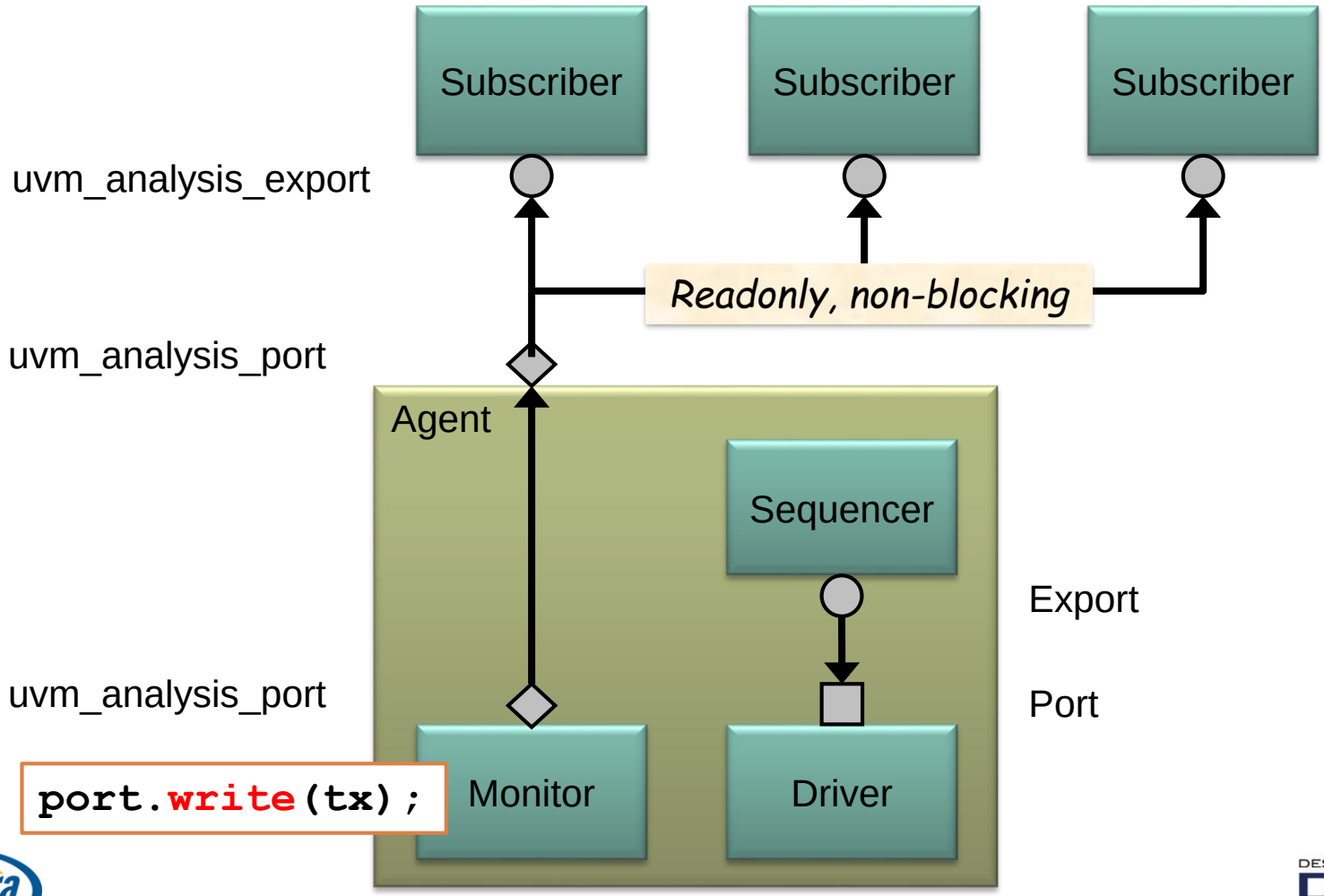
Agent Connect Phase

```
function void clkndata_agent::connect_phase(uvm_phase phase);

  if (m_config.vif == null)
    `uvm_fatal(get_type_name(), "clkndata virtual interface is not set!")

  m_monitor.vif = m_config.vif;
  m_monitor.analysis_port.connect(analysis_port);

  if (get_is_active() == UVM_ACTIVE)
  begin
    m_driver.seq_item_port.connect(m_sequencer.seq_item_export);
    m_driver.vif = m_config.vif;
  end
endfunction : connect_phase
```

TLM connections

Analysis Port

```
function void write(input data_tx t);
   ...
```



Monitor Class

```
class clkndata_monitor extends uvm_monitor;

  `uvm_component_utils(clkndata_monitor)

  virtual clkndata_if vif;

  uvm_analysis_port #(data_tx) analysis_port;

  data_tx m_trans;

  extern function new(string name, uvm_component parent);
  extern task run_phase(uvm_phase phase);
  extern task do_mon();

endclass : clkndata_monitor
```

TLM port

Monitor Class – Run Phase

```
task clkndata_monitor::run_phase(uvm_phase phase);
  `uvm_info(get_type_name(), "run_phase", UVM_HIGH)

  m_trans = data_tx::type_id::create("m_trans");
  do_mon();
endtask : run_phase
```

Factory method

Interface template file

```
monitor_inc = clkndata_do_mon.sv   inline
```

```
// Start of inlined include file generated_tb/tb/include/clkndata_do_mon.sv
task clkndata_monitor::do_mon;
  forever @(posedge vif.clk)
  begin
    m_trans.data = vif.data;
    analysis_port.write(m_trans);
  end
endtask
// End of inlined include file
```

Easier UVM:
Learning and Using UVM with a Code Generator

- Introduction to UVM
- Easier UVM?
- The Easier UVM Code Generator
- Reporting
- Phases and Configuration
- TLM Connections
- The Factory
- Sequences and Tests

The Factory

```
data_tx m_trans;

task clkndata_monitor::run_phase(uvm_phase phase);


  // m_trans = new;
```

Always creates a data_tx

The Factory

```
data_tx m_trans;

task clkndata_monitor::run_phase(uvm_phase phase);

  // m_trans = new;
  m_trans = data_tx::type_id::create("m_trans");
```

Always creates a data_tx

Can be overridden

```
class my_tx extends data_tx;
  ...
```

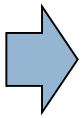
```
agent_factory_set = data_tx my_tx
```

Interface template file

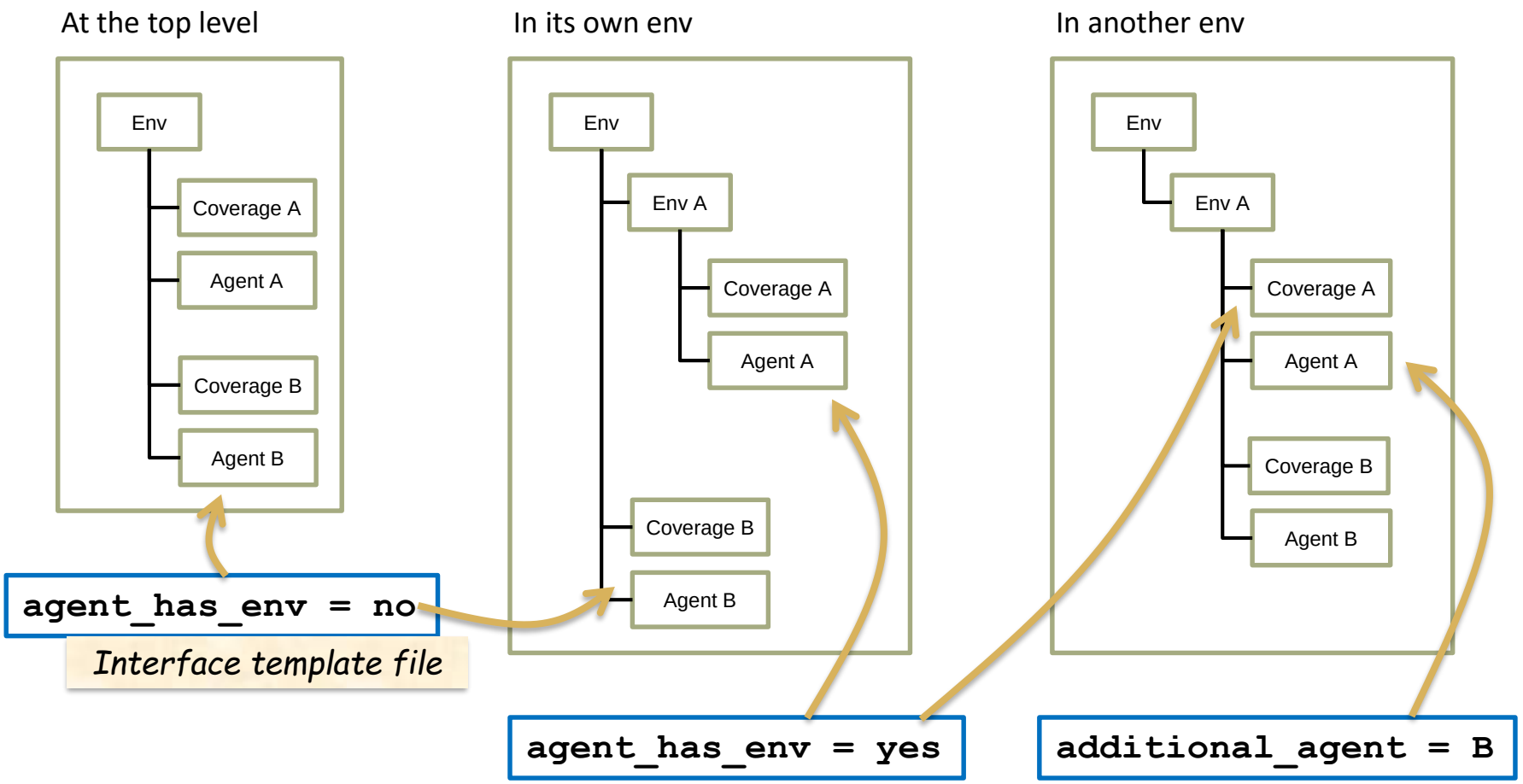
```
data_tx::type_id::set_type_override( my_tx::get_type() );
```

Easier UVM:
Learning and Using UVM with a Code Generator

- Introduction to UVM
- Easier UVM?
- The Easier UVM Code Generator
- Reporting
- Phases and Configuration
- TLM Connections
- The Factory
- Sequences and Tests



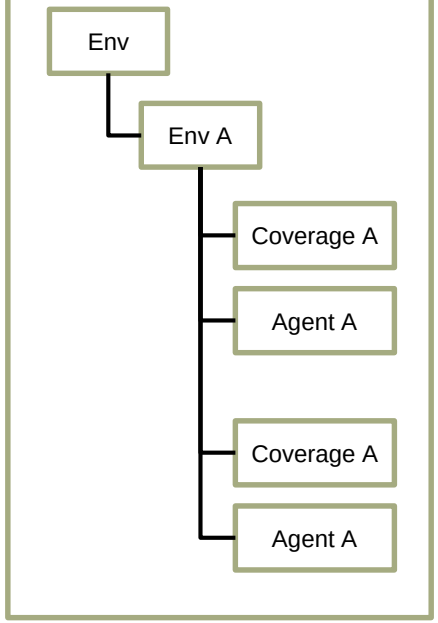
Ways to Instantiate an Agent



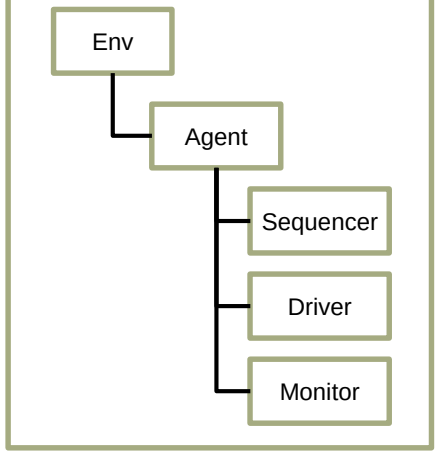
How the Agent is Instantiated

Interface template file

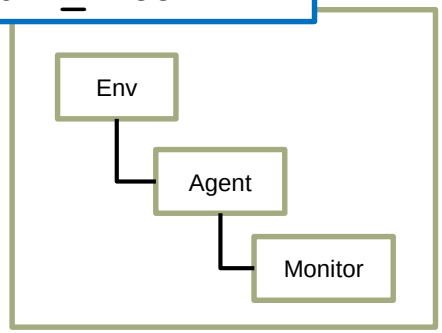
```
number_of_instances = 2
```



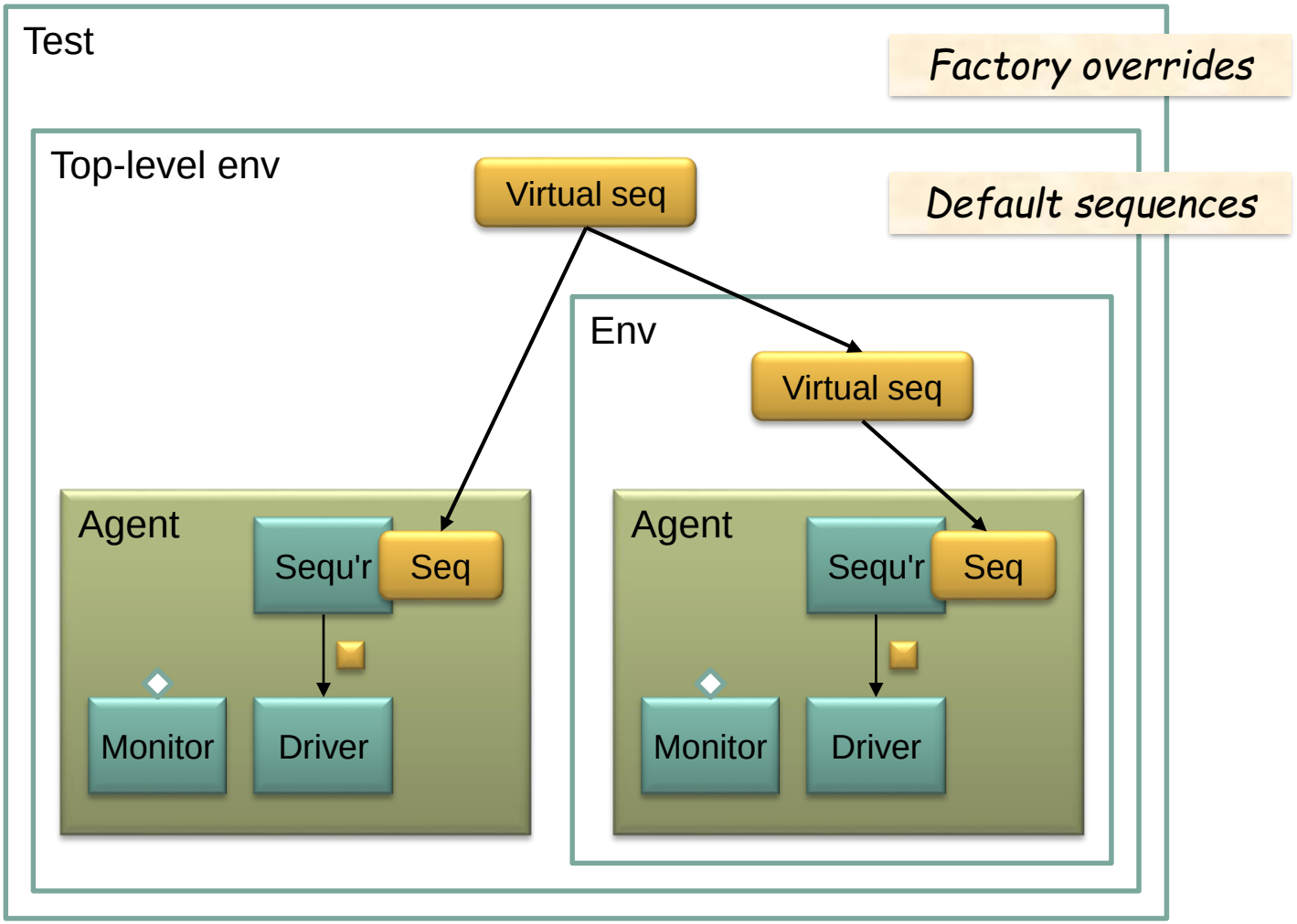

```
agent_is_active = UVM_ACTIVE
```




```
agent_is_active = UVM_PASSIVE
```



Sequence Organization



Test

Interface template file

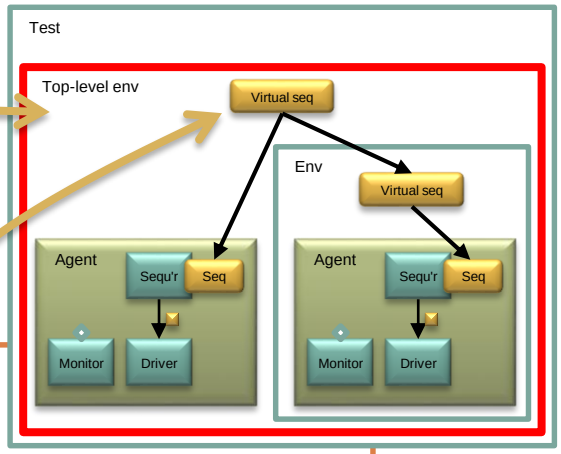
```
test_prepend_to_build_phase = test_prepend_to_build_phase.sv inline
agent_factory_set = clkndata_default_seq  my_clkndata_seq
```

```
function void top_test::build_phase(uvm_phase phase);

  // Start of inlined include file .../include/test_prepend_to_build_phase.sv

  if (!uvm_config_db #(top_config)::get(this, "m_env", "config", m_config))
    `uvm_error(get_type_name(), "Unable to get top_config")
  m_config.coverage_enable = 0;

  // End of inlined include file


  clkndata_default_seq::type_id::set_type_override(
                                        my_clkndata_seq::get_type());


  m_env = top_env::type_id::create("m_env", this);

endfunction : build_phase
```

Modify config

Factory override

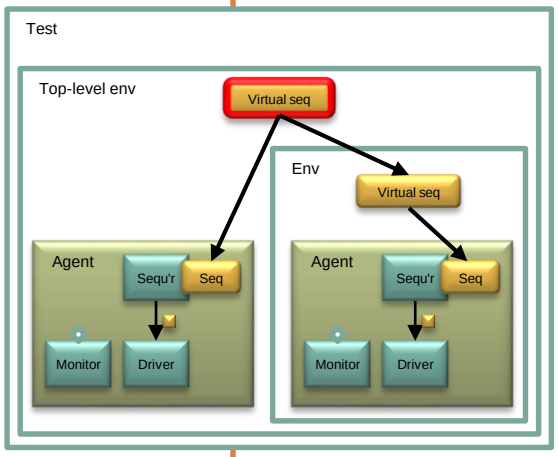
Top-Level Env – Run Phase



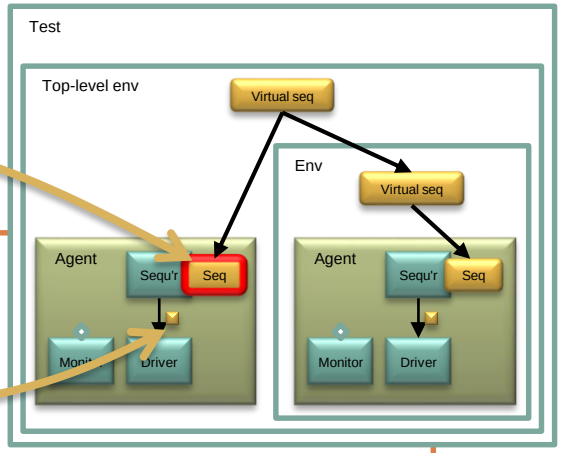

```
task top_env::run_phase(uvm_phase phase);
  top_default_seq vseq;
  vseq = top_default_seq::type_id::create("vseq");
  if ( !vseq.randomize )
    `uvm_fatal(get_type_name(), "Failed to randomize sequence")
  vseq.m_clkndata_agent = m_clkndata_agent;
  vseq.set_starting_phase(phase);
  vseq.start(null);
endtask : run_phases
```

Virtual Sequence

```
class top_default_seq extends uvm_sequence #(uvm_sequence_item);
  `uvm_object_utils(top_default_seq)

  extern function new(string name = "");
  extern task body();
endclass : top_default_seq
...

task top_default_seq::body();
  repeat(m_seq_count)
  begin
    fork
      if (m_clkndata_agent.m_config.is_active == UVM_ACTIVE)
      begin
        clkndata_default_seq seq;
        seq = clkndata_default_seq::type_id::create("seq");
        if ( !seq.randomize() )
          ...
        seq.start(m_clkndata_agent.m_sequencer, this);
      end
      ... // Other agents
    join
  end
endtask : body
```



Agent Sequence




```
task clkndata_default_seq::body();


  req = data_tx::type_id::create("req");
  start_item(req);
  if ( !req.randomize() )
    `uvm_error(get_type_name(), "Failed to randomize transaction")
  finish_item(req);


endtask : body
```

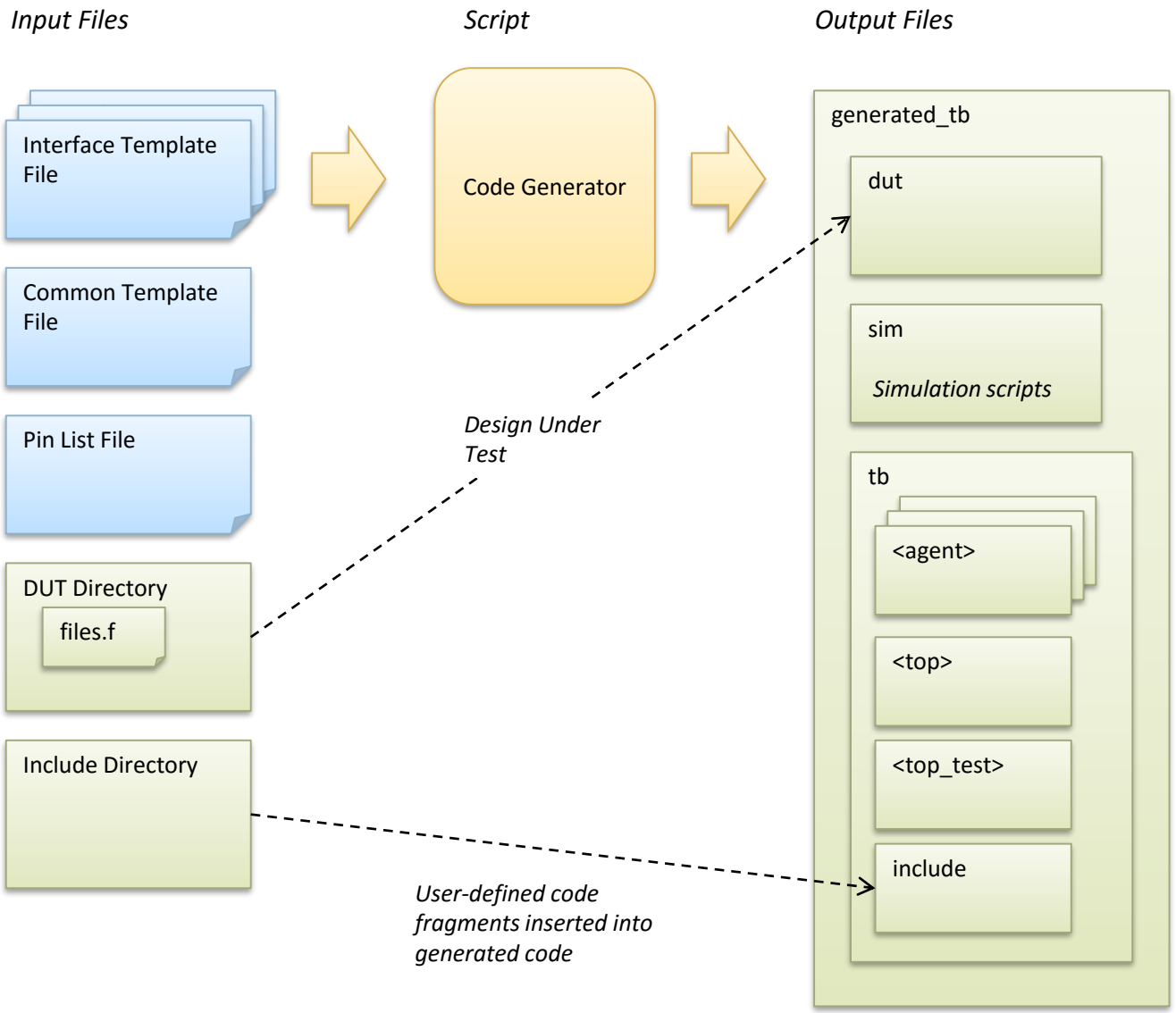
create - start_item - randomize - finish_item

Summary 1

- UVM is a widely used standard
- UVM captures best practice and enables reuse

- UVM is not easy!
- The *Easier UVM Code Generator* can help

Summary 2


Input Files
Script
Output Files
Interface Template File
Common Template File
Pin List File
DUT Directory
files.f
Include Directory
Code Generator
Design Under Test
User-defined code fragments inserted into generated code
generated_tb
dut
sim
Simulation scripts
tb
<agent>
<top>
<top_test>
include

Any Questions?