2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Mentor®
A Siemens Business

UVM IEEE Shiny Object

Rich Edelman

Moses Satyasekaran

Mentor, A Siemens Business

accellera SYSTEMS INITIATIVE

2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Agenda

- UVM Background
- Can I measure the UVM?
- New in UVM 1.2; enhanced in **UVM IEEE**: Core Services
- Replacing the Report Server
- Replacing the Factory
- Replacing the Configuration Database
- Conclusion

accellera SYSTEMS INITIATIVE

Getting the UVM IEEE 1800.2 2017-1.0

UVM 2017-1.0 Now Available

New reference implementation is aligned with IEEE 1800.2

The Accellera Universal Verification Methodology (UVM) Working Group has released the UVM 2017-1.0 reference implementation. UVM 2017-1.0 is aligned with the IEEE 1800.2 standard and the enhancements that make it more powerful and easier to use. The working group has addressed some inconsistencies between the UVM Register Layer and other standards. It received a lot of feedback on the 0.9 release, and they were able to fix the bugs that were reported. UVM 2017-1.0 also includes full documentation of the API that is provided in addition to 1800.2-2017.

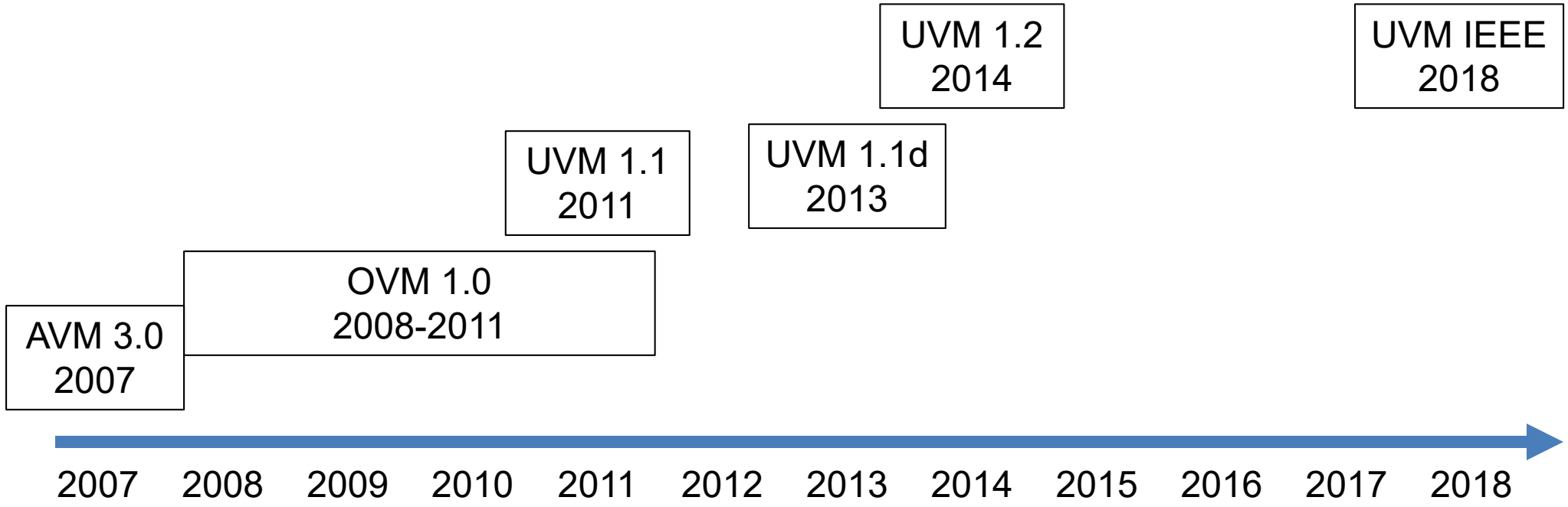
The UVM 2017-1.0 reference implementation can be downloaded for free from Accellera.

The IEEE 1800.2-2017 standard is available free of charge from the IEEE Get program, courtesy of Accellera. Visit the UVM forum to provide feedback, ask questions, and engage in discussions. For more information on UVM, visit the UVM community page.

What is the UVM?

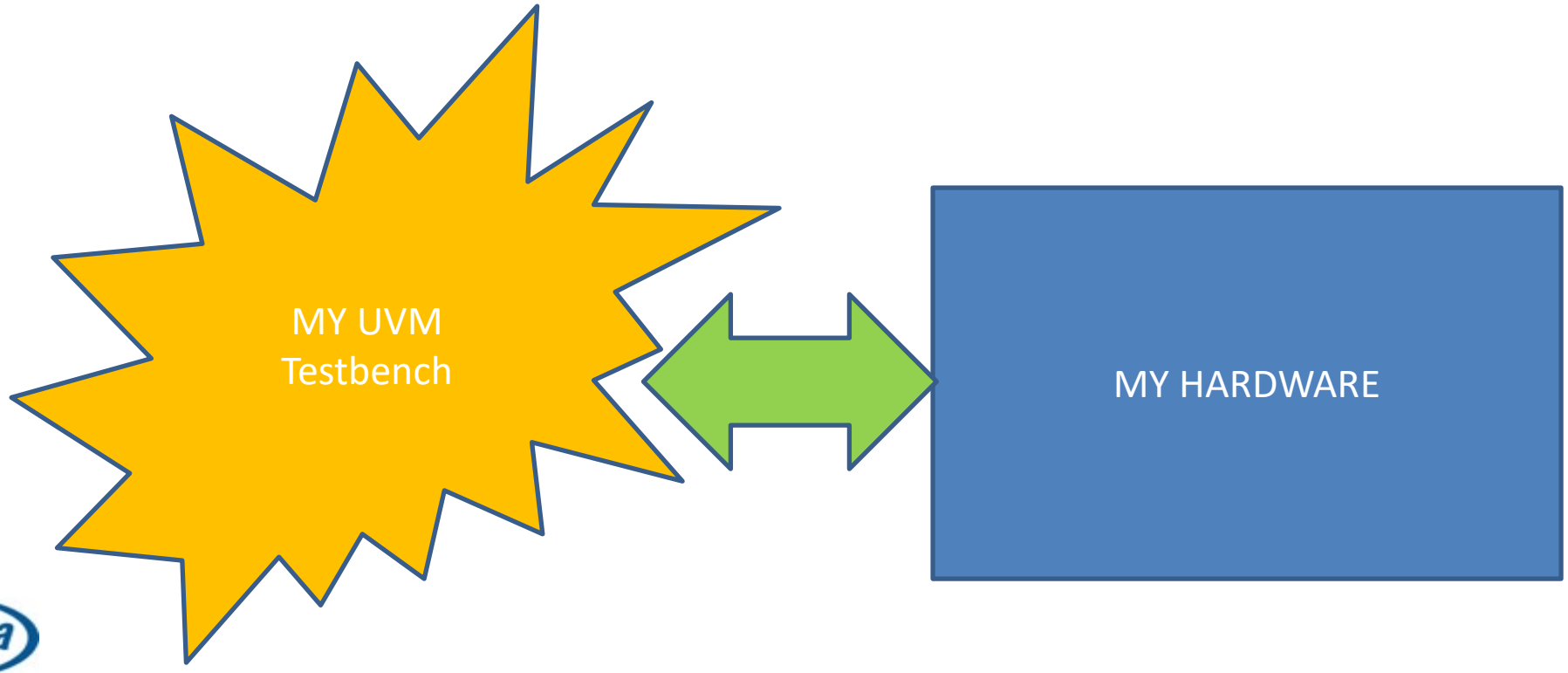
- Universal Verification Methodology
- A library to make SystemVerilog easier to use and to improve verification productivity

What is the UVM History?



Why should I use the UVM?

- Growing base of skilled engineers
- Standard
- Randomization helps explore states with little effort



Which UVM should I use?

- Use the UVM you have always used
  - You prefer it
- Use the UVM that “comes with the simulator”
  - You really don’t care which version
- Use the LATEST release
  - You want all the bug fixes
  - You want all the new features
- Use the one that hasn’t changed in years
  - You prefer stability

What if I want to change versions?

- It depends
- There are backward incompatibilities
- Best to pick a version and stick to it for the whole project

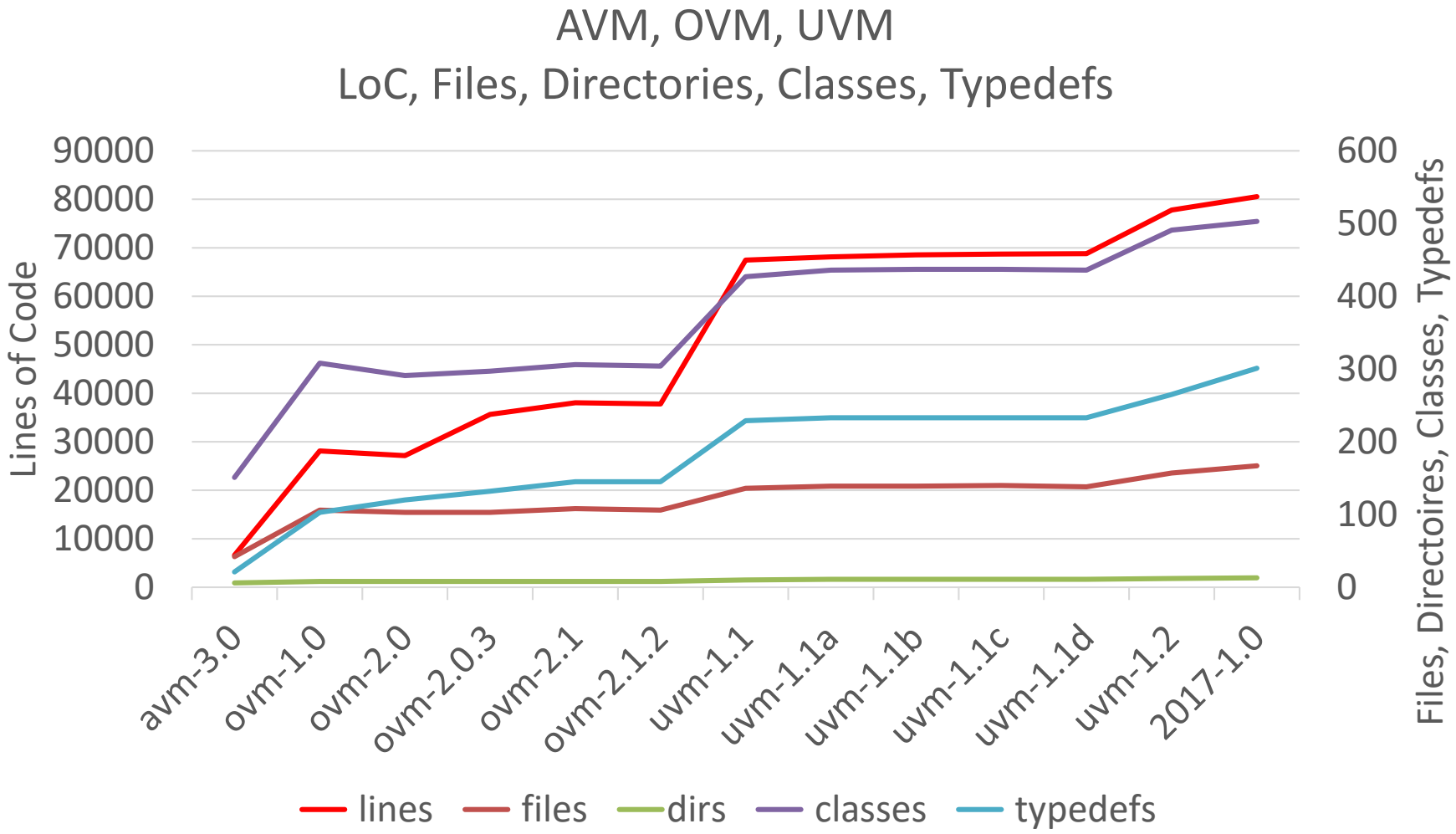
- Can I use two different versions together?
  - Not a good idea

Agenda

- UVM Background
- Can I measure the UVM?
- New in UVM 1.2; enhanced in UVM IEEE: Core Services
- Replacing the Report Server
- Replacing the Factory
- Replacing the Configuration Database
- Conclusion

UVM IEEE - Size

Name	lines	files	class
avm-3.0	6608	42	151
ovm-1.0	28,098	106	308
uvm-1.1d	68,799	138	436
uvm-1.2	77,781	157	491
2017-1.0	80,549	167	503



UVM IEEE - Speed

- Trivial Testcase
  - No RTL
  - Two Envs
    - 10M transactions each
  - 350 lines of code
- Not a real life verification example
- But it is a real example of generating 20M Transactions

UVM Version	Elapsed Time (Wall Clock)
UVM-1.1d	0:39:46
UVM-1.2	0:46:05
UVM-IEEE	0:54:02

Agenda

- UVM Background
- Can I measure the UVM?
- New in UVM 1.2; enhanced in UVM IEEE: Core Services
- Replacing the Report Server
- Replacing the Factory
- Replacing the Configuration Database
- Conclusion

Digging into the Core Services

- Classes introduced in UVM 1.2
- The idea is that there are CORE services which might be implemented in some different or better way.
- Seems like a good idea

```
//
// Class: uvm_coreservice_t

// @uvm-ieee 1800.2-2017 auto F.4.1.1
virtual class uvm_coreservice_t;
  ...
endclass
```

- Timing issues on overrides
  - i.e. Factory before time 0
- Some compatibility worries
  - Different vendors with different implementations?

What are the core services in the UVM?

- Factory
  - Allow dynamic changes to class construction
- Configuration
  - Allow settings to be set in one place and looked up in another place (set/get)
- Report server
  - Central place for standard messages
  - Verbosity and format control

*But the uvm_coreservice does NOT support replacing the **configuration** database*

Creating new factory and report server

- How we create new functionality in OOP

```
class my_factory extends uvm_default_factory;
  function void print (int all_types=1);
    ...
  endfunction
endclass
```

```
class my_report_server_t extends uvm_default_report_server;
  ...
endclass
```

```
my_coreservice_t   my_coreservice;
my_report_server_t my_report_server;
my_factory_t       my_factory;

my_coreservice = new();
uvm_init(my_coreservice);

my_report_server = new("my_report_server");
my_coreservice.set_report_server(my_report_server);

my_factory = new();
my_coreservice.set_factory(my_factory);
```

2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Agenda

- UVM Background
- Can I measure the UVM?
- New in UVM 1.2; enhanced in UVM IEEE: Core Services
- Replacing the Report Server
- Replacing the Factory
- Replacing the Configuration Database
- Conclusion

accellera SYSTEMS INITIATIVE

The Report Server

- What the report server does
  - Standard formatting
  - Access to the “report message structure”
- Why do you need a new report server?

```
case(report_message.get_verbosity())
  UVM_NONE:   verbosity = "NONE";
  UVM_LOW:    verbosity = "LOW";
  UVM_MEDIUM: verbosity = "MEDIUM";
  UVM_HIGH:   verbosity = "HIGH";
  UVM_FULL:   verbosity = "FULL";
  UVM_DEBUG:  verbosity = "DEBUG";
endcase
```

```
$display("MY FANCY REPORT SERVER: severity=%s fullname=%s, id=%0s, MSG=%s, \
   filename=%s, line=%0d, context=%s, action=%s, verbosity=%s, object=%p",
   report_message.get_severity().name(),
   report_message.get_report_handler().get_full_name(),
   report_message.get_id(),
   report_message.get_message(),
   report_message.get_filename(),
   report_message.get_line(),
   report_message.get_context(),
   uvm_report_handler::format_action(report_message.get_action()),
   verbosity,
   report_message.get_report_object());
```

A Message Gets Generated

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

```
$display("MY FANCY REPORT SERVER: severity=%s fullname=%s, id=%0s, MSG=%s, \
   filename=%s, line=%0d, context=%s, action=%s, verbosity=%s, object=%p",
   report_message.get_severity().name(),
   report_message.get_report_handler().get_full_name(),
   report_message.get_id(),
   report_message.get_message(),
   report_message.get_filename(),
   report_message.get_line(),
   report_message.get_context(),
   uvm_report_handler::format_action(report_message.get_action()),
   verbosity,
   report_message.get_report_object());
```

```
# MY FANCY REPORT SERVER: severity=UVM_INFO fullname=uvm_test_top.a1.sqr, id=my_sequence,
MSG=Finished, filename=t.sv, line=91, context=seq1-3, action=DISPLAY , verbosity=MEDIUM, object='
```

2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Agenda

- UVM Background
- Can I measure the UVM?
- New in UVM 1.2; enhanced in UVM IEEE: Core Services
- Replacing the Report Server
- Replacing the Factory
- Replacing the Configuration Database
- Conclusion

accellera SYSTEMS INITIATIVE

The Factory

- Registering with the factory
  - I have a class named “ABC”.
  - I register it with the factory.

```
`uvm_component_utils(ABC)
```

- Constructing a class of type ABC

```
ABC c1;
c1 = ABC::type_id::create(“c1”)
```

- Overriding ABC

```
ABC::type_id::set_type_override(newABC::get_type(), 1);
```

- Why do you need a **new** factory?

```
class ABC extends uvm_component;
  `uvm_component_utils(ABC)
  ...
endclass
```

```
class newABC extends ABC;
 `uvm_component_utils(newABC)
  ...
endclass
```

What can go wrong with the factory?

- Used ‘new’ directly, instead of ‘create’
- Wrong override
  - Instance
  - By Type

```
c1 = ABC::type_id::create("c1")

ABC::type_id::set_type_override(newABC::get_type(), 1);
```

Debugging the factory

- Print a message for a registration
  - Registering class ABC
  - Registering class newABC

Fancy Print

- Print a message for an override
  - newABC overrides ABC

- Print a message when a class is produced by the factory
  - Requested class ABC, creating class newABC

Agenda

- UVM Background
- Can I measure the UVM?
- New in UVM 1.2; enhanced in UVM IEEE: Core Services
- Replacing the Report Server
- Replacing the Factory
- Replacing the Configuration Database
- Conclusion

The Configuration Database

- Putting information into the Configuration Database

```
uvm_config_db#(int)::set(this, "*",      "simple_int", 12);
uvm_config_db#(int)::set(this, "a*.d",   "simple_int", 13);
uvm_config_db#(int)::set(this, "a*.sqr", "simple_int", 14);
```

- Getting information from a Configuration database

```
int simple_int;

function void build_phase(uvm_phase phase);
  if (!uvm_config_db#(int)::get(this, "", "simple_int", simple_int))
    `uvm_info(get_type_name(), "GET CONFIG FAILED", UVM_MEDIUM)
...
```

- Why do you need a new configuration database?

Config DB: What can go wrong?

- Putting information into the Configuration Database

```
uvm_config_db#(int)::set(this, "*",      "simple_int", 12);
uvm_config_db#(int)::set(this, "a*.d",   "simple_int", 13);
uvm_config_db#(int)::set(this, "a*.sqr", "simple_int", 14);
```

- Getting information from a Configuration database

```
int simple_int;

function void build_phase(uvm_phase phase);
  if (!uvm_config_db#(int)::get(this, "", "simple_int", simple_int))
    `uvm_info(get_type_name(), "GET CONFIG FAILED", UVM_MEDIUM)
...
```

Debugging the Configuration Database

- Using the Command Line Switches

  `+UVM_RESOURCE_DB_TRACE   +UVM_CONFIG_DB_TRACE`

- Generates large quantities of information

```
# UVM_INFO my_1800.2-2017-1.0/src/base/uvm_resource_db.svh(161) @ 0: reporter [CFGDB/SET] CFG scope='uvm_test_top.*' name='simple_int' (type int) set accessor=uvm_test_top = (int) 12
# UVM_INFO my_1800.2-2017-1.0/src/base/uvm_resource_db.svh(161) @ 0: reporter [CFGDB/SET] CFG scope='uvm_test_top.a*.d' name='simple_int' (type int) set accessor=uvm_test_top = (int) 13
# UVM_INFO my_1800.2-2017-1.0/src/base/uvm_resource_db.svh(161) @ 0: reporter [CFGDB/SET] CFG scope='uvm_test_top.a*.sqr' name='simple_int' (type int) set accessor=uvm_test_top = (int) 14
# UVM_INFO my_1800.2-2017-1.0/src/base/uvm_resource_db.svh(161) @ 0: reporter [CFGDB/GET] CFG scope='uvm_test_top.a1.d' name='simple_int' (type int) read accessor=uvm_test_top.a1.d = (int) 13
# UVM_INFO my_1800.2-2017-1.0/src/base/uvm_resource_db.svh(161) @ 0: reporter [CFGDB/GET] CFG scope='uvm_test_top.a2.d' name='simple_int' (type int) read accessor=uvm_test_top.a2.d = (int) 13
# UVM_INFO my_1800.2-2017-1.0/src/base/uvm_resource_db.svh(161) @ 0: reporter [CFGDB/GET] CFG scope='uvm_test_top.a2.sqr' name='simple_int' (type int) read accessor=uvm_test_top.a2.sqr = (int) 14
# UVM_INFO my_1800.2-2017-1.0/src/base/uvm_resource_db.svh(161) @ 0: reporter [CFGDB/GET] CFG scope='uvm_test_top.a1.sqr' name='simple_int' (type int) read accessor=uvm_test_top.a1.sqr = (int) 14
# UVM_INFO my_1800.2-2017-1.0/src/base/uvm_resource_db.svh(161) @ 0: reporter [CFGDB/GET] CFG scope='uvm_test_top.a2.sqr' name='simple_int' (type int) read accessor=uvm_test_top.a2.sqr = (int) 14
# UVM_INFO my_1800.2-2017-1.0/src/base/uvm_resource_db.svh(161) @ 0: reporter [CFGDB/GET] CFG scope='uvm_test_top.a1.sqr' name='simple_int' (type int) read accessor=uvm_test_top.a1.sqr = (int) 14
# UVM_INFO my_1800.2-2017-1.0/src/base/uvm_resource_db.svh(161) @ 0: reporter [CFGDB/GET] CFG scope='uvm_test_top.a2.sqr' name='simple_int' (type int) read accessor=uvm_test_top.a2.sqr = (int) 14
# UVM_INFO my_1800.2-2017-1.0/src/base/uvm_resource_db.svh(161) @ 0: reporter [CFGDB/GET] CFG scope='uvm_test_top.a1.sqr' name='simple_int' (type int) read accessor=uvm_test_top.a1.sqr = (int) 14
# UVM_INFO my_1800.2-2017-1.0/src/base/uvm_resource_db.svh(161) @ 0: reporter [CFGDB/GET] CFG scope='uvm_test_top.a2.sqr' name='simple_int' (type int) read accessor=uvm_test_top.a2.sqr = (int) 14
# UVM_INFO my_1800.2-2017-1.0/src/base/uvm_resource_db.svh(161) @ 0: reporter [CFGDB/GET] CFG scope='uvm_test_top.a1.sqr' name='simple_int' (type int) read accessor=uvm_test_top.a1.sqr = (int) 14
```

- Something better

uvm_resource::lookup_name(): Context

- Update lookup_name() to provide
  - Calling context – a class
  - File name
  - Line number

```
function uvm_resource_types::rsrc_q_t lookup_name(string scope = "",
                                                  string name,
                                                  uvm_resource_base type_handle = null,
                                                  bit rpterr = 1,
                                                  input uvm_object CALLING_CONTEXT = null,
                                                  input string FILE = "",
                                                  input int    LINE = 0
                                                  );
```

uvm_resource::lookup_name(): Info

- Annotate the decisions
  - Success
  - Failure
- Share the information for better debug
- "Didn't match"
  - What were they?

Types don't match

Scopes don't match

```
uvm_resource_types::rsrc_q_t rq;
uvm_resource_types::rsrc_q_t q;
uvm_resource_base rsrc, r;
string rsrcs;

if(name == "") return q;

if(!rtab.exists(name)) begin
    if(rpterr) void'(spell_check(name));
    return q;
end

rsrc = null;
rq = rtab[name];
for(int i=0; i<rq.size(); ++i) begin
  r = rq.get(i);
  rsrcs = ri_tab.exists(r) ? ri_tab[r].scope: "";
  if(((type_handle == null) ||
      (r.get_type_handle() == type_handle)) &&
      uvm_is_match(rsrcs, scope))
    q.push_back(r);
end
return q;
```

It doesn't exist

Agenda

- UVM Background
- Can I measure the UVM?
- New in UVM 1.2; enhanced in UVM IEEE: Core Services
- Replacing the Report Server
- Replacing the Factory
- Replacing the Configuration Database
- Conclusion

Where are we now…

- There are new features in UVM IEEE
- There are bug fixes in UVM IEEE

- Are there NEW bugs in UVM IEEE?
- Should I move to UVM IEEE?

Industry Bug Rate:
15-50 bugs per 1000 lines of code

Name	lines	# bugs @ 10 bugs per 1000 lines
avm-3.0	6608	66
ovm-1.0	28,098	280
uvm-1.1d	68,799	687
uvm-1.2	77,781	777
2017-1.0	80,549	805

Summary

- Use the UVM
  - Productivity
  - Standard
  - Common skill set for new hires
- Do NOT use “ALL” the features of the UVM
  - Hard to debug
  - Hard to understand
  - Keep things simple
  - See the authors for things to stay away from

Get on the UVM Planning and Development team