VERIFICATION ACADEMY

UVM Basics

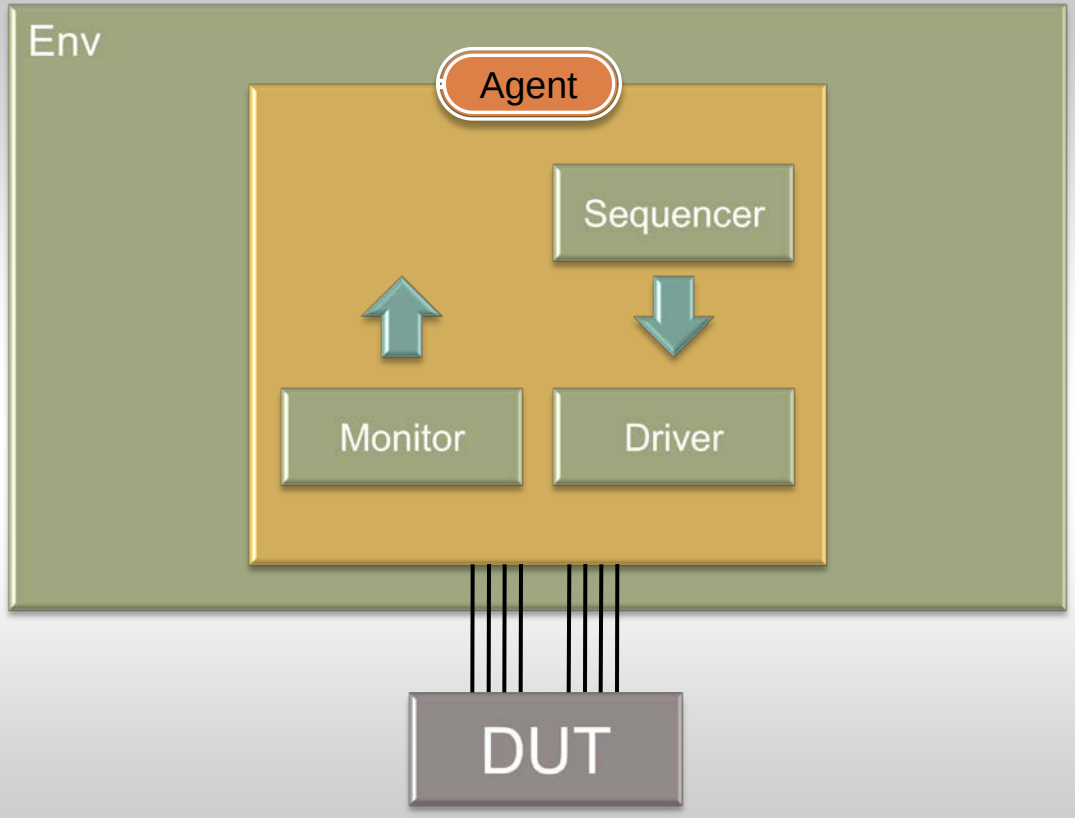
Connecting Components

John Aynsley
CTO, Doulos

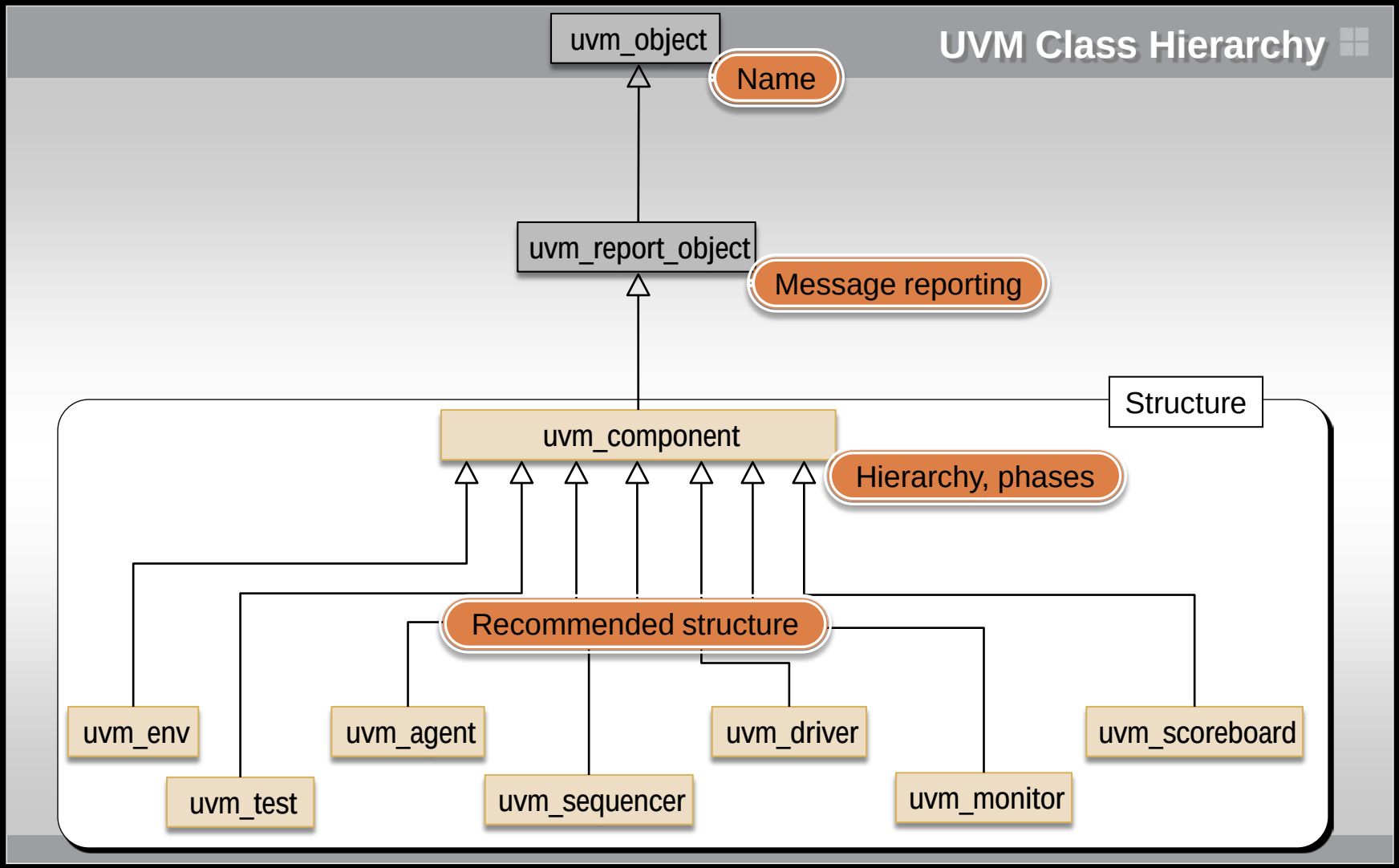
academy@mentor.com
www.verificationacademy.com

Mentor Graphics®

Verification Components



UVM Class Hierarchy



Component

```
class my_agent extends uvm_agent;
```

Component

```
class my_agent extends uvm_agent;
  `uvm_component_utils(my_agent)

  my_sequencer my_sequencer_h;
  my_driver    my_driver_h;
```

Component

```
class my_agent extends uvm_agent;
  `uvm_component_utils(my_agent)

  my_sequencer my_sequencer_h;
  my_driver    my_driver_h;

  function new(string name, uvm_component parent);
    super.new(name, parent);
  endfunction: new

  function void build_phase(uvm_phase phase);
    ...

  function void connect_phase(uvm_phase phase);
    ...
```

Create child components

Connect child components together

Factory Method

```
function void build_phase(uvm_phase phase);
  super.build_phase(phase);
  my_sequencer_h =
    my_sequencer::type_id::create("my_sequencer_h", this);

  my_driver_h    =
    my_driver   ::type_id::create("my_driver_h"   , this);
endfunction: build_phase
```

Instance name

Parent

"Factory methods" can be overridden in tests

Connect

```
function void build_phase(uvm_phase phase);
  super.build_phase(phase);
  my_sequencer_h =
    my_sequencer::type_id::create("my_sequencer_h", this);

  my_driver_h    =
    my_driver   ::type_id::create("my_driver_h"   , this);
endfunction: build_phase

function void connect_phase(uvm_phase phase);
  my_driver_h.seq_item_port.connect(
    my_sequencer_h.seq_item_export );
endfunction: connect_phase
```

Connect

```
function void build_phase(uvm_phase phase);
  super.build_phase(phase);
  my_sequencer_h =
    my_sequencer::type_id::create("my_sequencer_h", this);

  my_driver_h    =
    my_driver   ::type_id::create("my_driver_h"   , this);
endfunction: build_phase

function void connect_phase(uvm_phase phase);
  my_driver_h.seq_item_port.connect(
    my_sequencer_h.seq_item_export );
endfunction: connect_phase
```

Handles used to connect child components

Standard Phase Methods of a Component

```
class my_component extends uvm_component;
  ...
```

Standard Phase Methods of a Component

```
class my_component extends uvm_component;
  ...
  function new(string name, uvm_component parent);
    super.new(name, parent);
  endfunction: new
```

Standard Phase Methods of a Component

```
class my_component extends uvm_component;
  ...
  function new(string name, uvm_component parent);
    super.new(name, parent);
  endfunction: new
  function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    ...
```

Called top-down from type_id::create

Standard Phase Methods of a Component

```
class my_component extends uvm_component;
  ...
  function new(string name, uvm_component parent);
    super.new(name, parent);
  endfunction: new
  function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    ...
  function void connect_phase(uvm_phase phase);
    ...
```

Called bottom-up

Standard Phase Methods of a Component

```
class my_component extends uvm_component;
  ...
  function new(string name, uvm_component parent);
    super.new(name, parent);
  endfunction: new
  function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    ...
  function void connect_phase(uvm_phase phase);
    ...
  function void start_of_simulation;
    ...
```

Standard Phase Methods of a Component

```
class my_component extends uvm_component;
  ...
  function new(string name, uvm_component parent);
    super.new(name, parent);
  endfunction: new
  function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    ...
  function void connect_phase(uvm_phase phase);
    ...
  function void start_of_simulation_phase(uvm_phase phase);
    ...
  task run_phase(uvm_phase phase);
    ...
```

The only task amongst the phases shown here

run_phase can have sub-phases

Standard Phase Methods of a Component

```
class my_component extends uvm_component;
  ...
  function new(string name, uvm_component parent);
    super.new(name, parent);
  endfunction: new
  function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    ...
  function void connect_phase(uvm_phase phase);
    ...
  function void start_of_simulation_phase(uvm_phase phase);
    ...
  task run_phase(uvm_phase phase);
    ...
  function report_phase(uvm_phase phase);
    ...
```

Anatomy of an UVM Component

```
class my_component extends uvm_component;
  `uvm_component_utils(my_component)

  virtual dut_if dut_if_h;

  my_sequencer my_sequencer_h;
  my_driver    my_driver_h;

  function new(string name, uvm_component parent); ...
  function void build_phase(...); ...
  function void connect_phase(...); ...
  function void start_of_simulation_phase(...); ...
  task run_phase(...); ...
  function check_phase(...); ...
  function report_phase(...); ...
endclass
```

Factory registration

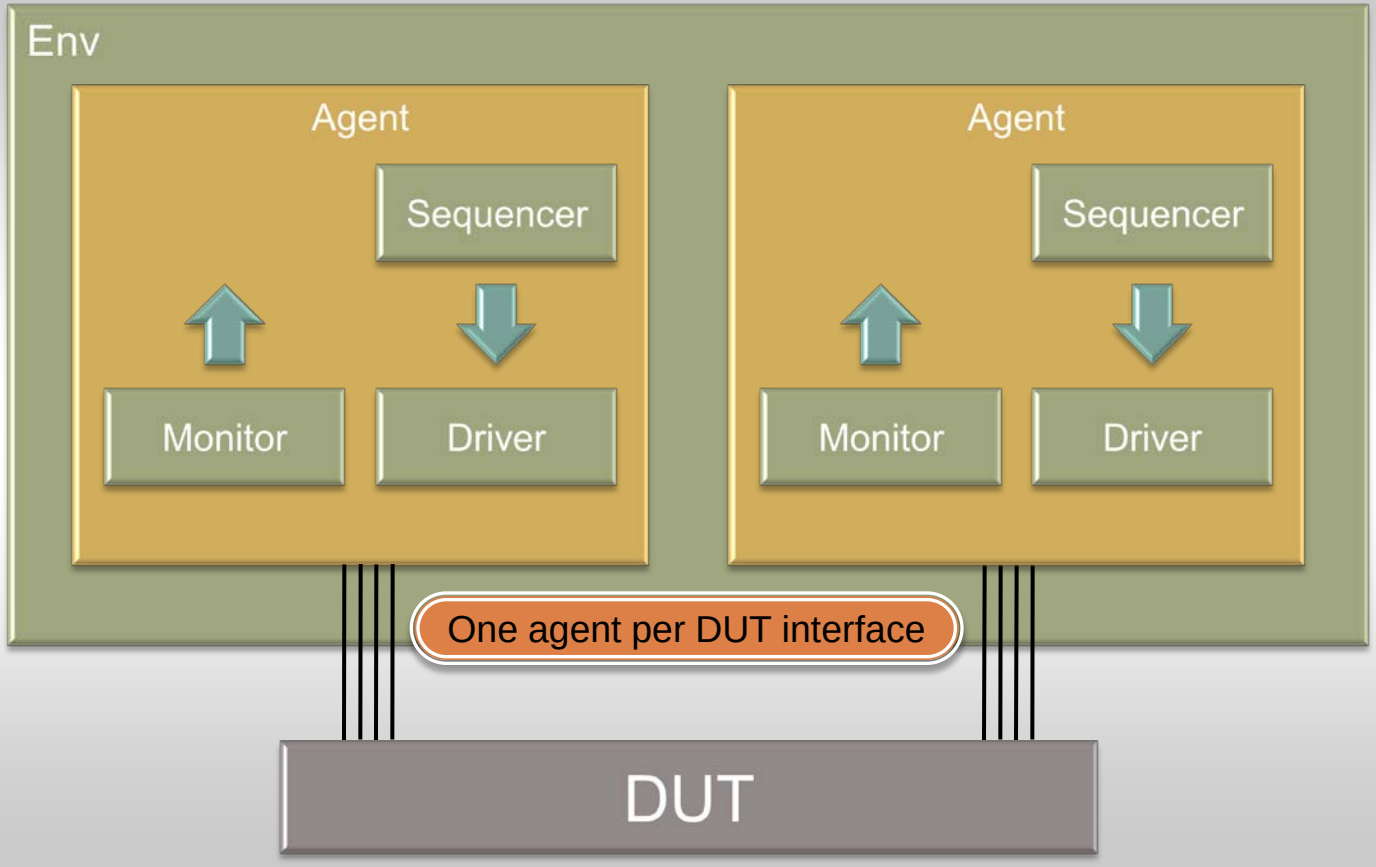
External interfaces, ports

Internal component handles

Standard phase methods

Recommended order

Summary


Env
Agent
Sequencer
Monitor
Driver
Agent
Sequencer
Monitor
Driver
One agent per DUT interface
DUT

VERIFICATION ACADEMY

UVM Basics

Connecting Components

John Aynsley
CTO, Doulos

academy@mentor.com
www.verificationacademy.com

Mentor Graphics®