CEC Continuing Education Center

DesignNews

Test Automation Design for Embedded Systems

DAY 5 : Automating System-Level Testing

Sponsored by

DigiKey

BENINGO EMBEDDED GROUP

informa markets

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.

CEC Continuing Education Center

BENINGO EMBEDDED GROUP

DigiKey

THE SPEAKER

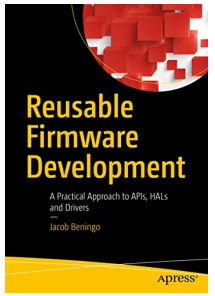
Jacob Beningo

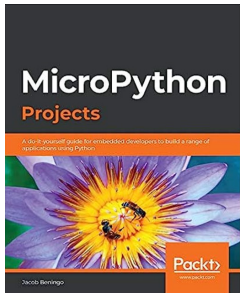
Jacob@beningo.com

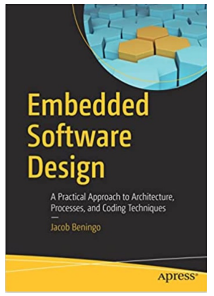
Beningo Embedded Group – CEO / Founder

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

CEC Continuing Education Center

BENINGO EMBEDDED GROUP

DigiKey

01

System-level Testing

System-level Testing

System-level testing is a type of testing in which the complete, integrated system is tested to ensure that it meets the specified requirements.

It encompasses both functional and non-functional testing, aiming to verify the system's:

- Behavior
- Performance
- Security
- Overall reliability in real-world scenarios

Types of System Testing

Clear box testing (White Box Testing) involves testing the internal structures or workings of an application, as opposed to its functionality. The tester selects inputs to exercise paths through the code and determines the appropriate outputs. This type of testing requires knowledge of the internal logic of the system's code.

Opaque Box Testing (Black Box Testing) is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This type of testing focuses on the input and output of the software system and is based on requirements and specifications.

Requirements Traceability

Requirement: Blinking an LED at 2 Hz

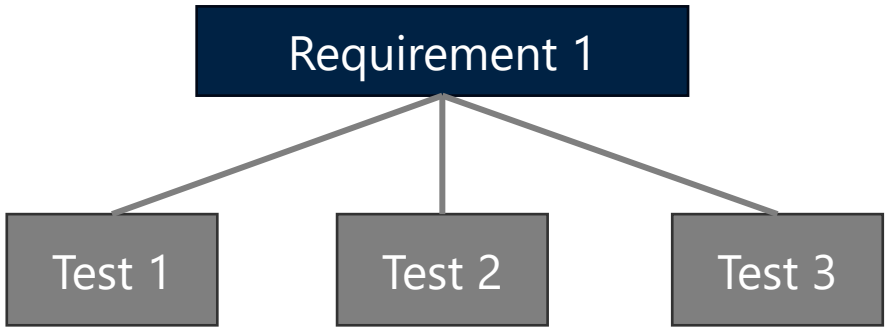
Description: The controller shall blink an LED at a frequency of 2 Hz

Functional Requirements:

1. The system shall turn the LED on and off at a frequency of 2 Hz
2. The LED shall have a duty cycle of 50%

Non-Function Requirements:

1. Timing Accuracy: The blinking frequency will have an accuracy of +/- 0.05 Hz
2. Durability: The system shall be capable of operating continuously for at least 10,000 hours of blinking without failure



Audience POLL Question

Do you trace your system-level requirements to the system-level tests that are executed to validate your system?

- Yes
- No
- Working on it
- Other

CEC Continuing Education Center

BENINGO EMBEDDED GROUP

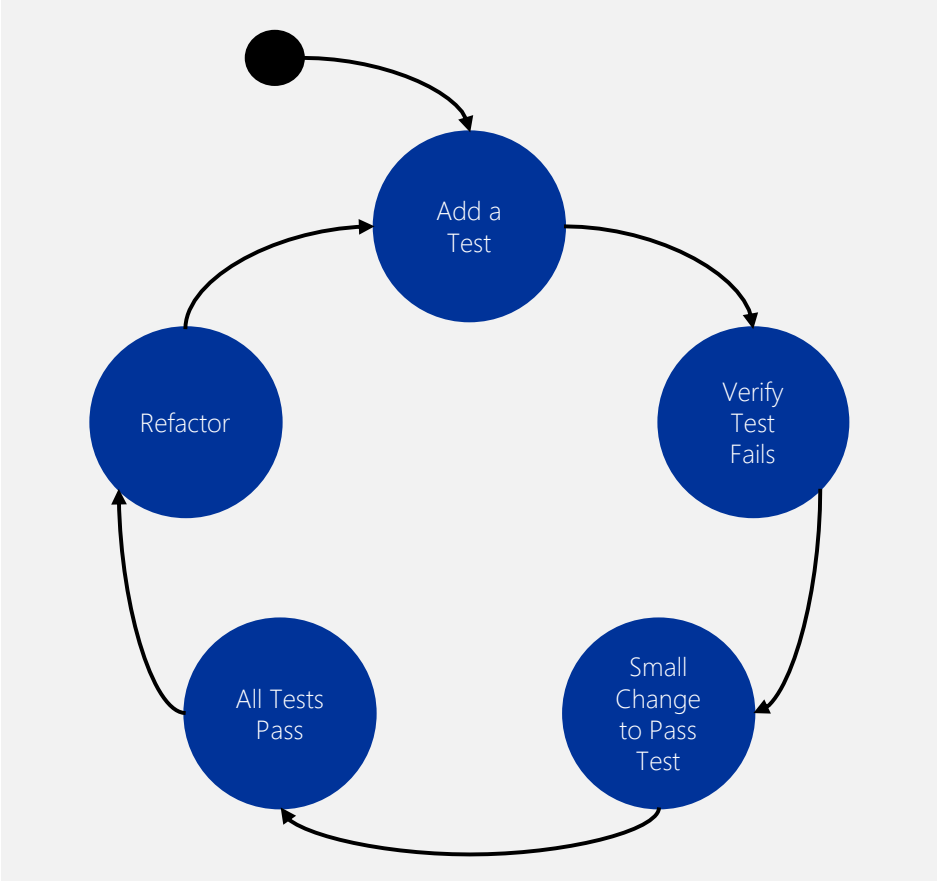
DigiKey

02

System-Level TDD

The TDD Microcycle

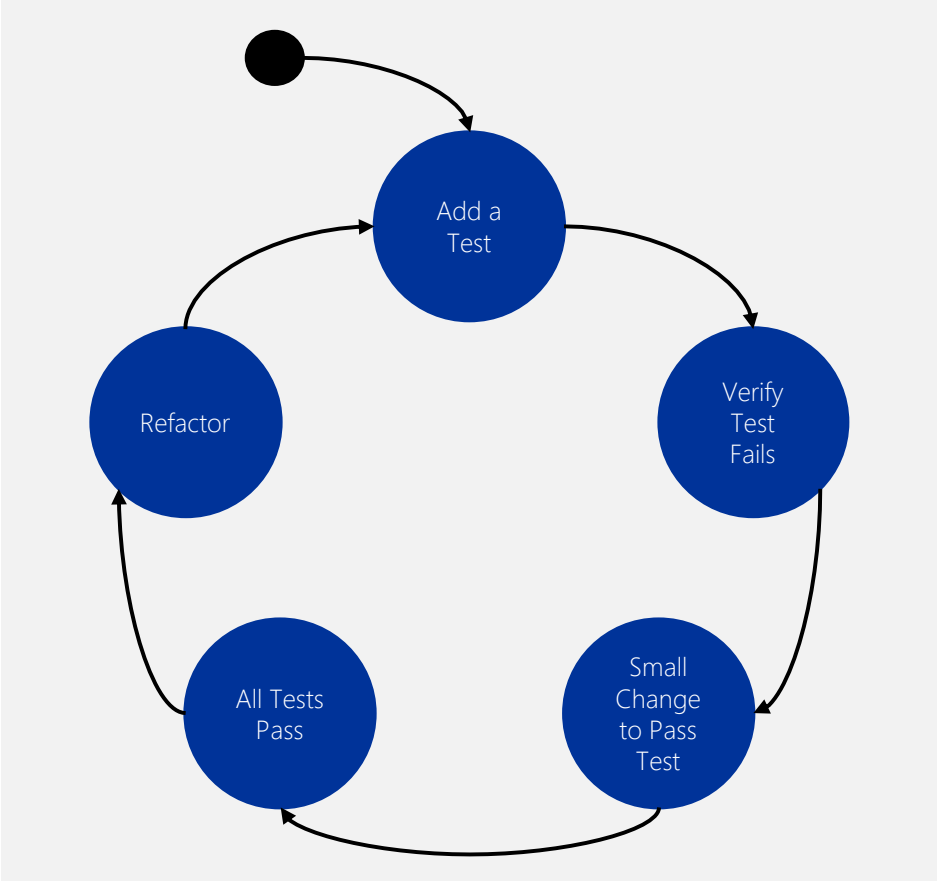
1. Add a small test
2. Run all the tests and see the new one fail. (Maybe not even compile!)
3. Make the small change(s) needed to pass the test
4. Run all the tests and see the new one pass
5. Refactor to remove duplication and improve the expressiveness of the tests



System-Level TDD Microcycle

1. Add a small test (from your requirements)
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

Instead of working from the bottom up, we work from the system-level in!



Audience POLL Question

Is system-level TDD something you would be interested in trying?

- Yes, I'm already doing it
- Yes, in time it's possible
- Maybe, it will be hard to convince my team
- No, I don't see the value in it

CEC Continuing Education Center

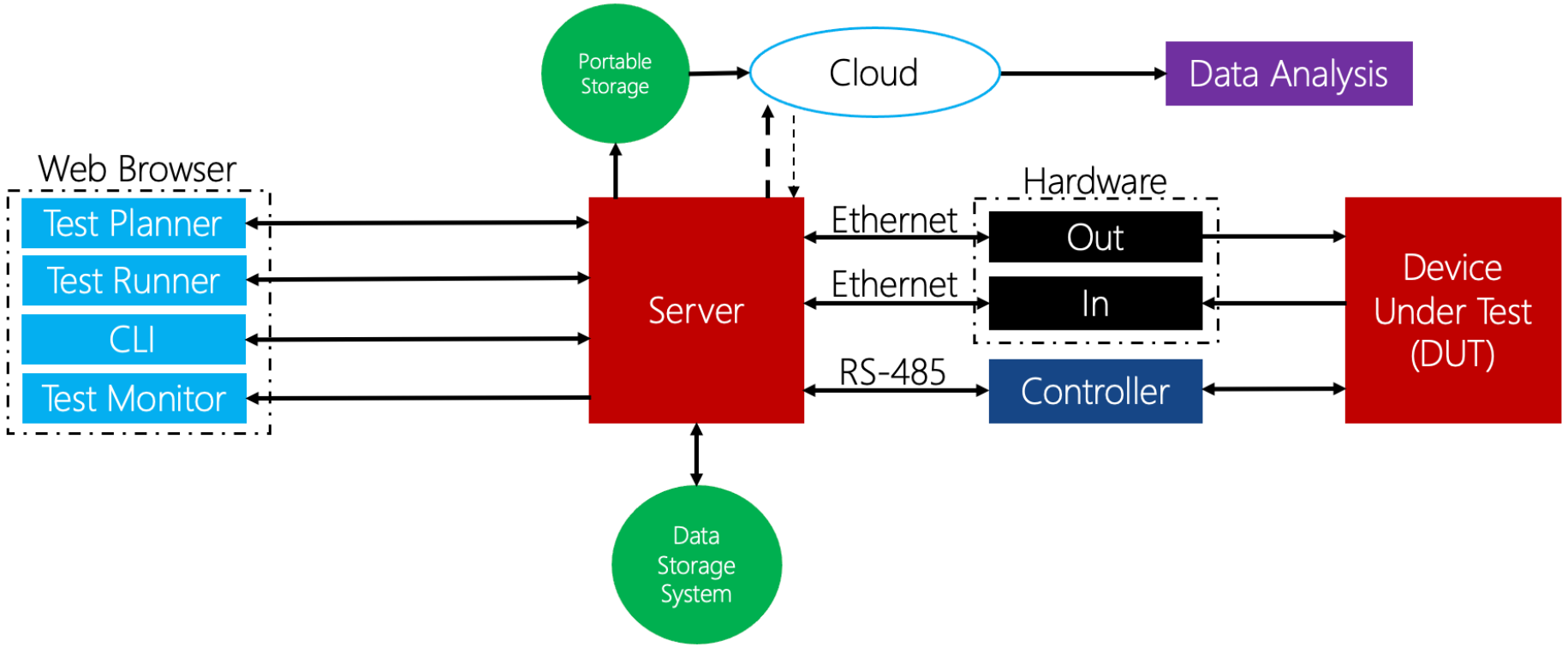
BENINGO EMBEDDED GROUP

DigiKey

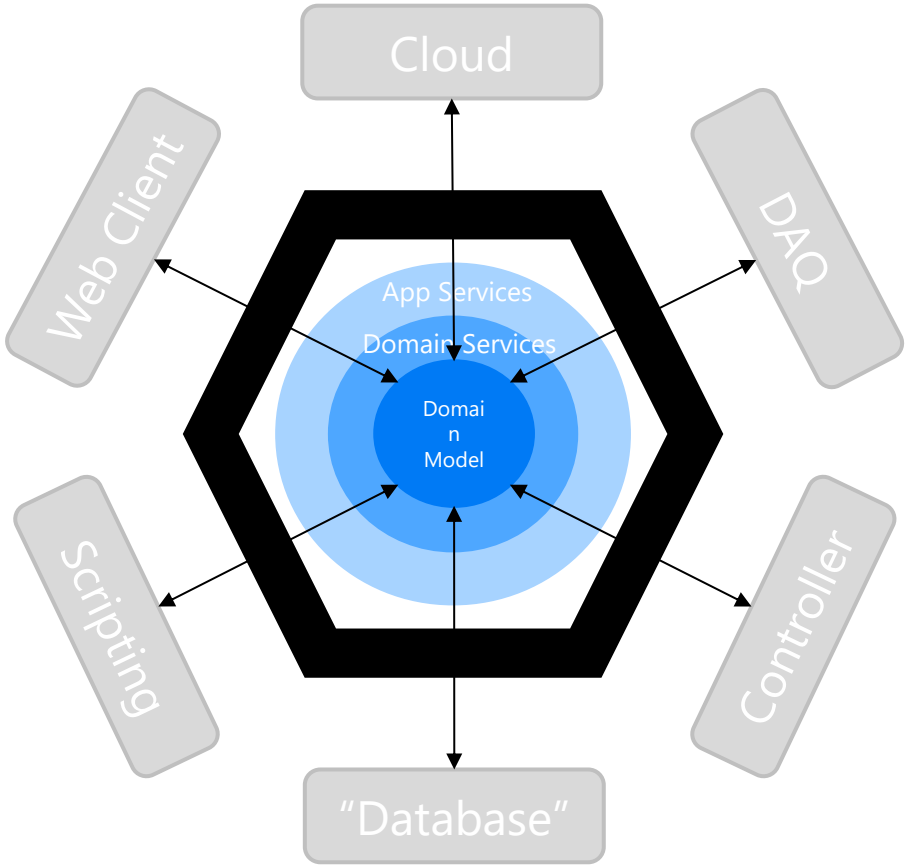
03

Automating your system tests

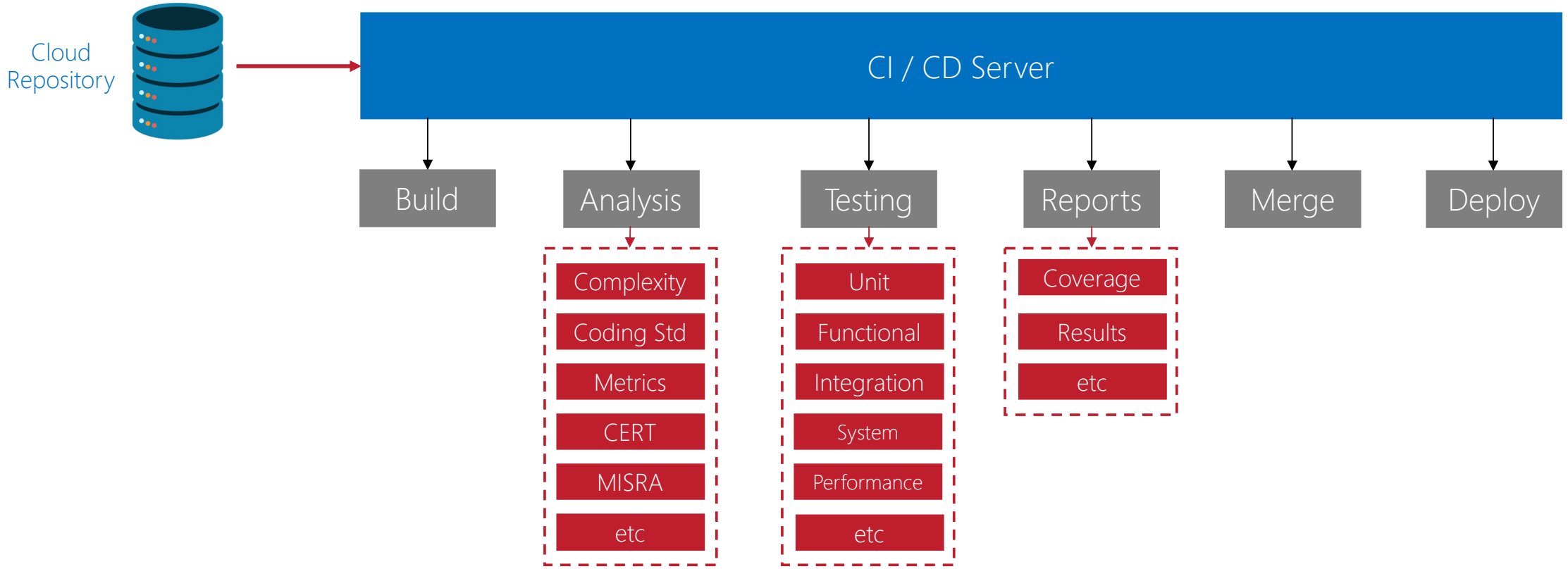
An Example HIL Test Setup



Ports and Adapters Architecture



Automating the Tests



Audience POLL Question

Is the extra effort worth it to automate system-level testing?
- Yes, I'm already doing it
- Yes, in time it's possible
- Maybe, it will be hard to convince my team
- No, I don't see the value in it

Next Steps

04

Key Take-A-Ways

- How you design your build system will dictate whether you can automate testing
- Test-driven development is a great technique to catch bugs as you build from the ground up or from the system in
- Automating system testing requires adding hardware to
  - Drive the system
  - Collect data
- Analyzing results may be difficult, although AI is now a possible helper!

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

Additional Resources

Please consider the resources below:

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

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

Consulting | Coaching | Training

www.beningo.com

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

CEC Continuing Education Center

DesignNews

Thank You

Sponsored by

DigiKey

BENINGO EMBEDDED GROUP

informa markets