

# Analysis of Critical Scientific Software

Jacques Carette, Chris George, Zarrin Langari, Mark Lawford, Tom Maibaum, Ned Nediakov, Vera Pantelic, Spencer Smith, Ali Taleghani, Alan Wassying<sup>1</sup>  
Jonathan Ostroff<sup>2</sup>

<sup>1</sup>McMaster Centre for Software Certification  
Department of Computing and Software  
McMaster University

<sup>2</sup>Department of Computer Science and Engineering  
York University

## 1 Introduction

## 2 Approaches

- Software Requirements Specification (SRS)
- Testing for Trapezoidal Input
- Noise robustness
- Interval Analysis
- Static Code Analysis
- Contracts

## 3 Conclusions

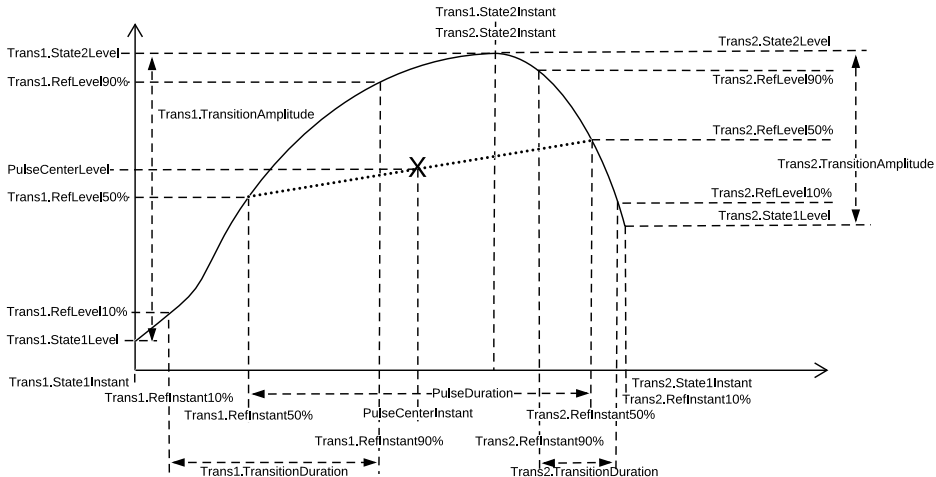
# Goal

- An industrial partner in need of FDA certification for a medical device.
- We were provided with a Matlab function from the software for a blood pressure cuff.
- Goal: identify approaches aimed at increasing confidence in code.

# Context

- A reading of a blood pressure cuff is decomposed into waveforms.
- Waveforms are input to the Matlab function.
- The function calculates parameters of a pulse as defined by *the IEEE Standard on Transitions, Pulses, and Related Waveforms IEEE Standard 181:2003*.
- Code is about 100 lines long and scarcely commented.
- Besides the code, we were provided with 5 sample pulses.

# IEEE Points for Pulse



# Proposed Approaches

- Requirements specification,
- Basic testing of the function for a specific input: trapezoidal signal,
- Investigate noise robustness,
- Interval analysis,
- Static analysis,
- Contracts.

# SRS

- Requirements for the function were poorly documented as comments in the code.
- Benefits of SRS: better communication, needed for assessment of software vs requirements, basis for testing, basis for future changes.

# Proposed Template

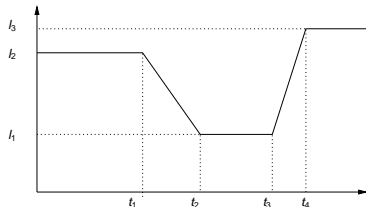
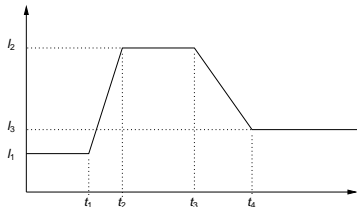
- 1 Introduction
  - Purpose of the Document
  - Scope of the Software Product
  - Terminology Definition
  - Organization of the Document
- 2 General Software Description
  - Software Context
  - User Characteristics
  - Software Constraints
- 3 Specific Software Description
  - Goal Statement
  - Data Definitions
  - Assumptions
  - Theoretical Model
  - Non-Functional Requirements
- 4 Software Validation Plan

# Example Assumptions

- 1 The signal consists of two single transition waveforms.
- 2 The first transition is positive-going.
- 3 The second transition is negative-going.
- 4 A given reference level may intersect with the signal at multiple points. Between multiple instants select, the one that occurs closest to the low, or first, state instant.
- 5 There are at least 3 sampled data points.
- 6 The sampling frequency is high enough to avoid aliasing.

# Trapezoidal input

- Easy to calculate what to test against.



# Inputting trapezoidal signals

- Tested negative pulse: returns NaN or incorrect.
- Tested some special signals: constant, the number of samples is 1 or 2, or there is only one transition, etc.
- For positive trapezoidal pulse, the IEEE parameters expressed in a simple closed form match those calculated by the code.

# Findings

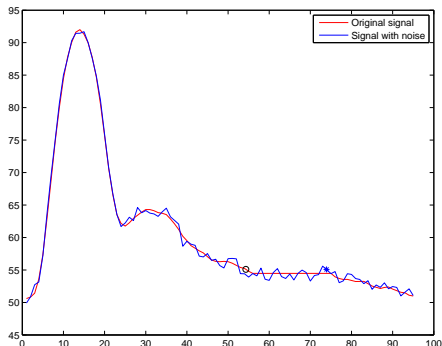
- To summarize, the code calculates what is supposed to under certain assumptions:
  - The number of samples is greater than 2,
  - The signal consists of two transition waveforms, where the first transition is positive-going, and second transition is negative-going.

# Adding Noise

- If a noise is added to a signal, what happens to calculated parameters?
- The white Gaussian noise was added to sample signals provided.
- The noise with signal-to-noise ratio of 40dB was added using Matlab's function `awgn`.

# Findings

- Even weak noise can make some parameters vary substantially.



**Figure:** For one of the provided samples, the point corresponding to 10% reference level of the negative-going transition shifts from the one marked with a circle to the one marked with an asterisk.

# What is Interval Analysis?

- Useful when uncertainties are present.
- Assume that value of a measured quantity is in interval  $\mathbf{x}$ .
- Intervals are propagated through computations using interval arithmetics, e.g.:

$$\mathbf{x} \circ \mathbf{y} = \{\mathbf{x} \circ \mathbf{y} \mid \mathbf{x} \in \mathbf{y}, \mathbf{y} \in \mathbf{y}\}, \quad \circ \in \{+, -, *, /\},$$
$$0 \notin \mathbf{y} \text{ for } \circ = /$$

- Allows for investigation of division by zero: if 0 is not in a calculated range for the divisor, it is a proof that the divisor cannot be 0.
- Tools: INTLAB, a Matlab toolbox.

# Uncertainties

- Interval arithmetics can be used for noise robustness analysis.
- Assume that, for each sample at  $t_j$ , the interval for the magnitude of pulse at  $t_j$  is  $\mathbf{p}_j$ .
- Propagate interval computations to IEEE parameters.
- Find ranges for 10%, 50%, and 90% reference values, and ranges for corresponding instances.
- The analysis found the same noise sensitivity of some parameters as classical noise robustness analysis presented earlier.

# Static Code Analysis

- No static analysis tools for Matlab (showing e.g., index out of bounds, null dereferencing, etc.).
- Converting to C did not work out.

# Specification in Eiffel

- Contracts describe each component's expectations and guarantees.
- Assertions: preconditions, postconditions, and invariants.
- Rewrote the Matlab code in Eiffel in a straightforward manner.
- Signal validity conditions are given as preconditions, while postconditions and invariants are derived from IEEE 181.

# Testing with Eiffel

- Assertions can be evaluated at run time.
- AutoTest, an automatic software testing tool for Eiffel.
- For one of the pulses provided by the industrial partner, a postcondition failed pointing to a problem:
  - According to IEEE 181 standard, if there are more than two instances corresponding to the 10% level, the one closest to the 50% should be chosen.
  - But, the code does not choose that one.

# Ideas for Future Work

- Implement contracts and AutoTest-like tool for Matlab,
- Investigate theorem proving,
- Explore Simulink Design Verifier.