VUnit

An open source unit testing framework for VHDL

Lars Asplund, Synective Labs
lars.asplund@synective.se

https://github.com/LarsAsplund/vunit

Best viewed on YouTube

Synective Labs

Control Theory – Self-Driving Car

Uncertainty in constant k

Tire Pressure?

Slippery surface?

Target destination

Controller

Power (P)
Direction (θ)

System/Process

$w = k * P$

$v = g * w * r$

$\vec{d} = \vec{D} - \int \vec{v}\, dt$

$\vec{v} = v * f(theta)$

Route

Sensor

Imperfect steering?

Noise

Noise

Noise

Best viewed on YouTube

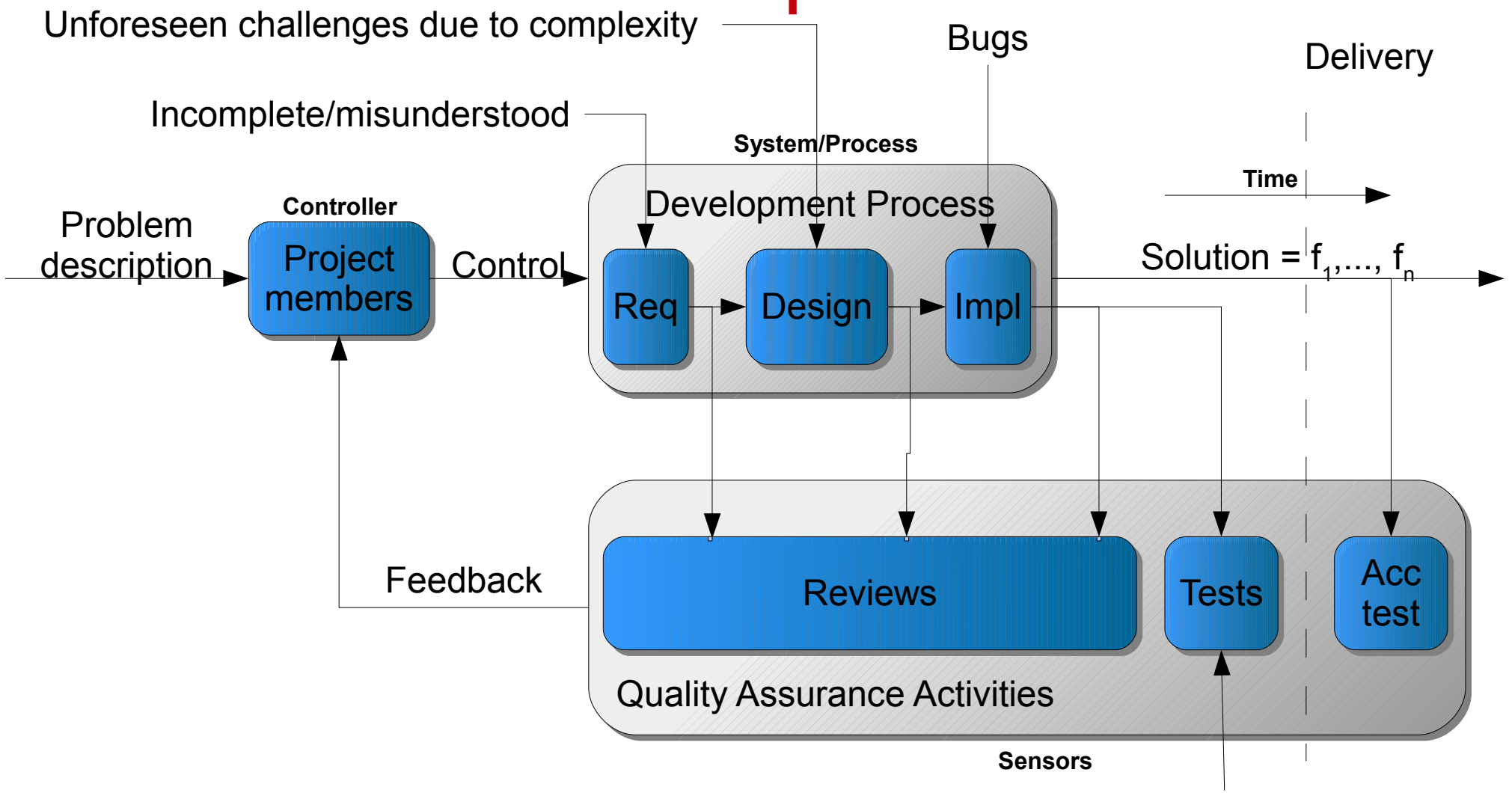
Synective Labs

Simulation with Feedback

Best viewed on YouTube

Synective Labs

Control Theory – VHDL Development



Best viewed on YouTube

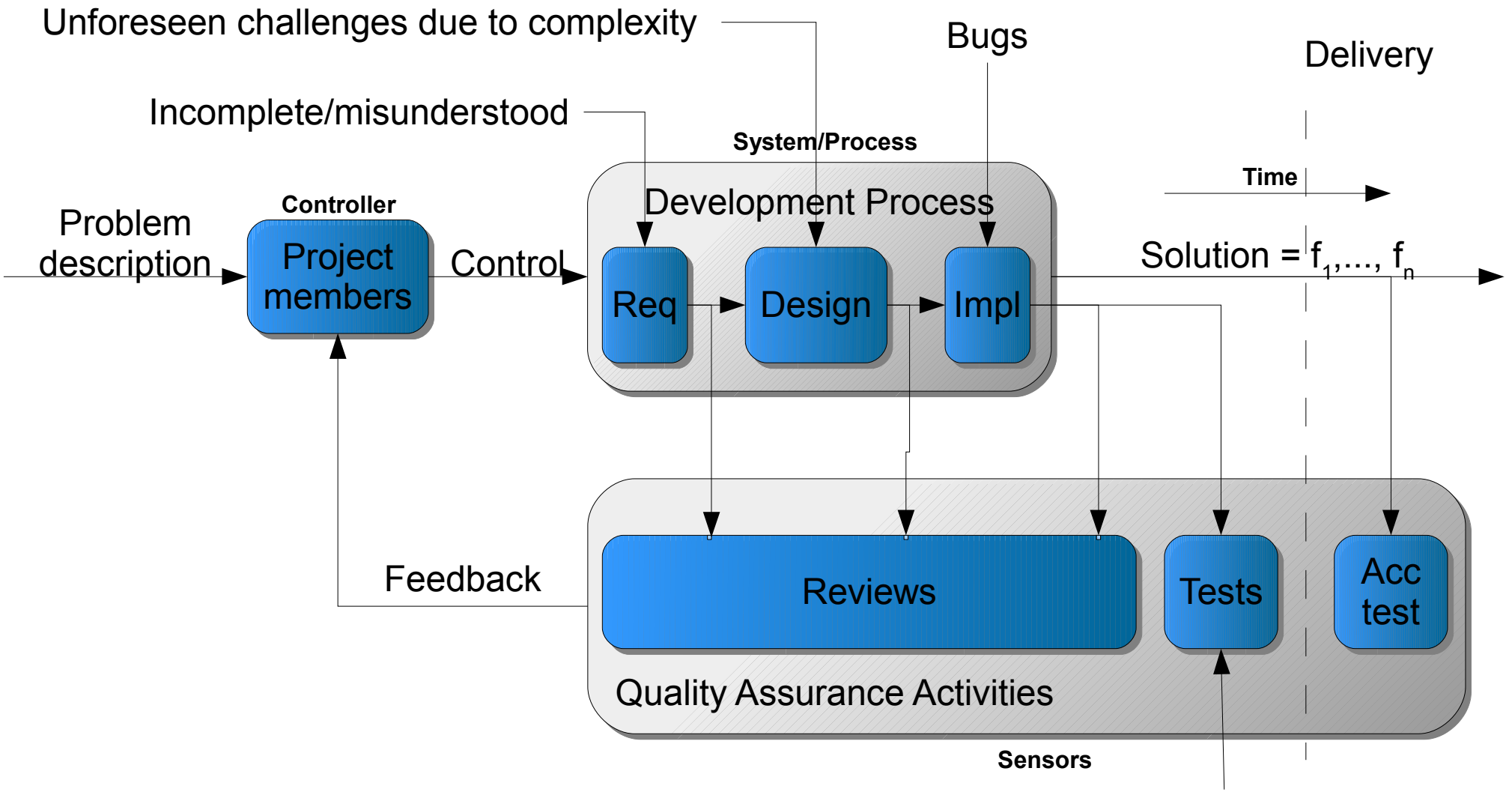
Synective Labs

Infrequent And Delayed Feedback

Best viewed on YouTube

Synective Labs

Process Feedback



Best viewed on YouTube

Synective Labs

Research Support

- Research indicates higher success rates with more iterative development[1]

	More iterative	More waterfall
Successful	70%	50%
Challenged	20%	30%
Failed	10%	20%

- Research indicates that small iterations are to prefer[2]

  "Research also indicates that smaller time frames, with delivery of software components early and often, will increase the success rate."

Best viewed on YouTube

Synective Labs

Checking

- Automation starts with self-checking testbenches
- VUnit works with VHDL asserts but provides a check package for extended functionality.
- A simple sequential check equivalent to a VHDL assert:

```
check(num_overflows = 0, "Overflow!");
```

Best viewed on YouTube

Synective Labs

Concurrent checks

```
entity uart_rx is
  port (
    ...
    -- AXI stream for output bytes
    tready : in std_logic;
    tvalid : out std_logic;
    tdata : out std_logic_vector(7 downto 0));
begin
  check_stable(clk, check_enabled, tvalid, tready, tvalid,
               "tvalid must be active until tready is active");
  check_not_unknown(clk, check_enabled, tvalid,
                    "tvalid must never be unknown");
end entity;
```

Best viewed on YouTube

Synective Labs

Checker Package

Checker Type	Checker Type
check	check_failed
check_true	check_passed
check_false	(check_match)
check_equal	(check_expect)
check_relation	
check_implication	
check_stable	
check_not_unknown	
check_zero_one_hot	
check_one_hot	
check_next	
check_sequence	

- Report errors using the VUnit logging package

Best viewed on YouTube

Synective Labs

Logging

```
entity uart_rx is
  port (
    ...
    -- AXI stream for output bytes
    tready : in std_logic;
    tvalid : out std_logic := '0';
    tdata : out std_logic_vector(7 downto 0));
begin
  -- pragma translate_off
  traffic_logger: process (clk) is
  begin
    if tvalid = '1' and tready = '1' and rising_edge(clk) then
      debug("Received " & to_string(to_integer(unsigned(tdata))));
    end if;
  end process traffic_logger;
  -- pragma translate_on
end entity;
```

- Resulting log entry

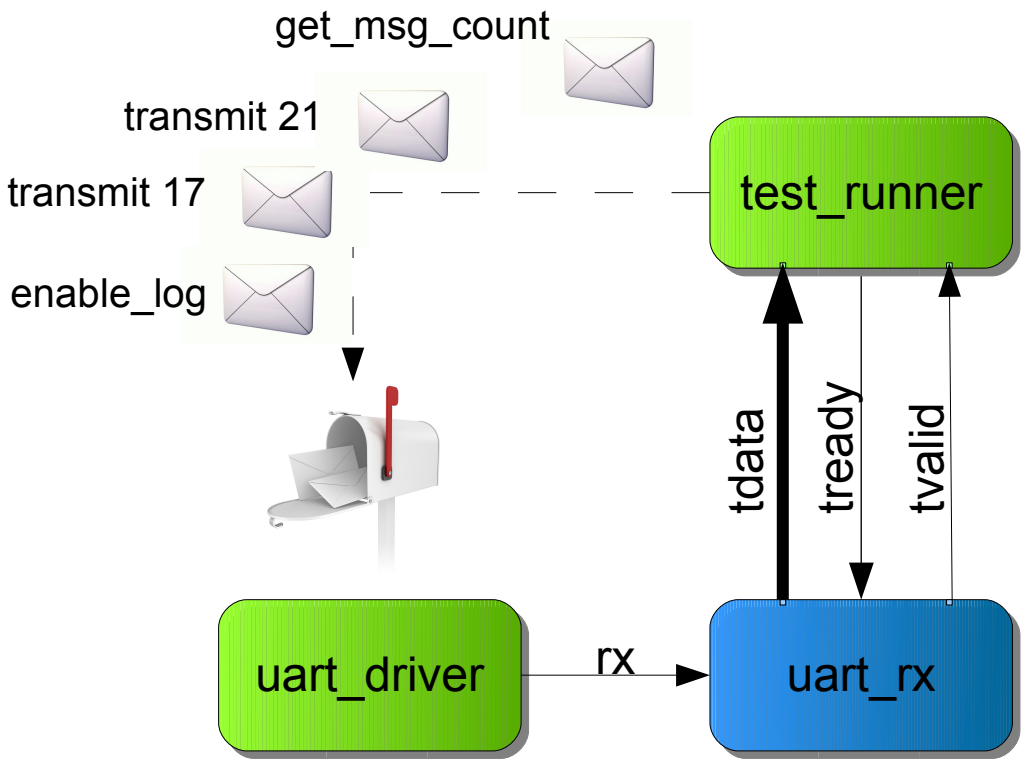
```
86810000 ps: DEBUG in traffic_logger (uart_rx.vhd:39): Received 77
```

Best viewed on YouTube

Synective Labs

Test Runner

Automating Execution



Best viewed on YouTube

Synective Labs

Test Runner

```
entity tb_uart_rx is
  generic (
    runner_cfg : runner_cfg_t := runner_cfg_default);
end entity;

-- <uart_driver, uart_rx, clock generator,...> --

test_runner : process
begin
  test_runner_setup(runner, runner_cfg);
    -- <test code> --
  test_runner_cleanup(runner);
end process;

test_runner_watchdog(runner, 10 ms);
```

Best viewed on YouTube

Synective Labs

Bitvis Utility Library Integration

```
test_runner : process
  impure function get_error_statistics
    return checker_stat_t is
    variable ret_val : checker_stat_t;
  begin
    ret_val.n_failed := get_alert_counter(TB_ERROR);
    return ret_val;
  end function get_error_statistics;
begin
  test_runner_setup(runner, runner_cfg);
    -- <test code> --
  test_runner_cleanup(runner, get_error_statistics);
end process;
```

Best viewed on YouTube

Synective Labs

Test Runner

```
test_runner : process
  variable filter : log_filter_t;
begin

  checker_init(
    display_format => verbose,
    file_name => join(output_path(runner_cfg),"error.csv"),
    file_format => verbose_csv);
  logger_init(...);
  stop_level(debug, display_handler, filter);
  uart_driver := find("UART Driver");
  check(uart_driver /= null, "Failed to find UART Driver",
        level => failure);

  test_runner_setup(runner, runner_cfg);
    -- <test code> --
  test_runner_cleanup(runner);
end process;
test_runner_watchdog(runner, 10 ms);
```

Best viewed on YouTube

Synective Labs

Test Suite

```
test_runner_setup(runner, runner_cfg);

while test_suite loop
  if run("Test that streaming data is handled") then
    ...
  elsif run("Test that overflow is detected") then
    ...
  elsif run("...") then
    ...
  end if;
end loop;

test_runner_cleanup(runner);
```

Best viewed on YouTube

Synective Labs

Test Suite

```
while test_suite loop
  if run("Test that streaming data is handled") then

    test_seq := rv.RandIntV(0, 255, test_seq'length);
    for i in test_seq'range loop
      send(uart_driver, transmit(test_seq(i)));
    end loop;

    tready <= '1';
    for i in test_seq'range loop
      wait until tvalid = '1' and rising_edge(clk);
      check_equal(unsigned(tdata), test_seq(i));
    end loop;

  elsif run("Test that overflow is detected") then
    ...
  elsif run("...") then
    ...
  end if;
end loop;
```

Best viewed on YouTube

Synective Labs

Python Run Script

```
ui = VUnit.from_argv()

src_path = join(dirname(__file__), "src")

uart_lib = ui.add_library("uart_lib")
uart_lib.add_source_files(join(src_path, "*.vhd"))

tb_uart_lib = ui.add_library("tb_uart_lib")
tb_uart_lib.add_source_files(join(src_path, "test", "*.vhd"))

ui.add_osvvm()

ui.main()
```

Best viewed on YouTube

Synective Labs

Passing Test

```
c:\git\vunit\examples\uart> python run.py

...


pass tb_uart_lib.tb_uart_tx.Test sending one byte after 1.0 seconds
pass tb_uart_lib.tb_uart_tx.Test sending two bytes after 1.2 seconds
pass tb_uart_lib.tb_uart_tx.Test sending many bytes after 1.2 seconds
pass tb_uart_lib.tb_uart_rx.Test that streaming data is handled after 1.0 seconds
pass tb_uart_lib.tb_uart_rx.Test that overflow is detected after 1.0 seconds
pass tb_uart_lib.tb_uart_rx.Test that tvalid is low at start after 1.5 seconds

Total time 6.9 seconds
6 of 6 passed
All passed!

c:\git\vunit\examples\uart>
```

Best viewed on YouTube

Synective Labs

Failing Test

```
pass tb_uart_lib.tb_uart_tx.Test sending two bytes after 1.2 seconds
pass tb_uart_lib.tb_uart_tx.Test sending many bytes after 1.2 seconds
pass tb_uart_lib.tb_uart_rx.Test that overflow is detected after 1.0 seconds
pass tb_uart_lib.tb_uart_rx.Test that tvalid is low at start after 1.5 seconds
fail tb_uart_lib.tb_uart_tx.Test sending one byte after 1.0 seconds
fail tb_uart_lib.tb_uart_rx.Test that streaming data is handled after 1.0 seconds

Total time 6.9 seconds
4 of 6 passed
2 of 6 passed
Some failed!

c:\git\vunit\examples\uart>
```

- Test cases are isolated
  - One failing test case doesn't fail the entire test suite
  - Failure states are not propagated between test cases

Best viewed on YouTube

Synective Labs

First Failing Test Case

- Error message

```
# ** Fatal: (vsim-3738) (vcom-1220) Integer divide by zero.
#    ...
#
# Surrounding code from 'see' command
#   75 :          for i in test_sequence'range loop
#   76 :            send(uart_driver, transmit(test_sequence(i)));
#   77 :          end loop;
#   78 :
# ->79 :          report to_string(1/0);
#   80 :          tready <= '1';
#   81 :          for i in test_sequence'range loop
#   82 :            wait until tvalid = '1' and rising_edge(clk);
#
```

- Simulator crashes are also handled

Best viewed on YouTube

Synective Labs

Second Failing Test Case

- Error message

```
# 173670000 ps: ERROR in (tb_uart_tx.vhd:92): Relation received_byte =
expected_byte failed! Left is 165. Right is 90.
```

- Caused by the following code

```
check_relation(received_byte = expected_byte);
```

- Preprocessing can help with location and error message generation

Best viewed on YouTube

Synective Labs

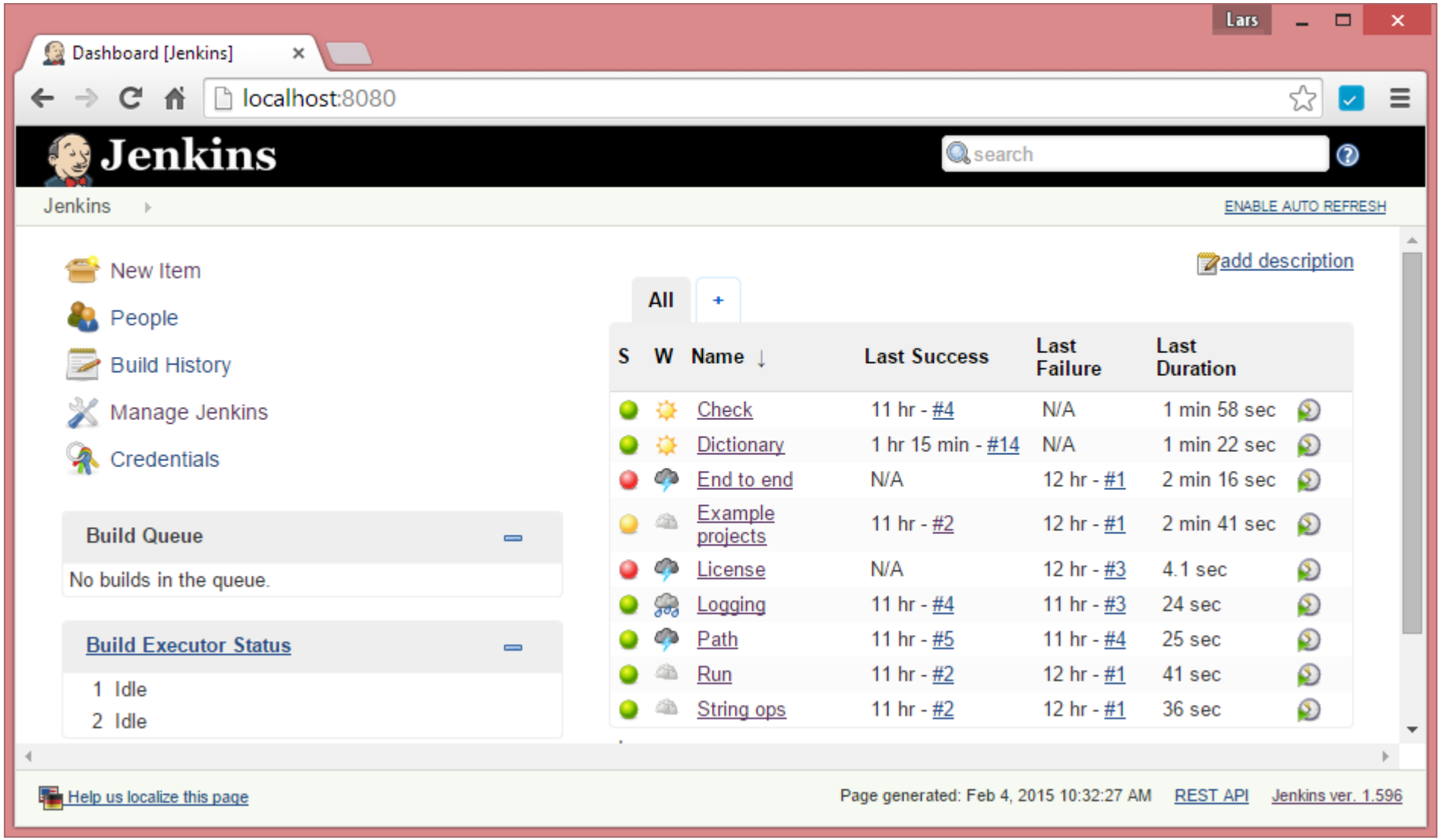
Jenkins

- Continuously run the test suites you think are affected by your changes
- Run all test suites occasionally to make sure you didn't miss any dependencies
- The open source tool Jenkins can do that
  - Manage when tests are run (e.g. nightly or when code is checked in to your code repository)
  - Distribute the workload on many computers to reduce feedback delay
  - Present results
  - Continuously build releases to enable customer feedback

Best viewed on YouTube

Synective Labs

Jenkins – Overview



Best viewed on YouTube

Synective Labs

Getting Started

- Download at https://github.com/LarsAsplund/vunit
- Examples and video available
- Introduce VUnit step by step
- Active and responsive community
- Participate by
  - Asking questions
  - Reporting bugs
  - Suggesting enhancement
  - Contribute code (pull requests)

Best viewed on YouTube

Synective Labs

Thank You

https://github.com/LarsAsplund/vunit

Best viewed on YouTube

Synective Labs