Sensors
Converge

MODERNIZING EMBEDDED SOFTWARE

June 20–22, 2023 | Santa Clara, CA

#SensorsConverge

THE SPEAKER

Jacob Beningo

Beningo Embedded Group
President

Focus: Embedded Software Consulting

An independent consultant who specializes in the design of real-time, microcontroller based embedded software.

He has published three books;

- **Reusable Firmware Development**
- **MicroPython Projects**
- **Embedded Software Design**

Writes a weekly blog for DesignNews.com focused on embedded system design techniques and challenges.

Visit **www.beningo.com** to learn more ...

EMBEDDED ONLINE CONFERENCE

www.embeddedonlineconference.com

Preconference
Organized By:

Embedded
Online
Conference

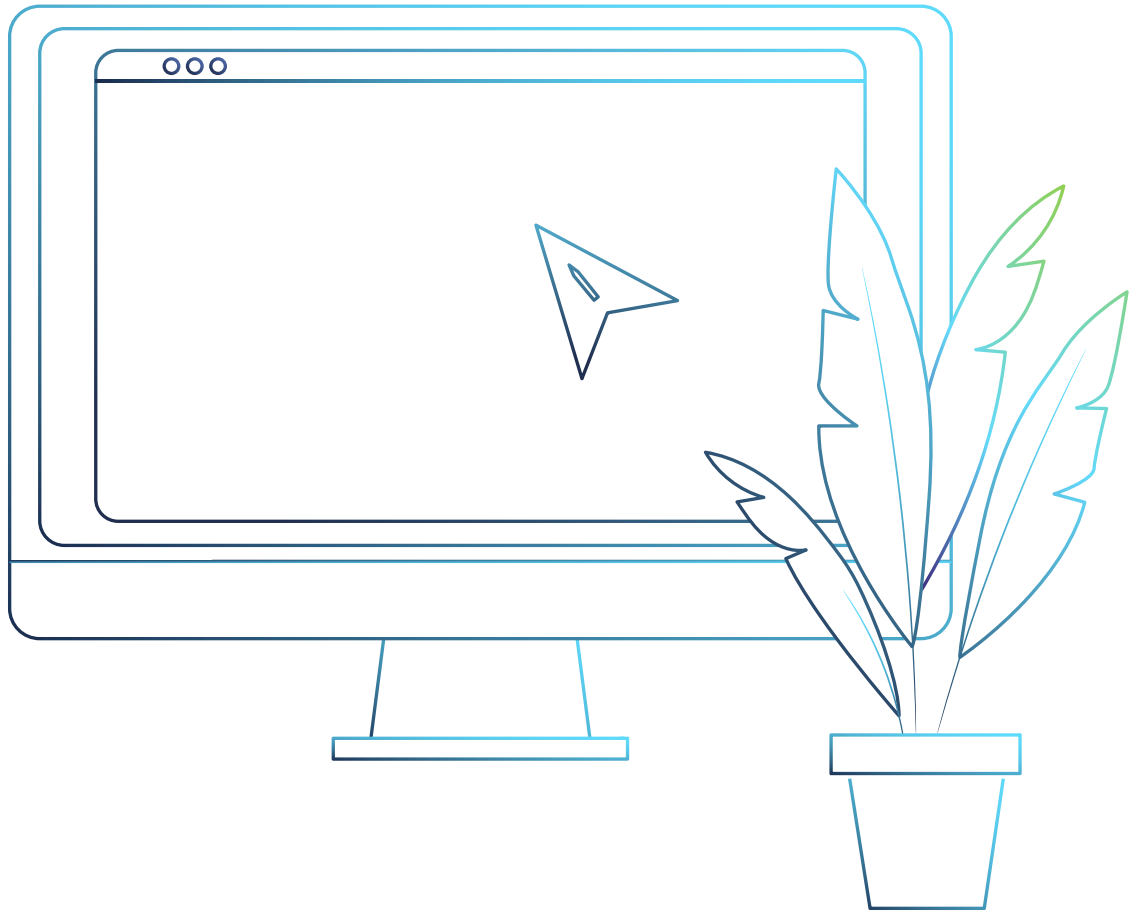
1

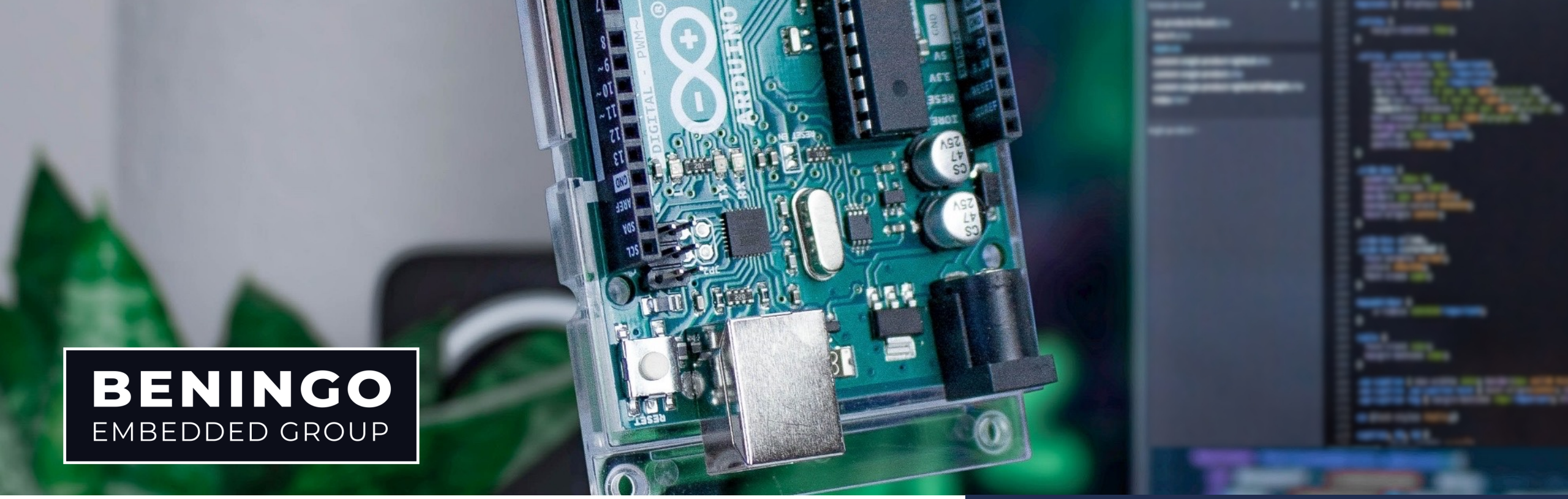
Embedded Systems Preconference Track Overview

SESSION OVERVIEW

Agenda

9:00 – 9:30	Modern Embedded Software Development Strategies and Processes
9:30 – 10:15	Designing Secure Embedded Devices for an Interconnected World
10:25 – 11:10	Sensing on a Mars Analogue Astronaut Mission
11:15 – 12:00	The Best Defense is Offensive Programming
12:00 – 12:50	Lunch
1:00 – 1:50	Where's the killer app for TinyML?
2:00 – 2:50	A Hands-on Introduction to The Zephyr Project RTOS





BENINGO
EMBEDDED GROUP

Simplifying Concepts, Delivering Success℠

MODERNIZING EMBEDDED SOFTWARE

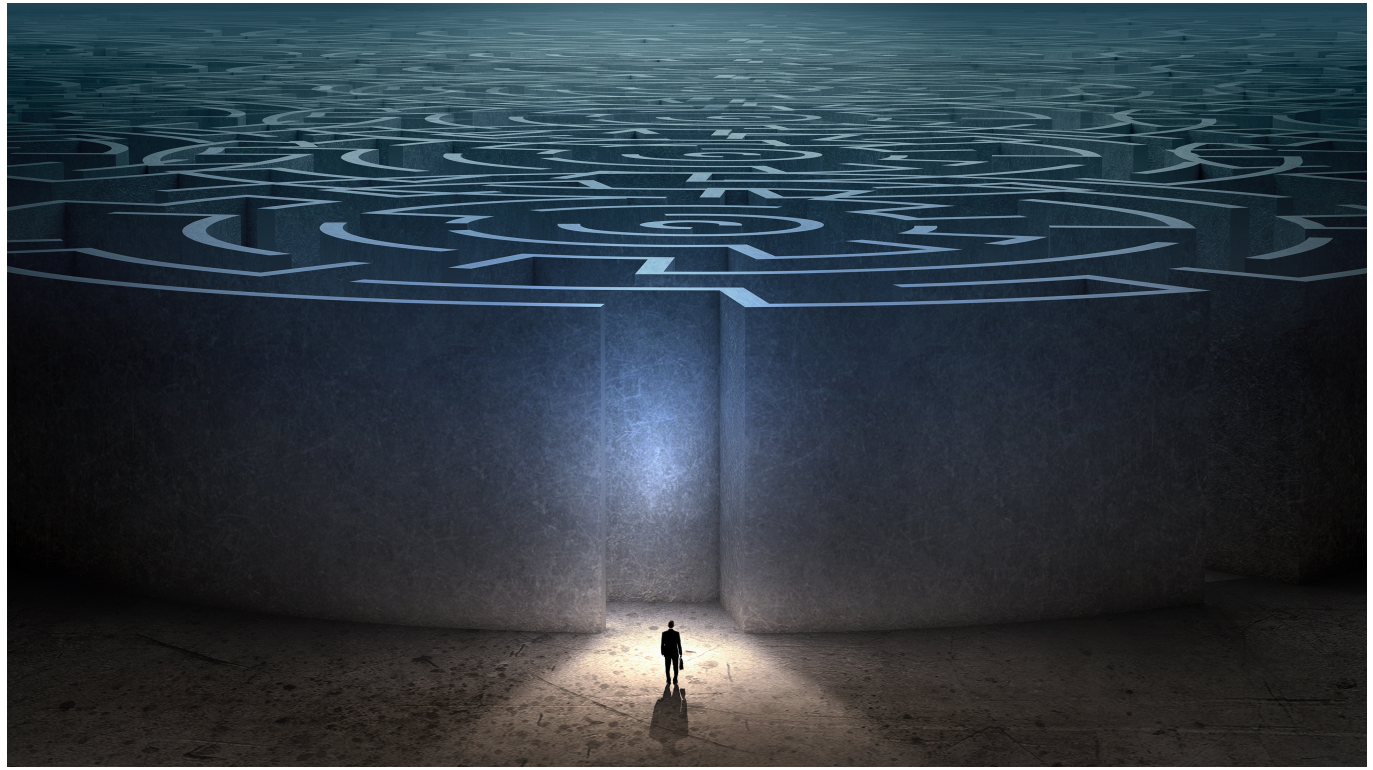
Session 0000 – Modern Embedded Software Development Strategies and Processes

2

Embedded Software Challenges

EMBEDDED SOFTWARE CHALLENGES

What challenges are you facing?



MODERN CHALLENGES

Challenges Facing Embedded Developers

Quality

- Buggy software
- Constant bug fixes
- Customer complaints

Development Costs

- Smaller budgets
- More features
- Increased complexity

Time to Market

- More debugging
- Missed deadlines
- Integration woes

Scalable Solutions

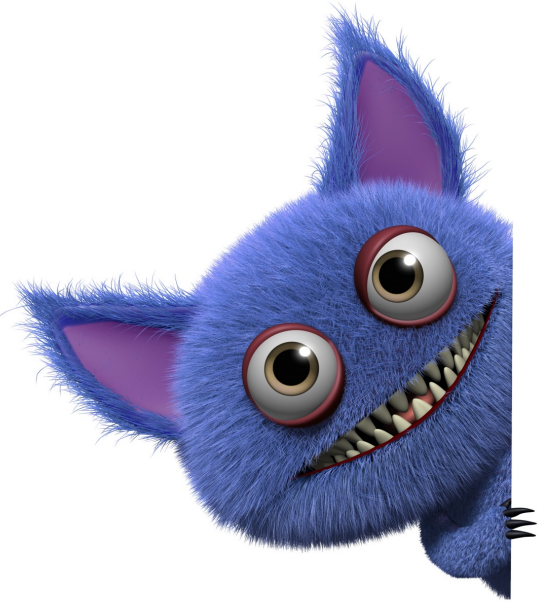
- Tightly coupled code
- Vendor dependency
- Inflexible architecture

2

Software Design Philosophy

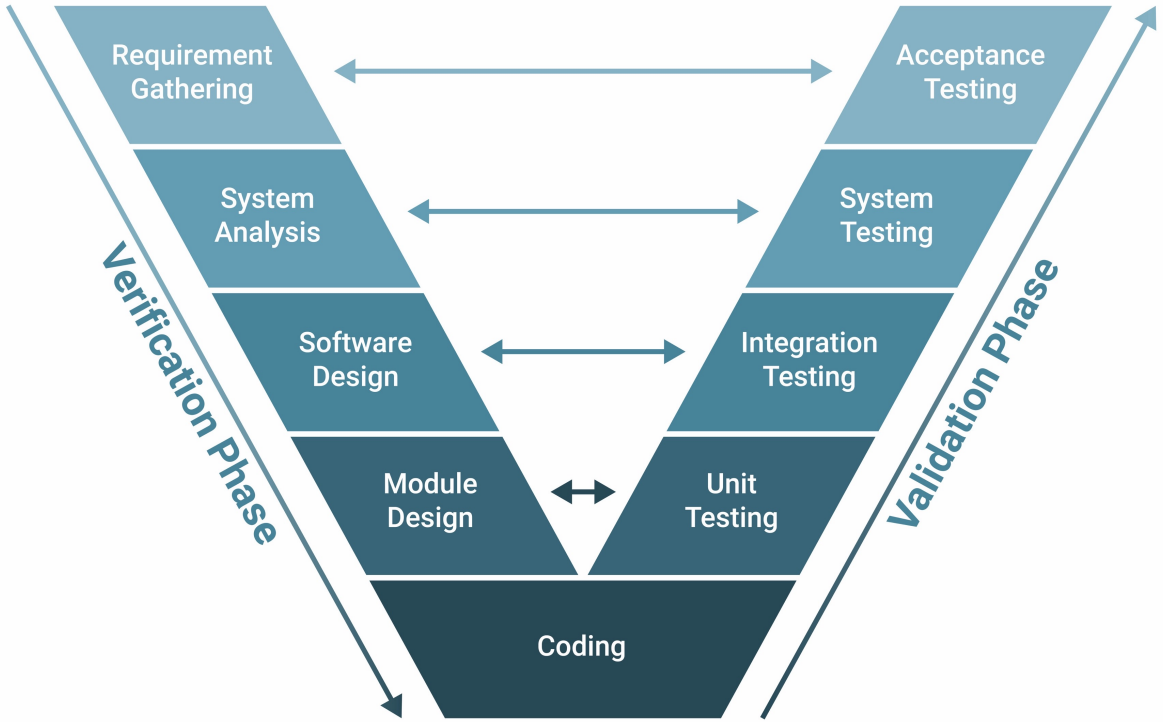
SOFTWARE DESIGN PHILOSOPHY

Fight the biggest fire



SOFTWARE DESIGN PHILOSOPHY

Traditional Thinking



SOFTWARE DESIGN PHILOSOPHY

Agile Methodologies

At its core, Agile is about[1]:

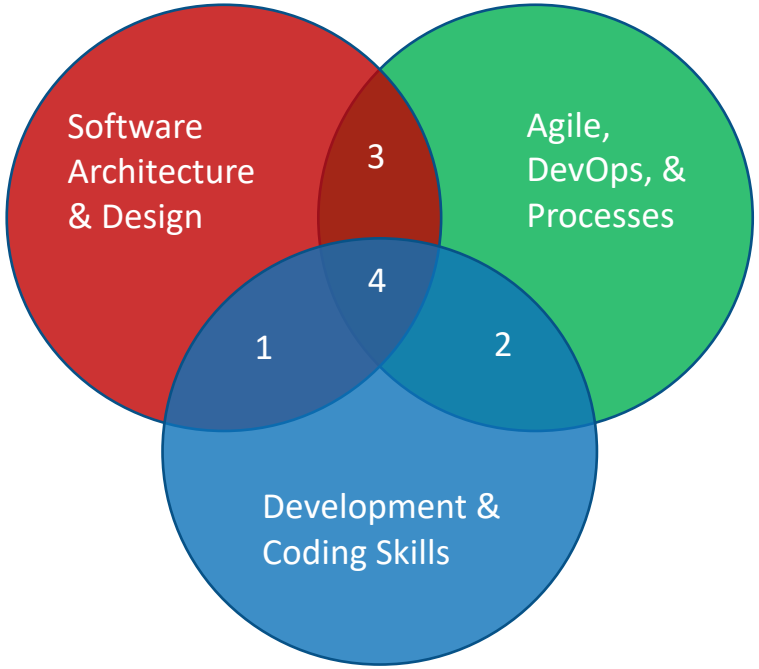
1) **Individuals and interactions** over processes and tools
2) **Working software** over comprehensive documentation
3) **Customer collaboration** over contract negotiation
4) **Responding to change** over following a plan

> "Some folks think that Agile is about going fast. It's not. It's never been about going fast. Agile is about knowing, as early as possible, just how screwed we are."
>
> -- Bob Martin

[1]Clean Agile, Bob Martin

SOFTWARE DESIGN PHILOSOPHY

Successful Embedded Software



1 - Late, Inconsistent, Quality Issues
2 - Late, Rework, Lost / Meandering
3 - Never completed
4 - Successful Delivery

SOFTWARE DESIGN PHILOSOPHY

Define the principles that are most important to you.

- Principle #1 – Data Dictates Design
- Principle #2 – There is No Hardware (only data)
- Principle #3 – KISS the Software
- Principle #4 – Practical, Not Perfect
- Principle #5 – Scalable and Configurable
- Principle #6 – Test, Test, and Test
- Principle #7 – Security is King

Action Item:

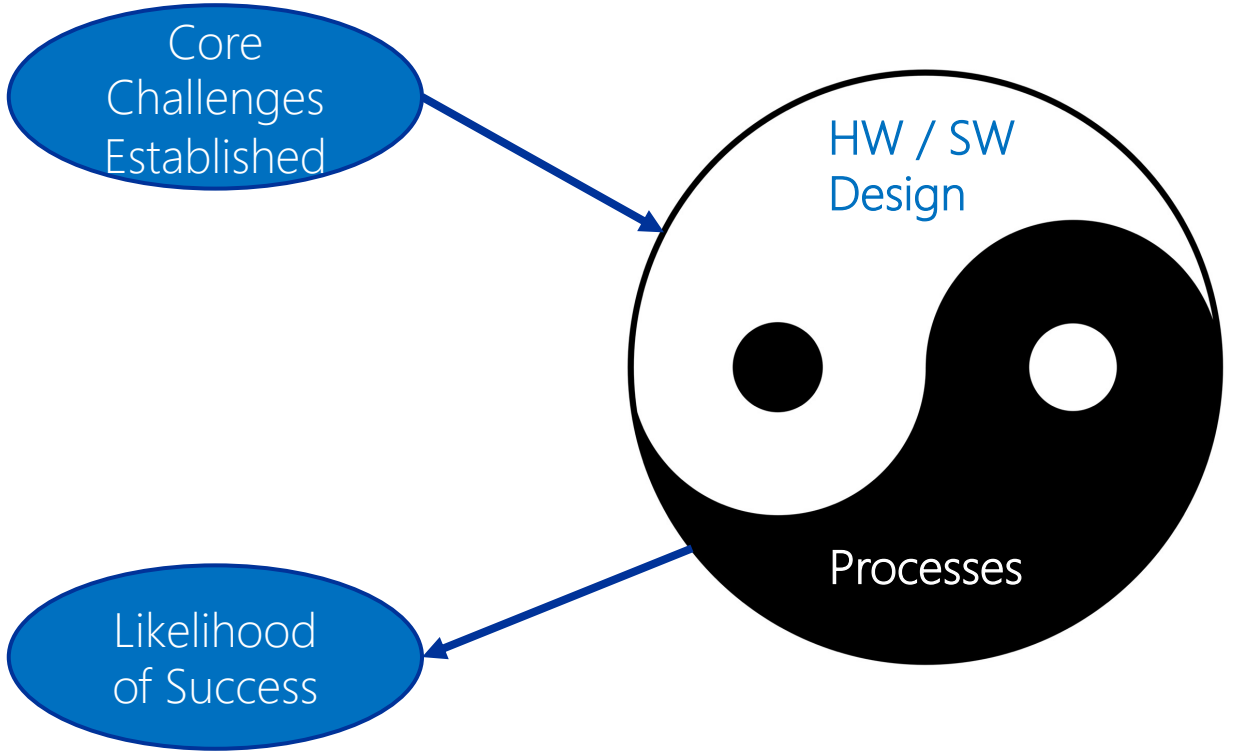
Identify and write down your top 7 principles.

Put them in a place you can readily see them!

SOFTWARE DESIGN PHILOSOPHY

Successful Embedded Software

- Process dictates design
- Design dictates Process
- What is being accomplished?
  - Proof-of-concept
  - Production system?
  - Simulation?



3

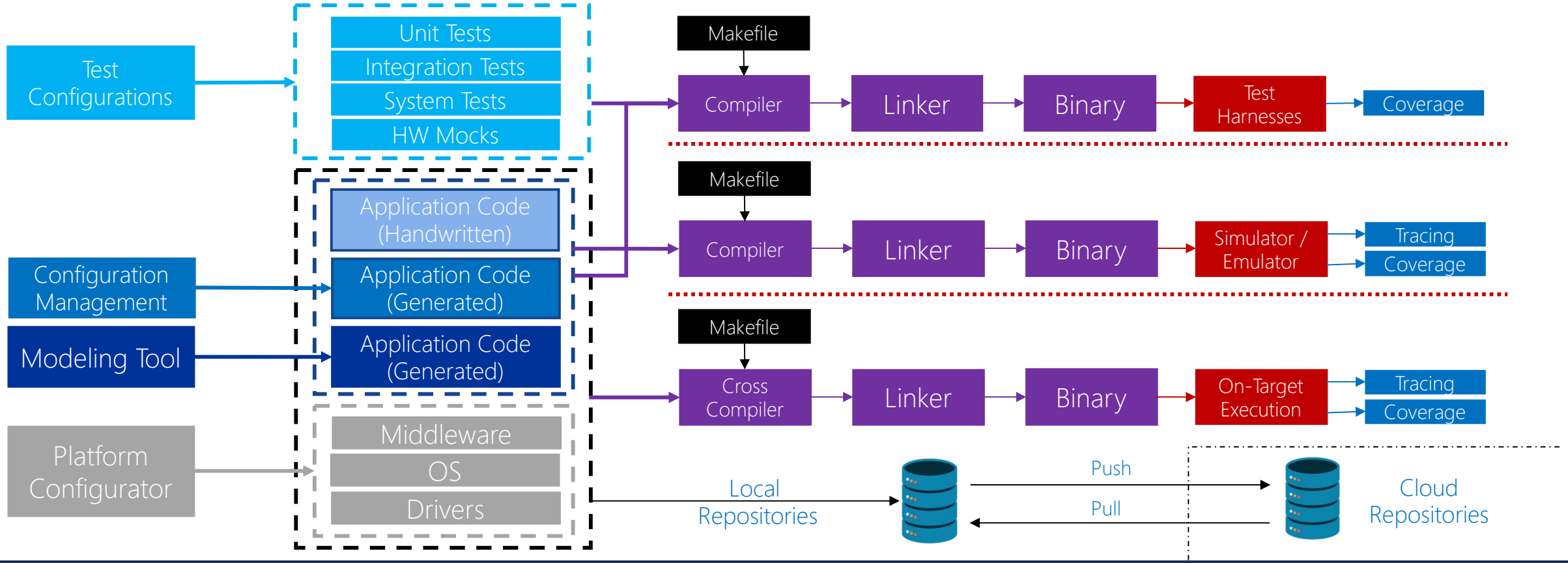
Modern Build Processes

MODERN BUILD PROCESSES

What is your vision of the ultimate build process?

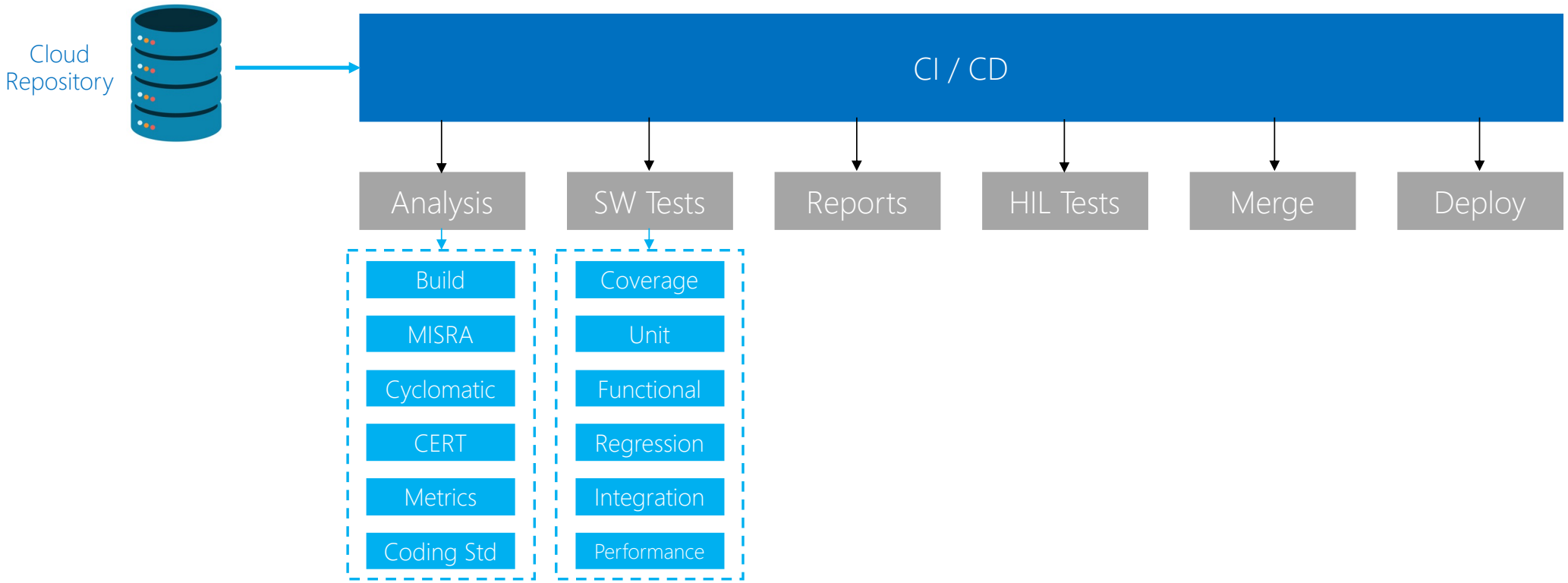
MODERN BUILD PROCESSES

An Example Modern Build Process



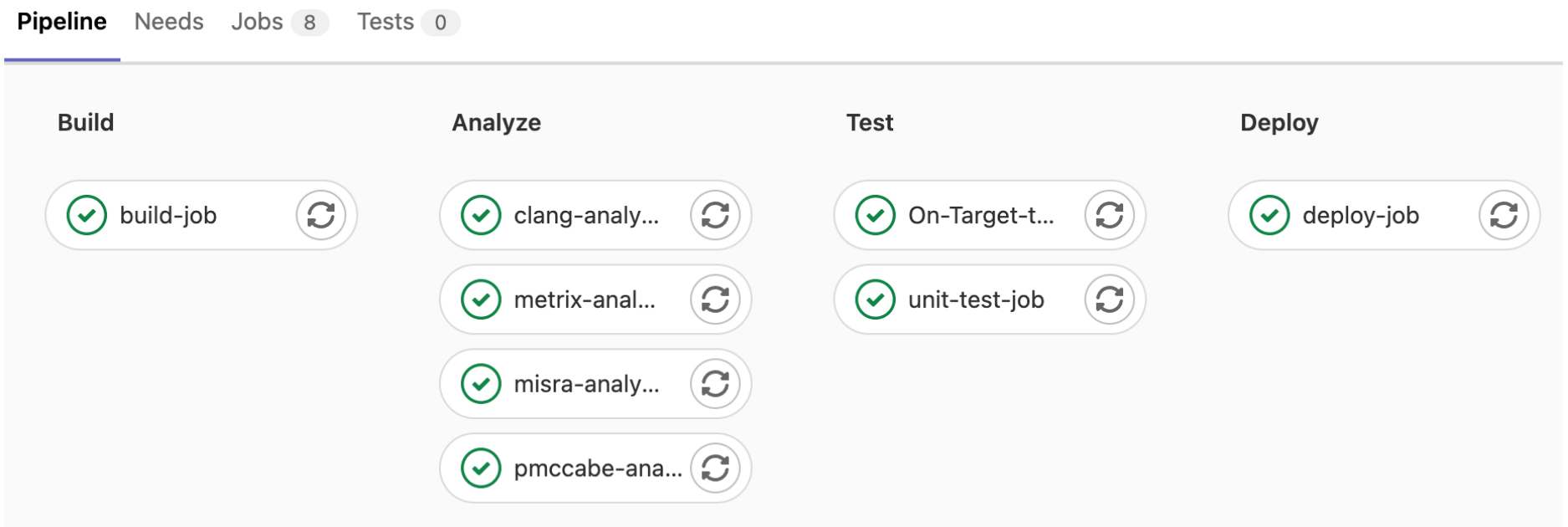
MODERN BUILD PROCESSES

CI/CD



MODERN BUILD PROCESSES

Example Starter Pipeline

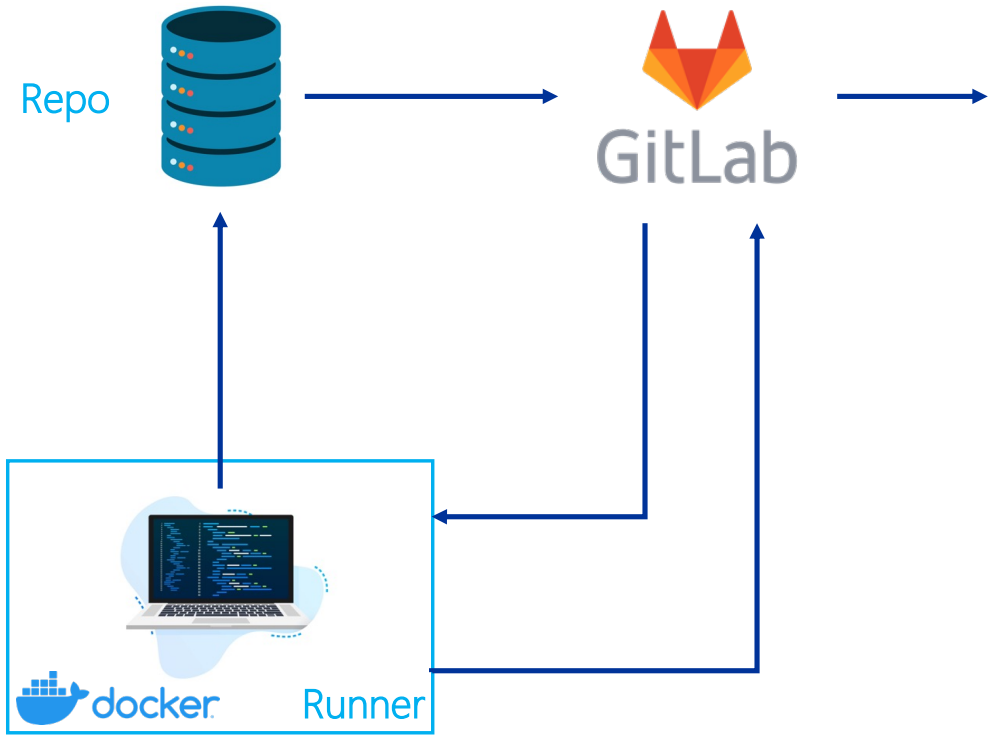


3

Docker-based Build System

DOCKER-BASED BUILD SYSTEM

With CI/CD



Status	Pipeline ID	Triggerer	Commit	Stages
passed	#417379990 latest		main 832d8075 Fixed build error.	

DOCKER-BASED BUILD SYSTEM

Example Dockerfile

```
FROM ubuntu:latest
ENV REFRESHED_AT 2023-03-24

# Download Linux support tools
RUN apt-get update && \
    apt-get clean &&  \
    apt-get install -y autoconf && \
    apt-get install -y libtool && \
    apt-get install -y \
        build-essential \
        wget \
        curl \
        git  \
    python3 python-setuptools python3 python3-pip python3-setuptools

# Set up a development tools directory
WORKDIR /home/dev
ADD . /home/dev

# Install the arm toolchain and C++ support
RUN apt-get install -y --no-install-recommends gcc-arm-none-eabi
ENV PATH $PATH:/home/dev/gcc-arm-none/bin
RUN apt-get install -y --no-install-recommends libstdc++-arm-none-eabi-newlib
```

```
# Download Static analyzer and formatting tools
RUN apt-get install -y clang clang-tools clang-format clang-tidy cppcheck
RUN pip install cpplint

# Metric analyzers and code coverage tools
RUN apt-get install -y pmccabe
RUN pip install metrixpp
RUN pip install gcovr

# Install the debugging tools
RUN apt-get install -y --no-install-recommends stlink-tools

# Install doxygen and graphviz for documentation
RUN apt-get install -y doxygen graphviz

# Clean up
RUN apt-get clean all
```

DOCKER-BASED BUILD SYSTEM

Dockerfile with Test Harness

```
# Install and configure CppUTest
WORKDIR /home/cpputest

RUN git clone --depth 1 --branch v4.0 https://github.com/cpputest/cpputest.git .
RUN autoreconf . -i
RUN ./configure
RUN make install

ENV CPPUTEST_HOME=/home/cpputest

# Set our working directory back to app
WORKDIR /home/app
```

3

Tuning Up Your Design Cycle

TUNING UP YOUR DESIGN CYCLE

Development is Dynamic not Static

A simple process:

1. Identify your ideal build, CI/CD, and processes
2. Audit where you are today
3. Define a roadmap to your ideal design cycle

Important Notes:

- Will require a lot of thought
- May require outside assistance
- Will be evolutionary
- Won't be accomplished in the short term

QUESTIONS

THANK YOU