# Figuring out the Testing of Configurable Software

Michael Stahl, Intel
Mar 2019

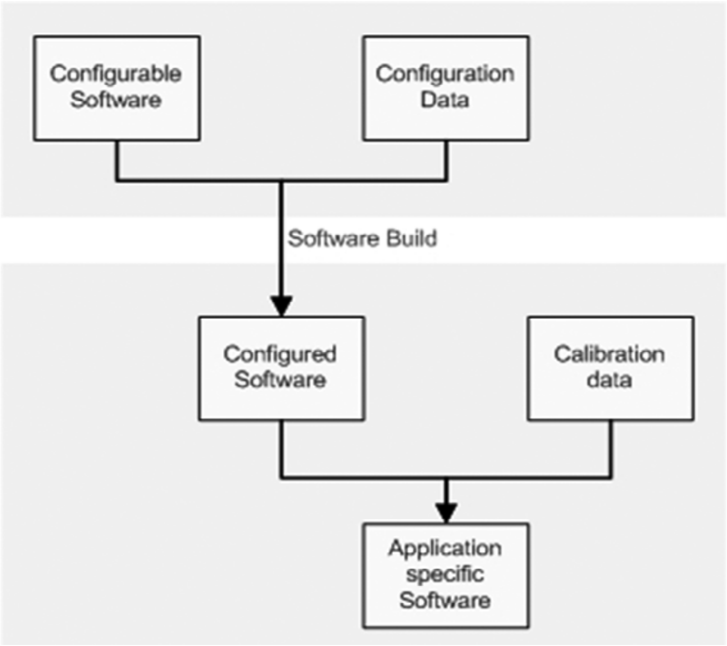
- Names and brands referenced herein may be claimed as the property of third parties
- The views expressed in this presentation are solely my own, and do not in any manner represent the views of my employer
- Information in this presentation is provided “AS IS” without any warranties or representations of any kind

# 1.

# ISO 26262, Annex C

Road Vehicles — Functional Safety
1st installment

# C.2 General




**Annex C** (normative) **Software configuration**

# 2.

# Terms

1st installment

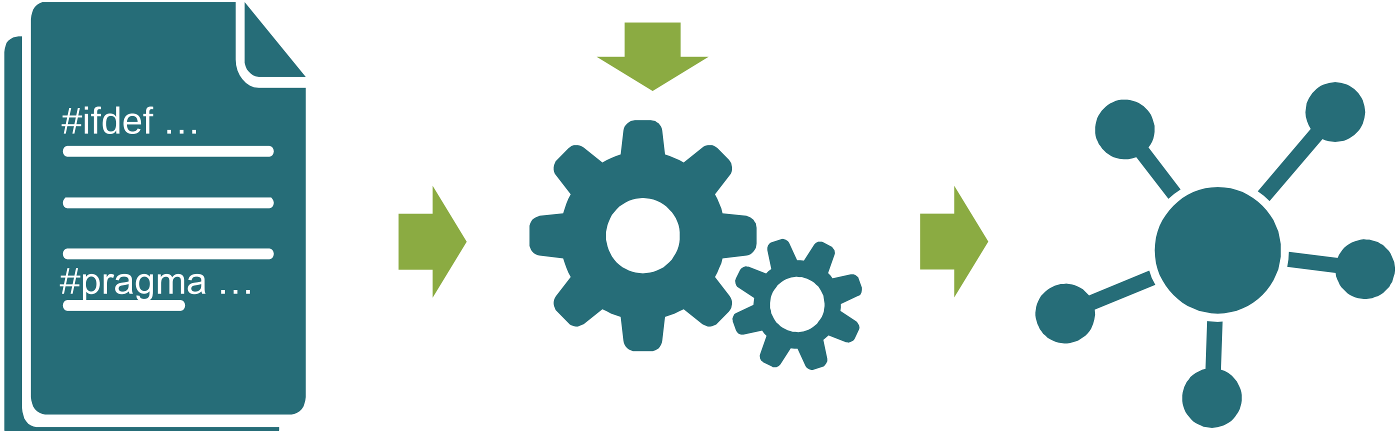
# Warning!

The standard's terminology won't always match to what you expect or are using!

## Configurable Software

## Configured Software

Build script parameters

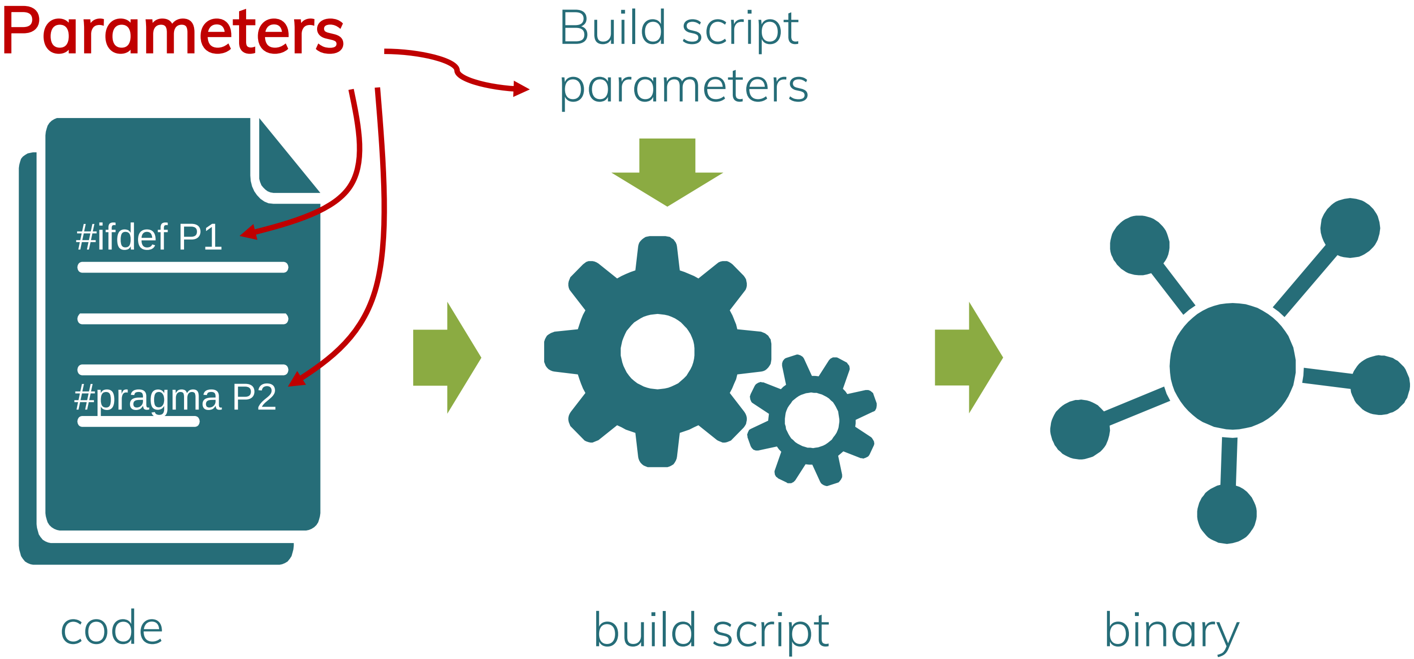


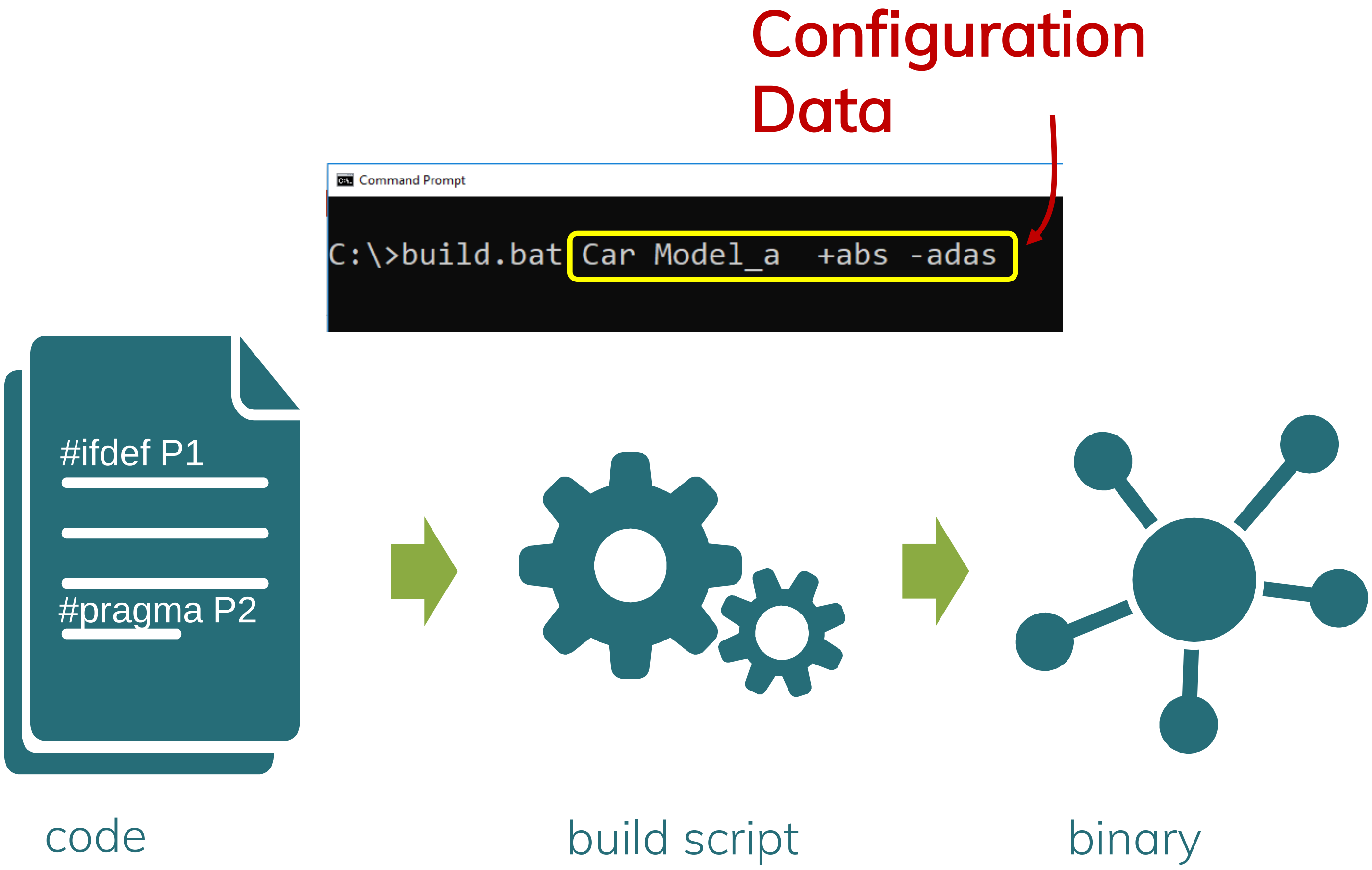

code

build script

binary

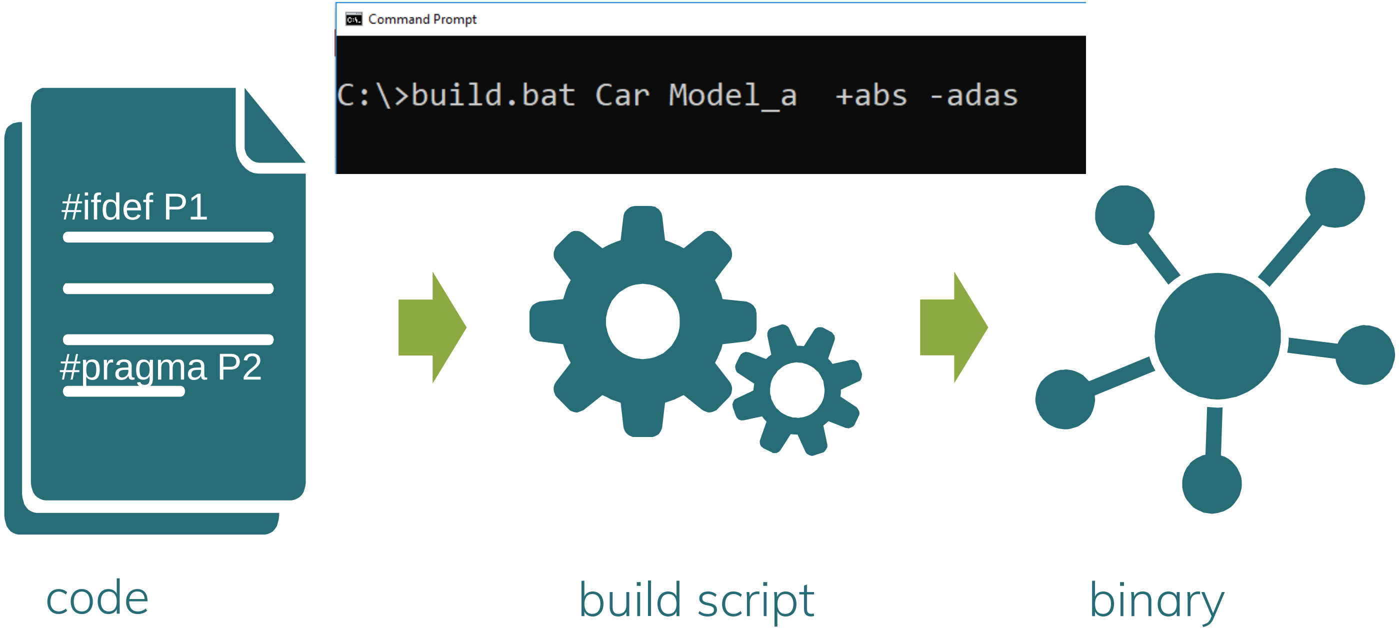
# Configuration Parameters



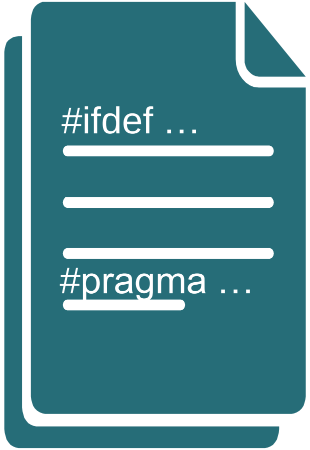

Configuration Data
Command Prompt
C:\>build.bat Car Model_a +abs -adas
#ifdef P1
#pragma P2
code
build script
binary

# Configuration Data:

Compiler version; Defines; Optimization directives; …



# Configurable Software

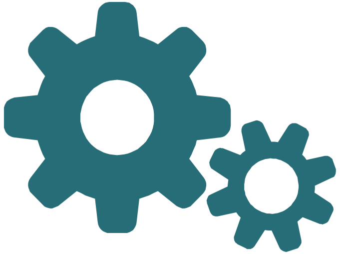



code

Build script parameters

Build script parameters

Build script parameters



build script

# Configured Software

binary

# 3.

# ISO 26262 Annex C

## 2nd Instalment

# C.4.5

"A combination of the following verification activities can achieve the complete verification of the configured software"

a. "verification of the configurable software",
b. "verification of the configuration data", and
c. "verification of the configured software".

# Verification (in ISO 26262)

ver·i·fi·ca·tion | \ ˌver-ə-fə-ˈkā-shən

1) Static testing (reviews; static analysis)
2) Dynamic testing

# C.4.5

"A combination of the following verification activities can achieve the complete verification of the configured software"

a. "verification of the configurable software", — Dynamic testing
b. "verification of the configuration data", and — Static testing
c. "verification of the configured software". — Dynamic testing

## Acceptable options:

- (a) + (b)
- (b) + (c)

## Remember!

The discussion is about safety-relevant requirements and features*

* Largely applicable to non-safety-relevant features

# Example: Cruise Control Module

- Configuration parameters:
  - Legacy Cruise Control (CC)
  - Adaptive CC (ACC)
  - Speed limit (SL)
- Configuration Data = Y/N
  - ⇨ Eight possible configurations

| Legacy | Adaptive | Speed Limit |
|---|---|---|
| N | N | N |
| N | N | Y |
| N | Y | N |
| N | Y | Y |
| Y | N | N |
| Y | N | Y |
| Y | Y | N |
| Y | Y | Y |

Build command:

```
Build [CC] [ACC] [SL]   (default: CC)
```

# Example: Cruise Control Module

## Release 12.0.0.3 supports CC+SL

Option A:

a) Test the module with each of the combinations

b) Verify that the parameters used for version 12.0.0.3 are CC+SL

Eight versions

Option B:

b) Verify that the parameters used for version 12.0.0.3 are CC+SL

c) Test version 12.0.0.3

One version

# When to use what?

(a)+(b) seems to **always** call for more work.

Why would you **ever** want to go that way?

# When to use what?

Option (a)+(b):

- Configuration data is a **range**
  … and you have many versions of the binary
- Open Source; Code delivery

Option (b)+(c): Binary delivery

- All other cases

The rest of this presentation assumes Option (b) + (c)

# 4.

# Test Strategy

## Configuration Parameters

# What are we validating?

- **Validity**
  - Valid values are accepted
  - Invalid values rejected (the binary is not built)

- **Functionality**
  - The resulting binary and functionality is as required

# Parameter Data Validity

- **Validity**
  - All parameters assigned valid values

Static testing

- **Error protection measures**
  - Invalid values or invalid combinations are rejected by the build script

Dynamic testing

Static testing

- **Build System Change Control**
  - Prevent unintended changes to the parameters values

Static testing

# Functionality

If parameter A enables feature X

- ☐ X exists when A is set
- ☐ X does not exist when A is not set
- ☐ X is functional when it exists

X is tested dynamically on the compiled binary

The binary (may) use **Calibration Parameters** that are set with **Calibration Data**

# 5.

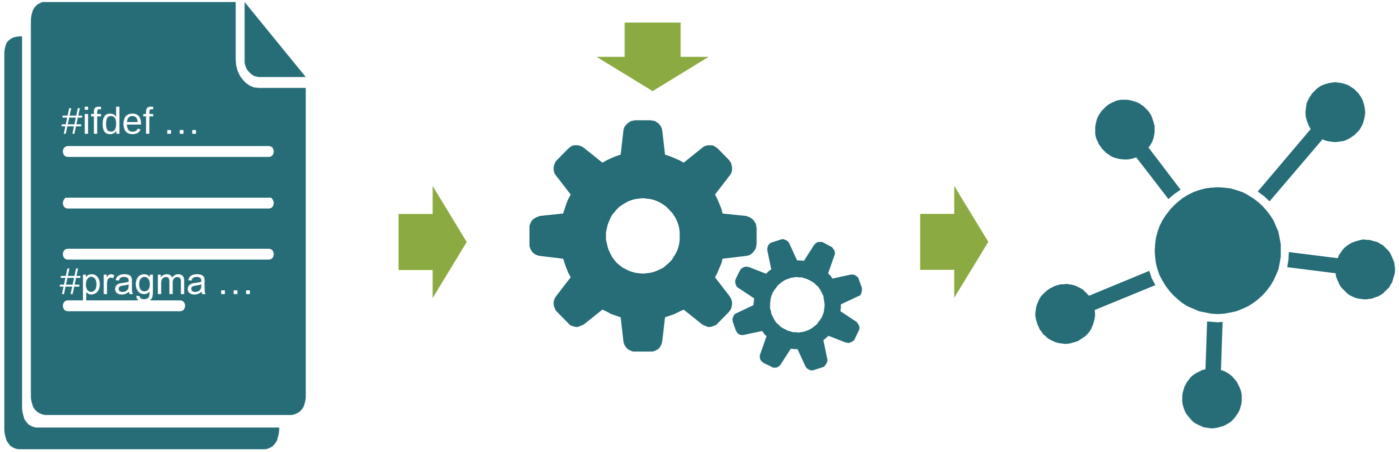
# Terms

## 2nd Instalment

## Configurable Software

Build script parameters

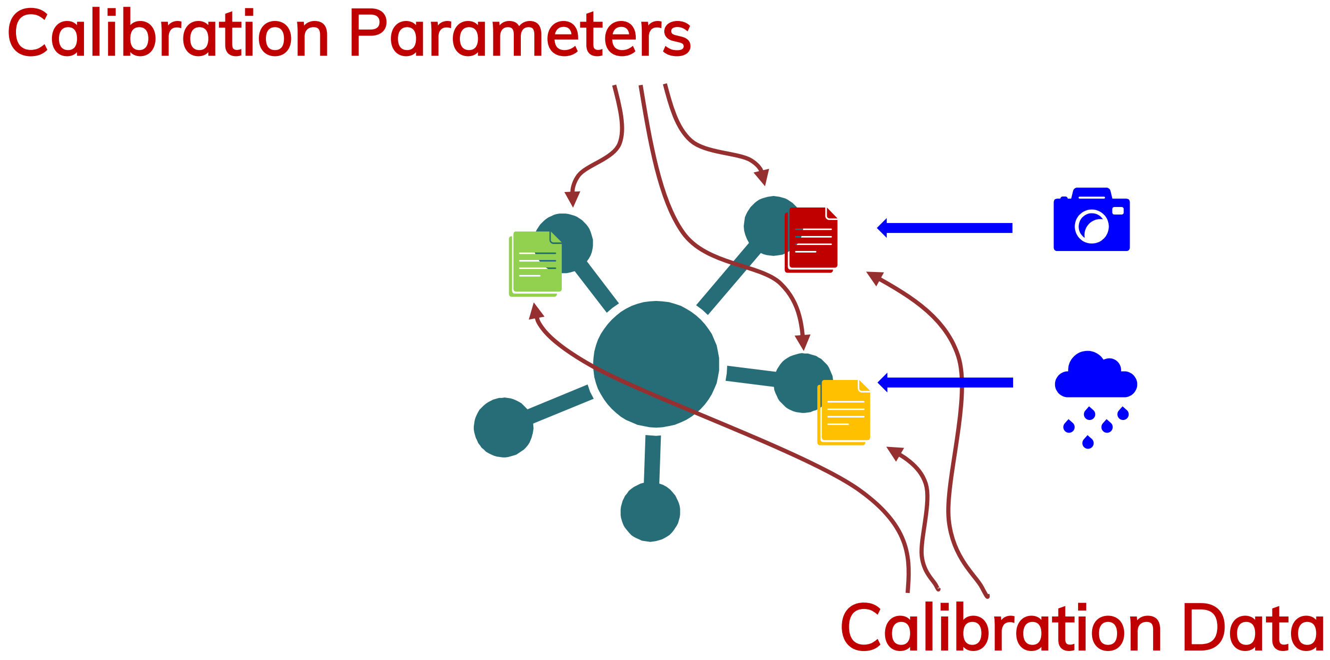
## Configured Software



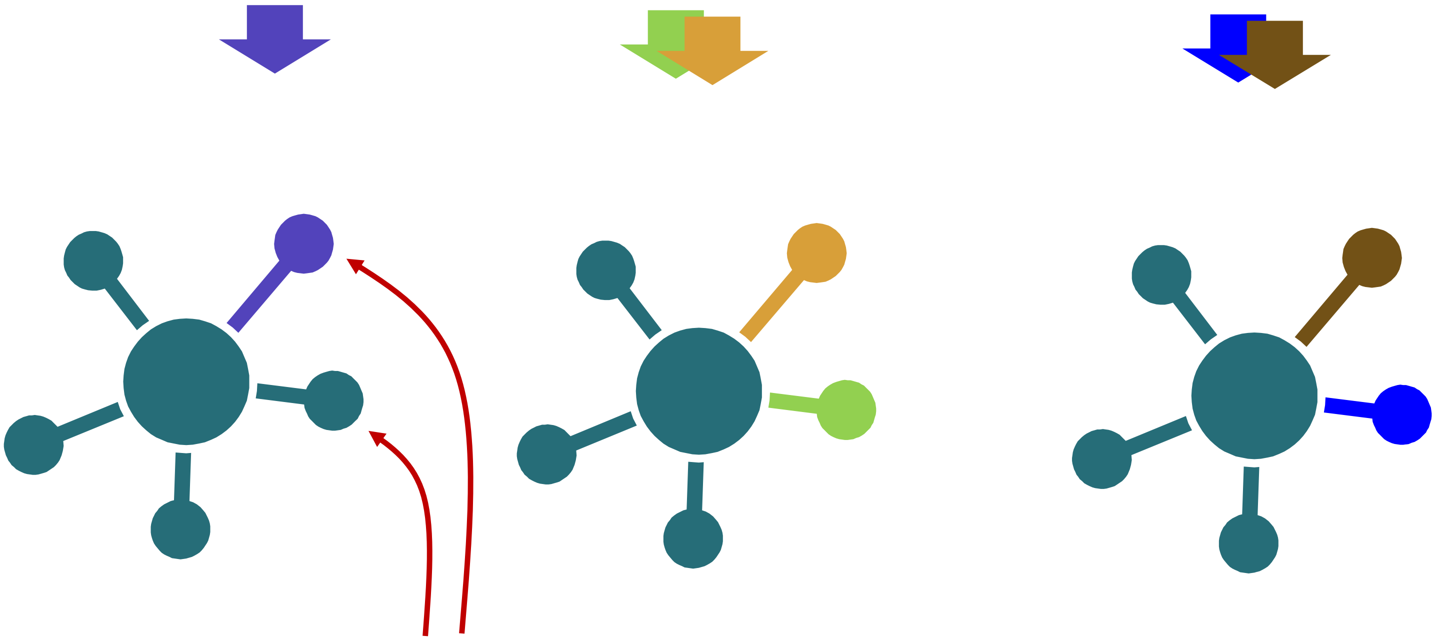

code

build script

binary


Calibration Parameters
Calibration Data

## Calibration Data



## Calibration Parameters

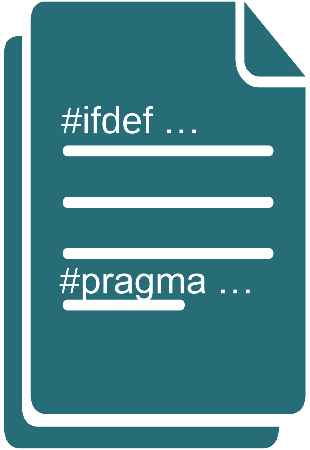
## Configurable Software

Build script parameters

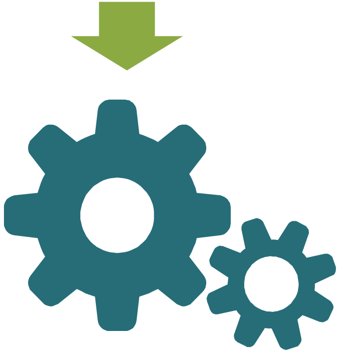
Build script parameters

Build script parameters

## Configured Software

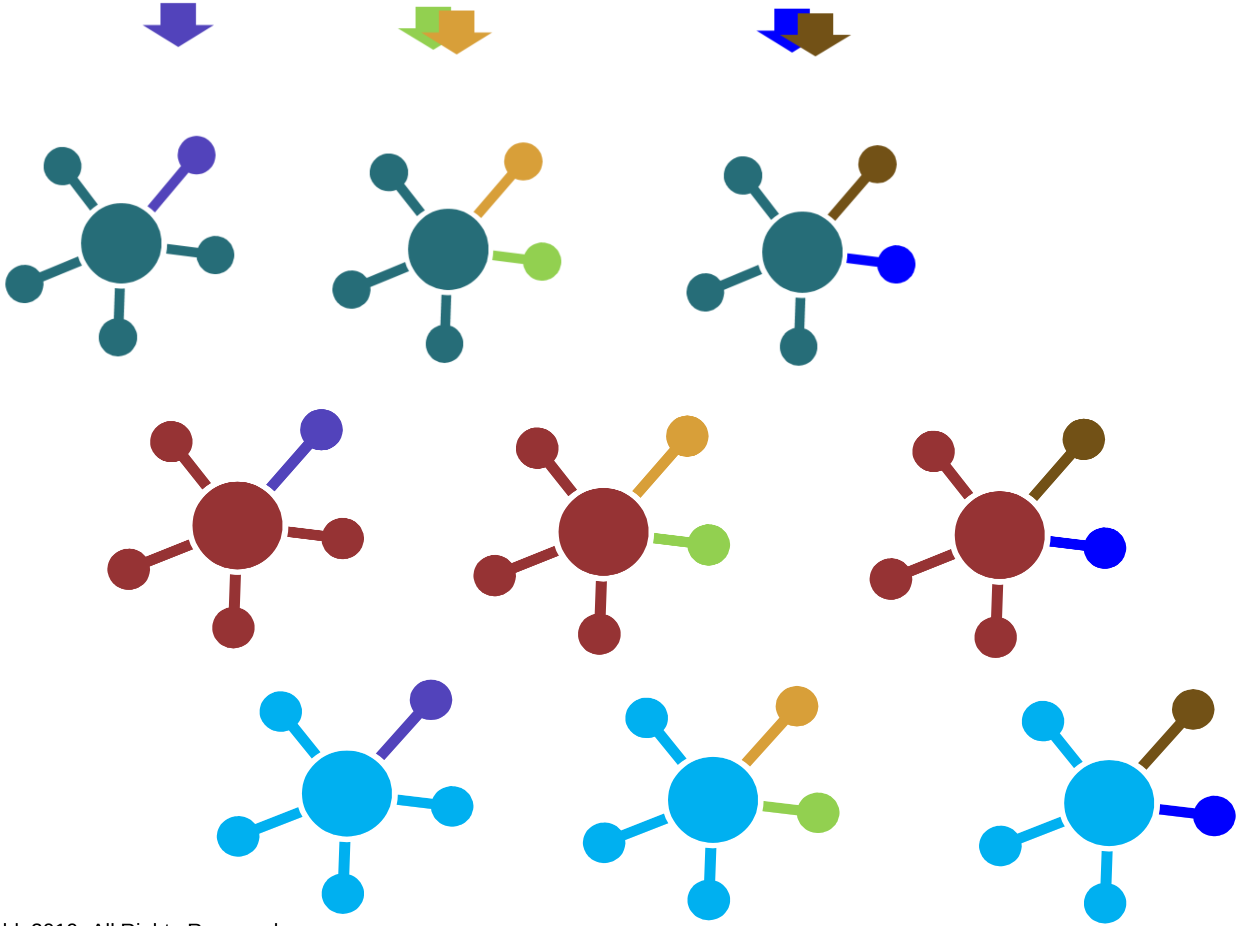


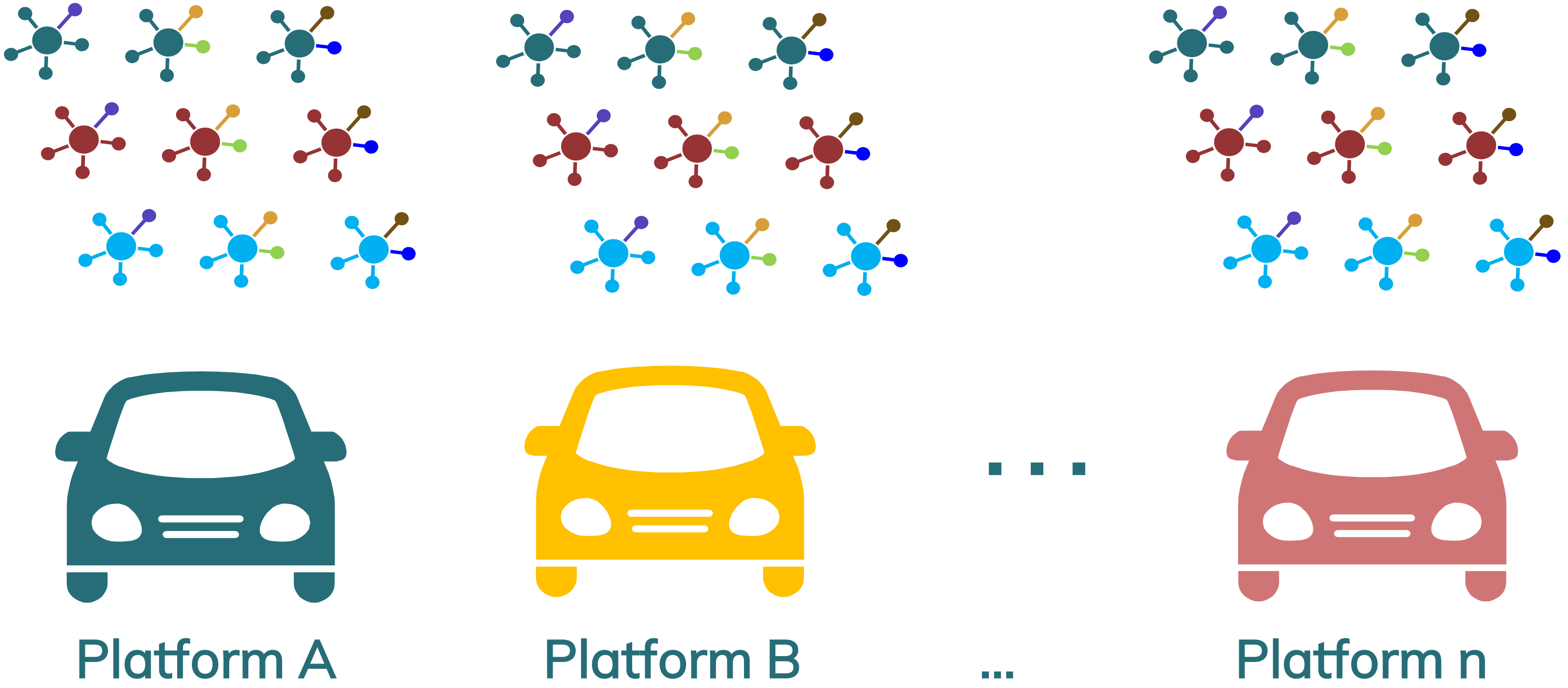

code

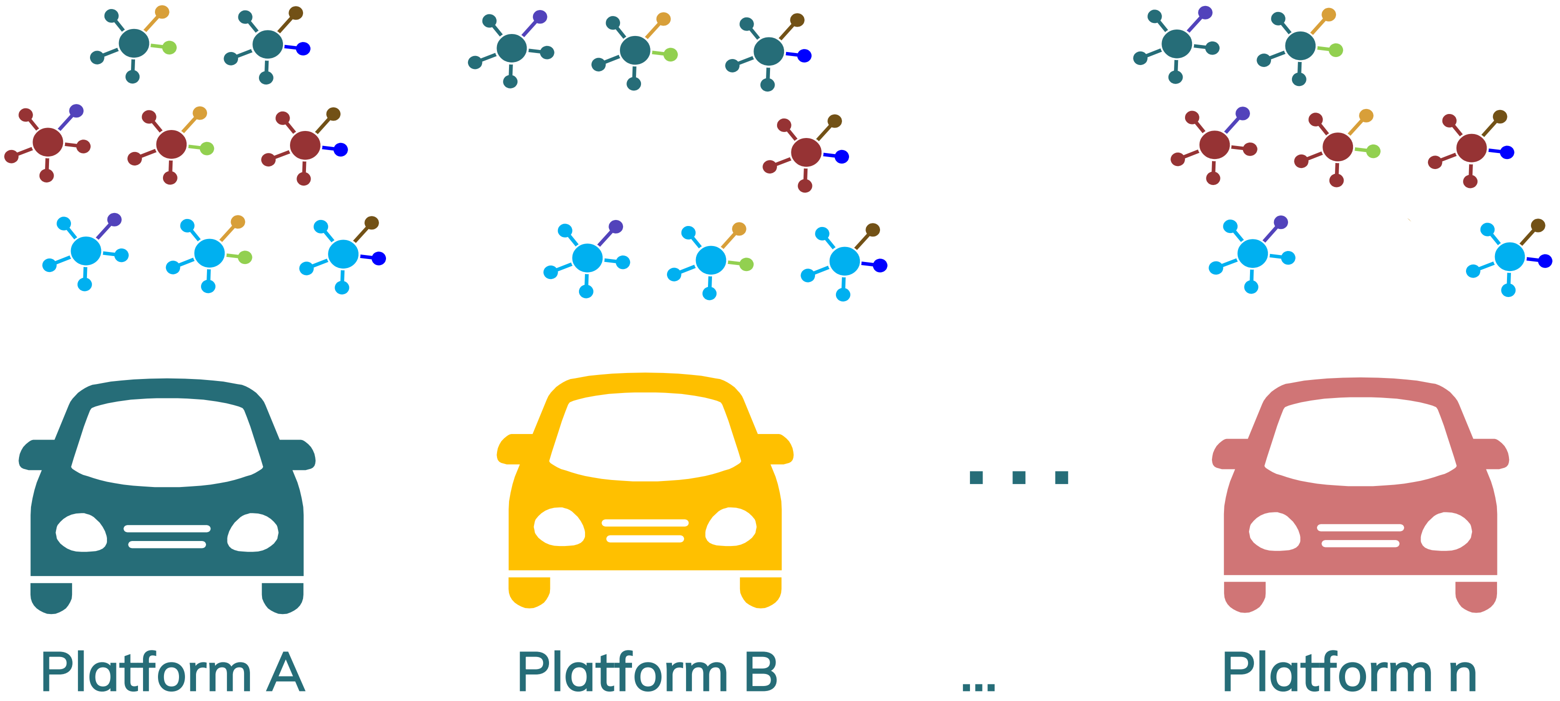


build script

binary




Platform A
Platform B
...
Platform n


Platform A
Platform B
...
Platform n

# 6.

# Analysis of configurations

Selecting the HW-SW combinations to test

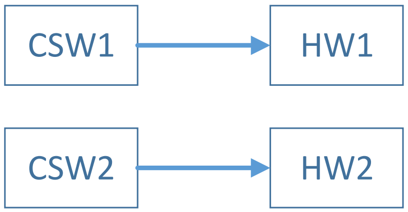
# Platform-dependent Configuration Parameters

- Take different values based on the HW
- For a given HW, a parameter may have:
  - Single viable value
  - Multiple viable values

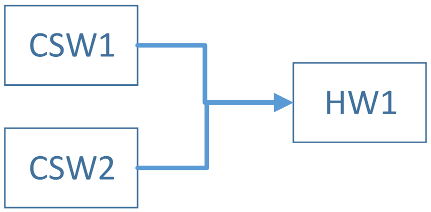
- Each set of configuration parameters creates a different "Configured Software" (CSW)
- A specific CSW may fit a number of HW configurations

# Types of SW-HW configurations

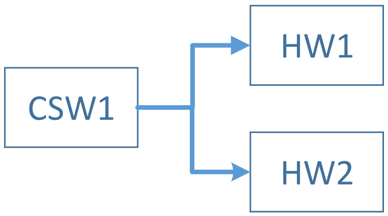
**1 to 1**



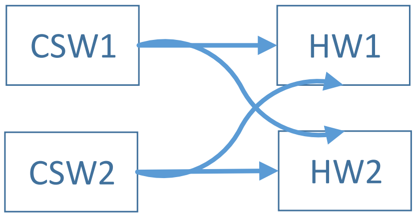

**Many to 1**



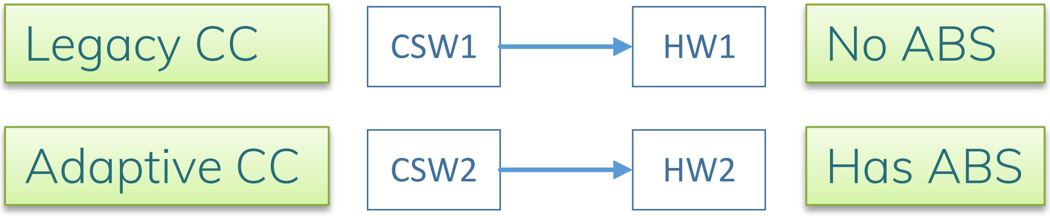

**1 to Many**




**Many to Many**

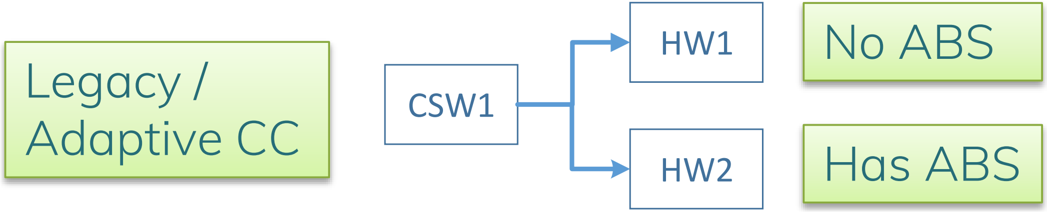


# 1 to 1

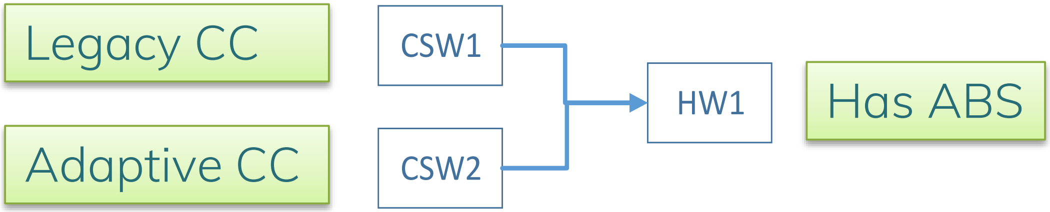



CC = Cruise Control

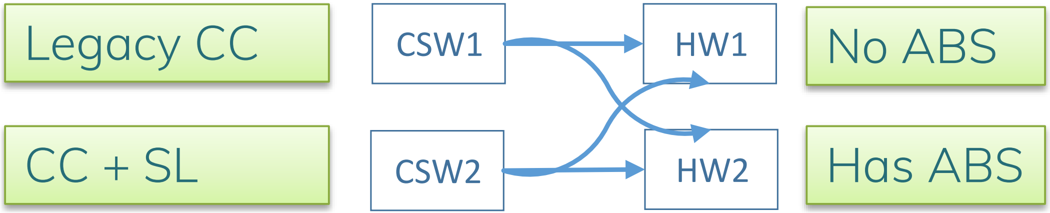
# 1 to Many



# Many to 1

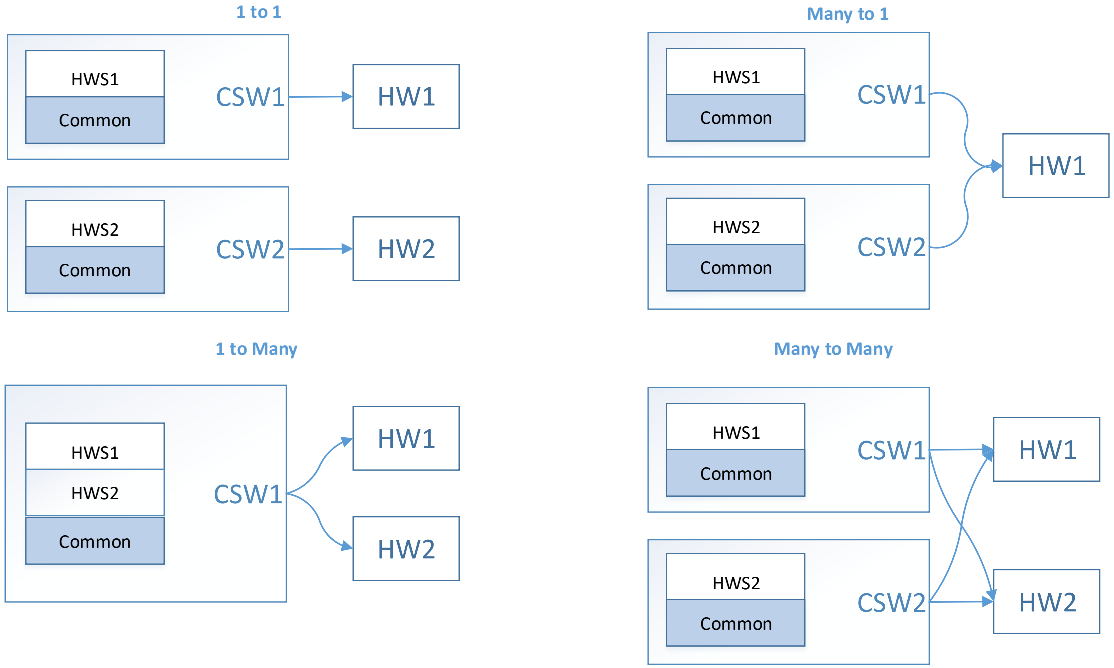


# Many to Many



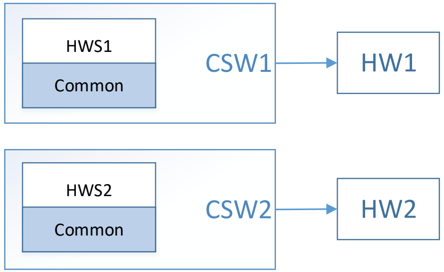

CC = Cruise Control
SL = Speed Limit

# Analysis of SW-HW configurations

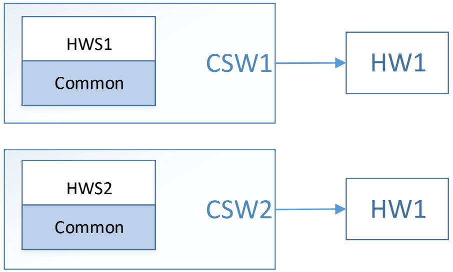



HWS = HW-Specific

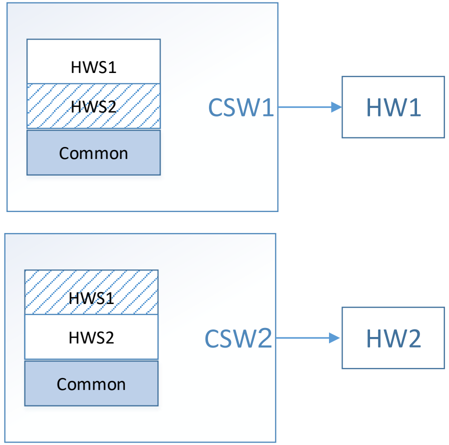
## 1 to 1



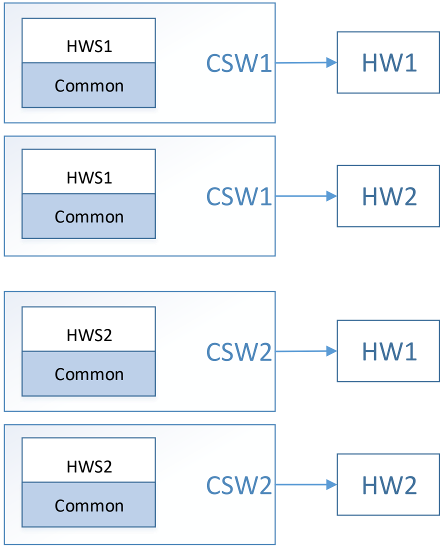

## Many to 1



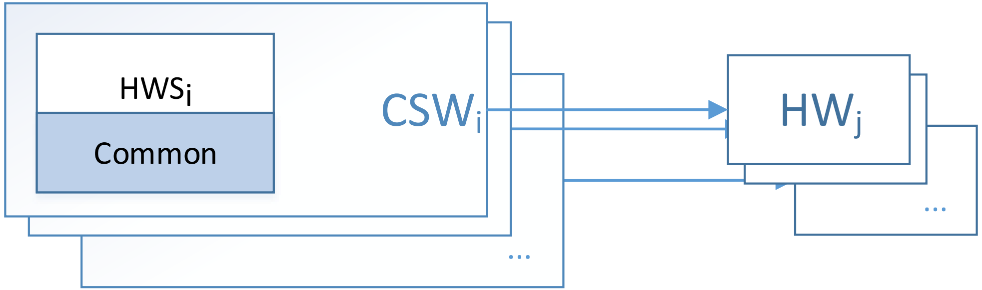

## 1 to Many




## Many to Many



# Analysis of SW-HW configurations



# 7.

## Lead Configuration Selection

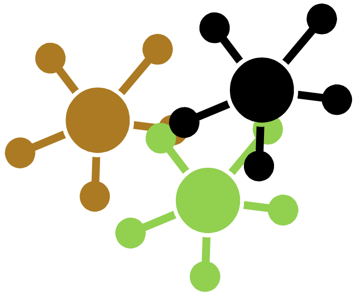
# Lead Configuration selection

**For each Safety feature:**

Split the **Configured SW** versions to two groups

**SW group A**

Supports the feature



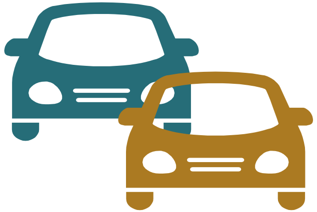
**SW group B**

Does not support the feature

# Lead Configuration selection

**For each Safety feature:**

Split the HW versions to two groups



**HW group A**

Supports the feature

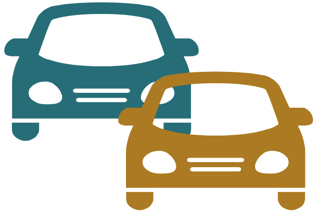
**HW group B**

Does not support the feature

# Lead Configuration selection

**For each Safety feature:**

Split the HW versions to two groups



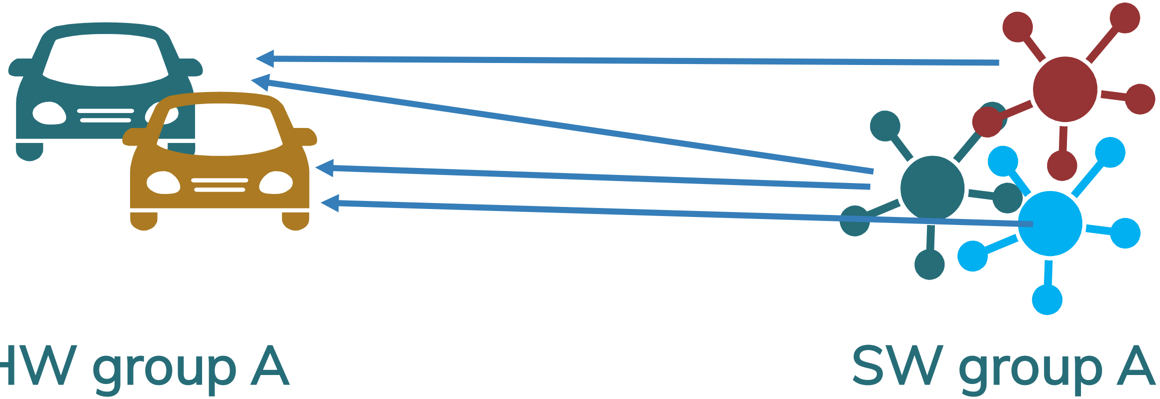
## HW group A

Fits at least one binary of SW group A

## HW group B

Fits no binary from group A

# Lead Configuration selection



# Lead Configuration selection

HW group B

SW group A

# Lead Platform selection

## Prioritize:

- Full feature support over partial
- Expected market-share leader
- Worst-case for the feature-under-test
  - e.g. Time constrained feature => slowest platform
  - e.g. Positioning feature => platform with worst sensors

# Lead Platform selection

## Considerations may contradict each other

- May need more than a single “Lead Platform”
- Can switch Lead Platform between test cycles

## Performance requirements may need to be tested on each platform

# Lead Binary selection

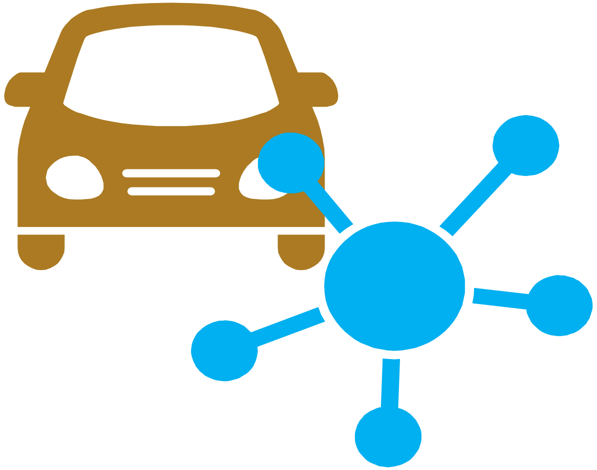
## Prioritize:

- Full feature support over partial (on the Lead Platform!)
- Expected market-share leader
- Most diverse support of Calibration parameters affecting the feature-under-test (number of params; range)

## Considerations may contradict each other

- May need more than a single “Lead Binary”
- Can switch Lead Binary between test cycles

# Result: Lead Configuration

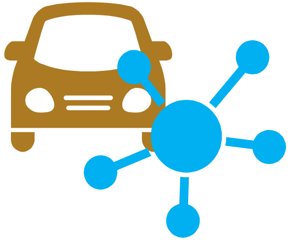


# 8.

## Test Strategy

### Calibration Parameters

# The Role of the Lead Configuration



## Strategy

- Most testing is done on the Lead Configuration
- The results are considered applicable to all configurations

# The Role of the Lead Configuration

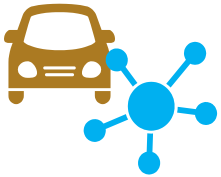
## Assumptions / Justification

- The Lead Binary is a superset of the other binaries code
- The Lead Platform is a superset of the other HW options
- Testing covers the full range of config & calib values

⇒ You may need more than a single Lead Configuration

⇒ Must document the selection rationale

⇒ Must document how config & calib values are covered



# What are we validating?

- **Validity**
  - Valid values are accepted
  - Invalid values rejected properly

- **Runtime validity**
  - Ensure the validity of values before using them

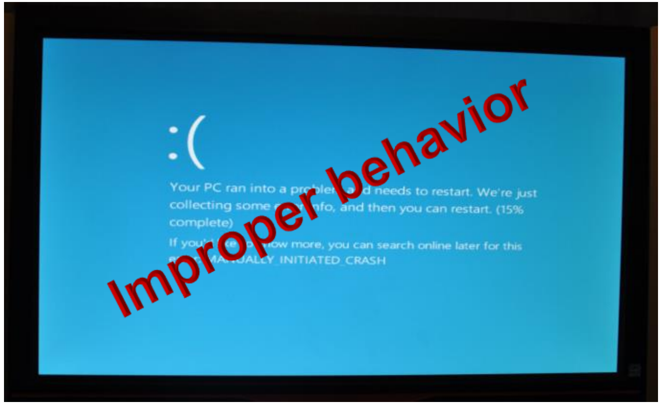
- **Functionality**
  - The parameters affect program behavior as required

# What are we validating?

- **Validity**
  - Valid values are accepted
  - Invalid values rejected properly
- **Runtime validity**
  - Ensure the validity of values before using them
- **Functionality**
  - The parameters affect program behavior as required

# Parameter Data Validity

- Dynamic testing
  - Equivalence Class Partitioning
  - Boundary Value analysis
- Test both valid and invalid values; combinations
- Ensure **proper behavior** for invalid cases



# What are we validating?

- Validity
  - Valid values are accepted
  - Invalid values rejected properly
- Runtime validity
  - Ensure the validity of values before using them
- Functionality
  - The parameters affect program behavior as required

# Runtime Validity Checks

## Unique to Functional Safety!

**Table 17 — Mechanisms for the detection of unintended changes of data**

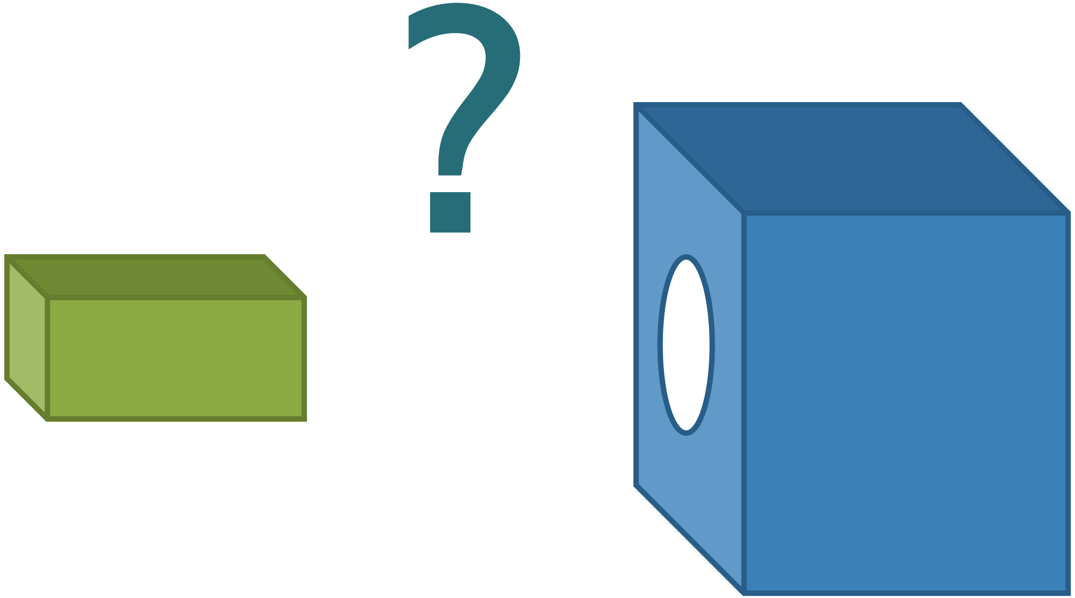
| | **Mechanisms** | |
|---|---|---|
| 1a | Plausibility checks on calibration data | Recommended! |
| 1b | Redundant storage and comparison of calibration data | |
| 1c | Calibration data checks using error detecting codes [a] | |

[a] Error detecting codes may also be implemented in the hardware in accordance with ISO 26262-5:2018.

ISO 26262:6-2018, Annex C.4.10

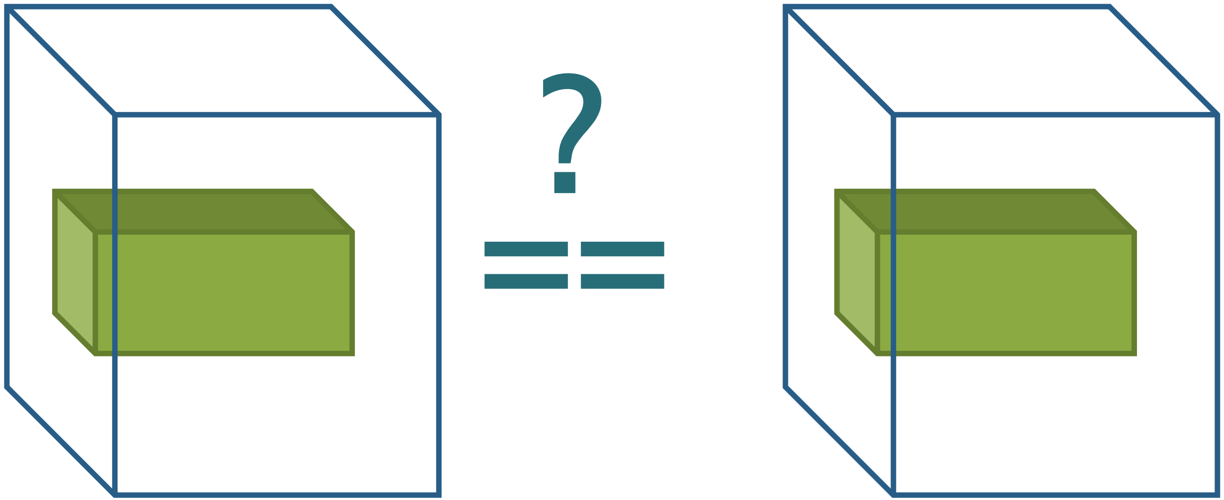
# Runtime Parameter Data Validity

## Plausibility checks



# Runtime Parameter Data Validity

## Redundant storage



# Runtime Parameter Data Validity

## Error detection mechanisms

# Runtime Parameter Data Validity Test Techniques

**General approach:**

- ☐ Access the calibration parameter prior to use
- ☐ Change it to plausible or non-plausible value
- ☐ Verify correct behavior

- **Debugger:** Breakpoint – modify param value
- **Test code:** Change param values at known time or on event
- **Direct Memory access:** Modify params in memory

# What are we validating?

- Validity
  - Valid values are accepted
  - Invalid values rejected properly
- Runtime validity
  - Ensure the validity of values before using them
- Functionality
  - The parameters affect program behavior as required
  - Functional, non-functional, stress, load, regulatory, etc. etc.

# Functionality

## Independent Calibration Parameter

- **Full validation on the Lead Configuration**
  - All values for enumerated parameters
  - EC / BV for ranges

- **"Touch testing" on all the other binaries in SW group A**
  - Show the feature is alive
  - Calibration parameter set to expected "popular" values
  - Lead Platform or any other HW that supports the feature

- **Basic negative test**
  - Ensure proper behavior when calling the non-existing feature
  - Run on each of the binaries is SW group B
  - Use any HW platform that fits the tested binary

# Functionality

## Dependent Calibration Parameters

- **Define the combinations to cover**
- **Select the Lead Combinations: most important due to**
  - Worst-cases; Expected Popularity; Etc.
- **On the Lead Configuration**
  - Full testing on the Lead Combinations
    - Including interaction between features
  - Touch testing on the less important combinations
- **On non-lead configurations**
  - Touch testing on the Lead Combinations
  - Range and validity check on the less important combinations
- **Basic negative test**

# Done!...

Almost…

# Let's Summarize…

# Disclaimer

This was MY interpretation of the standard

Backed by a few expert reviewers

…but certainly not an official view

## Remember!

The discussion is about safety-relevant requirements and features*

* Largely applicable to non-safety-relevant features

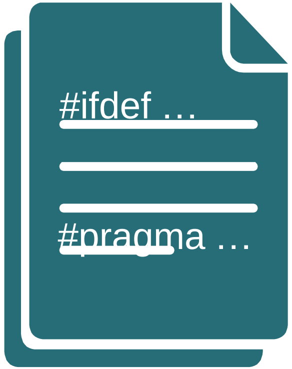
Software Code

+

Configuration Parameters

=

Configurable Software



Configurable Software

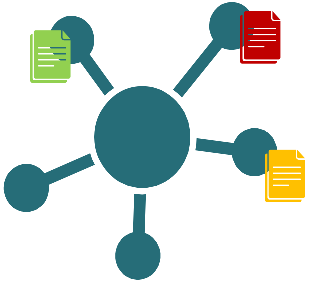
+

Configuration Data

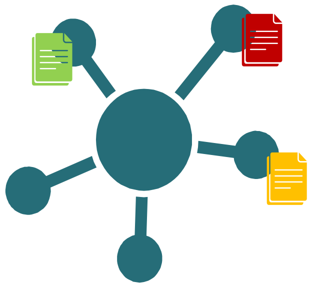
=

Configured Software

# Configured Software (may) have ______________ Parameters



# Configured Software (may) have Calibration Parameters



Configured Software

+

Calibration ___________

=

Specific SW Application

Configured Software

+

Calibration Data

=

Specific SW Application

There are

—

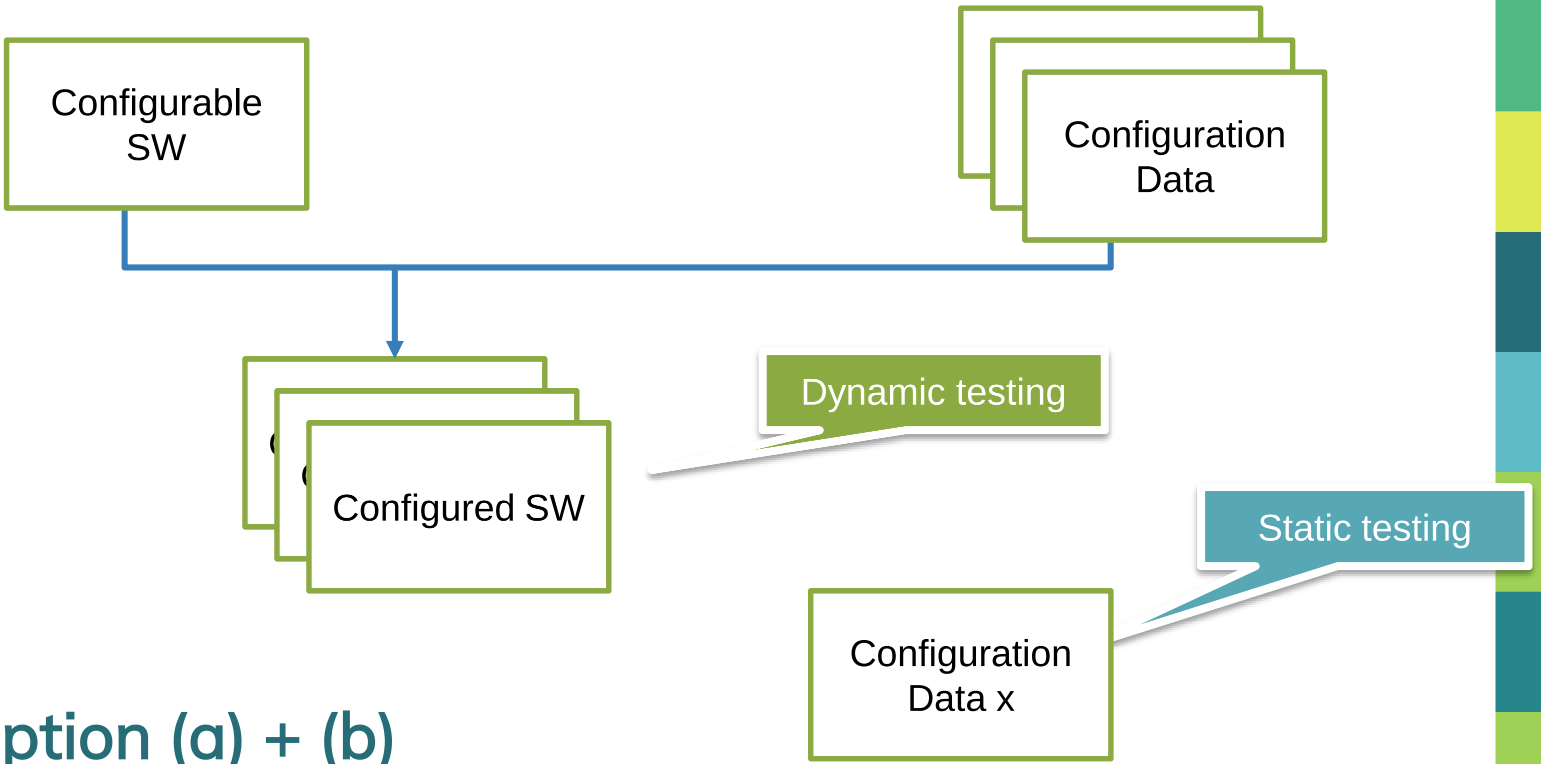
options to verify the **configurable software**

There are

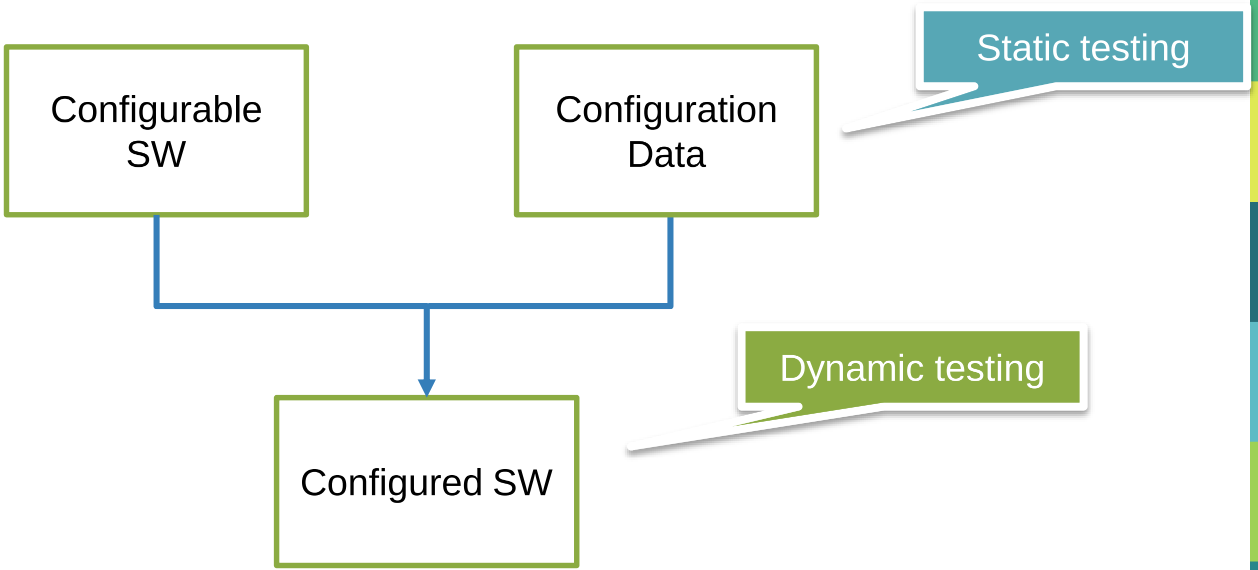
2

options to verify the

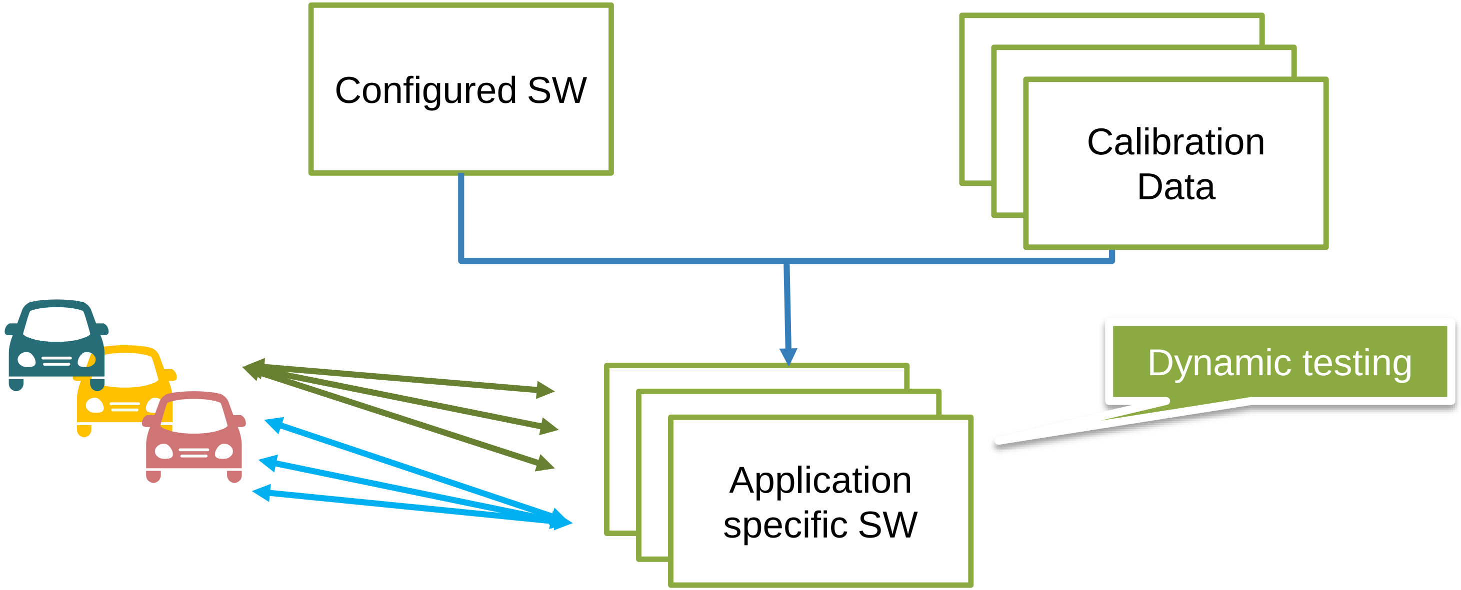
**configurable software**


Configurable SW
Configuration Data
Dynamic testing
Configured SW
Static testing
Configuration Data x

# Option (a) + (b)


Configurable SW
Configuration Data
Static testing
Dynamic testing
Configured SW

# Option (b) + (c)

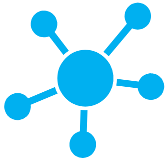



# Both options

# Select

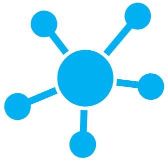
&

______ ______



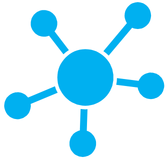
# Select

Lead Platform & ____________



# Select

Lead Platform & Lead Binary

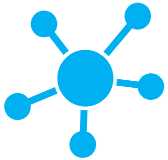


to make the

____________________

Select

Lead Platform & Lead Binary



to make the

Lead Configuration

# What is tested?

Configuration Parameters

V__________

E_________ P__________

# What is tested?

Configuration Parameters

**Validity**

E________ P

# What is tested?

Configuration Parameters

Validity
Error Protection

Calibration Parameters

V________
R________ V________
F_______________

# What is tested?

Configuration Parameters

Validity

Error Protection

Calibration Parameters

Validity

R________ V_________

F________________

# What is tested?

| Configuration Parameters | Calibration Parameters |
| --- | --- |
| Validity | **Validity** |
| Error Protection | **Runtime Validity** |
| | F_____________ |

# What is tested?

## Configuration Parameters

Validity

Error Protection

## Calibration Parameters

Validity

Runtime Validity

**Full Validation**

(functional, non-functional, stress, load, regulatory, etc. etc.)

# Where do we test?

Mostly on the L_____ C__________

# Where do we test?

Mostly on the Lead Configuration

Less on the other HW-SW combinations

# Now we are really Done!

# Thanks!

**Any questions?**

You can find me at

- michael.stahl@intel.com
- LinkedIn: michaelmstahl

You can find this presentation + paper at www.testprincipia.com

# Credits

Thanks **Simone Fabris**, **Maurizio Iacaruso**, **Gabriele Paoloni** and **Itamar Sharoni**, from Intel and Mobileye, for their invaluable advice and review while writing this paper.

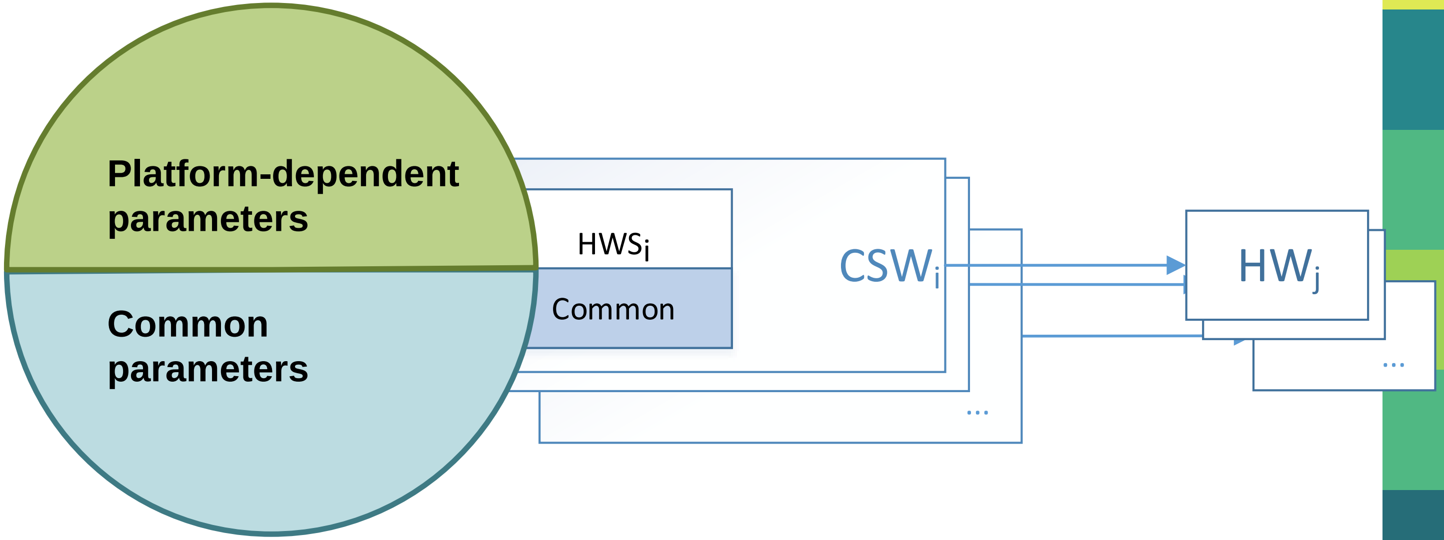
Special thanks to all the people who made and released these awesome resources for free:

- Presentation template by SlidesCarnival
- Photographs by Unsplash

# Backup

## Static, Independent and Dependent Configuration Parameters

# Analysis of SW-HW configurations



# Static Parameters

- **#pragma** directives that are always used
- **#ifdef** clauses that are always True (or always False)
- A build-script parameter that is used with only a single value
- Calibration parameters that get only one value

**Platform selection:** Ignore these

**=>** Test on the Lead Platform

# Platform Dependent Parameters

**Assumption:** Per-feature, the configured SW versions are more-or-less the same:

- Much is common
- Some differences to accommodate different HW platforms

**Platform selection:** The differences are what influence the Lead Platform selection

**=>** Test on the Lead Platform

If the assumption is wrong, treat each unique implementation of the feature as a separate feature

# Common Parameters

**Any** of the SW-HW combinations (from groups A) are good for testing these parameters

**Platform selection:** Any platform that supports the feature

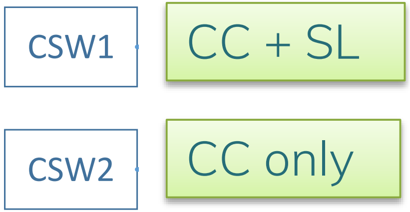
**=>** Test on the Lead Platform

# Interaction Between Parameters

**Independent parameter:** The impact on the Configured SW is the same, regardless to other Config Params

```
C:\>build.bat cc ON speedLimit ON
```

```
C:\>build.bat cc ON speedLimit OFF
```



# Interaction Between Parameters

**Independent parameter:** The impact on the Configured SW is the same, regardless to other Config Params

**Platform selection:** Any platform that supports the feature

**=>** Test on the Lead Platform

# Interaction Between Parameters

**Dependent parameter:** The impact on the Configured SW is dependent on other Config Params

```
C:\>build.bat cc ON abs ON
```

CSW1 → Adaptive CC

```
C:\>build.bat cc ON abs OFF
```

CSW2 → CC only

# Interaction Between Parameters

**Dependent parameter:** The impact on the Configured SW is dependent on other Config Params

**Platform selection:**

- Create ALL CSW permutations
- Preferred platform will support all permutations

**=>** Test on the Lead Platform

**Assumption:** In the real world, there won't be *that* many CSW's

Since multiple variations are easier to do with Calibration Parameters