DesignNews

Test Automation Design for Embedded Systems

# DAY 3 : Unit-Testing Using Test-Driven Development (TDD) Part 1

Sponsored by

DigiKey

©2023 Beningo Embedded Group, LLC. All Rights Reserved.

# Webinar Logistics

- Turn on your system sound to hear the streaming presentation.
- If you have technical problems, click “Help” or submit a question asking for assistance.
- Participate in ‘Group Chat’ by maximizing the chat widget in your dock.


©2023 Beningo Embedded Group, LLC. All Rights Reserved.

CEC Continuing Education Center

BENINGO EMBEDDED GROUP

DigiKey

# THE SPEAKER

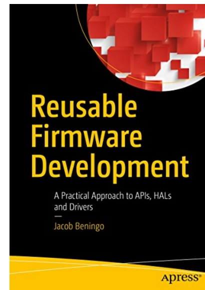
## Jacob Benigno

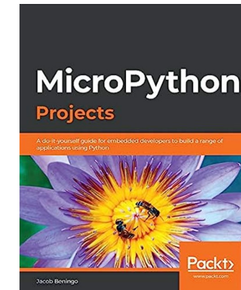
Jacob@beningo.com

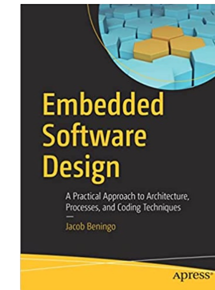
# Beningo Embedded Group – CEO / Founder

Focus: Embedded Software Consulting and Training

Help teams deliver higher-quality embedded software faster. We specialize in creating and promoting embedded software excellence in businesses around the world.




Blogs for:

- DesignNews.com
- Embedded.com
- EmbeddedRelated.com
- MLRelated.com

Visit www.beningo.com to learn more



©2023 Beningo Embedded Group, LLC. All Rights Reserved.

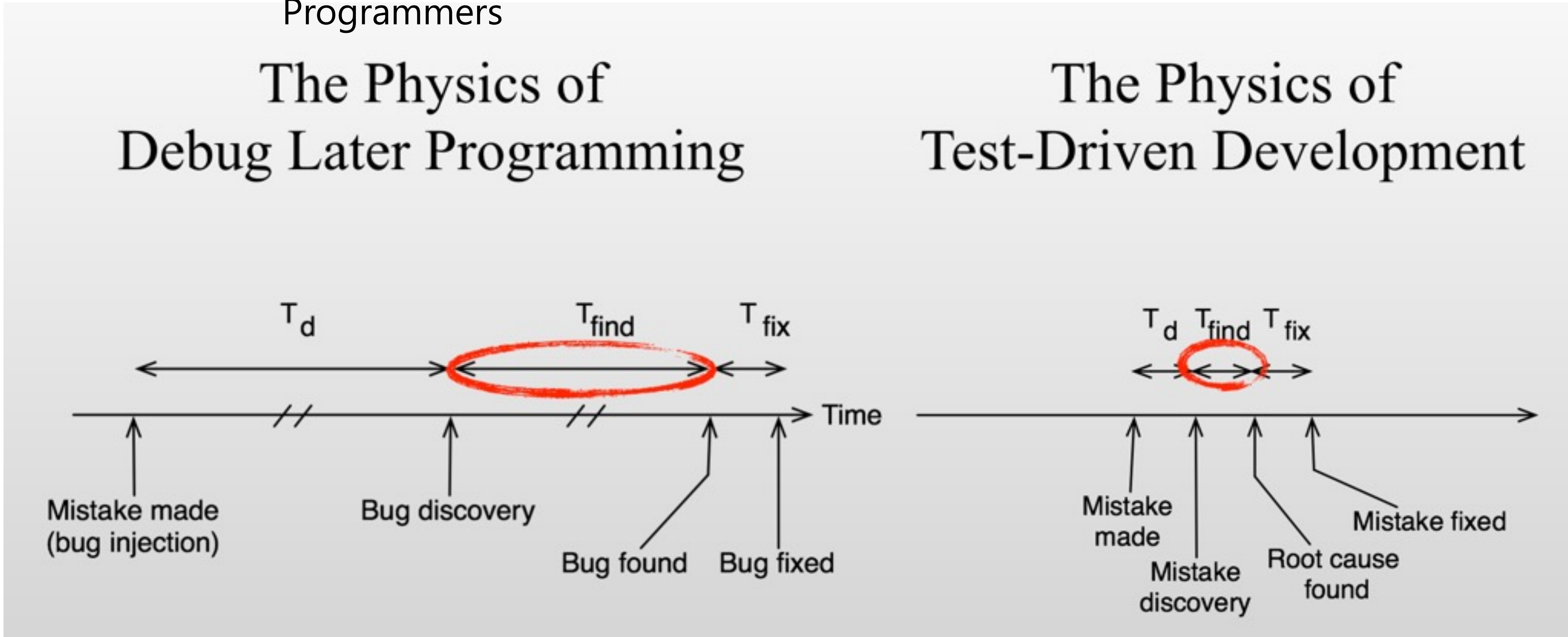
01

# Test-Driven Development TDD


©2023 Beningo Embedded Group, LLC. All Rights Reserved.

# TDD Physics

Source: James Grenning; TDD for Embedded C; Pragmatic Programmers






©2023 Beningo Embedded Group, LLC. All Rights Reserved.

# Test-Driven Development (TDD)

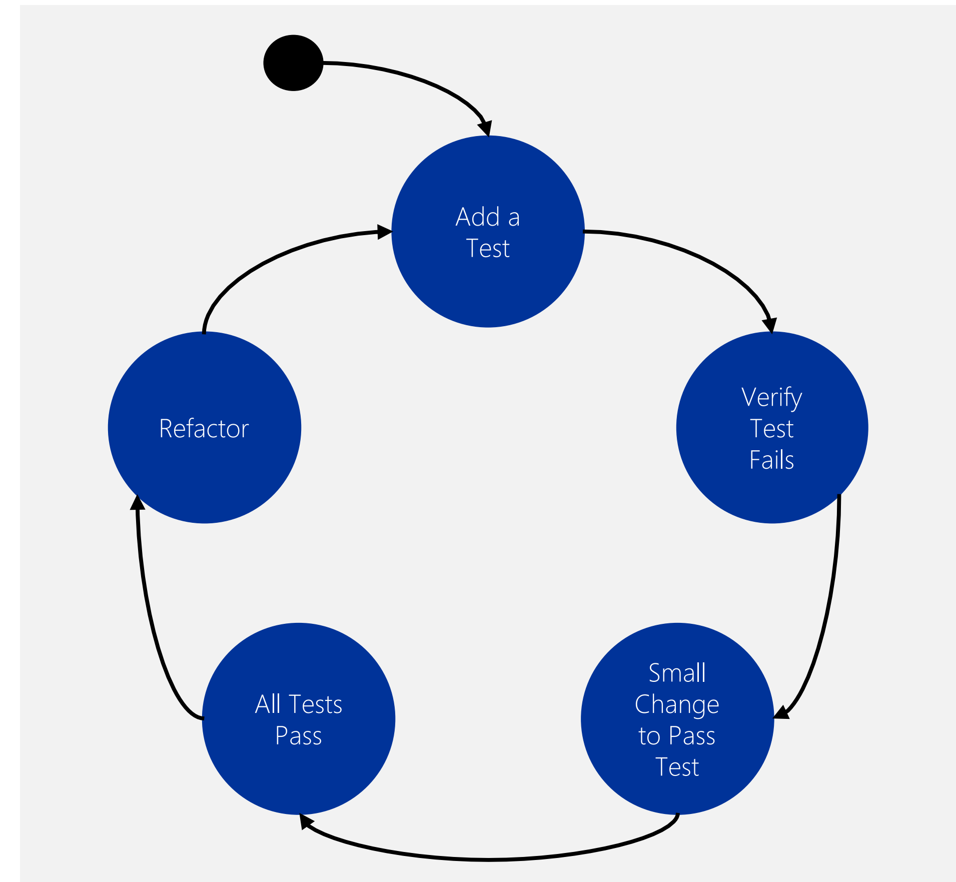
**TDD** is a technique for building software incrementally that allows the test cases to drive the production code development.

- TDD improves code quality through cleaner, less buggy code
- Improves design due to developers thinking more carefully about what they are doing
- Code is debugged more efficiently due to failing tests that pinpoint exactly what the problem is
- Reduced development time and cost by catching issues earlier in the development cycle
- Developers can refactor with confidence due to existing tests


©2023 Beningo Embedded Group, LLC. All Rights Reserved.

# The TDD Microcycle

1. Add a small test
2. Run all the tests and see the new one fail. (Maybe not even compile!)
3. Make the small change(s) needed to pass the test
4. Run all the tests and see the new one pass
5. Refactor to remove duplication and improve the expressiveness of the tests






©2023 Beningo Embedded Group, LLC. All Rights Reserved.

# Audience POLL Question

How do you feel about TDD?

- For TDD
- Cautiously optimistic
- Skeptical, but see the value in it
- Rubbish! I can’t support this concept



©2023 Beningo Embedded Group, LLC. All Rights Reserved.

02

# What makes a great test?


©2023 Beningo Embedded Group, LLC. All Rights Reserved.

# What makes a great test?

Edsger W. Dijkstra stated that:

"Testing shows the presence, not the absence of bugs"


©2023 Beningo Embedded Group, LLC. All Rights Reserved.

# What makes a great test?

Great tests:

- Are limited in scope and test only one thing at a time
- Communicate their intent clearly
- Be self-documenting
- Automated
- Easily change with time (maintainable)

```
TEST(PacketDecoding, crc_correct)
```

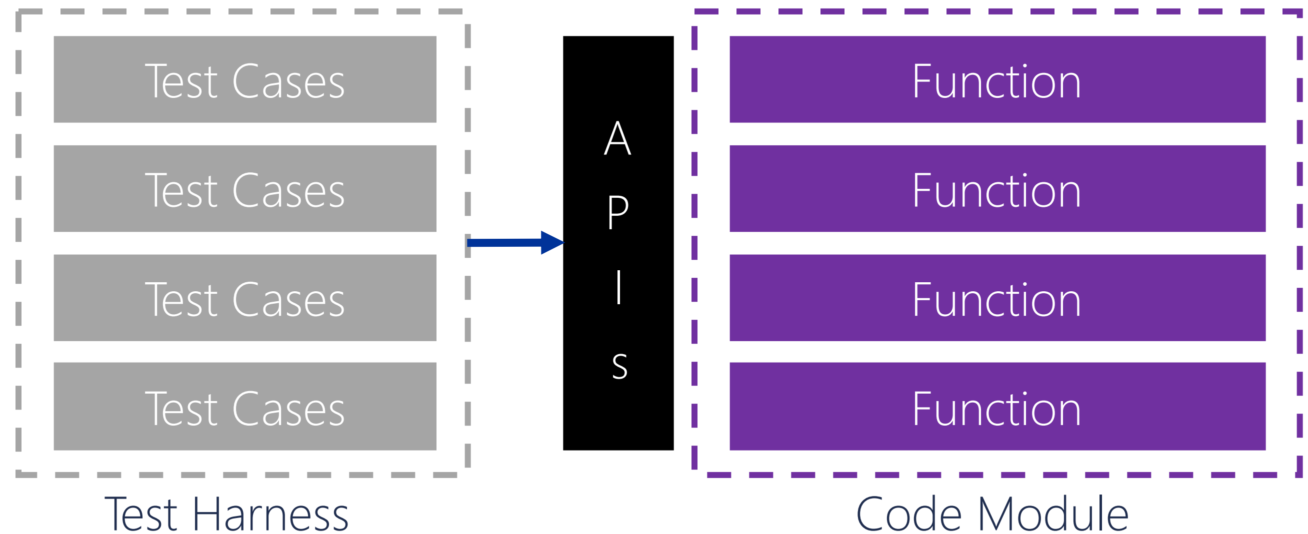


©2023 Beningo Embedded Group, LLC. All Rights Reserved.

# What makes a great test?

## Questions to ask yourself:

- What are the inputs to the interface?
- What are the outputs from the interface?
- Are there errors that are reported or can be tracked?
- What ranges can inputs and outputs be within?






©2023 Beningo Embedded Group, LLC. All Rights Reserved.

# Audience POLL Question

Testing can prove that there are no bugs in the system:

- True
- False



©2023 Beningo Embedded Group, LLC. All Rights Reserved.

CEC Continuing Education Center

BENINGO EMBEDDED GROUP

DigiKey

03

# Writing your first unit test



©2023 Beningo Embedded Group, LLC. All Rights Reserved.

# Writing Your First Unit Test

A few simple steps to get started:

1) Start Docker Desktop
2) Open Visual Studio Code
3) Open the controller project
4) Click Terminal->New Terminal
5) In the terminal, type make docker_image
6) In the terminal, type make docker_run
7) Write your first test


©2023 Beningo Embedded Group, LLC. All Rights Reserved.

# Writing Your First Unit Test





©2023 Beningo Embedded Group, LLC. All Rights Reserved.

# Writing Your First Unit Test

Start with a design:

- Add 2 integers of type uint8_t

The interface might look like:

```
uint8_t add(uint8_t a, uint8_t b);
```

Make a list of tests:

- Zero values: a = 0, b = 0
- Zero and non-zero: a = 0, b = 10
- Zero and non-zero: a = 10, b = 0
- Positive Numbers: a = 50, a = 70
- Max without overflow: a = 125, b = 130
- Overflow: a = 200, b = 100
- "Typical" Values: a = 23, b = 77
- Symmetrical values: a = 123, b = 123
- Boundary values: a = 254, b = 1
- Overflow by 1: a = 255 + 1



©2023 Beningo Embedded Group, LLC. All Rights Reserved.

# Writing Your First Unit Test

1) Open adder_tests.cpp
2) Add the test:

```
TEST(adder, onePlusOneEqualsTwo)
{
    CHECK_EQUAL(2, add(1, 1));
}
```

3) Run "make unit_tests", watch it fail:

```
/usr/bin/ld: tests/obj/tests/adder_tests.o: in function `TEST_adder_onePlusOneEqualsTwo_Test::testBody()':
/home/app/tests/adder_tests.cpp:25: undefined reference to `add'
/usr/bin/ld: /home/app/tests/adder_tests.cpp:25: undefined reference to `add'
/usr/bin/ld: /home/app/tests/adder_tests.cpp:25: undefined reference to `add'
/usr/bin/ld: /home/app/tests/adder_tests.cpp:25: undefined reference to `add'
collect2: error: ld returned 1 exit status
make[1]: *** [/home/cpputest/build/MakefileWorker.mk:521: tests/main_tests] Error 1
make[1]: Leaving directory '/home/app'
make: *** [Makefile:106: unit_tests] Error 2
root@de25010e3006:/home/app#
```

4) Add the function, and compile:

```
uint8_t add(uint8_t a, uint8_t b) {

}
```

5) Compile, and watch the test fail

```
        Thus the value in the error message is probably incorrect.
tests/adder_tests.cpp:25: error: Failure in TEST(adder, onePlusOneEqualsTwo)
        expected <2>
        but was  <4>
        difference starts at position 0 at: <          4         >
                                                       ^

.
Errors (1 failures, 2 tests, 2 ran, 2 checks, 0 ignored, 0 filtered out, 0 ms)
```



©2023 Beningo Embedded Group, LLC. All Rights Reserved.

# Writing Your First Unit Test

6) Write some code to make the test pass:

```
uint8_t add(uint8_t a, uint8_t b) {
    return 2;
}
```

7) Verify the test passes:

```
Running tests/main_tests
..
OK (2 tests, 2 ran, 2 checks, 0 ignored, 0 filtered out, 0 ms)

make[1]: Leaving directory '/home/app'
mkdir -p tests/coverage
(INFO) Reading coverage data...
(INFO) Writing coverage report...
------------------------------------------------------------------------------
                           GCC Code Coverage Report
Directory: .
------------------------------------------------------------------------------
File                                       Lines    Exec  Cover   Missing
------------------------------------------------------------------------------
firmware/app/adder.c                           2       2   100%
------------------------------------------------------------------------------
TOTAL                                          2       2   100%
------------------------------------------------------------------------------
```



©2023 Beningo Embedded Group, LLC. All Rights Reserved.

# Audience POLL Question

Can you trust that 100% test coverage means there are no bugs in the code?

a) Yes
b) No



©2023 Beningo Embedded Group, LLC. All Rights Reserved.

CEC Continuing Education Center

BENINGO EMBEDDED GROUP

DigiKey

# Next Steps

04

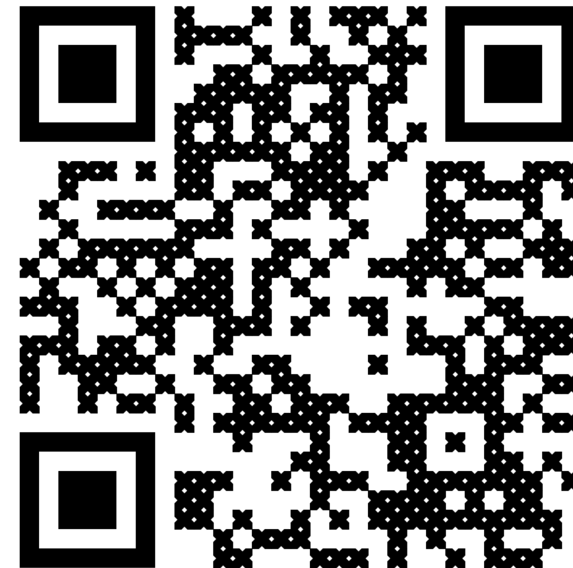


©2023 Beningo Embedded Group, LLC. All Rights Reserved.

# Test Automation Build System

Build System Example

- Docker container build system
- Makefile-based
- Cmake with Ninja Example
- Compilation scripts
- Integrated tools like cpputest



https://mailchi.mp/beningo/beningo-devops



©2023 Beningo Embedded Group, LLC. All Rights Reserved.

# Additional Resources

Please consider the resources below:

- Jacob's Blogs
- Jacob's CEC courses
- Embedded Software Academy

- Embedded Bytes Newsletter
  - http://bit.ly/1BAHYXm

www.beningo.com

Consulting | Coaching | Training



©2023 Beningo Embedded Group, LLC. All Rights Reserved.

# Next Steps

- ✓ Introduction to Test Automation Design
- ✓ Using Docker for a Test Automation Environment
- ✓ Unit-Testing Using Test-Driven Development Part 1
- Unit-Testing Using Test-Driven Development Part 2
- Automating System-Level Testing



©2023 Beningo Embedded Group, LLC. All Rights Reserved.

DesignNews

# Thank You

Sponsored by

DigiKey