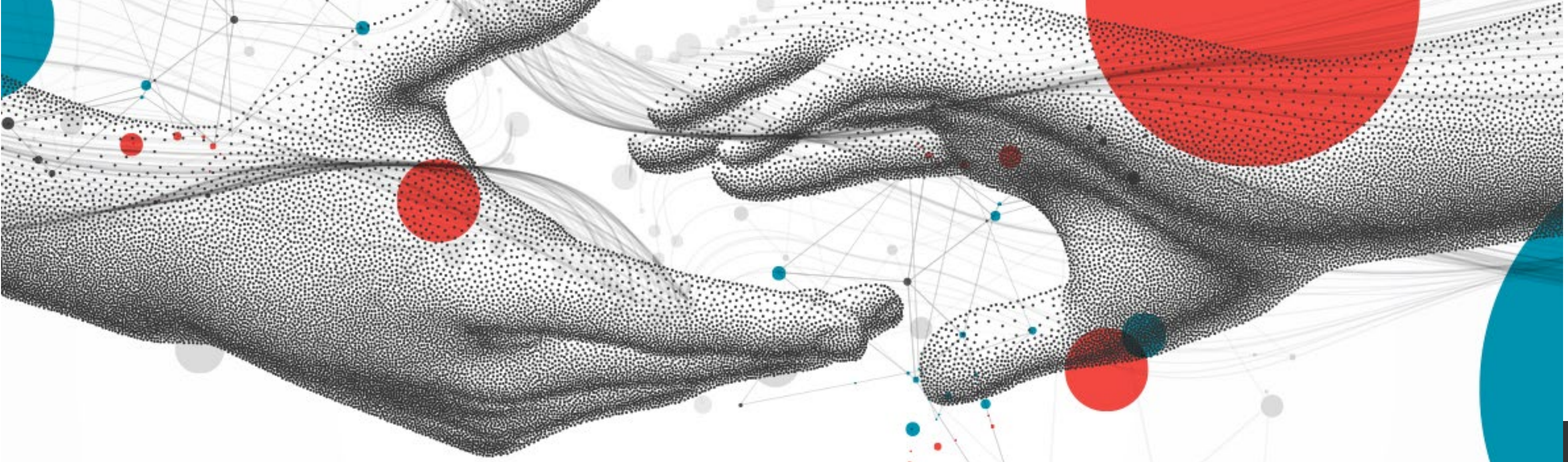
RESEARCH REVIEW 2024



Carnegie Mellon University Software Engineering Institute

Explainable Verification for Rapid Certification

NOVEMBER 13, 2024

Bjorn Andersson

Carnegie Mellon University Software Engineering Institute

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.

DM24-1371

Bottom Line Up Front

Background

- Certification is expensive; testing is expensive.
- Formal methods (FM) potential make it less expensive.
- Practitioners do not trust FM.

Theorem 4. *If* $\forall \tau_i \in \tau\ \forall q \in \mathrm{QSET}_i\ \exists t \in [0, D_i]\ \mathrm{reqlpARINC653}(i, t) \leq \mathrm{sbf}(i, q, t)$, *then the system is schedulable.*

Our work

- Create Explanation Methods (XMs) for FM.
- 88% believe XMs increase confidence in FMs.

DoD Context


N280BH
Photo Credit: U.S. Army photo by Jay Miller


U.S. Airforce Illustration:
Staff Sgt. Shelby Thurman

DoD Context

System includes software that reads sensors and controls physical objects to conduct its mission.

DoD Context

Critical Systems: Need to be certified/approved before being deployed.

DoD Context

Correct timing of software in execution is essential.

DoD Context

From page 24 in *Army Military Airworthiness Certification Criteria (AMACC), Revision A Change 2 (C2)*, 2021:

3.1.36 Performance. A quantitative or qualitative measure characterizing a physical or functional attribute relating to the execution of an operation or function. Performance attributes include quantity (how many or how much), quality (how well), coverage (how much area, how far), timeliness (how responsive, how frequent), and readiness (availability, mission/operational readiness). Performance is an attribute for all systems, people, products, and processes including those for development, production, verification, deployment, operations, support, training, and disposal. Supportability parameters, manufacturing process variability, reliability, and so forth are all performance measures.

DoD Context

From page 33 in
Army Military Airworthiness Certification Criteria (AMACC), Revision A Change 2 (C2), 2021:

WCET	Worst-Case Execution Time

DoD Context

Carnegie Mellon University Software Engineering Institute

From page 46 in *DAFMAN91-119_DAFGM2023-01*, June 8 2023:

8.10.5 The software shall detect and appropriately respond to Critical Signal-impacting hardware and software faults during all stages of execution; including startup, operation, and shutdown; within the lesser of (1) the maximum operationally acceptable time and prior to the time limit; and (2) the time to any associated irreversible adverse system event. **(T-1)** Verification activities should prove that the software detects hardware and software faults during all stages of execution within an acceptable time and responds to those within an acceptable time.

8.10.6 The software shall report the events defined in the following subparagraphs to the operator within the lesser of (1) the maximum operationally acceptable time and prior to the time limit, and (2) the time to any associated irreversible adverse system event. **(T-1)** Verification activities should prove that the software reports the listed events and automated actions to the operator within an acceptable time.

DoD Context

From page 52 in *DAFMAN91-119_DAFGM2023-01*, June 8, 2023:

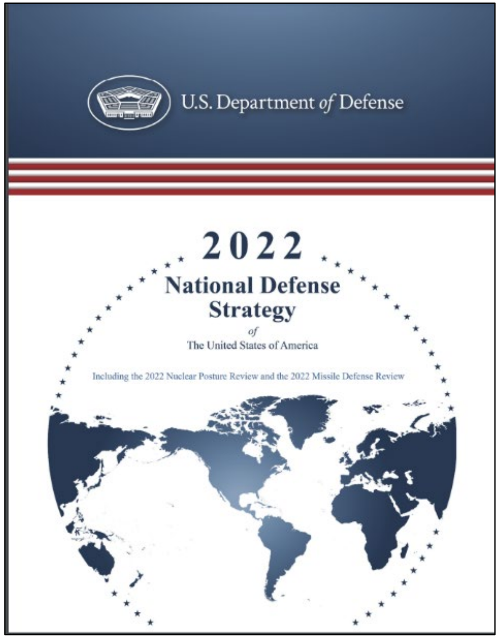
9.4 Real-Time Embedded Systems. An embedded system is a processor with software that usually serves a dedicated function within a larger system. Embedded systems rarely have traditional user Interfaces typical of computing systems and are not as multipurpose as other computing systems. Embedded systems have specific software safety concerns. Real-time software has strict timing constraints that require the system to react to inputs within a certain amount of time. Real-time software uses operating systems and software paradigms to maintain the timing constraints.

9.4.1. Embedded systems in the nuclear mission usually contain real-time software. Per **paragraph 8.16**, Real-Time Processing is only nuclear critical if a failure of the software can lead to the failure to cancel a Critical Signal after an operator has withdrawn human intent or if a failure of the software can lead to a failure to issue a Safing Command. If the design precludes these scenarios, then Real-Time Processing requirements are not applicable.

DoD Context

Critical Systems: Need to be certified/approved before being deployed. Time consuming and expensive.

The Need for Speed (of Software Development of Weapon Systems)

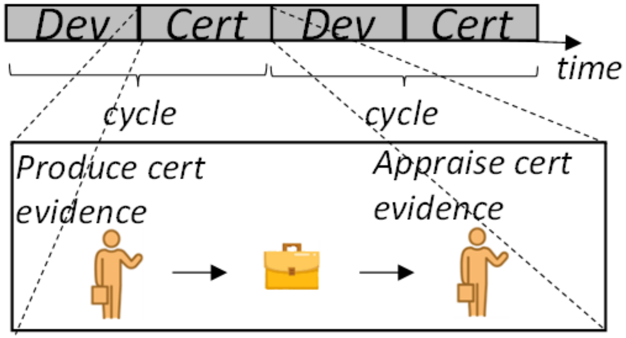


"The Department will instead reward rapid experimentation, acquisition, and fielding…."

How do we combine the need for safety with the need for speed?

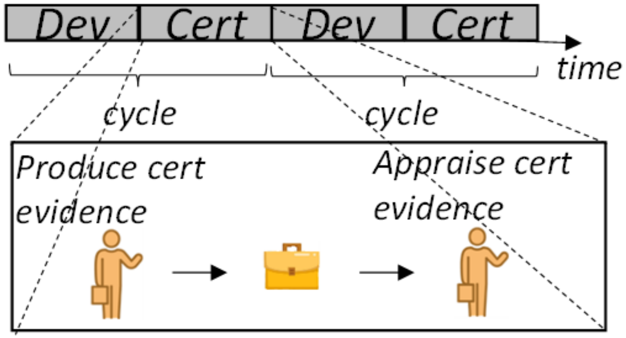
Carnegie Mellon University Software Engineering Institute

How do we combine the need for safety with the need for speed?



Carnegie Mellon University Software Engineering Institute

How do we combine the need for safety with the need for speed?



Even if we decrease the development time to zero, then certification time is still there and becomes a bottleneck for our ability to field new systems rapidly.

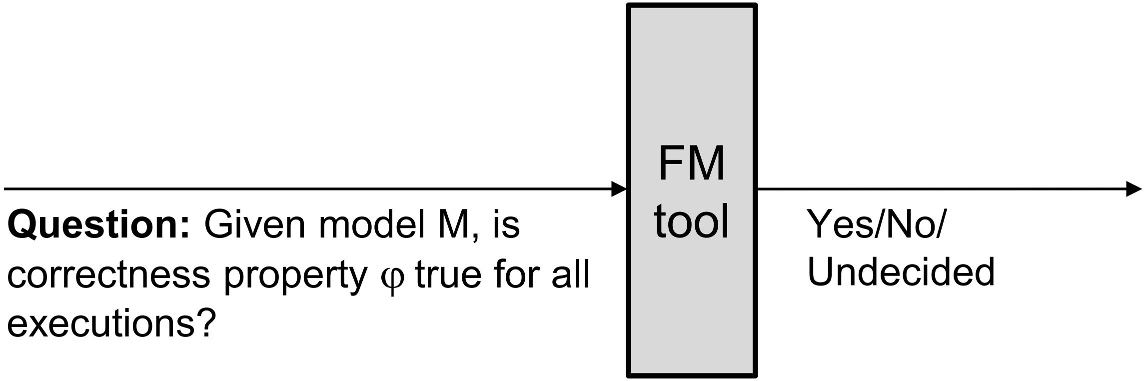
How do we combine the need for safety with the need for speed?

Test data as cert evidence

- Exhaustive takes too long
- Non-exhaustive is unsafe

Formal methods have the potential to avoid these.

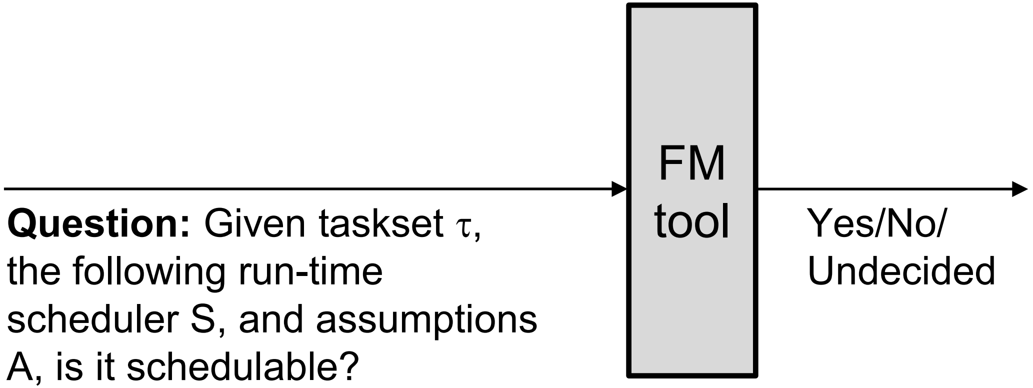
Formal Methods



Input: (i) a model of a system and (ii) a correctness condition.

Output: True/False/Undecided

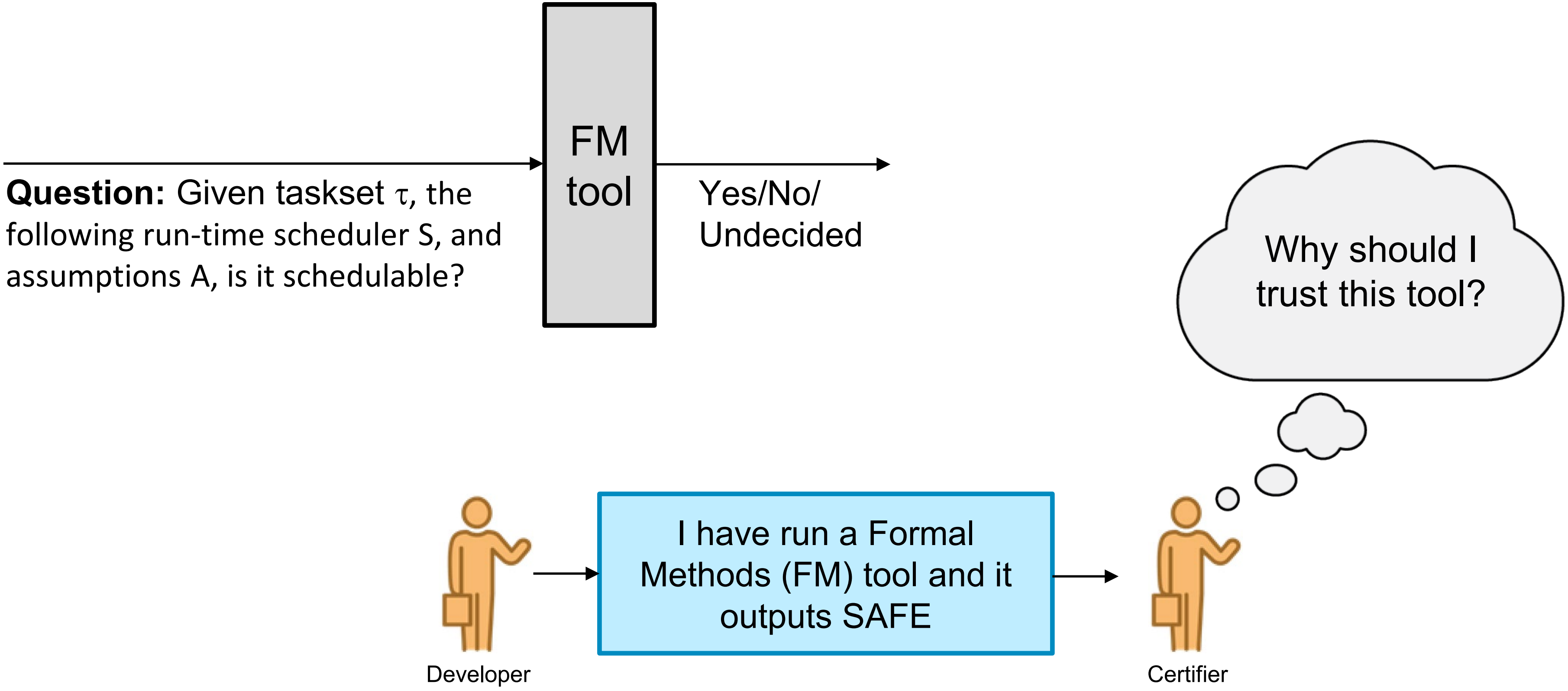
Formal Methods for Real-Time Systems



Real-Time Systems

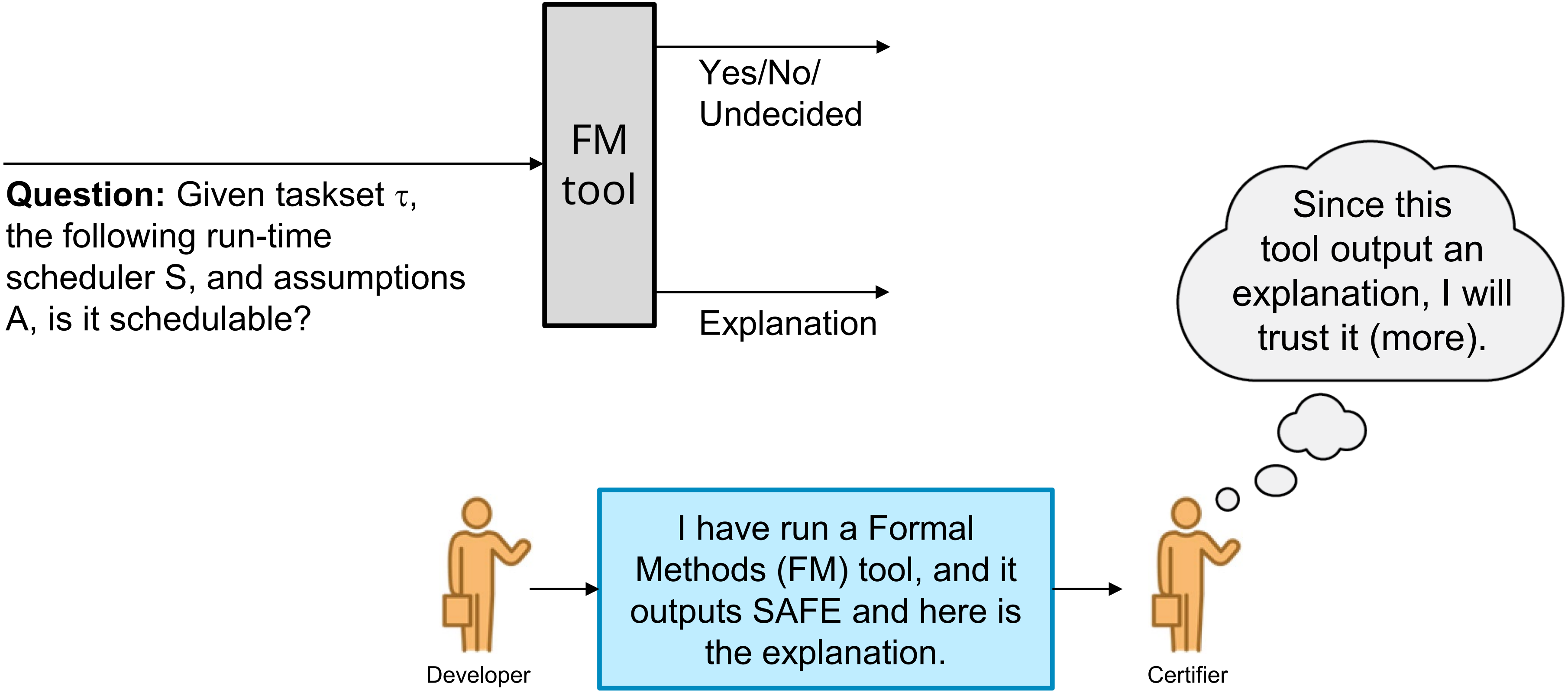
- Software systems where computing the right result is not enough. It needs to be computed at the right time.
- It is often implemented as a set of concurrent tasks.

The Role of Explanation in Certification

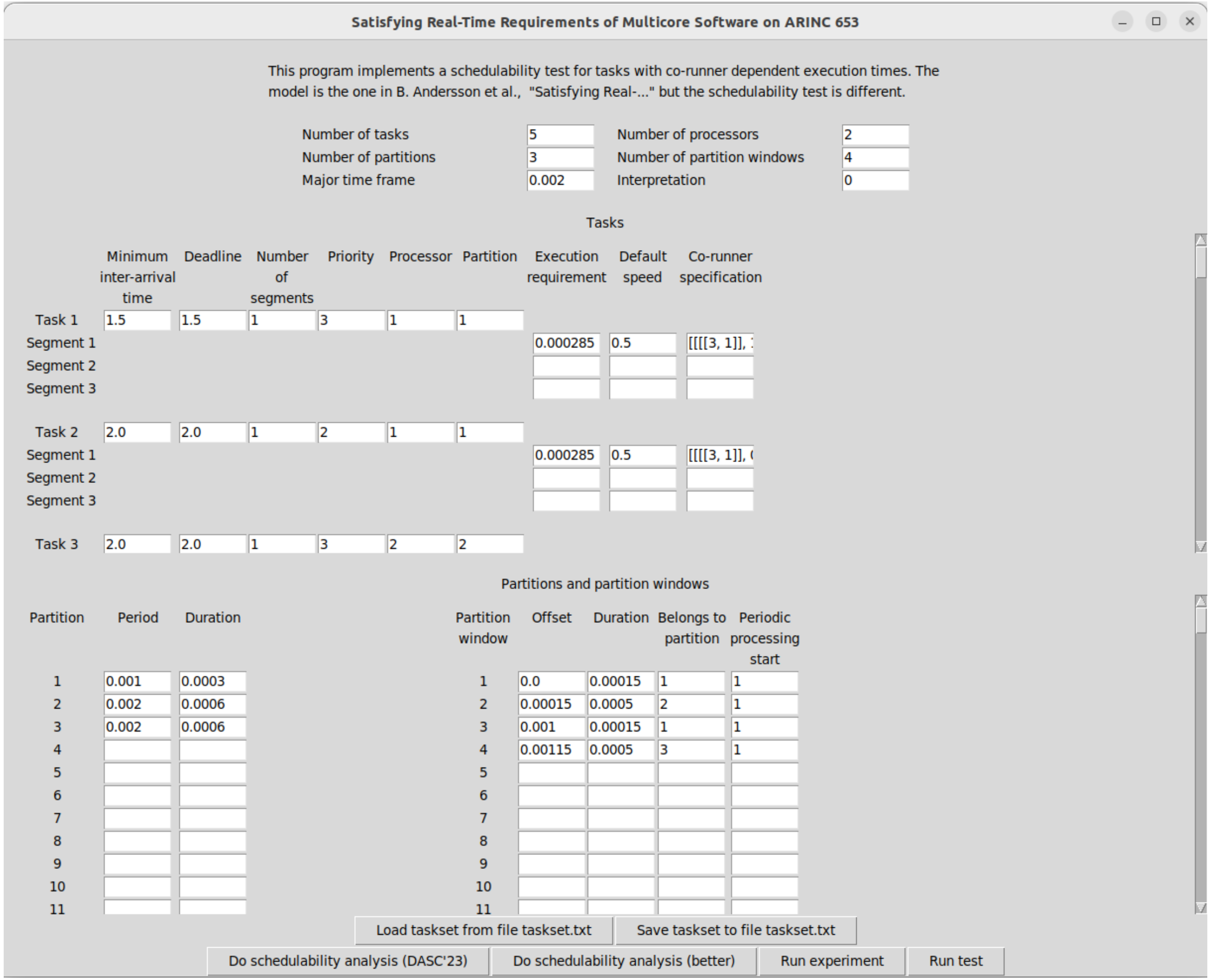


The Role of Explanation in Certification

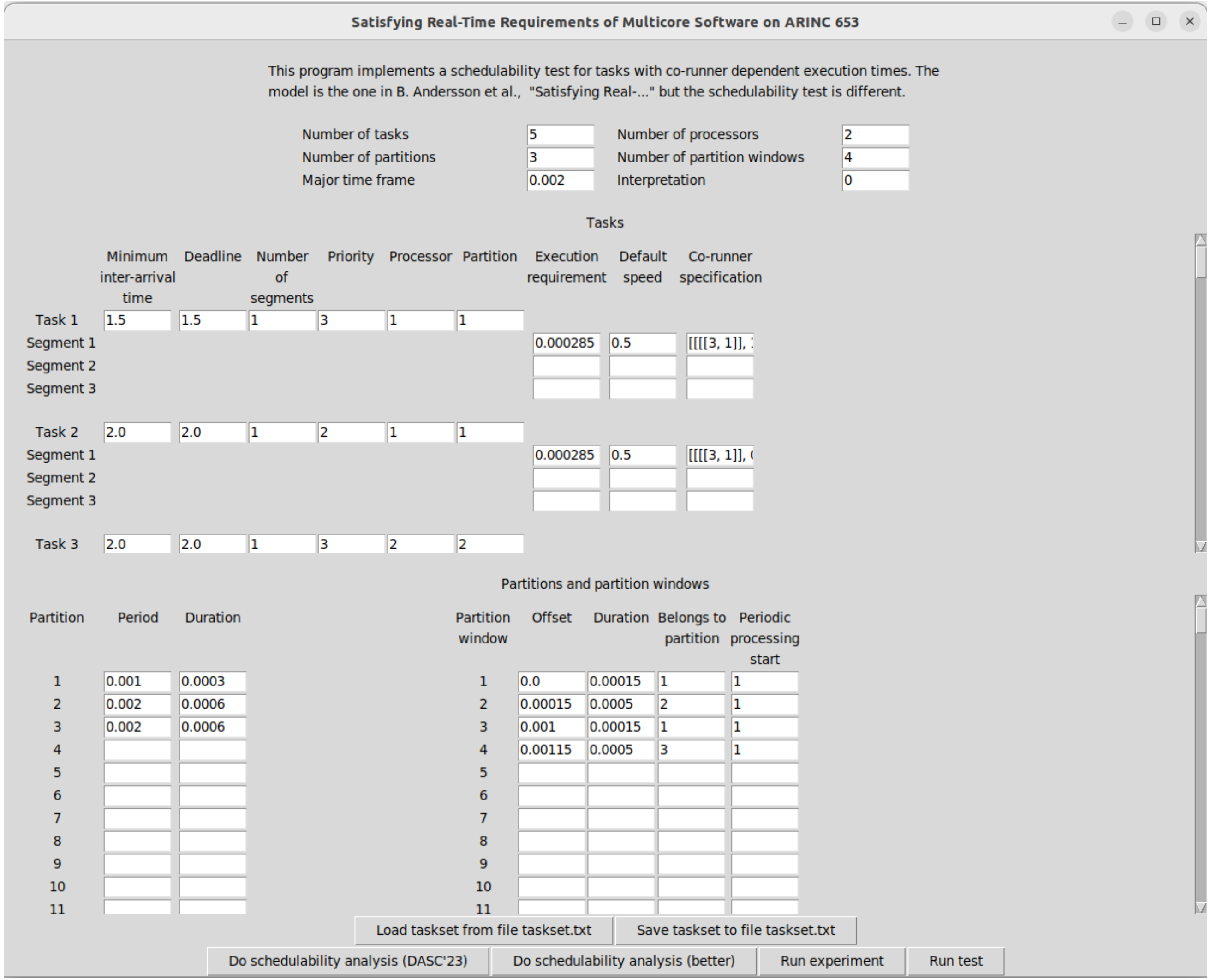
Question: Given taskset τ, the following run-time scheduler S, and assumptions A, is it schedulable?



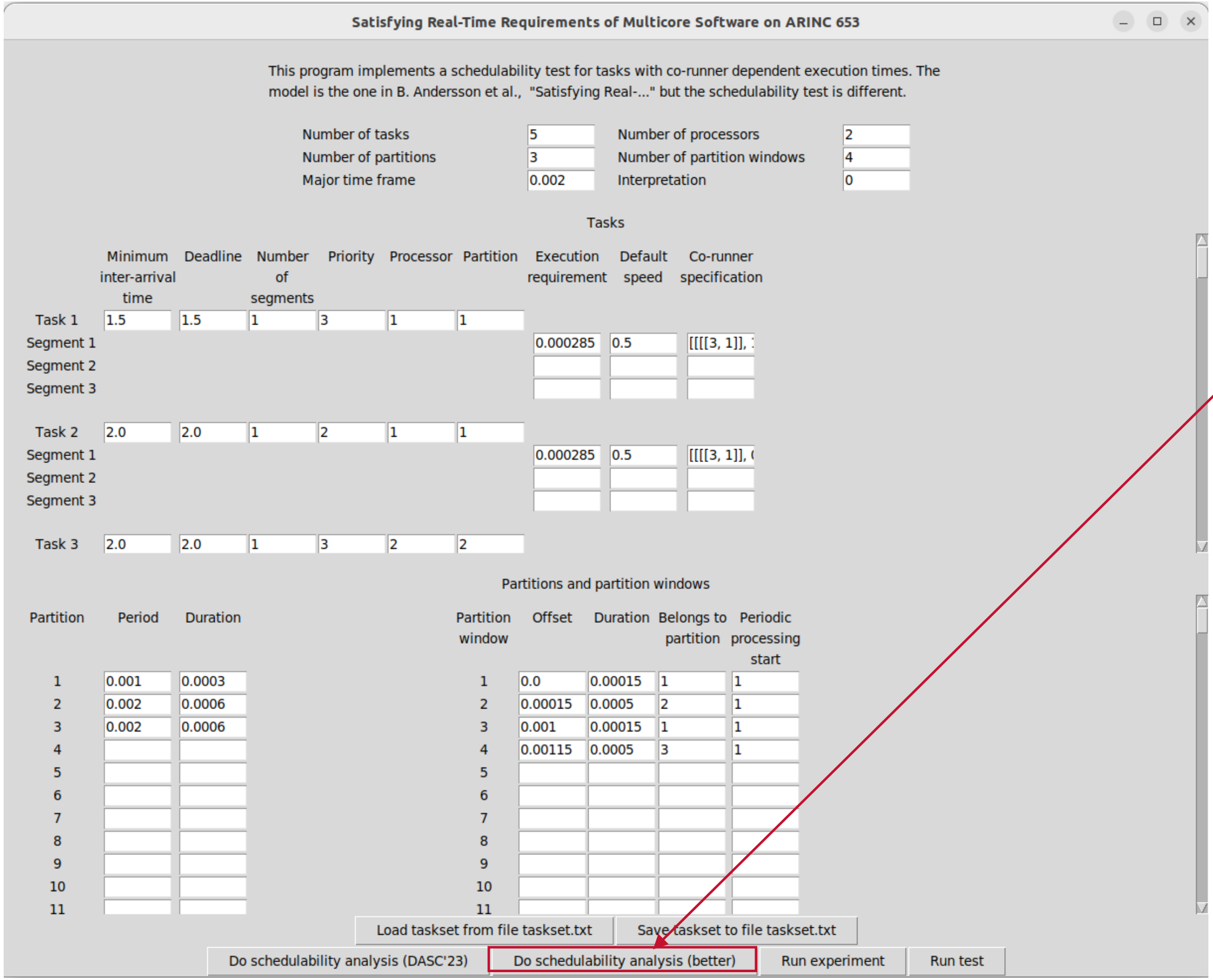
Carnegie Mellon University Software Engineering Institute



Our New Tool

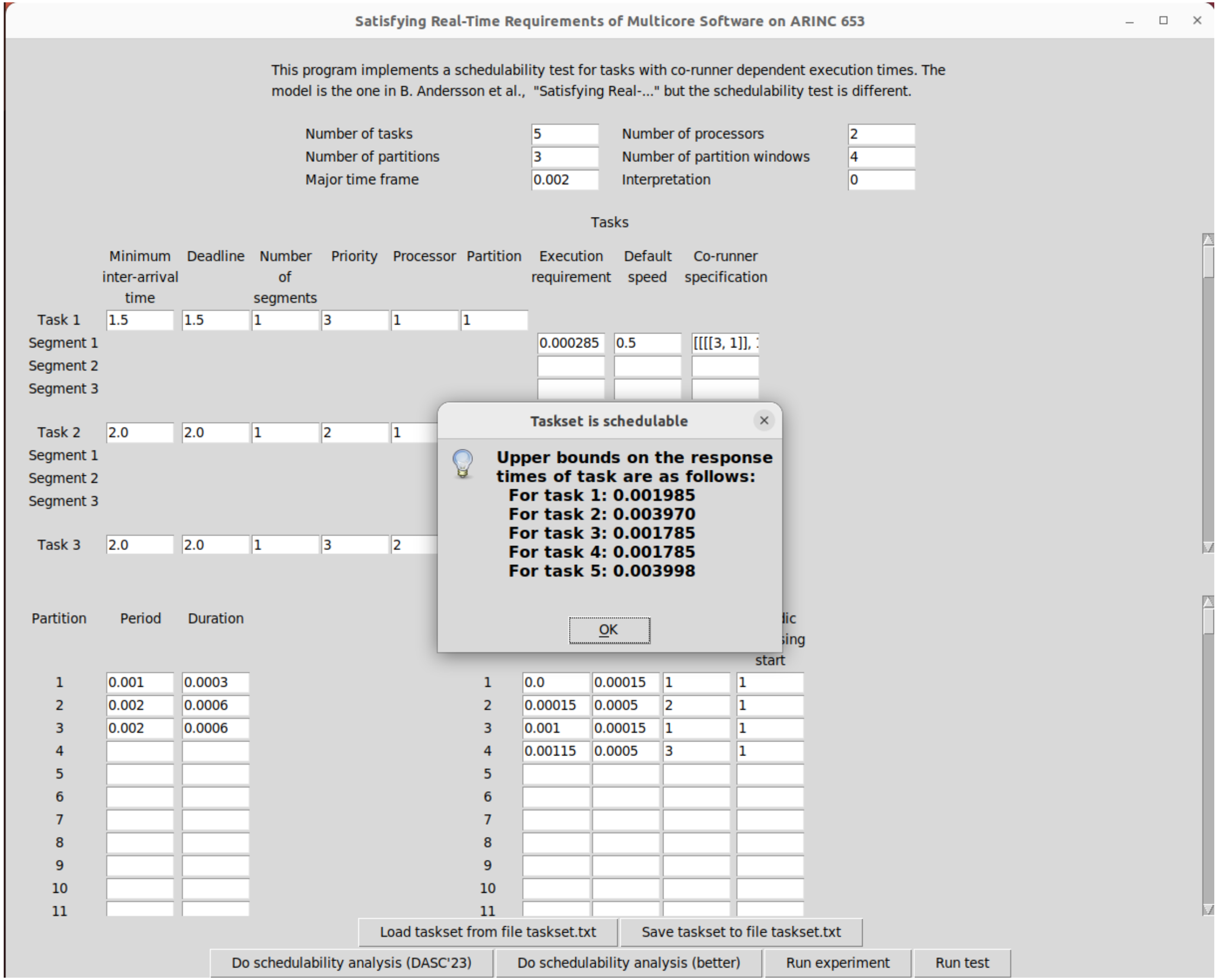


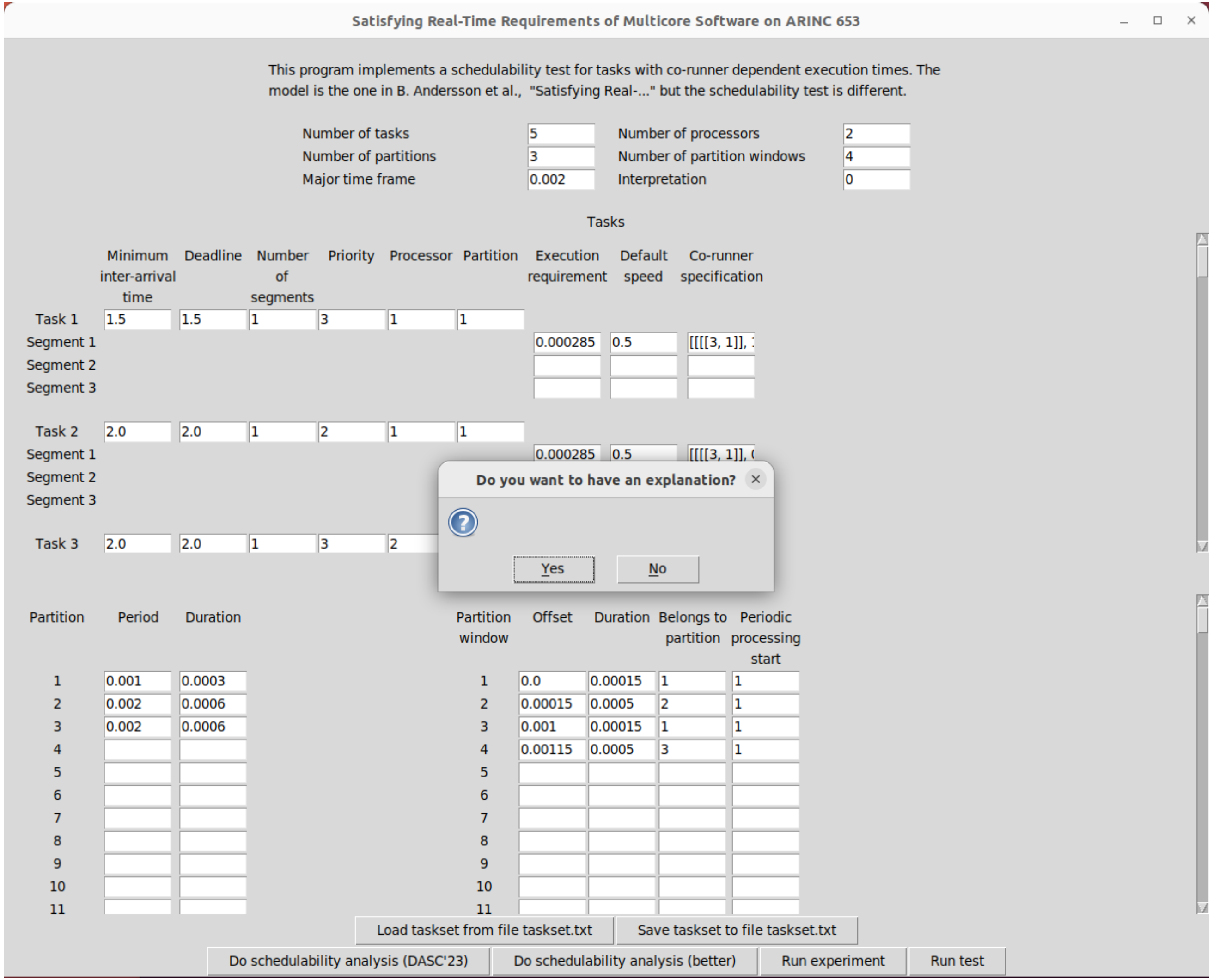
Enter data to describe system.

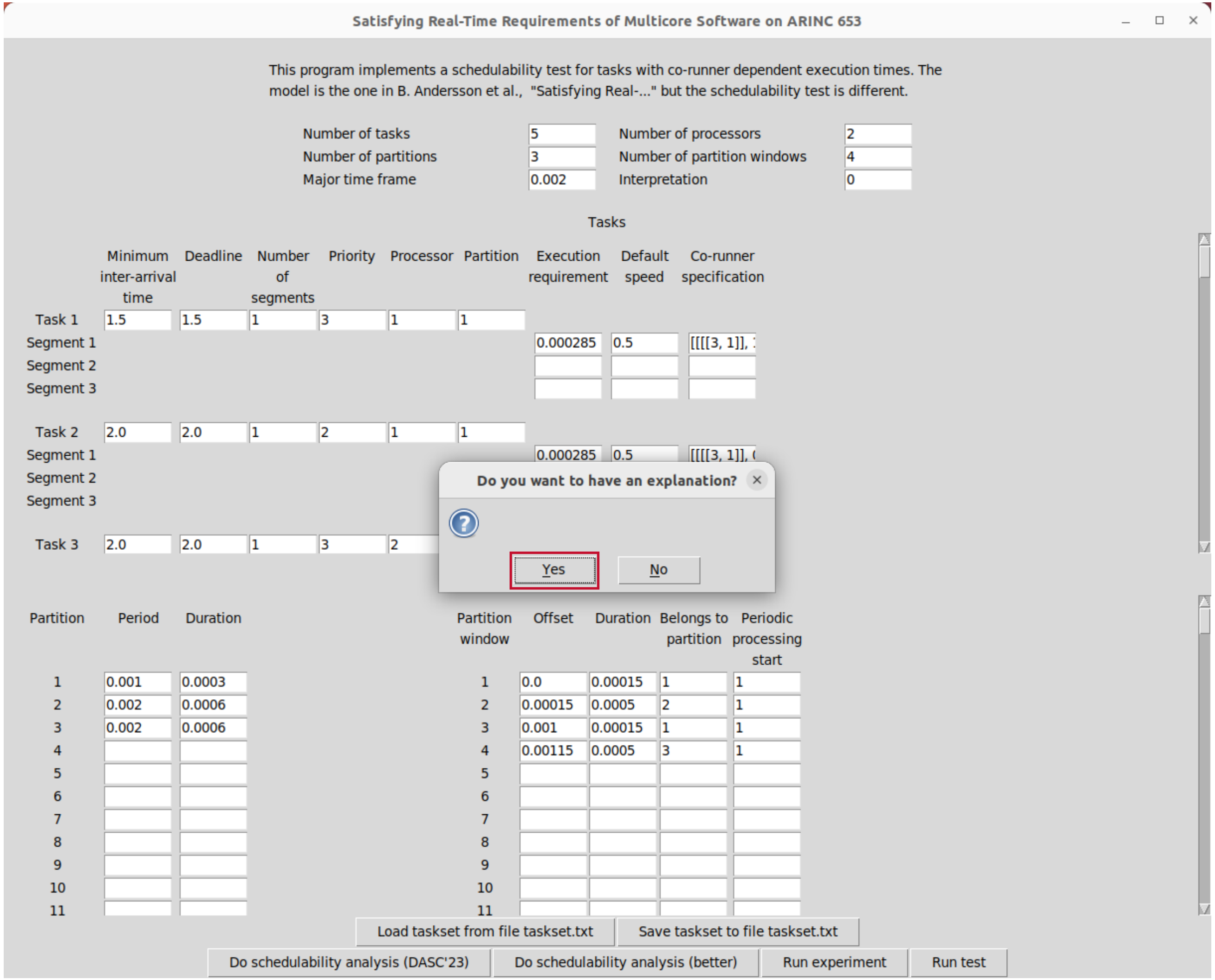


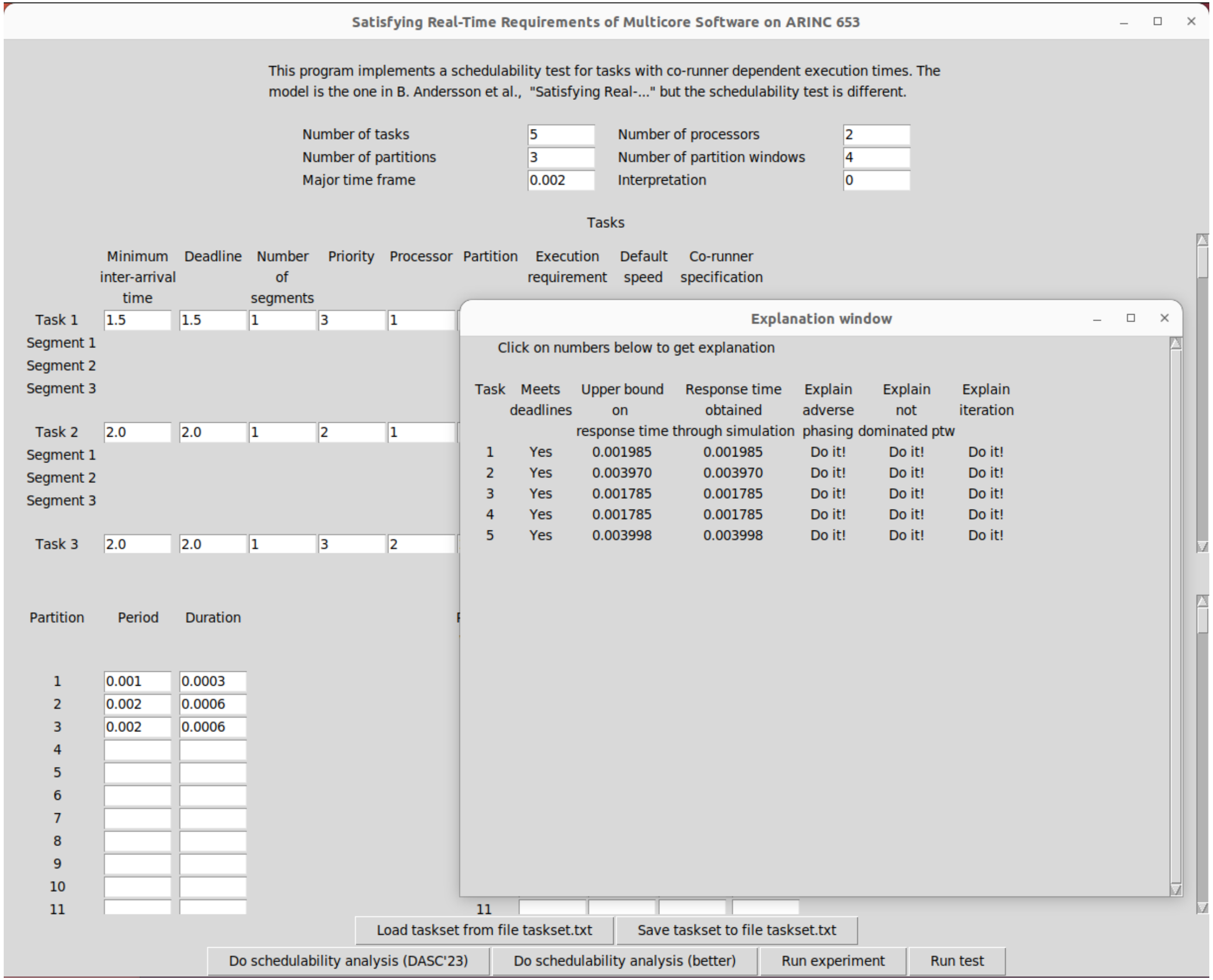
Click button.

Carnegie Mellon University Software Engineering Institute

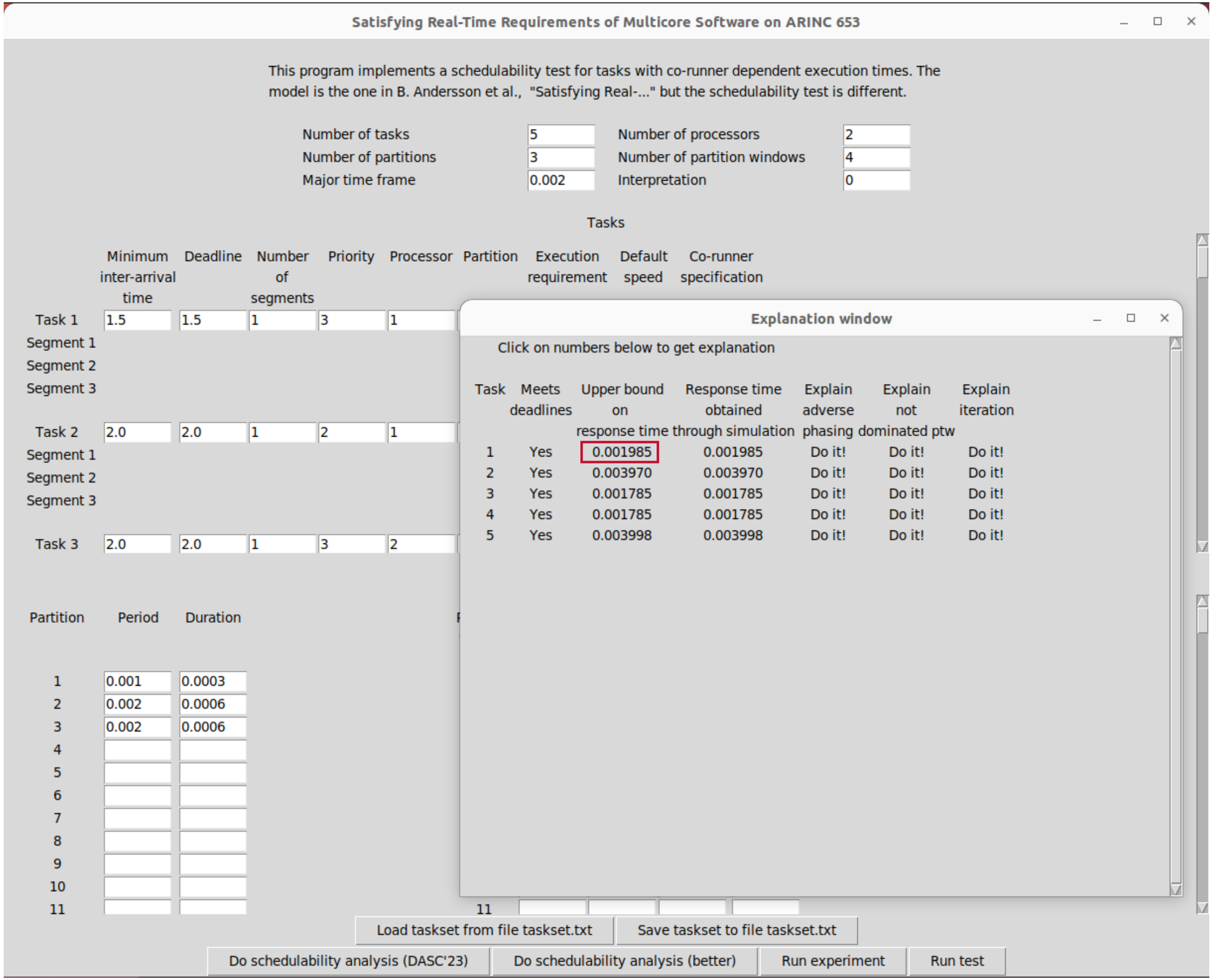


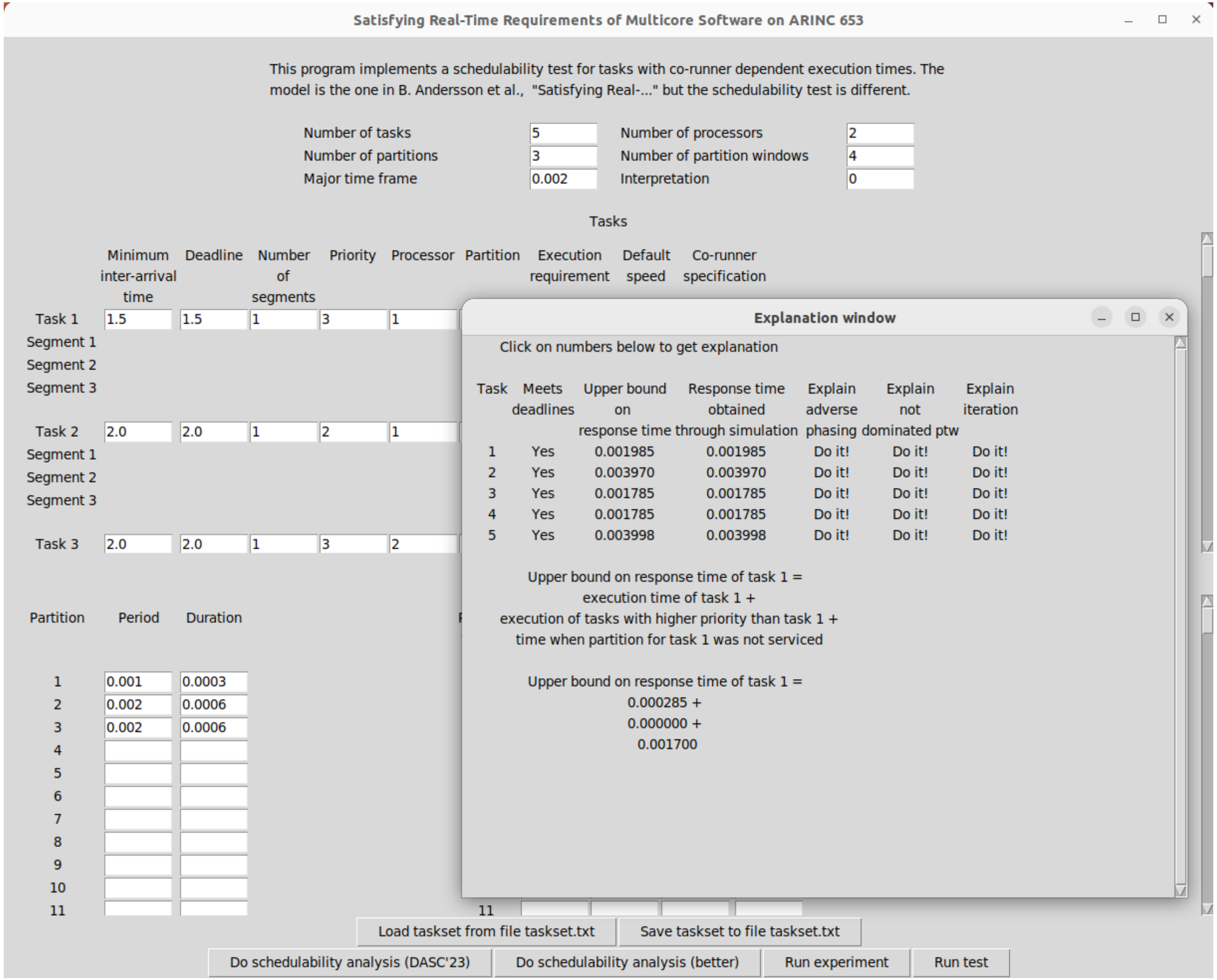

Satisfying Real-Time Requirements of Multicore Software on ARINC 653
This program implements a schedulability test for tasks with co-runner dependent execution times. The model is the one in B. Andersson et al., "Satisfying Real-..." but the schedulability test is different.
Number of tasks 5
Number of partitions 3
Major time frame 0.002
Number of processors 2
Number of partition windows 4
Interpretation 0
Tasks
Minimum inter-arrival time
Deadline
Number of segments
Priority
Processor
Partition
Execution requirement
Default speed
Co-runner specification
Task 1 1.5 1.5 1 3 1 1
Segment 1 0.000285 0.5 [[[3, 1]],
Segment 2
Segment 3
Task 2 2.0 2.0 1 2 1 1
Segment 1 0.000285 0.5 [[[3, 1]],
Segment 2
Segment 3
Task 3 2.0 2.0 1 3 2
Do you want to have an explanation?
Yes
No
Partition
Period
Duration
1 0.001 0.0003
2 0.002 0.0006
3 0.002 0.0006
4
5
6
7
8
9
10
11
Partition window
Offset
Duration
Belongs to partition
Periodic processing start
1 0.0 0.00015 1 1
2 0.00015 0.0005 2 1
3 0.001 0.00015 1 1
4 0.00115 0.0005 3 1
5
6
7
8
9
10
11
Load taskset from file taskset.txt
Save taskset to file taskset.txt
Do schedulability analysis (DASC'23)
Do schedulability analysis (better)
Run experiment
Run test

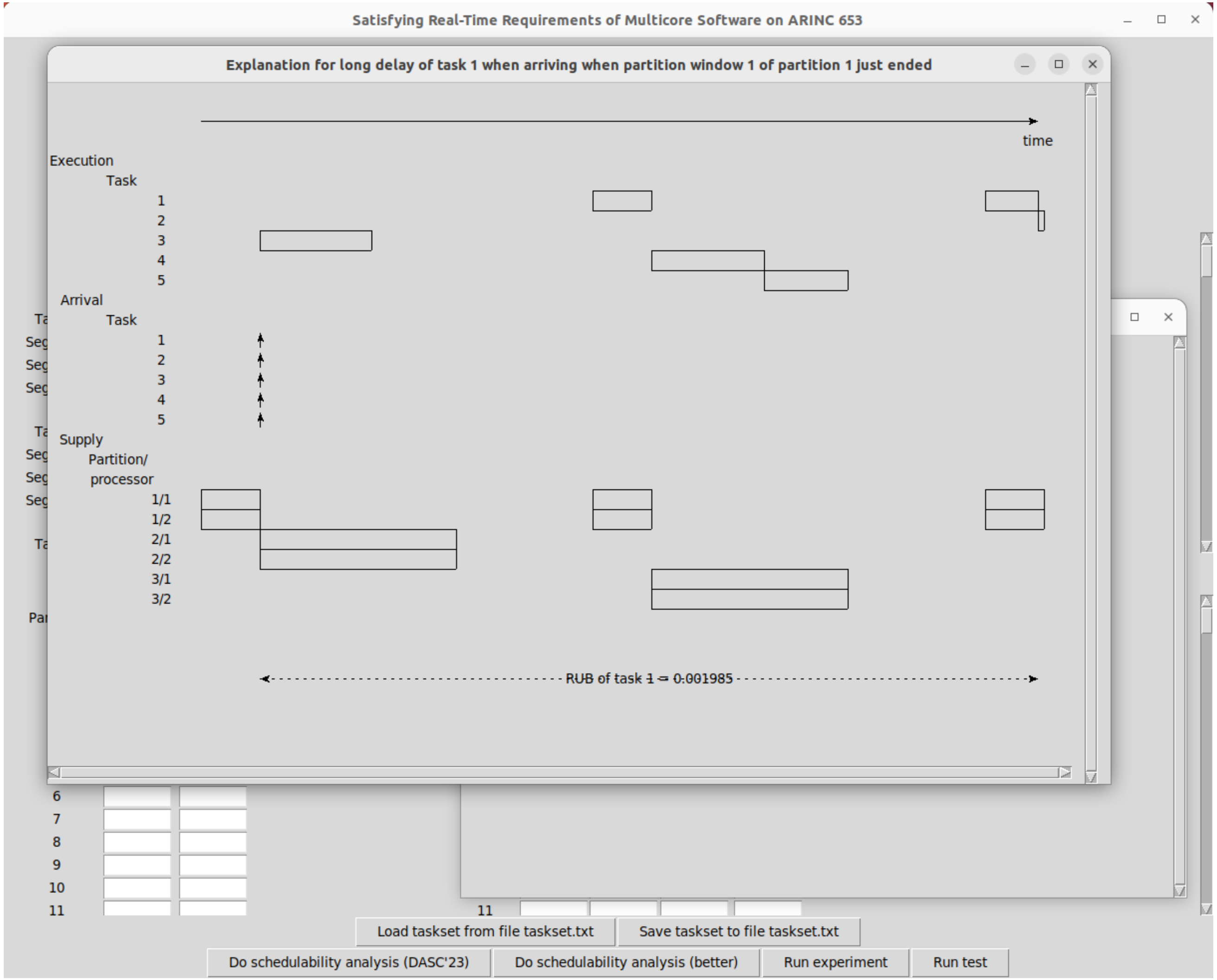

Satisfying Real-Time Requirements of Multicore Software on ARINC 653
This program implements a schedulability test for tasks with co-runner dependent execution times. The model is the one in B. Andersson et al., "Satisfying Real-..." but the schedulability test is different.
Number of tasks 5
Number of partitions 3
Major time frame 0.002
Number of processors 2
Number of partition windows 4
Interpretation 0
Tasks
Minimum inter-arrival time
Deadline
Number of segments
Priority
Processor
Partition
Execution requirement
Default speed
Co-runner specification
Task 1 1.5 1.5 1 3 1 1
Segment 1 0.000285 0.5 [[[[3, 1]],
Segment 2
Segment 3
Task 2 2.0 2.0 1 2 1 1
Segment 1 0.000285 0.5 [[[[3, 1]],
Segment 2
Segment 3
Task 3 2.0 2.0 1 3 2
Do you want to have an explanation?
Yes
No
Partition Period Duration
1 0.001 0.0003
2 0.002 0.0006
3 0.002 0.0006
Partition window Offset Duration Belongs to partition Periodic processing start
1 0.0 0.00015 1 1
2 0.00015 0.0005 2 1
3 0.001 0.00015 1 1
4 0.00115 0.0005 3 1
Load taskset from file taskset.txt
Save taskset to file taskset.txt
Do schedulability analysis (DASC'23)
Do schedulability analysis (better)
Run experiment
Run test


Satisfying Real-Time Requirements of Multicore Software on ARINC 653
This program implements a schedulability test for tasks with co-runner dependent execution times. The model is the one in B. Andersson et al., "Satisfying Real-..." but the schedulability test is different.
Number of tasks 5
Number of processors 2
Number of partitions 3
Number of partition windows 4
Major time frame 0.002
Interpretation 0
Tasks
Minimum inter-arrival time
Deadline
Number of segments
Priority
Processor
Partition
Execution requirement
Default speed
Co-runner specification
Task 1 1.5 1.5 1 3 1
Segment 1
Segment 2
Segment 3
Task 2 2.0 2.0 1 2 1
Segment 1
Segment 2
Segment 3
Task 3 2.0 2.0 1 3 2
Partition
Period
Duration
1 0.001 0.0003
2 0.002 0.0006
3 0.002 0.0006
4
5
6
7
8
9
10
11
Explanation window
Click on numbers below to get explanation
Task
Meets deadlines
Upper bound on response time
Response time obtained through simulation
Explain adverse phasing
Explain not dominated ptw
Explain iteration
1 Yes 0.001985 0.001985 Do it! Do it! Do it!
2 Yes 0.003970 0.003970 Do it! Do it! Do it!
3 Yes 0.001785 0.001785 Do it! Do it! Do it!
4 Yes 0.001785 0.001785 Do it! Do it! Do it!
5 Yes 0.003998 0.003998 Do it! Do it! Do it!
11
Load taskset from file taskset.txt
Save taskset to file taskset.txt
Do schedulability analysis (DASC'23)
Do schedulability analysis (better)
Run experiment
Run test

Carnegie Mellon University Software Engineering Institute




Satisfying Real-Time Requirements of Multicore Software on ARINC 653
This program implements a schedulability test for tasks with co-runner dependent execution times. The model is the one in B. Andersson et al., "Satisfying Real-..." but the schedulability test is different.
Number of tasks 5
Number of partitions 3
Major time frame 0.002
Number of processors 2
Number of partition windows 4
Interpretation 0
Tasks
Minimum inter-arrival time
Deadline
Number of segments
Priority
Processor
Partition
Execution requirement
Default speed
Co-runner specification
Task 1 1.5 1.5 1 3 1
Segment 1
Segment 2
Segment 3
Task 2 2.0 2.0 1 2 1
Segment 1
Segment 2
Segment 3
Task 3 2.0 2.0 1 3 2
Partition
Period
Duration
1 0.001 0.0003
2 0.002 0.0006
3 0.002 0.0006
4
5
6
7
8
9
10
11
Explanation window
Click on numbers below to get explanation
Task
Meets deadlines
Upper bound on response time
Response time obtained through simulation
Explain adverse phasing
Explain not dominated ptw
Explain iteration
1 Yes 0.001985 0.001985 Do it! Do it! Do it!
2 Yes 0.003970 0.003970 Do it! Do it! Do it!
3 Yes 0.001785 0.001785 Do it! Do it! Do it!
4 Yes 0.001785 0.001785 Do it! Do it! Do it!
5 Yes 0.003998 0.003998 Do it! Do it! Do it!
Upper bound on response time of task 1 =
execution time of task 1 +
execution of tasks with higher priority than task 1 +
time when partition for task 1 was not serviced
Upper bound on response time of task 1 =
0.000285 +
0.000000 +
0.001700
Load taskset from file taskset.txt
Save taskset to file taskset.txt
Do schedulability analysis (DASC'23)
Do schedulability analysis (better)
Run experiment
Run test



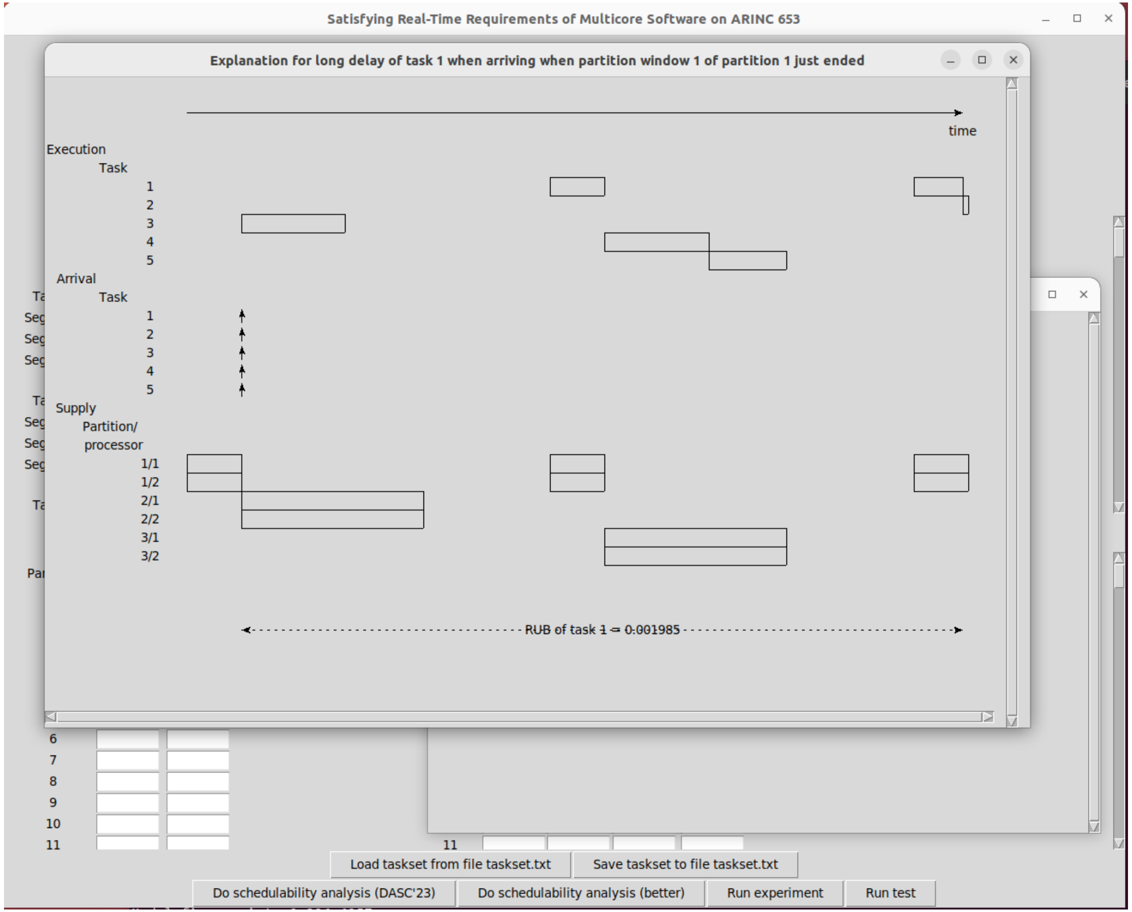
Satisfying Real-Time Requirements of Multicore Software on ARINC 653
Explanation for long delay of task 1 when arriving when partition window 1 of partition 1 just ended
time
Execution
Task
1
2
3
4
5
Arrival
Task
1
2
3
4
5
Supply
Partition/
processor
1/1
1/2
2/1
2/2
3/1
3/2
RUB of task 1 = 0.001985
6
7
8
9
10
11
11
Load taskset from file taskset.txt
Save taskset to file taskset.txt
Do schedulability analysis (DASC'23)
Do schedulability analysis (better)
Run experiment
Run test

Evaluation of Our Tool

Three different groups of people

Certification authorities and experienced software developers of safety-critical systems

These groups are representative users of our tool, and they include: Galois, AvMC, AFSEC.



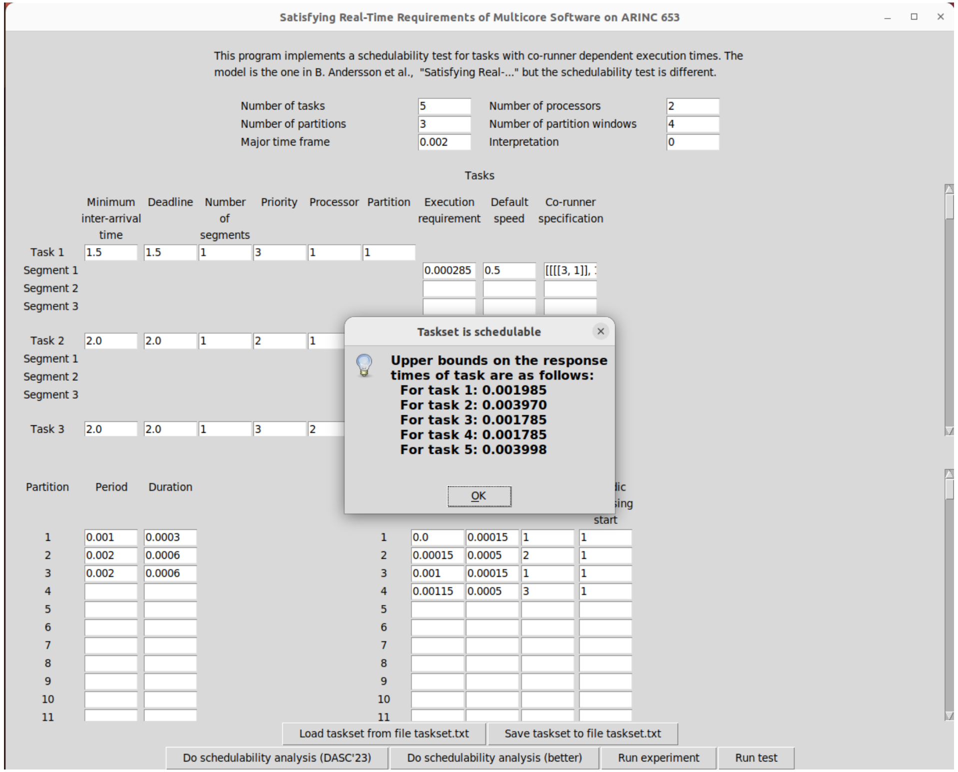
Number of persons: 35

Number of persons who found our explanations helpful to gain confidence in output of FM tool: 31

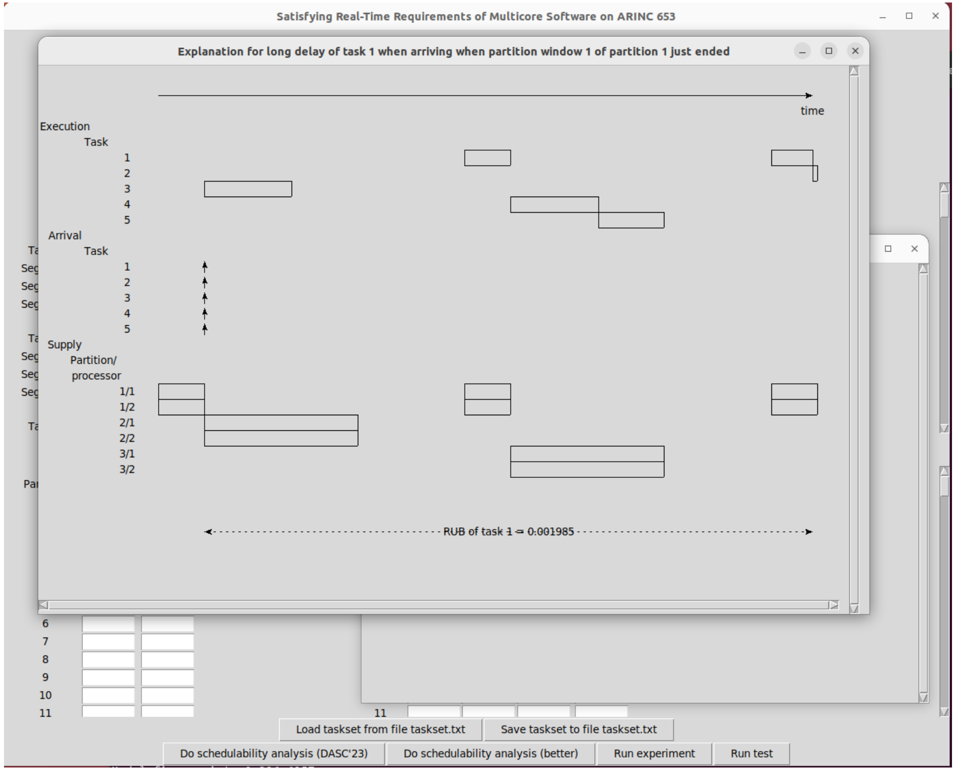
31/35 ≈ 88.5%

Carnegie Mellon University
Software Engineering Institute

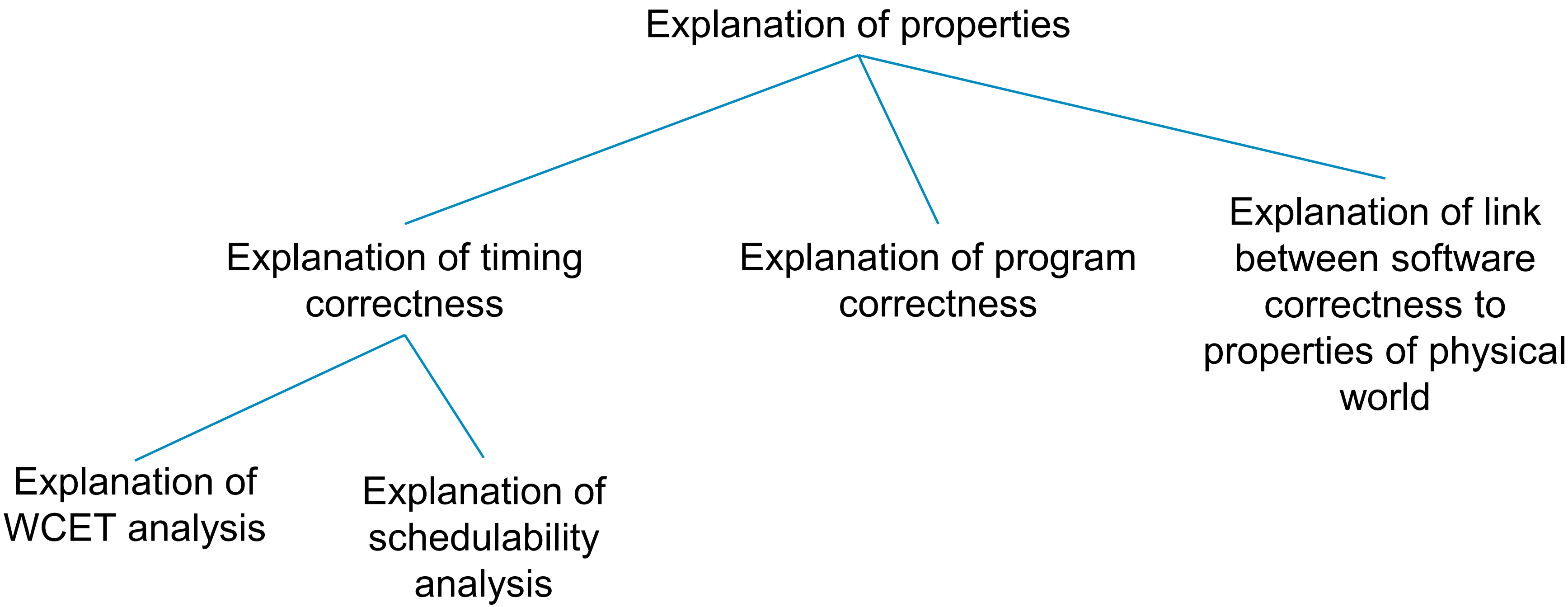
Without Explanation



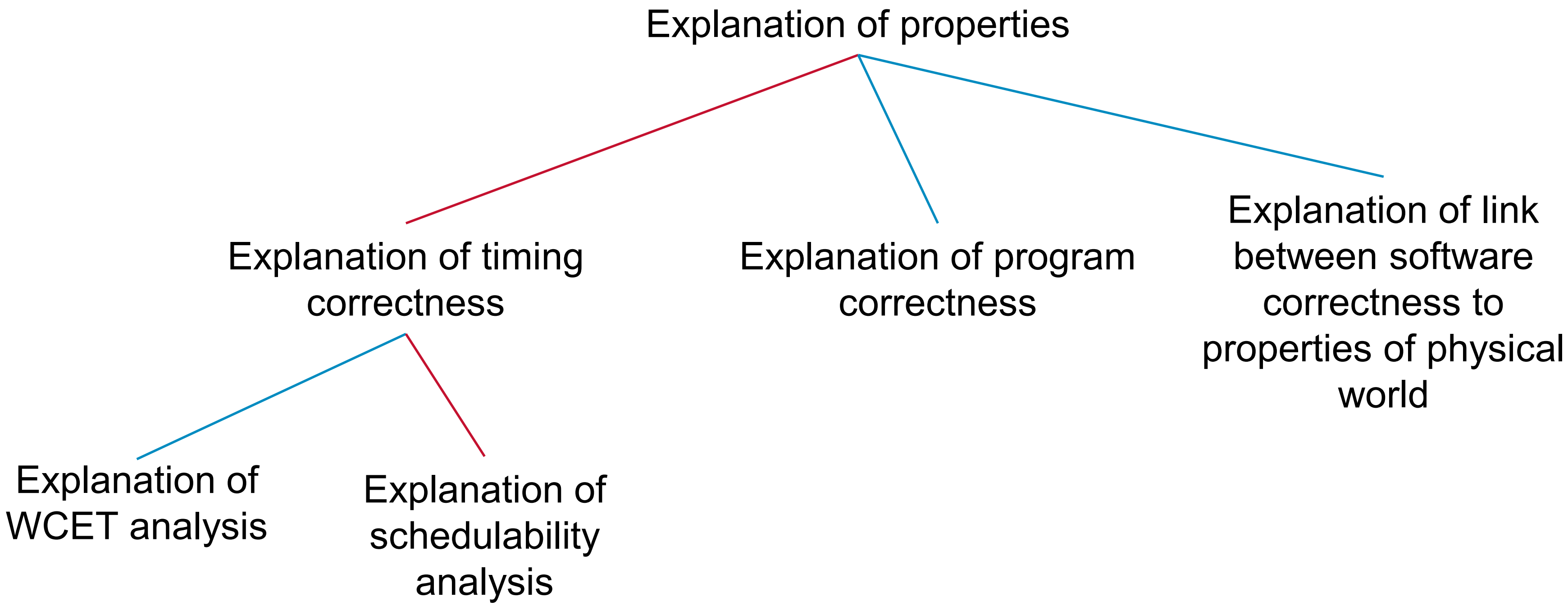
With Explanation



The Big Picture and this Project



The Big Picture and this Project



This project

The Big Picture and this Project

Carnegie Mellon University Software Engineering Institute

Explanation of Yes or No

Explanation of Yes

Explanation of No

The Big Picture and this Project

Carnegie Mellon University
Software Engineering Institute

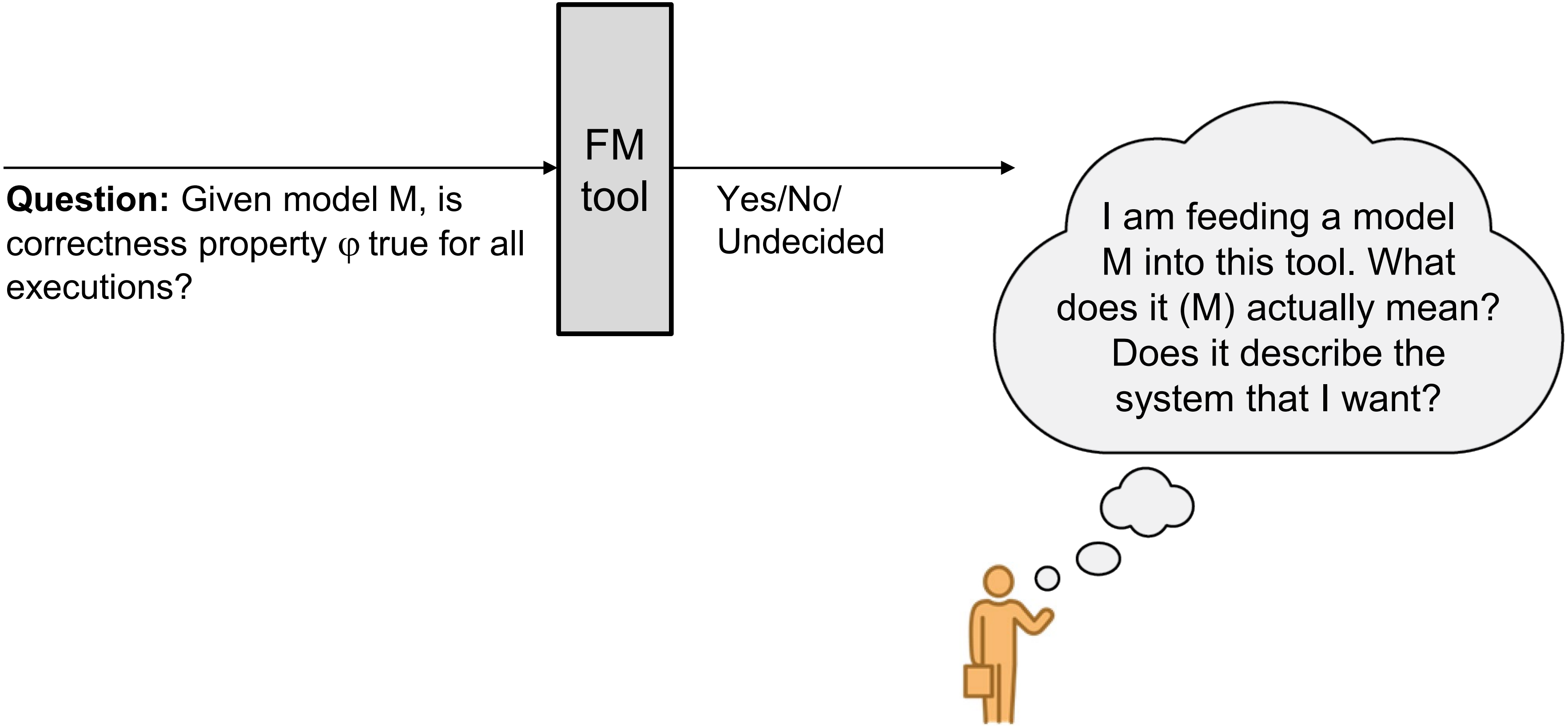
Explanation of Yes or No

Explanation of Yes

Explanation of No

This project

The Big Picture and this Project



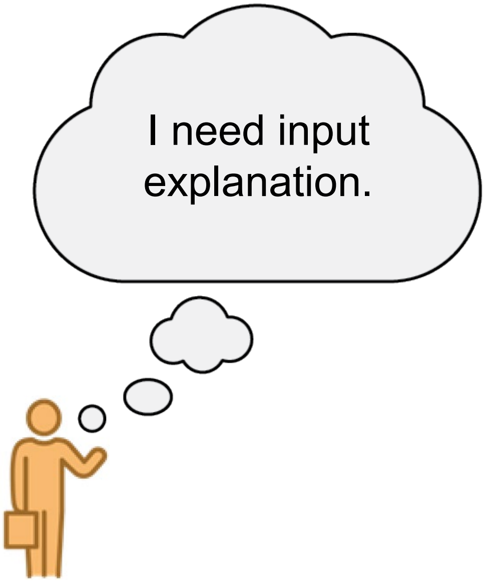
The Big Picture and this Project

Carnegie Mellon University Software Engineering Institute

Question: Given model M, is correctness property φ true for all executions?

→ FM tool →

Yes/No/
Undecided



The Big Picture and this Project

Question: Given model M, is correctness property φ true for all executions?

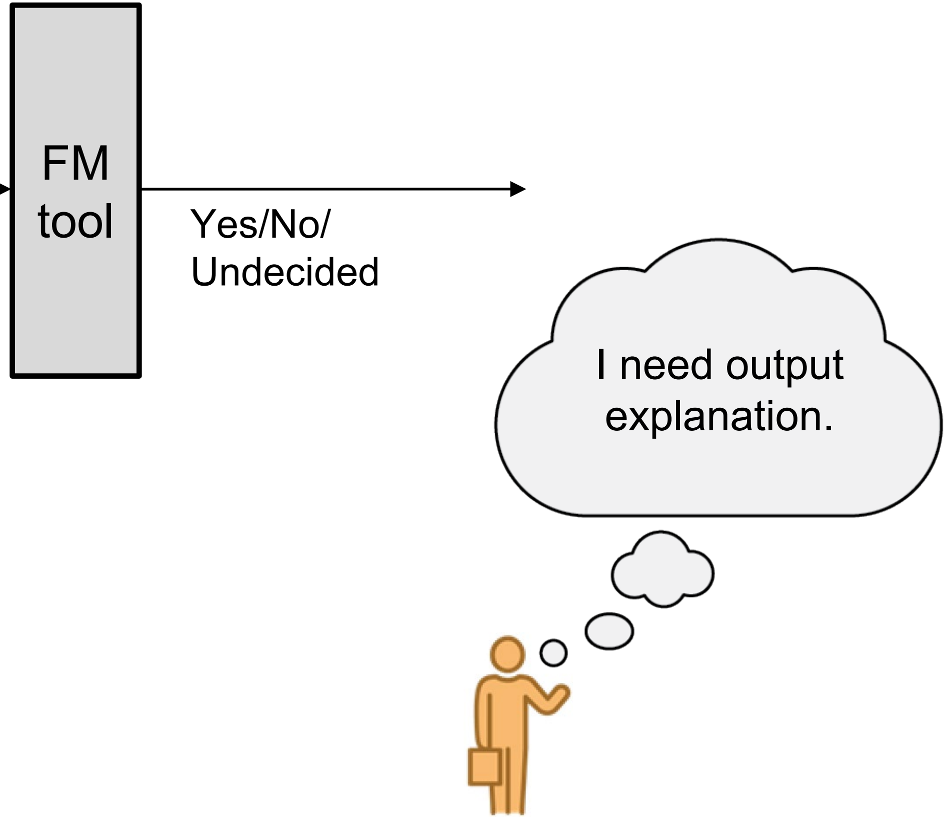
FM tool

Yes/No/
Undecided

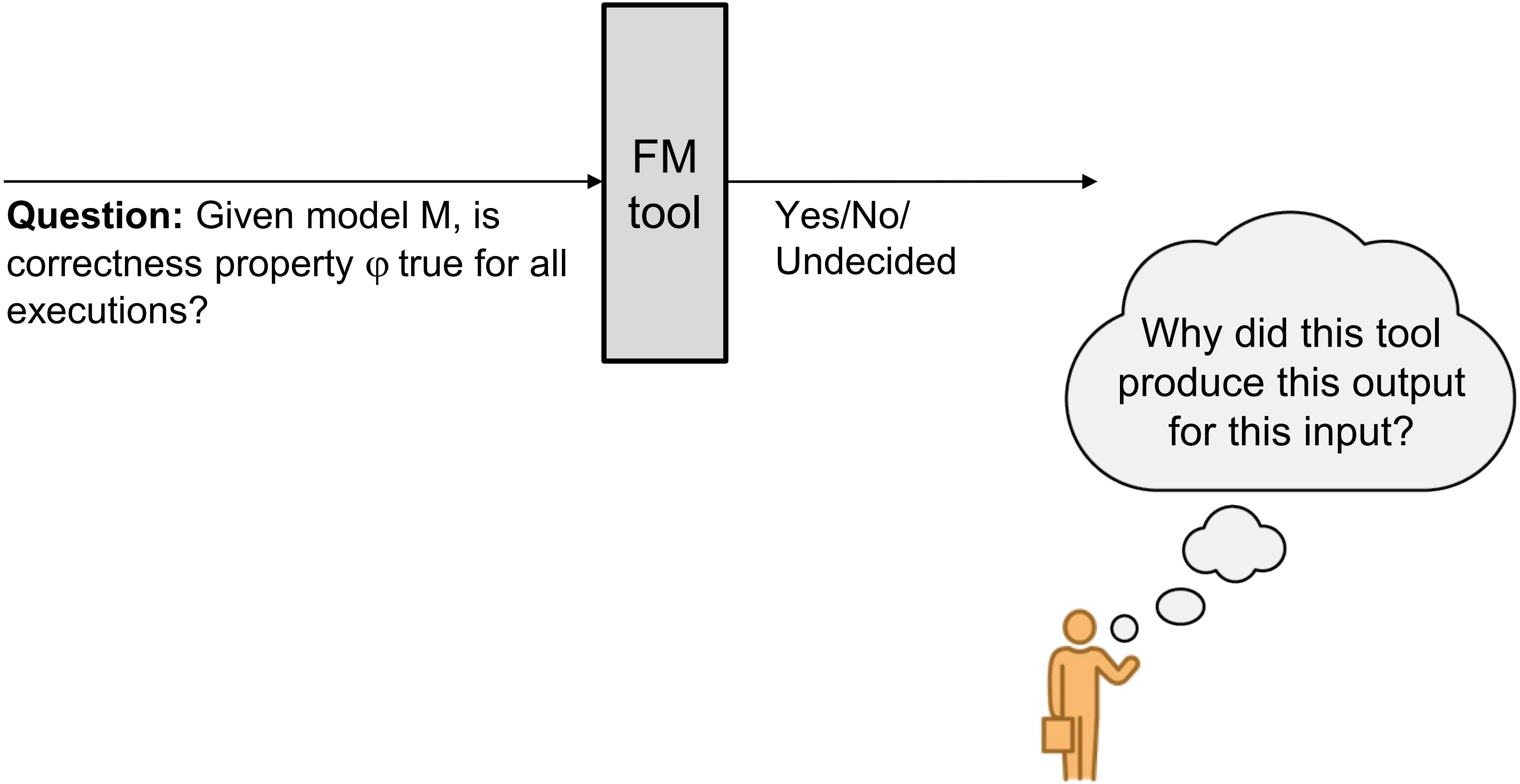
This tool gives me an answer to the question whether φ holds. What does it actually mean? Does it describe the correctness property that I care about?

The Big Picture and this Project

Question: Given model M, is correctness property φ true for all executions?

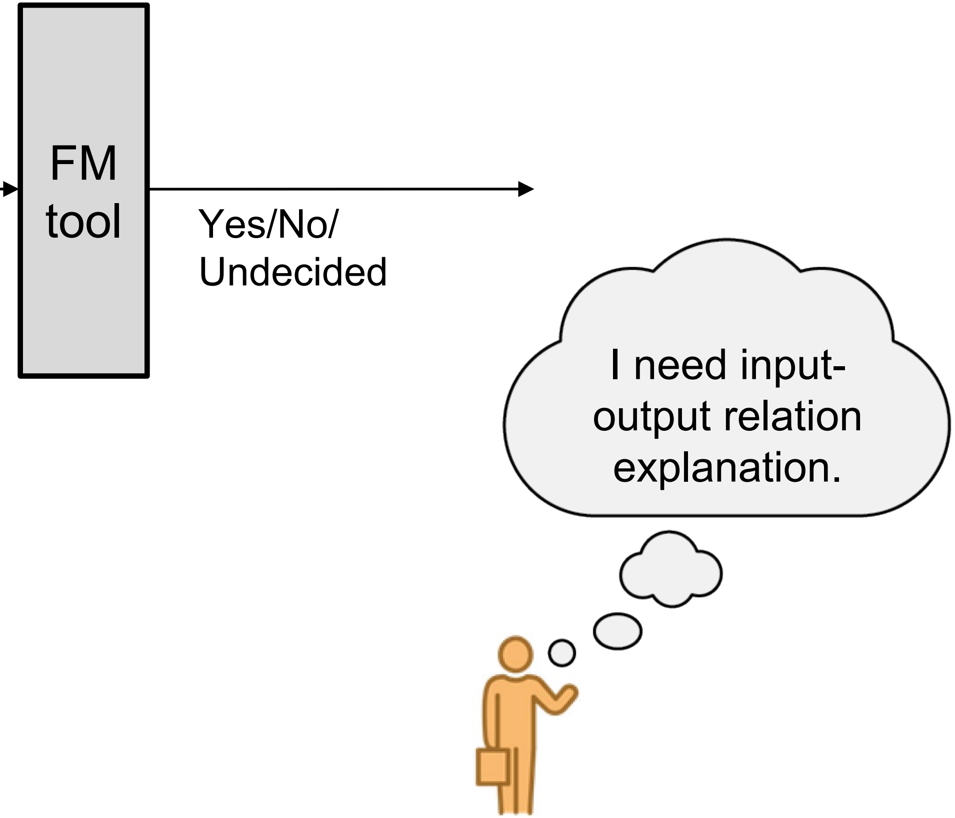


The Big Picture and this Project



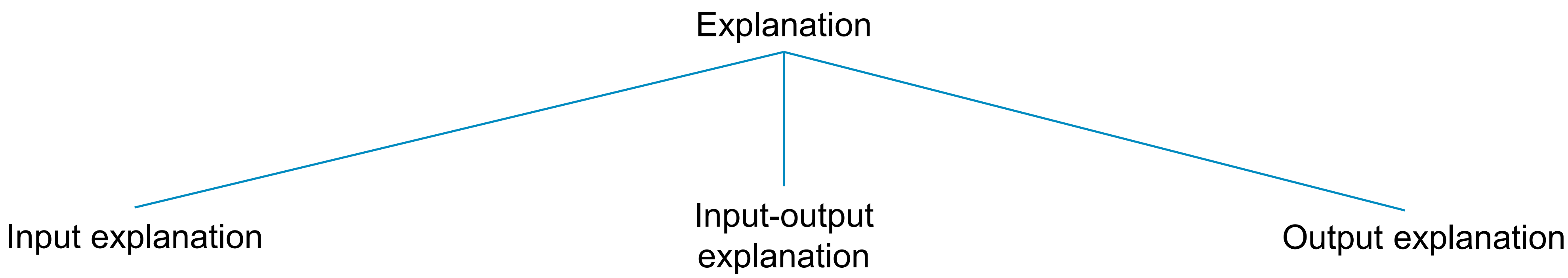
The Big Picture and this Project

Question: Given model M, is correctness property φ true for all executions?



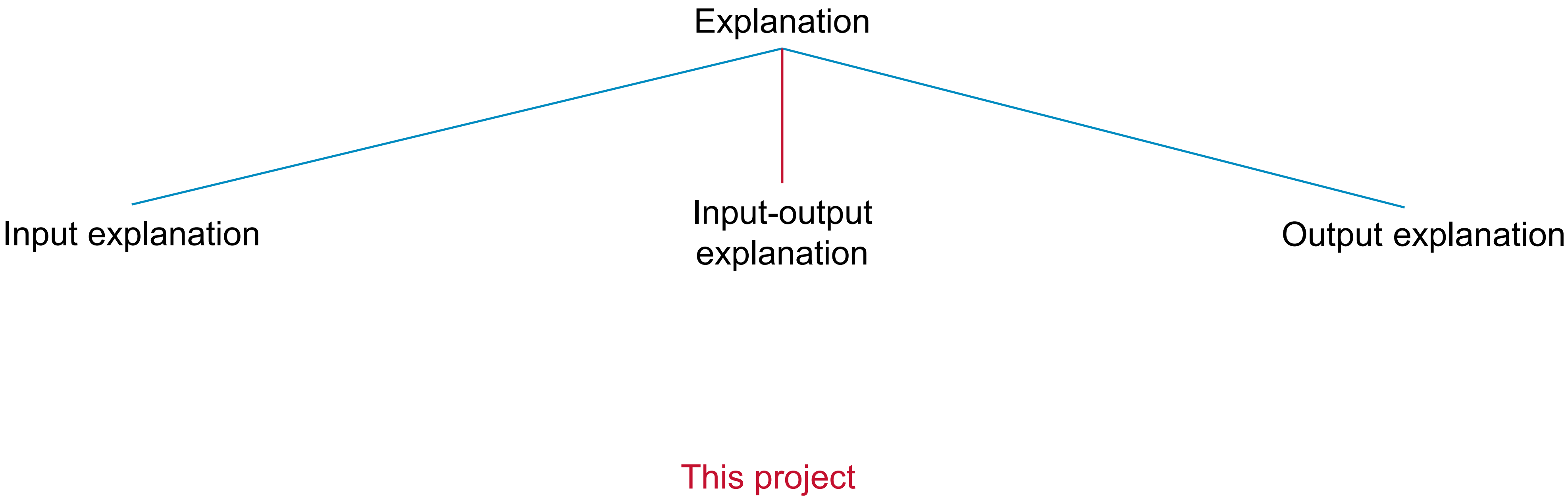
Carnegie Mellon University Software Engineering Institute

The Big Picture and this Project



Carnegie Mellon University Software Engineering Institute

The Big Picture and this Project



Conclusion

- Cert expensive; testing expensive
- Formal methods (FM) potential make it less expensive
- Practitioners do not trust FM

Theorem 4. *If* $\forall \tau_i \in \tau \ \forall q \in \mathrm{QSET}_i \ \exists t \in [0, D_i]$ $\mathrm{reqlpARINC653}(i,t) \leq \mathrm{sbf}(i,q,t)$, *then the system is schedulable.*

Our work

- Create Explanation Methods (XMs) for FM
- 88% believe XMs increase confidence in FMs
- Focus on timing
- Broad applicability for critical systems

Team

Bjorn Andersson

Dionisio de Niz

Mark Klein

Carnegie Mellon University Software Engineering Institute

Thanks!