RGB ASSURANCE

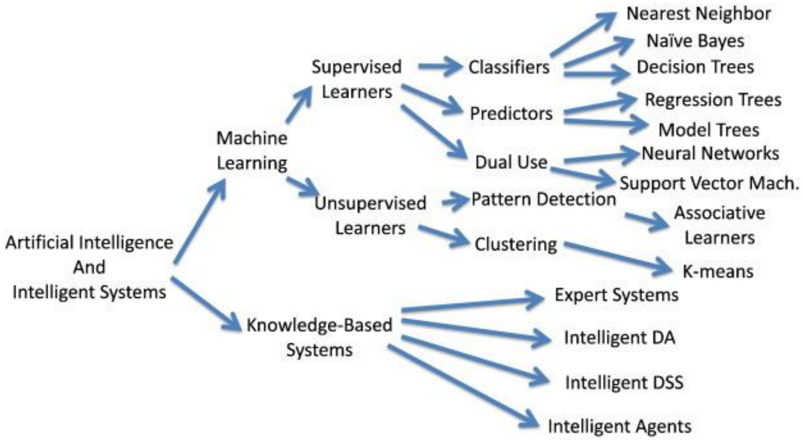
Artificial Intelligence and Safety Standard Compliance: Challenges

Tim McComb
RGB Assurance

ASSC’19
24th May 2019

Scope

- Safety function demanded of AI



Artificial Intelligence And Intelligent Systems

Machine Learning

Supervised Learners

Classifiers

Nearest Neighbor

Naïve Bayes

Decision Trees

Predictors

Regression Trees

Model Trees

Dual Use

Neural Networks

Support Vector Mach.

Unsupervised Learners

Pattern Detection

Associative Learners

Clustering

K-means

Knowledge-Based Systems

Expert Systems

Intelligent DA

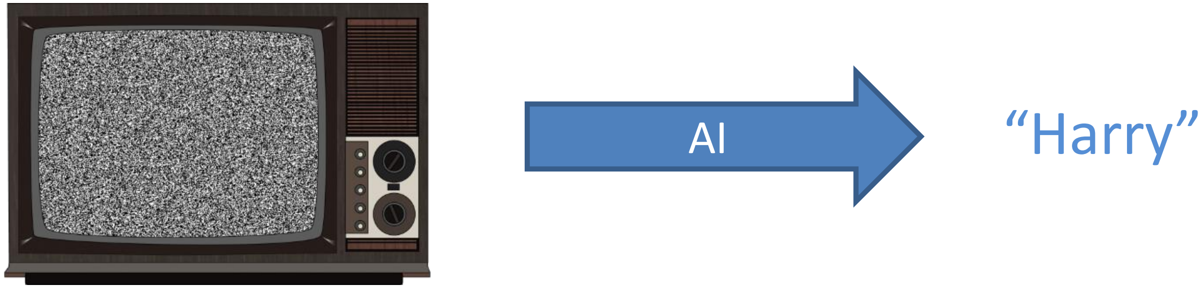
Intelligent DSS

Intelligent Agents

- No “safety bag”

AI Reliability

- Measured accuracy of the model against the validation data, e.g. “99.99% accurate”
- The environment is not random, and the behaviour is systematic
  - – Random process → Reliability prediction
  - – Systematic process → Lifecycle rigour



AI

“Harry”

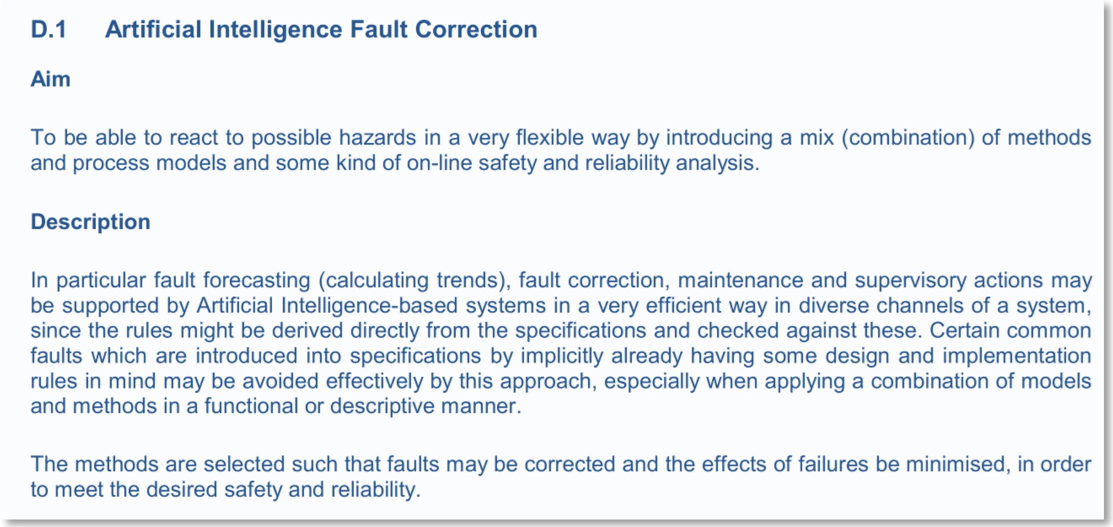
Lifecycle Rigour

- DEF STAN 00-55/6
  - AI not called out
- RTCA DO-178 (Air), RTCA DO-278 (Ground)
  - AI not called out
- IEC/AS 61508
  - Medical: IEC 62304
  - Process control: IEC 61511
  - Automotive: ISO 26262
  - Mining: IC 9460
  - Nuclear: IEC 61513
  - Rail: EN 50128

Artificial Intelligence
– Fault Correction

61508 Family

- Positively Not Recommended for SILs ≥ 2



D.1 Artificial Intelligence Fault Correction

Aim

To be able to react to possible hazards in a very flexible way by introducing a mix (combination) of methods and process models and some kind of on-line safety and reliability analysis.

Description

In particular fault forecasting (calculating trends), fault correction, maintenance and supervisory actions may be supported by Artificial Intelligence-based systems in a very efficient way in diverse channels of a system, since the rules might be derived directly from the specifications and checked against these. Certain common faults which are introduced into specifications by implicitly already having some design and implementation rules in mind may be avoided effectively by this approach, especially when applying a combination of models and methods in a functional or descriptive manner.

The methods are selected such that faults may be corrected and the effects of failures be minimised, in order to meet the desired safety and reliability.

- Not used for that purpose
- Standards are not written to anticipate specific technologies

Where are we?

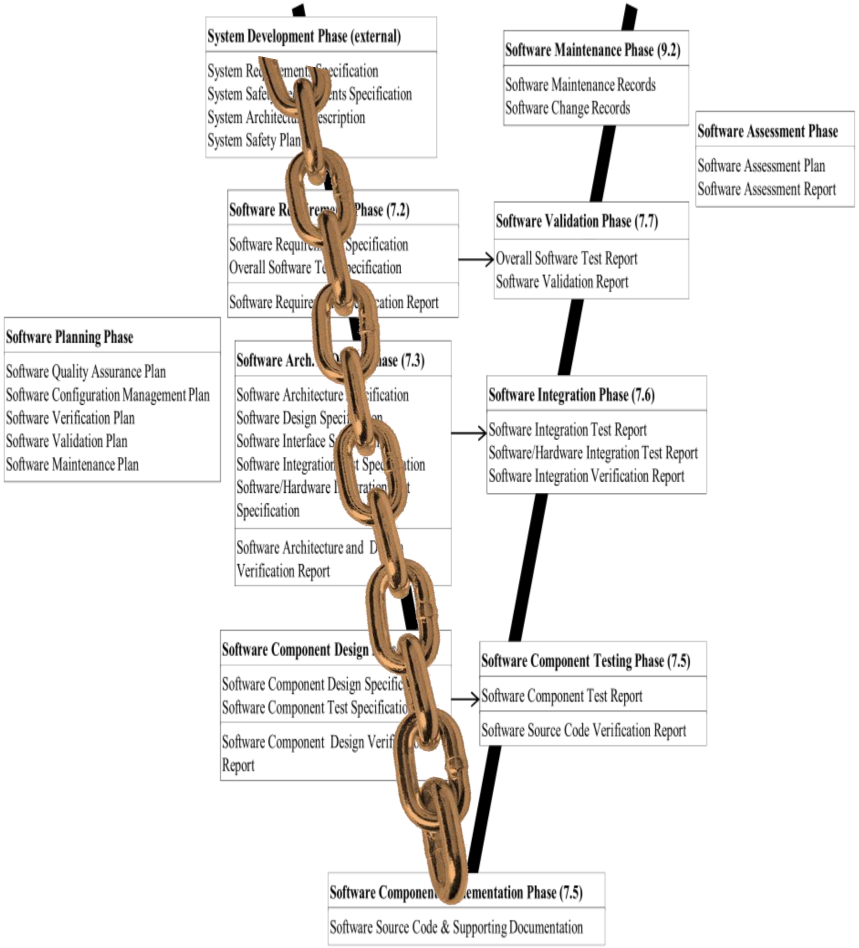
- Systematic behaviour of AI requires lifecycle rigour
- Functional safety standards describe the engineering processes to achieve that
- They are mostly silent on AI as a particular technology
- Can we employ AI and still comply with these standards?
  - Let’s start at the top!

Reduction + Integration

Abstract Specification of Intent

Breakdown/ Integration of subsystems and components

Concrete Implementation (AI)



System Development Phase (external)

System Req ement Specification
System Safe ents Specification
System Architectu escription
System Safety Plan

Software Maintenance Phase (9.2)

Software Maintenance Records
Software Change Records

Software Assessment Phase

Software Assessment Plan
Software Assessment Report

Software R reme Phase (7.2)

Software Requir Specification
Overall Software Te ecification

Software Require cation Report

Software Validation Phase (7.7)

Overall Software Test Report
Software Validation Report

Software Planning Phase

Software Quality Assurance Plan
Software Configuration Management Plan
Software Verification Plan
Software Validation Plan
Software Maintenance Plan

Software Arch. ase (7.3)

Software Architecture ification
Software Design Speci on
Software Interface S
Software Integration t Spec ation
Software/Hardware I ration t
Specification

Software Architecture and D
Verification Report

Software Integration Phase (7.6)

Software Integration Test Report
Software/Hardware Integration Test Report
Software Integration Verification Report

Software Component Design

Software Component Design Specific
Software Component Test Specificatio

Software Component Design Verif
Report

Software Component Testing Phase (7.5)

Software Component Test Report

Software Source Code Verification Report

Software Componen ementation Phase (7.5)

Software Source Code & Supporting Documentation

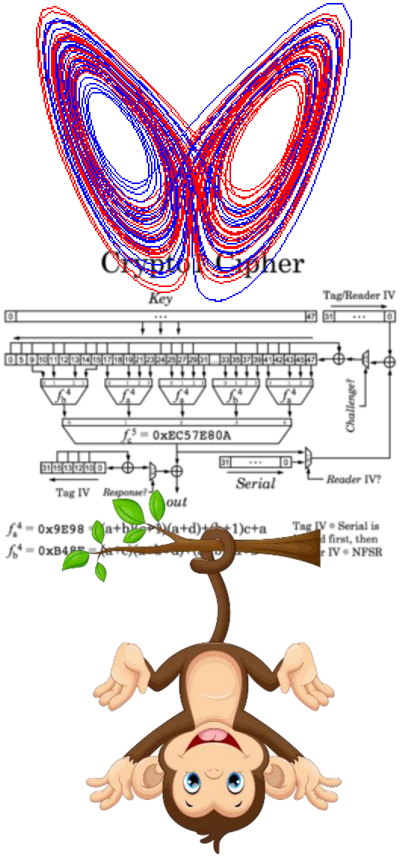
What can break the chain?

- Specifications that can’t be implemented

- Implementations that can’t be abstracted

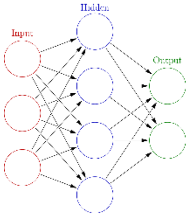
Abstraction Problem

- Software can be inherently unpredictable
  - Non-linear computation
    - Chaotic systems
    - Cryptographic functions
  - Nondeterminism
    - Concurrency
    - Unpredictable input
  - Complexity
    - Human comprehension
    - Bugs (i.e. unknowns)



- Who cares? Isn't this backwards?

Specifying AI



General specification of Neural Network behaviour – topology, mathematical functions, learning algorithms, etc.

Face Recognition Software = Neural Network +

Verification and Validation

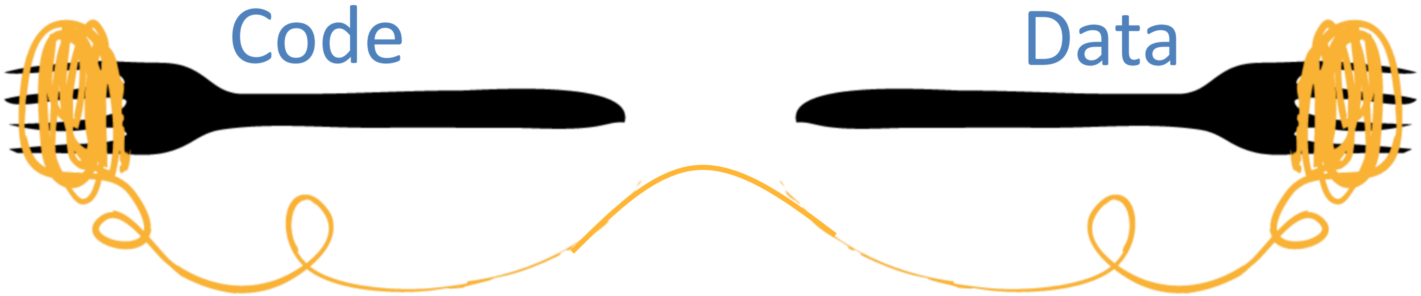
- Verification: “doing the thing right”
  - Implementation ‘perfectly’ traces to, and implements, the specification (since they are the same thing)
    - … absence of verification is not verification
- Validation: “doing the right thing”
  - This is tricky. It’s too complex for us to determine in general, which is why abstraction is needed.
  - All we know for sure is performance on the validation data set

Where are we?

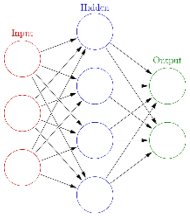
- We couldn’t apply reliability modelling
  - Fundamental problem with *systematic* behaviour
- We couldn’t apply development rigour
  - Fundamental problem with *specification* of that systematic behaviour
- Are there any options left?

Before we continue…

- There is a *duality* between code and data
  - – Code is Data
  - – Data is Code
- Complexity can be transferred but not reduced



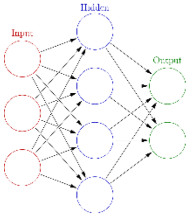
Application Data



General specification of Neural Network behaviour – topology, mathematical functions, learning algorithms, etc.

Face Recognition Software = Neural Network +

Application Data

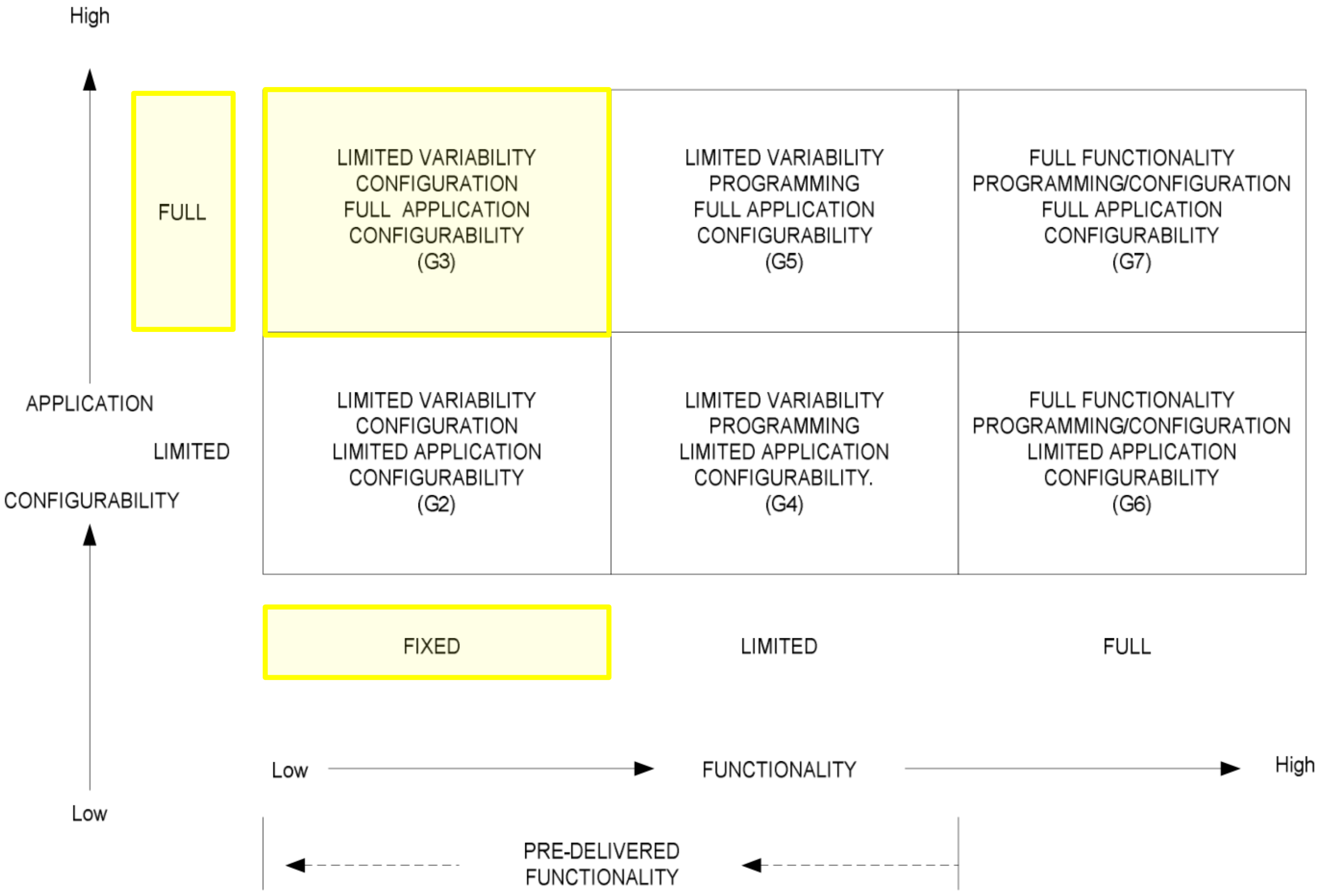


General specification of Neural Network behaviour – topology, mathematical functions, learning algorithms, etc.

“Configuration Language”

Face Recognition Software = Neural Network +

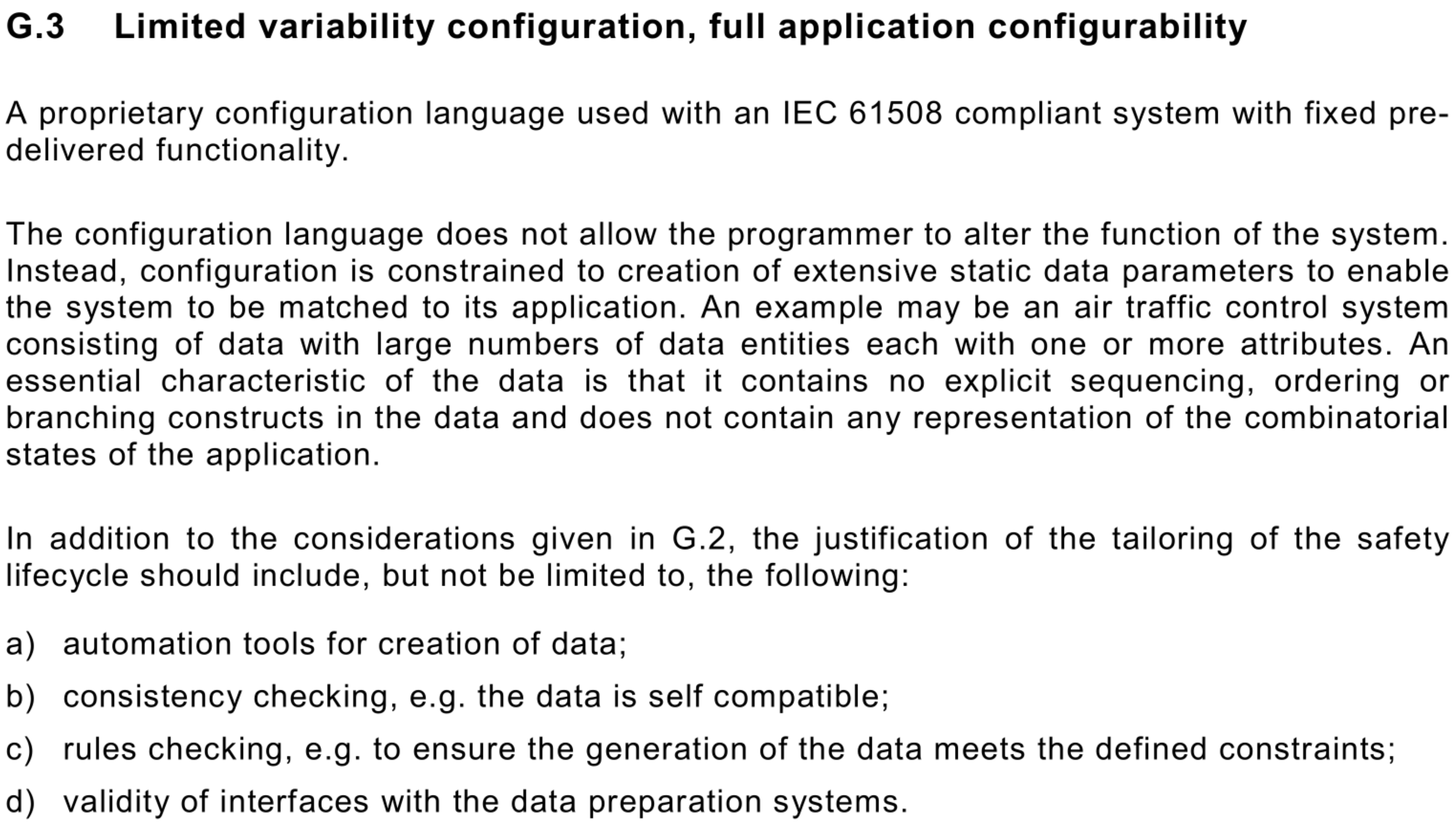
61508 Lifecycle Tailoring



APPLICATION CONFIGURABILITY	FIXED	LIMITED	FULL
FULL	LIMITED VARIABILITY CONFIGURATION FULL APPLICATION CONFIGURABILITY (G3)	LIMITED VARIABILITY PROGRAMMING FULL APPLICATION CONFIGURABILITY (G5)	FULL FUNCTIONALITY PROGRAMMING/CONFIGURATION FULL APPLICATION CONFIGURABILITY (G7)
LIMITED	LIMITED VARIABILITY CONFIGURATION LIMITED APPLICATION CONFIGURABILITY (G2)	LIMITED VARIABILITY PROGRAMMING LIMITED APPLICATION CONFIGURABILITY. (G4)	FULL FUNCTIONALITY PROGRAMMING/CONFIGURATION LIMITED APPLICATION CONFIGURABILITY (G6)

Figure G.1 – Variability in complexity of data driven systems

61508 Lifecycle Tailoring



G.3 Limited variability configuration, full application configurability

A proprietary configuration language used with an IEC 61508 compliant system with fixed pre-delivered functionality.

The configuration language does not allow the programmer to alter the function of the system. Instead, configuration is constrained to creation of extensive static data parameters to enable the system to be matched to its application. An example may be an air traffic control system consisting of data with large numbers of data entities each with one or more attributes. An essential characteristic of the data is that it contains no explicit sequencing, ordering or branching constructs in the data and does not contain any representation of the combinatorial states of the application.

In addition to the considerations given in G.2, the justification of the tailoring of the safety lifecycle should include, but not be limited to, the following:

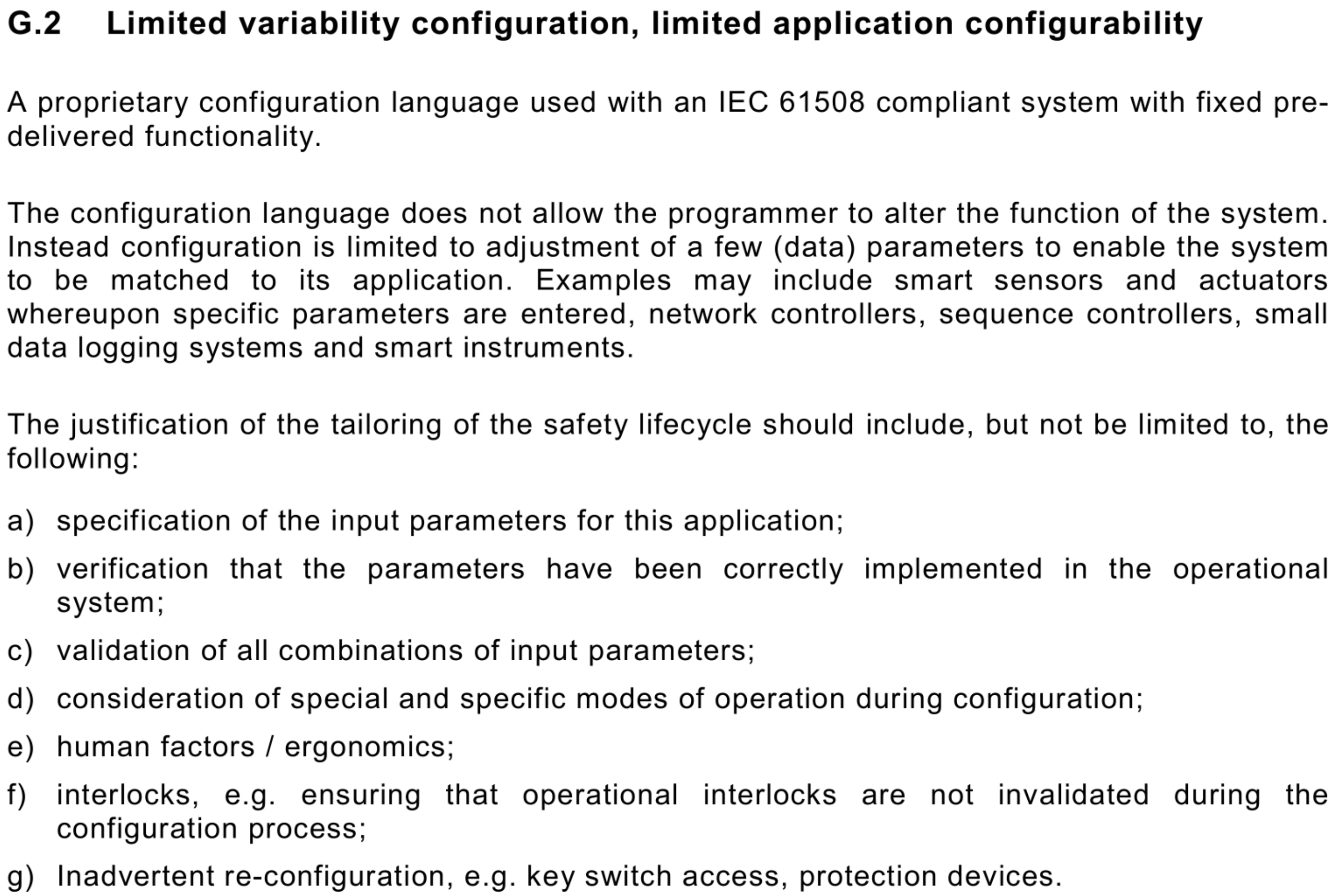
a) automation tools for creation of data;

b) consistency checking, e.g. the data is self compatible;

c) rules checking, e.g. to ensure the generation of the data meets the defined constraints;

d) validity of interfaces with the data preparation systems.

61508 Lifecycle Tailoring



G.2 Limited variability configuration, limited application configurability

A proprietary configuration language used with an IEC 61508 compliant system with fixed pre-delivered functionality.

The configuration language does not allow the programmer to alter the function of the system. Instead configuration is limited to adjustment of a few (data) parameters to enable the system to be matched to its application. Examples may include smart sensors and actuators whereupon specific parameters are entered, network controllers, sequence controllers, small data logging systems and smart instruments.

The justification of the tailoring of the safety lifecycle should include, but not be limited to, the following:

a) specification of the input parameters for this application;

b) verification that the parameters have been correctly implemented in the operational system;

c) validation of all combinations of input parameters;

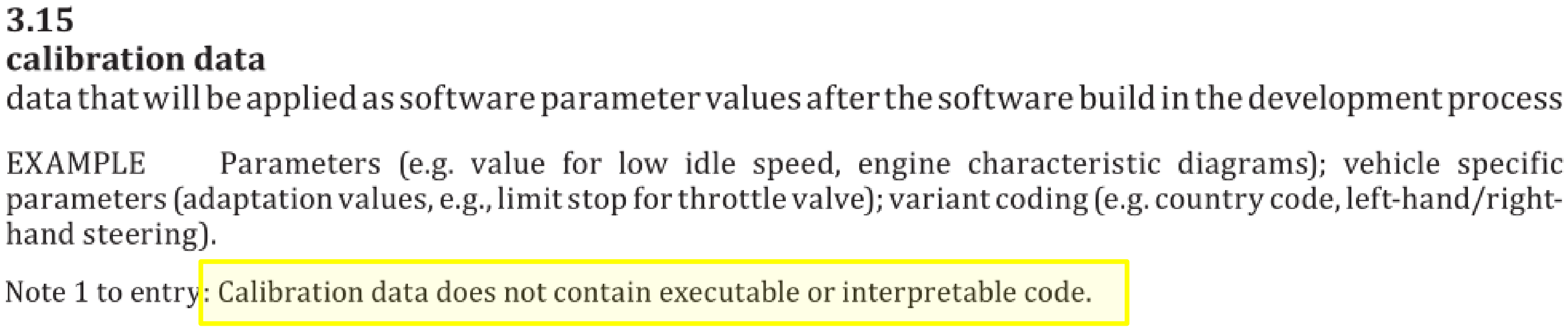
d) consideration of special and specific modes of operation during configuration;

e) human factors / ergonomics;

f) interlocks, e.g. ensuring that operational interlocks are not invalidated during the configuration process;

g) Inadvertent re-configuration, e.g. key switch access, protection devices.

ISO 26262

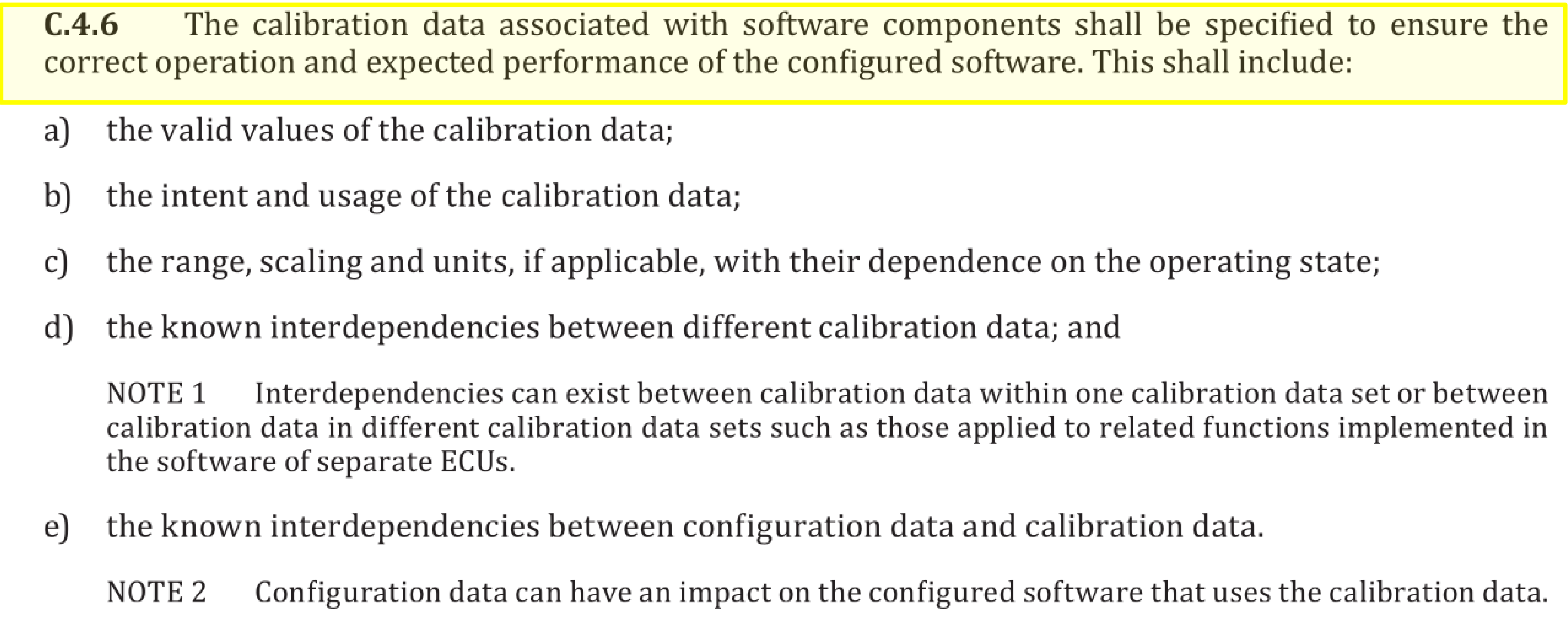


3.15
calibration data
data that will be applied as software parameter values after the software build in the development process

EXAMPLE Parameters (e.g. value for low idle speed, engine characteristic diagrams); vehicle specific parameters (adaptation values, e.g., limit stop for throttle valve); variant coding (e.g. country code, left-hand/right-hand steering).

Note 1 to entry: Calibration data does not contain executable or interpretable code.

ISO 26262



C.4.6 The calibration data associated with software components shall be specified to ensure the correct operation and expected performance of the configured software. This shall include:

a) the valid values of the calibration data;

b) the intent and usage of the calibration data;

c) the range, scaling and units, if applicable, with their dependence on the operating state;

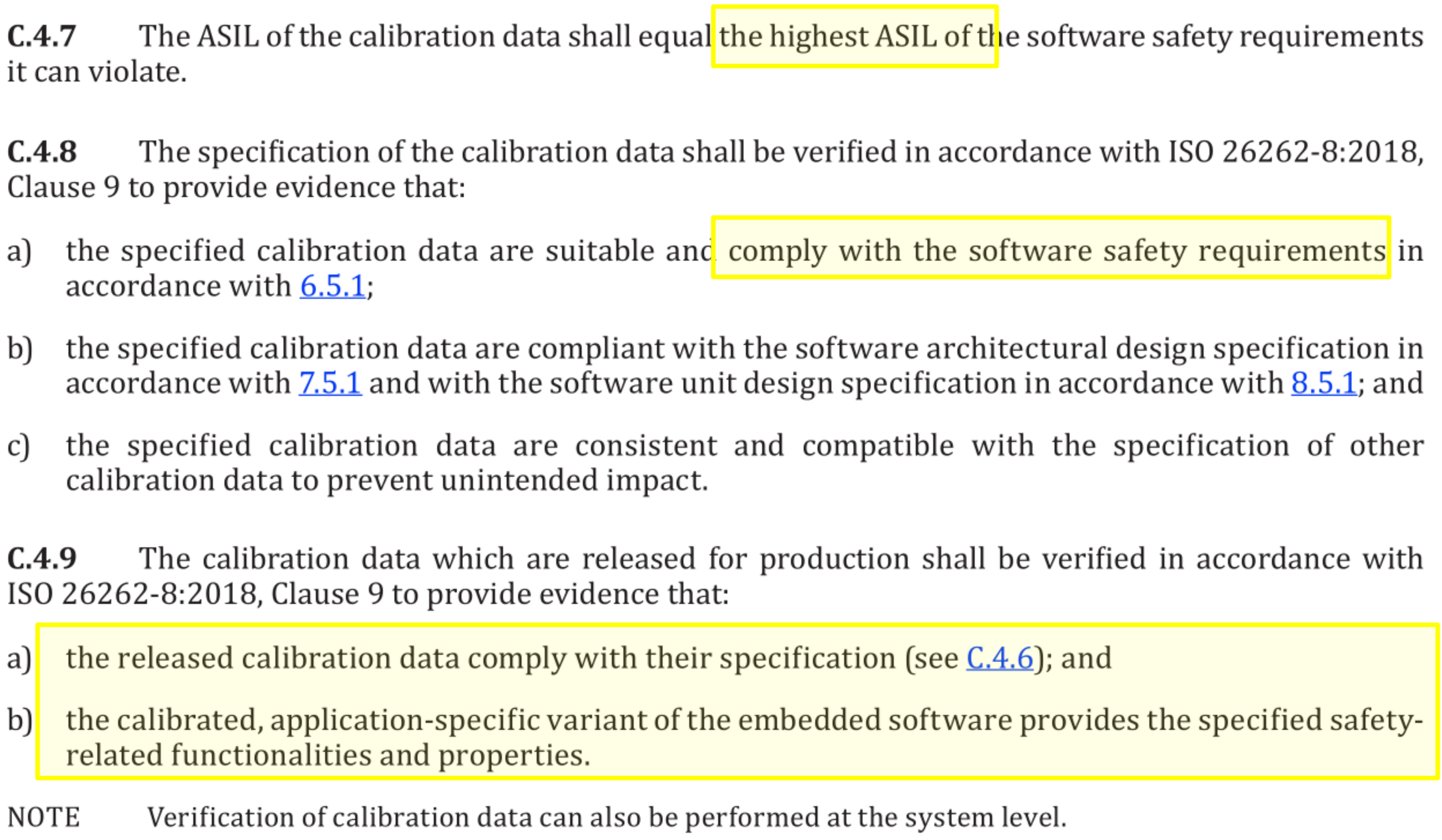
d) the known interdependencies between different calibration data; and

NOTE 1 Interdependencies can exist between calibration data within one calibration data set or between calibration data in different calibration data sets such as those applied to related functions implemented in the software of separate ECUs.

e) the known interdependencies between configuration data and calibration data.

NOTE 2 Configuration data can have an impact on the configured software that uses the calibration data.

ISO 26262



C.4.7 The ASIL of the calibration data shall equal the highest ASIL of the software safety requirements it can violate.

C.4.8 The specification of the calibration data shall be verified in accordance with ISO 26262-8:2018, Clause 9 to provide evidence that:

a) the specified calibration data are suitable and comply with the software safety requirements in accordance with 6.5.1;

b) the specified calibration data are compliant with the software architectural design specification in accordance with 7.5.1 and with the software unit design specification in accordance with 8.5.1; and

c) the specified calibration data are consistent and compatible with the specification of other calibration data to prevent unintended impact.

C.4.9 The calibration data which are released for production shall be verified in accordance with ISO 26262-8:2018, Clause 9 to provide evidence that:

a) the released calibration data comply with their specification (see C.4.6); and

b) the calibrated, application-specific variant of the embedded software provides the specified safety-related functionalities and properties.

NOTE Verification of calibration data can also be performed at the system level.

Conclusion

- We couldn’t apply reliability modelling
- We couldn’t apply lifecycle rigour
- We couldn’t make the System / Calibration Data argument
- This is good!

“design and implement software that fulfils the specified requirements for safety-related software with respect to the required safety integrity level, which is analysable and verifiable, and which is capable of being safely modified.”