2015 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

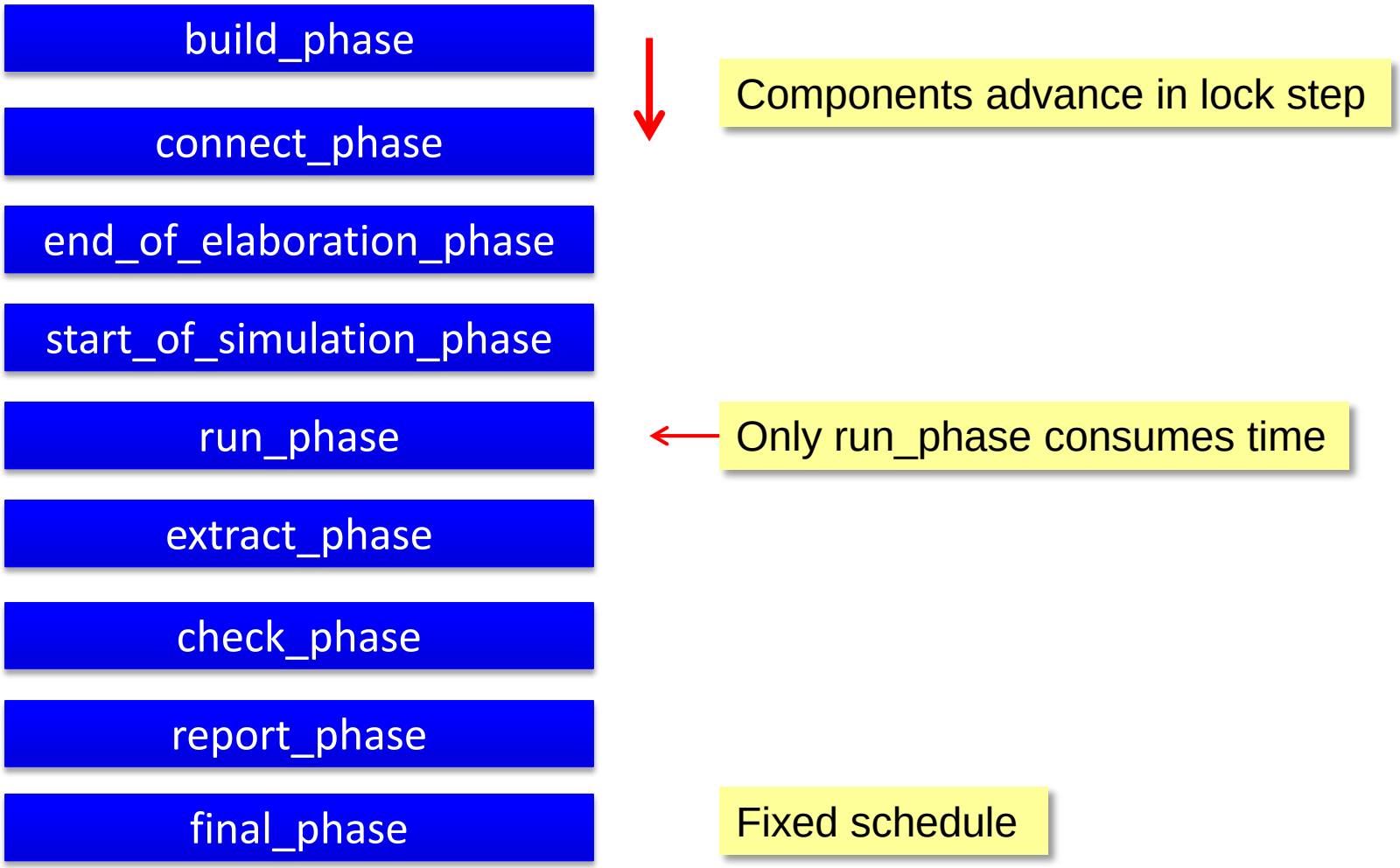
Run-Time Phasing in UVM: Ready for the Big Time or Dead in the Water?

John Aynsley, Doulos

Agenda

- The concepts and jargon of phases
- Phase synchronization
- Domains
- User-defined phases
- VIP integration
- RECOMMENDATIONS

The *Common Phases* of UVM



Phase Methods & Objects

Method

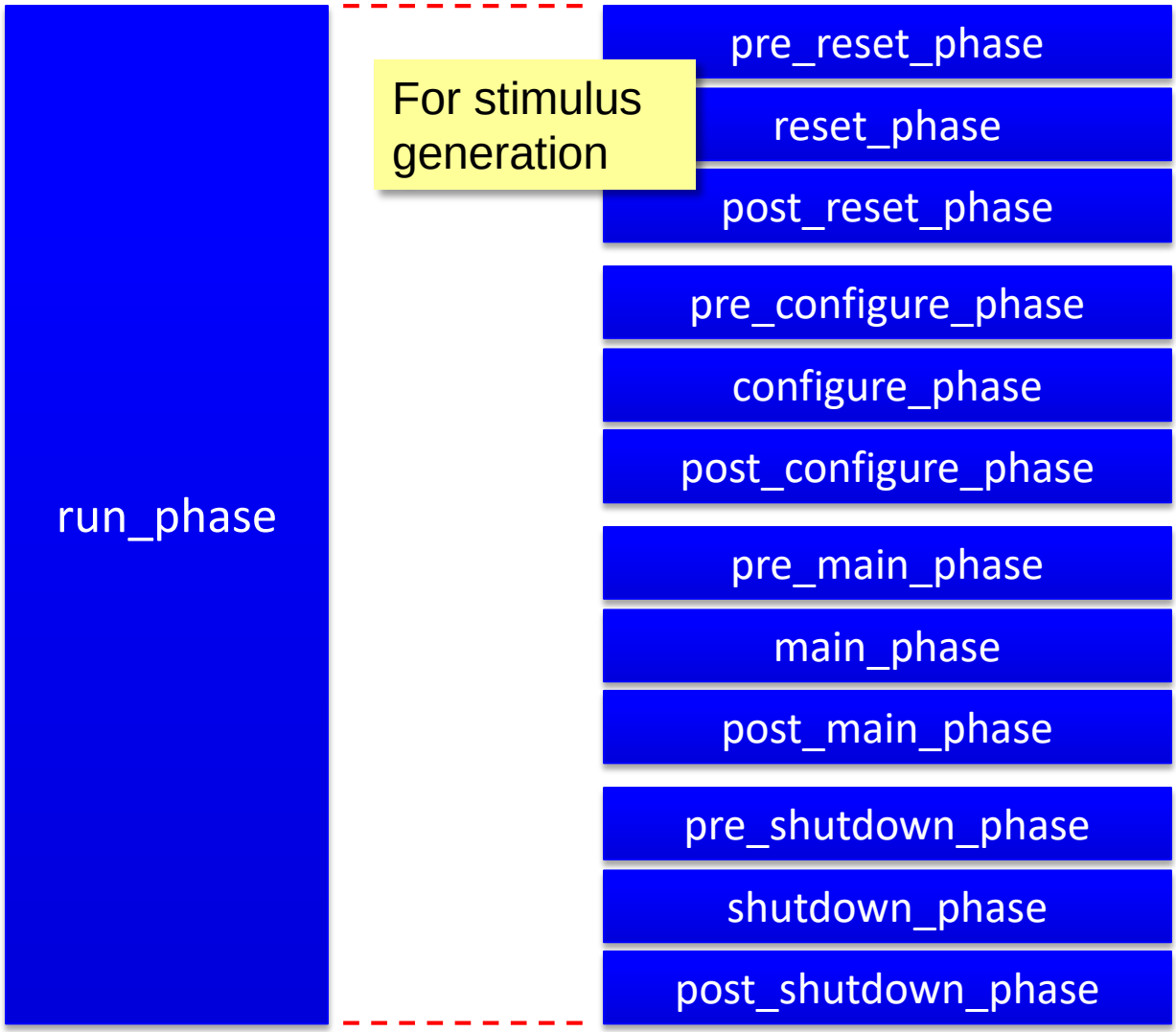
Reference to proxy object

```
function void build_phase(uvm_phase phase);

  assert ( phase.is( uvm_build_phase::get() ) );
  ...
endfunction
```

Class

```
task run_phase(uvm_phase phase);

  phase.raise_objection(this);
  ...
  phase.drop_objection(this);

endtask
```

The *UVM Run-Time Phases*

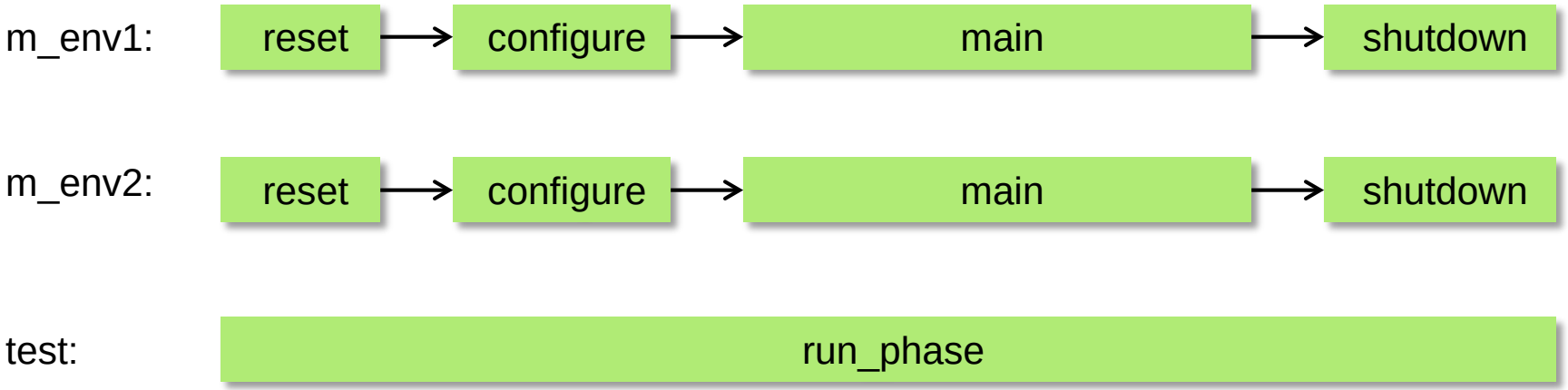


Default Synchronization

```
class env extends uvm_env;
  ...
  task reset_phase(uvm_phase phase);
    ...
  task configure_phase(uvm_phase phase);
    ...
  task main_phase(uvm_phase phase);
    ...
  task shu
    ...
endclass
```

```
class test extends uvm_test;
  ...
  env m_env1;
  env m_env2;
  function void build_phase(uvm_phase phase);
    m_env1 = env::type_id::create("m_env1", this);
    m_env2 = env::type_id::create("m_env2", this);
    ...
  task run_phase(uvm_phase phase);
    ...
```

Default Synchronization



Phase Method Synch

```
task main_phase(uvm_phase phase);
  phase.raise_objection(this);
  ...
  phase.drop_objection(this);
endtask
```

```
task main_phase(uvm_phase phase);
  ...
endtask
```

Both methods called in the same time step

The methods may return at different times

The phase only ends when all objections are dropped

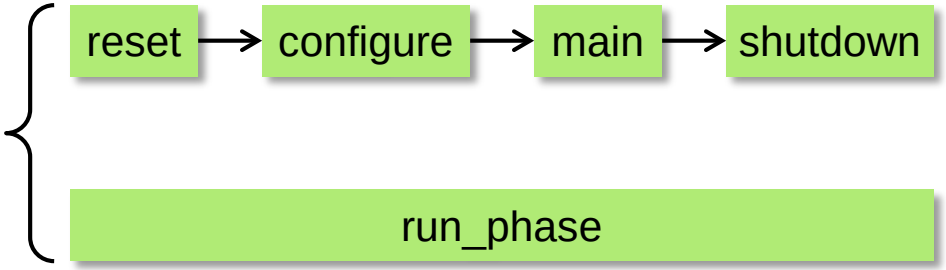
Domains

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

  uvm_domain domain1;

  m_env1 = env::type_id::create("m_env1", this);
  m_env2 = env::type_id::create("m_env2", this);

  domain1 = new("domain1");

  m_env1.set_domain(domain1);

endfunction
```

User-defined domain

Unsynchronized Domains

domain1: reset → configure → main → shutdown



"The UVM domain" contains the "UVM Run-time Phases"

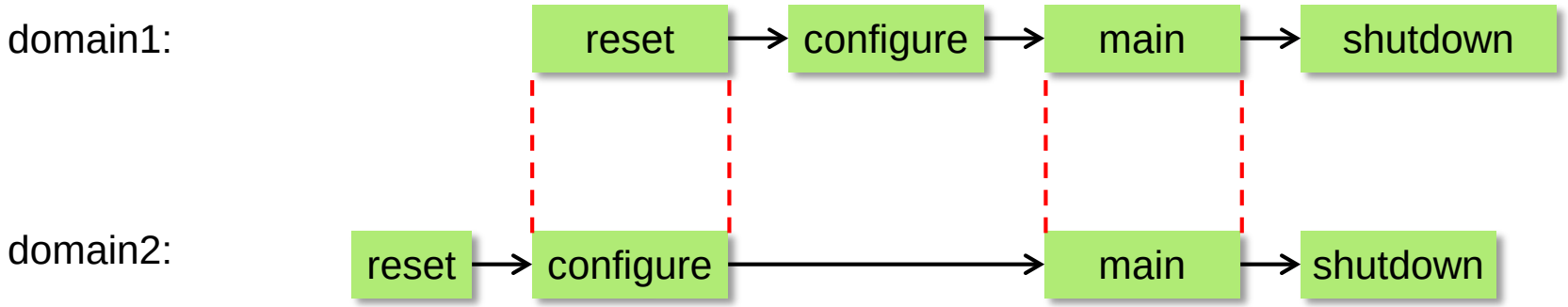
The user-defined domain gets its own copy of those phases

Explicit Synchronization

```
function void test::build_phase(uvm_phase phase);
  uvm_domain domain1, domain2;

  m_env1 = env::type_id::create("m_env1", this);
  m_env2 = env::type_id::create("m_env2", this);

  domain1 = new("domain1");
  m_env1.set_domain(domain1);

  domain2 = new("domain2");
  m_env2.set_domain(domain2);

  domain1.sync(domain2);
  domain1.unsync(domain2);

  domain1.sync(domain2, uvm_main_phase::get());
  domain1.sync(domain2, uvm_reset_phase::get(),
                        uvm_configure_phase::get());
```

For illustration!

Synchronized Phases



User-Defined Phases 1

```
class extended_component extends uvm_component;

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


  virtual task training_phase(uvm_phase phase);
  endtask

endclass
```

Base class

Boilerplate

User-defined phase task

User-Defined Phases 2

```
class my_training_phase extends uvm_task_phase;
  protected function new (string name = "");
    super.new(name);
  endfunction

  static local my_training_phase m_singleton_inst;

  static function my_training_phase get;
    if (m_singleton_inst == null)
      m_singleton_inst = new("my_training_phase");
    return m_singleton_inst;
  endfunction

  task exec_task(uvm_component comp, uvm_phase phase);
    extended_component c;
    if ($cast(c, comp))
      c.training_phase(phase);
  endtask
endclass
```

Phase class

Returns proxy object

Calls the overridden user-defined phase task

User-Defined Phases 3

```
class env extends extended_component;
  ...

  task training_phase(uvm_phase phase);

    phase.raise_objection(this);

    // Consume time

    phase.drop_objection(this);

  endtask

  ...
```

Override the phase task

User-Defined Phases 4

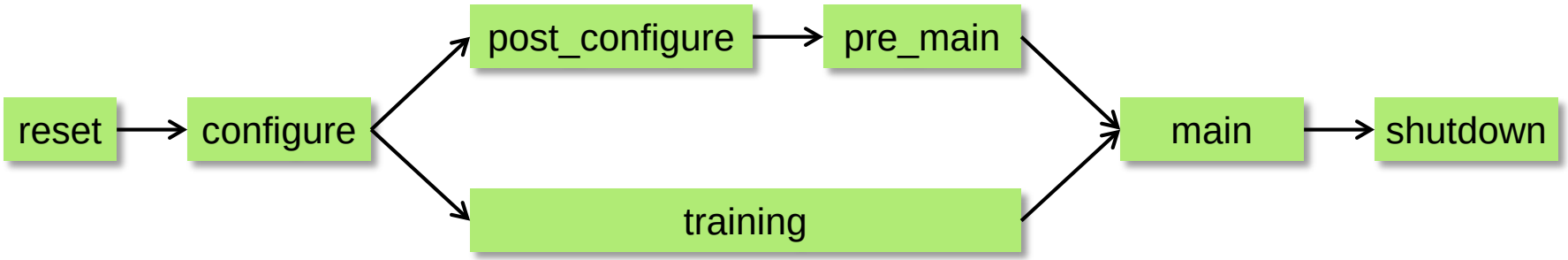
```
function void build_phase(uvm_phase phase);

  uvm_phase schedule;

  m_env1 = env::type_id::create("m_env1", this);


  schedule = uvm_domain::get_uvm_schedule();


  schedule.add(my_training_phase::get(),

               .after_phase(uvm_configure_phase::get()),

               .before_phase(uvm_main_phase::get()));
  ...
```

Pre-defined schedule of “The UVM domain”

Insert the phase into the schedule

Extended Schedule

"The UVM domain"



A Schedule from Scratch

See the paper

- `schedule = new("uvm_sched", UVM_PHASE_SCHEDULE);`
-
```
schedule.add( .before_phase(…),
              .after_phase(…),
              .with_phase(…) );
```

- `function void define_domain(uvm_domain domain);`

Start and End of each Phase

See the paper

- `function void phase_started(uvm_phase phase);`
- `function void phase_ready_to_end(uvm_phase phase);`
- `function void phase_ended(uvm_phase phase);`

Phase Jumping

See the paper

The jump might break the target component. No safeguards, no guarantees.

VIP Integration

- VIP creation
  - Where possible, keep VIP phase-agnostic
  - Provide sequences for reset, configuration, and so forth
- VIP integration - writing envs and tests
  - Where possible, start sequences from the “natural” run-time phase method
- VIP integration - integrating envs and tests
  - Use domains to orchestrate phases across multiple envs

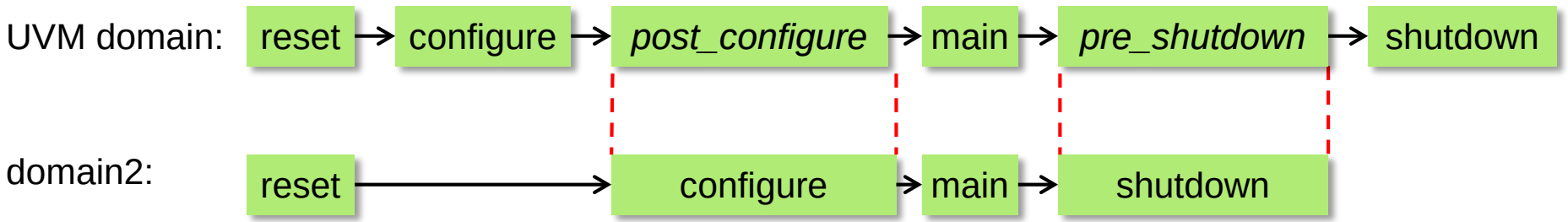
User-Defined Phases

- Do use the reset, configure, main, shutdown phases where “natural”
- Always beware assumptions when integrating VIP
- Do introduce user-defined phases where meaning is non-obvious

- Can sync pre-defined and user-defined phases alike using domain.sync(…)
- But domain.sync(…) cannot specify order of phases
- Reserve the pre-defined pre_ and post_ phases for synchronization

Sync with pre/post phases

```
domain2.sync(uvm_domain::get_uvm_domain(),
                  uvm_configure_phase::get(),
                  uvm_post_configure_phase::get());

domain2.sync(uvm_domain::get_uvm_domain(),
                  uvm_shutdown_phase::get(),
                  uvm_pre_shutdown_phase::get());
```



Let's you order phases across domains!

Recommendations

- Sequences for reset, configuration, and so forth
- Use the pre-defined reset, configure, main, and shutdown phase methods
- Only for stimulus
- Add user-defined phases where pre-defined phases are insufficient
- Use domains and sync phases when integrating VIP
- Only use the pre- and post- phases for synchronization
- Do not use phase jumping casually. There are no built-in safeguards

Downloads

Download a full set of examples from

http://www.doulos.com/downloads/dvcon15/