Two ways to use AI for assurance of critical software

JULY 15, 2024

Bjorn Andersson

Carnegie Mellon University Software Engineering Institute

[Distribution Statement A] Approved for public release and unlimited distribution.

Copyright 2024 Carnegie Mellon University.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

DM24-0796

Carnegie

Mellon

© 2024 Carnegie Mellon University

Agenda

- My background (and how it relates to space)
- Using Large-Language Model (LLM) for Hazard Analysis
- Using Artificial Intelligence for Worst-Case Execution Time Analysis

My background (and how it relates to space)

Carnegie Mellon University Software Engineering Institute

[Distribution Statement A] Approved for public release and unlimited distribution.

Apollo program: We need to schedule realtime tasks on a single processor.



Carnegie Mellon University Software Engineering Institute

time

[1] Liu, C. L. Scheduling algorithms for hard-real-time multiprogramming of a single processor. JPL Space Programs Summary 37,60, Vol. II, Jet Propulsion Lab., Calif. Inst. of Tech., Pasadena, Calif., Nov. 1969 Two ways to use AI for assurance of critical software © 2024 Carnegie Mellon University

Apollo program: Fixed-priority preemptive scheduling is a good idea [1].

Carnegie Mellon

1969

time

[1] Liu, C. L. Scheduling algorithms for hard-real-time multiprogramming of a single processor. JPL Space Programs Summary 37,60, Vol. II, Jet Propulsion Lab., Calif. Inst. of Tech., Pasadena, Calif., Nov. 1969 Two ways to use AI for assurance of critical software © 2024 Carnegie Mellon University

Apollo program: Assign priorities to process according to rate-monotonic (short period yields high priority) [1].



time

1969

1969 [1] Liu, C. L. Scheduling algo Summary 37,60, Vol. II, Jet Two ways to use AI for assurance of critical software © 2024 Carnegie Mellon University

[1] Liu, C. L. Scheduling algorithms for hard-real-time multiprogramming of a single processor. JPL Space Programs Summary 37,60, Vol. II, Jet Propulsion Lab., Calif. Inst. of Tech., Pasadena, Calif., Nov. 1969

Apollo program: Single processor system: Rate-Monotonic has utilization bound 69% [1].



time

8

[1] Liu, C. L. Scheduling algo Summary 37,60, Vol. II, Jet Two ways to use AI for assurance of critical software © 2024 Carnegie Mellon University

1969

Single processor system: Rate-Monotonic has utilization bound 69% [1].

Apollo program:

A rich research literature and practice was developed for Rate-Monotonic.

Carnegie Mellon University Software Engineering Institute

time

9

[1] Liu, C. L. Scheduling algorithms for hard-real-time multiprogramming of a single processor. JPL Space Programs Summary 37,60, Vol. II, Jet Propulsion Lab., Calif. Inst. of Tech., Pasadena, Calif., Nov. 1969



[2] C. Liu, "Scheduling algorithms for multiprocessors in a hard real-time environment," in JPL Space Programs Summary, vol. 37-60. JPL, Pasadena, CA, 28-31, 1969. Two ways to use AI for assurance of critical software © 2024 Carnegie Mellon University

Apollo program: Multi-processor system: Rate-Monotonic has utilization bound approaching 0% [2].

Carnegie Mellon

time

10

1969

Two ways to use AI for assurance of critical software © 2024 Carnegie Mellon University

Apollo program: Multi-processor system: Rate-Monotonic has utilization bound approaching 0% [2].

Multi-processor system: There is another way of assigning priorities; this yields utilization bound 33% [3].

Carnegie Mellon University

1969

2001

time

11

[3] B. Andersson, S. Baruah, and J. Jonsson, "Static-priority scheduling on multiprocessors," IEEE RTSS, 2001.

Two ways to use AI for Assurance

Using Large-Language Model (LLM) for Hazard Analysis

Carnegie Mellon University Software Engineering Institute

[Distribution Statement A] Approved for public release and unlimited distribution.

Environment (that we can't control)



A mishap occurs when (i) a certain condition is true about the technical system, and (ii) a certain condition is true about the environment.

Two ways to use AI for assurance of critical software © 2024 Carnegie Mellon University

Environment (that we can't control)



Since we can't control the environment, let us focus on what we can control; that is, let us focus on the technical system.

Mellor

Environment (that we can't control)



A hazard of a technical systems is a condition such that if this condition is true, and the environment is in a bad state, then a mishap occurs.

Mellon Univers Softwa Enginee Institut

Environment (that we can't control)



Carnegie Mellon University Software Engineering Institute

If we could find all hazards and eliminate them, then we would eliminate all mishaps.

Environment (that we can't control)



Mellon University Software Engineerii Institute

In practice, we can't find all hazards but we can find many of them and we can try to eliminate them.

Environment (that we can't control)



Hazard analysis is about finding hazards.

Environment (that we can't control)



Hazard analysis is about looking at documentation to find hazards.

Environment (that we can't control)



Hazard analysis is about looking at documentation from various perspectives to find hazards. One perspective yields one hazard analysis. Another perspective yields another hazard analysis.

Environment (that we can't control)



Hazard analysis is not about proving correctness properties. It is about discovering issues.

Hazard Analysis is Laborious and Expensive

Environment (that we can't control)



Carnegie Mellon University Software Engineering Institute

It takes a lot of time and money for humans to read and analyze documents.

Hazard Analysis is Hard to Automate

Environment (that we can't control)



Carnegie Mellon University Software Engineering Institute

It requires common-sense reasoning, contextual knowledge about the technical system and its environment, and background knowledge (that most humans have).

Hazard Analysis is Hard to Automate

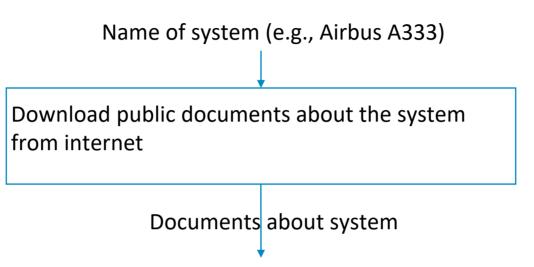
Environment (that we can't control)



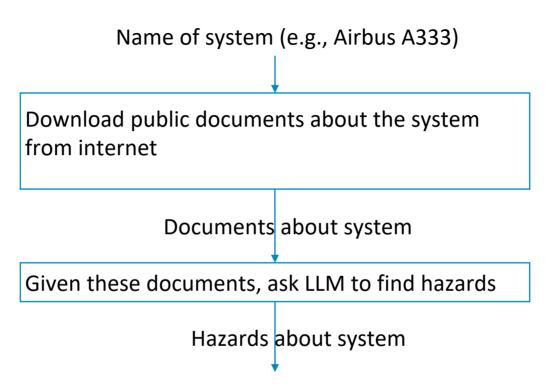
Carnegie Mellon University Software Engineering Institute

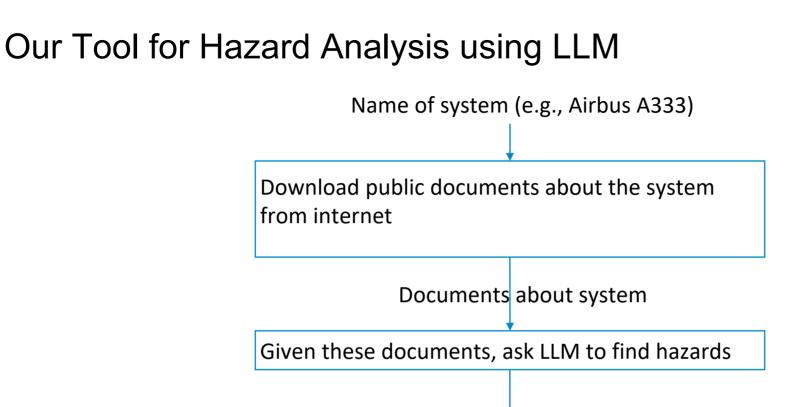
Idea: An LLM can "simulate" human thinking. Hence, using an LLM for hazard analysis seems worthwhile.

Our Tool for Hazard Analysis using LLM



Our Tool for Hazard Analysis using LLM

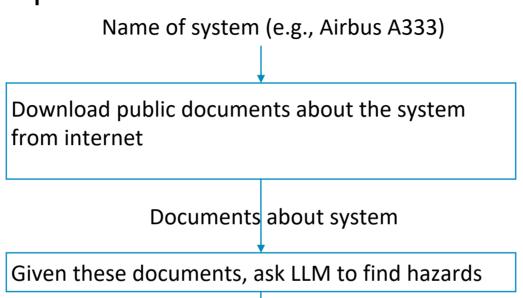




"The Airbus A330 carries various hazards related to operational, maintenance, and environmental factors, including its electrical, fuel, engine, hydraulic, and wastewater systems."

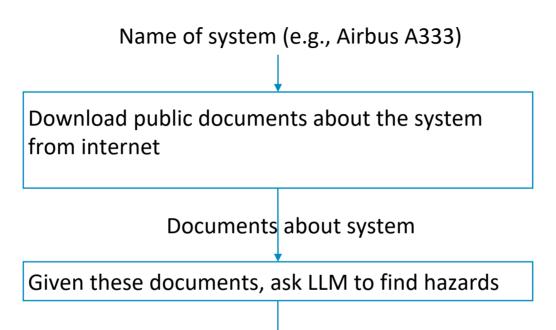
Aellor

The Quality of the Output from Hazard Analysis Depends on the Quality of Input



"The Airbus A330 carries various hazards related to operational, maintenance, and environmental factors, including its electrical, fuel, engine, hydraulic, and wastewater systems."

Mellon University Hazard Analysis based on Detailed Documents (Proprietary) yields better output than Superficial Documents (Publicly Available)



"The Airbus A330 carries various hazards related to operational, maintenance, and environmental factors, including its electrical, fuel, engine, hydraulic, and wastewater systems."

Carnegie

Mellon University Two ways to use AI for assurance

Using Artificial Intelligence for Worst-Case Execution Time Analysis

Carnegie Mellon University Software Engineering Institute

[Distribution Statement A] Approved for public release and unlimited distribution.

What is Worst-Case Execution Time?

Execution time

Different inputs to a program

Mellon



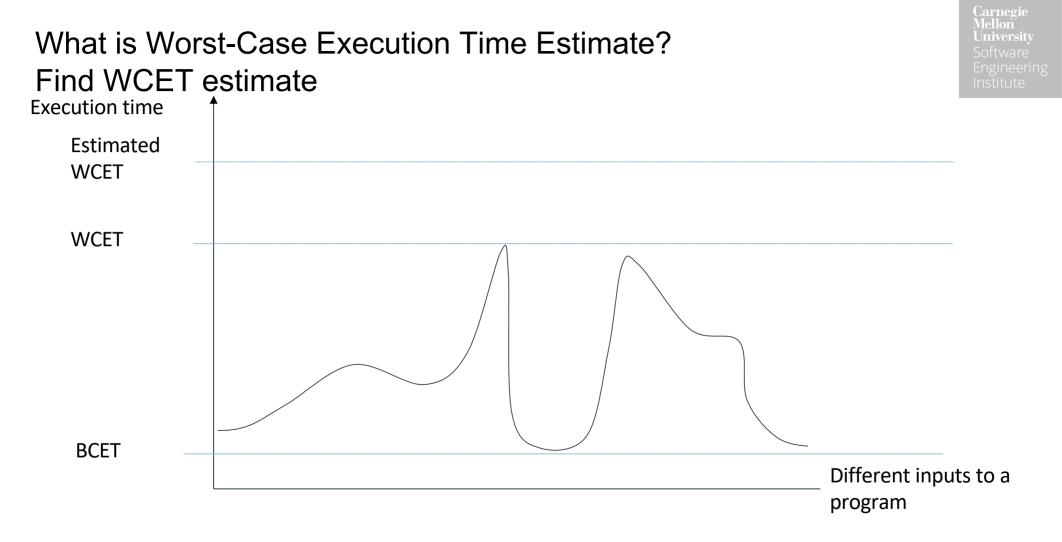
Two ways to use AI for assurance of critical software © 2024 Carnegie Mellon University

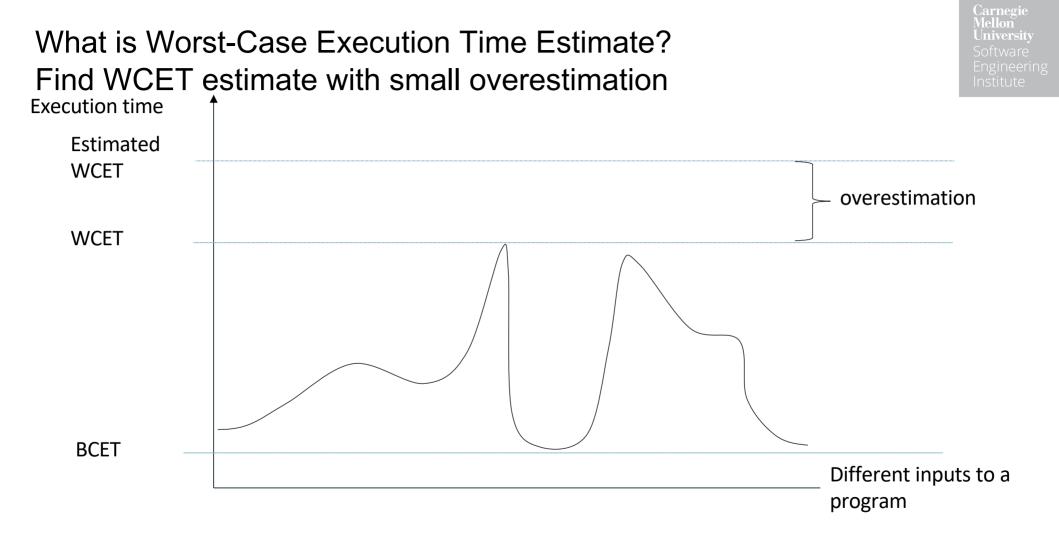
Mellon

Mellon What is Worst-Case Execution Time Estimate? Execution time Estimated WCET WCET BCET Different inputs to a program

Two ways to use AI for assurance of critical software © 2024 Carnegie Mellon University

Carnegie





Carnegie Mellon University Software Engineering Institute

Software Complexity

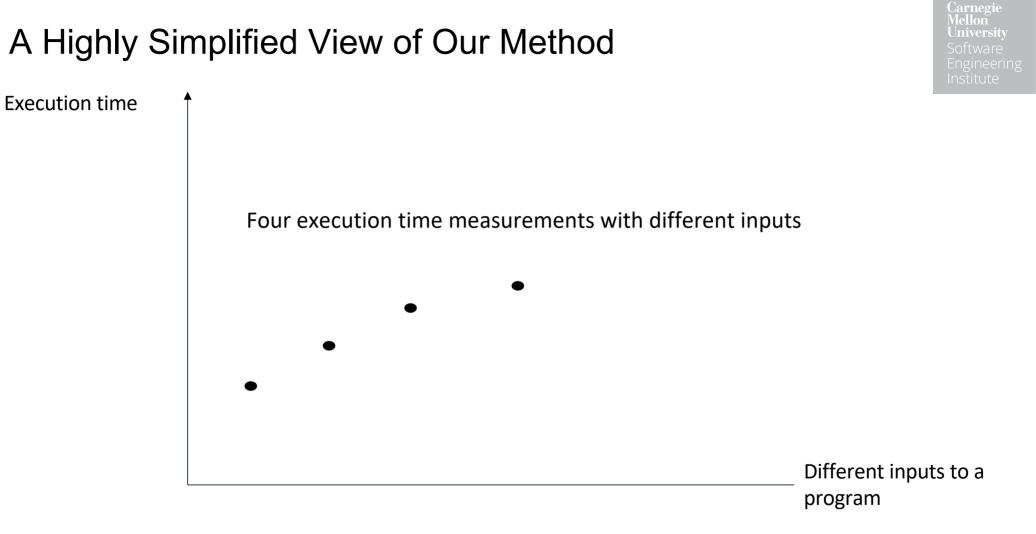
• The number of execution paths in a program tends to be very large and input dependent. We cannot explicitly enumerate all of them.

Hardware Complexity

- Even for a single path in a program, the execution time depends on (i) initial state (variable initialization), (ii) state of the hardware (dirty cache blocks initially), (iii)behavior of hardware (cache, pipelining, etc).
- On a multicore, it gets even more complex because the execution time of a program depends on co-runners.

Carnegie Mellon University Software Engineering Institute

Our Method



A Highly Simplified View of Our Method

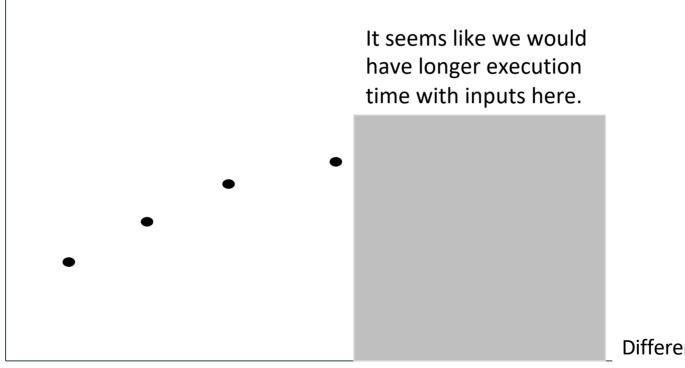
We don't have any execution time measurements here Different inputs to a program

Carnegie Mellon University Software Engineering Institute

Execution time

A Highly Simplified View of Our Method

Execution time

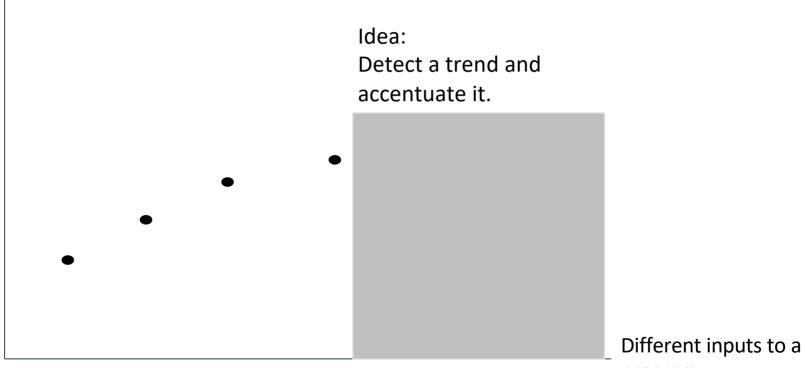


Different inputs to a program

Carnegie Mellon

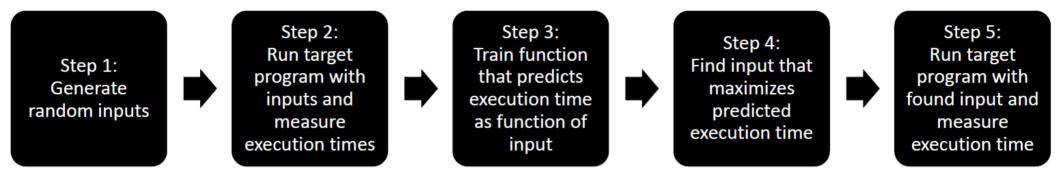
A Highly Simplified View of Our Method

Execution time



program

Carnegie Mellon



Step 3:

Train function

that predicts

execution time

as function of

input

Step 2:

Run target

program with

inputs and

measure

execution times

Step 4: Find input that maximizes predicted

Step 5: Run target program with found input and measure execution time

Carnegie Mellon University

Our method can find a WCET estimate even if (i) we do not have source code of program, and (ii) we do not have documentation of hardware.

Step 1:

Generate

random inputs

execution time

Step 2:

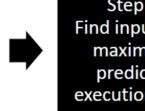
Run target

program with

inputs and

measure

execution times



Step 4: Find input that maximizes predicted execution time

Step 5: Run target program with found input and measure execution time

Carnegie Mellon University

Our method can find counter-intuitive effects that humans can't find and other WCET analysis tools can't find.

Step 3:

Train function

that predicts

execution time

as function of

input

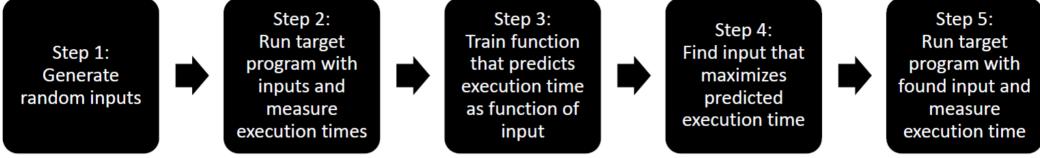
Step 1:

Generate

random inputs

University Software Engineeri Institute

Carnegie Mellon



Our method can find counter-intuitive effects that humans can't find and other WCET analysis tools can't find.

Details at:

DOT/FAA/TC-23/06 Assessing the Use of Machine Learning to Find the Worst-Case Execution Time of Avionics Software https://www.faa.gov/aircraft/air_cert/design_approvals/air_software/r esearch/TC2306



Bjorn Andersson Principal Researcher

Telephone: +1 412.268.9243 Email: <u>baandersson@sei.cmu.edu</u>