CS 6156
Runtime Verification

Owolabi Legunsen

Fall 2020

On the state of software quality

The New York Times

Airline Blames Bad Software in San Francisco Crash

GOOGLE SELF-DRIVING CAR CAUSED FREEWAY CRASH AFTER ENGINEER MODIFIED ITS SOFTWARE

BY JASON MURDOCK ON 10/17/18 AT 11:34 AM

Newsweek

~9% of 2017 US GDP

Report: Software failure caused $1.7 trillion in financial losses in 2017

TechRepublic.

Software testing company Tricentis found that retail and consumer technology were the areas most affected, while software failures in public service and healthcare were down from the previous year.

By Scott Matteson | January 26, 2018, 7:54 AM PST

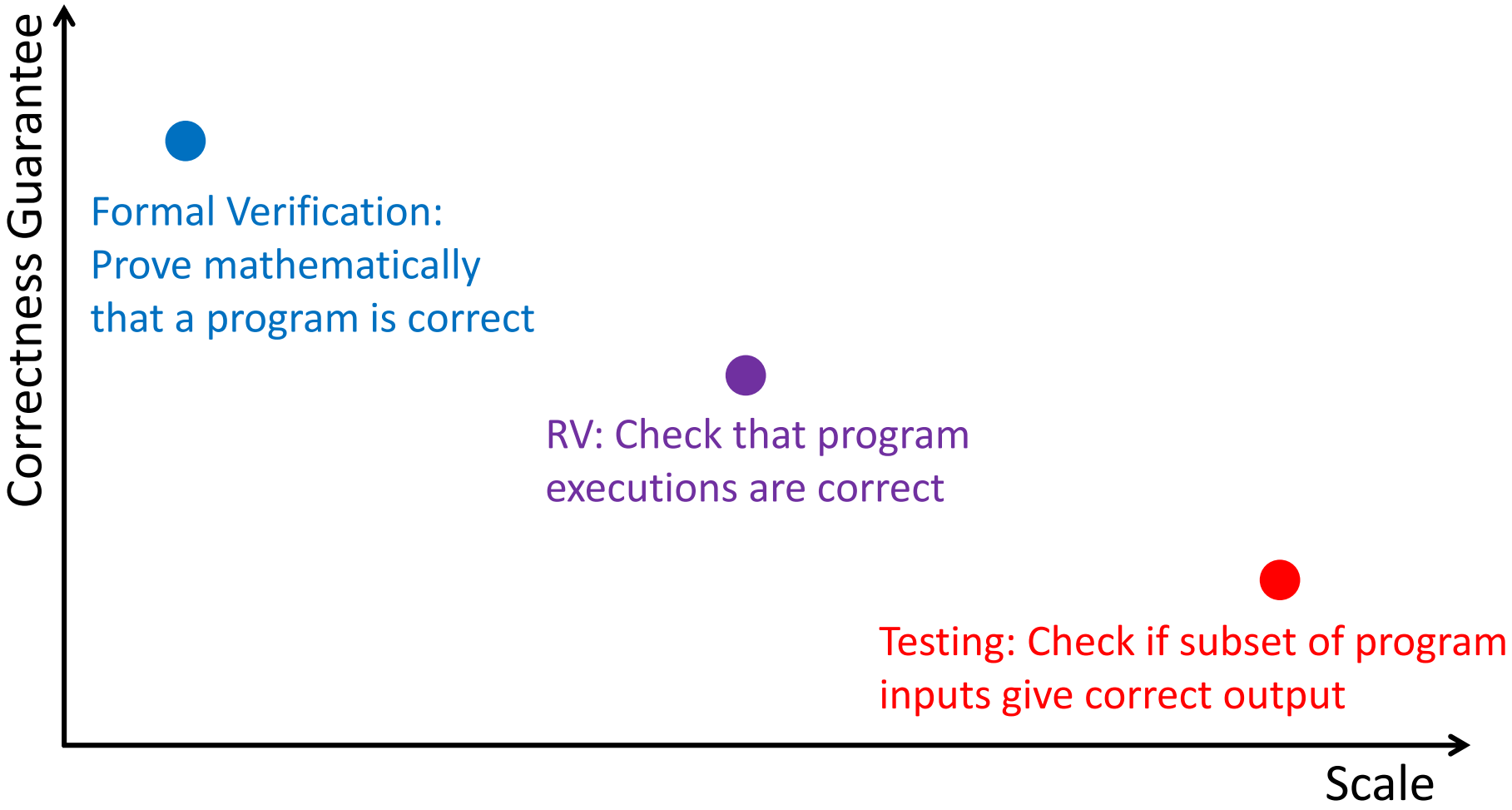
Hard Questions Raised When A Software 'Glitch' Takes Down An Airliner

Forbes

Intro to Runtime Verification (RV)

- RV is an emerging discipline for checking that ~~software~~ system executions satisfy some specifications
  - e.g., this is one of only ~3 RV courses in the world
- RV brings the mathematical rigor of formal verification to everyday ~~software~~ system development

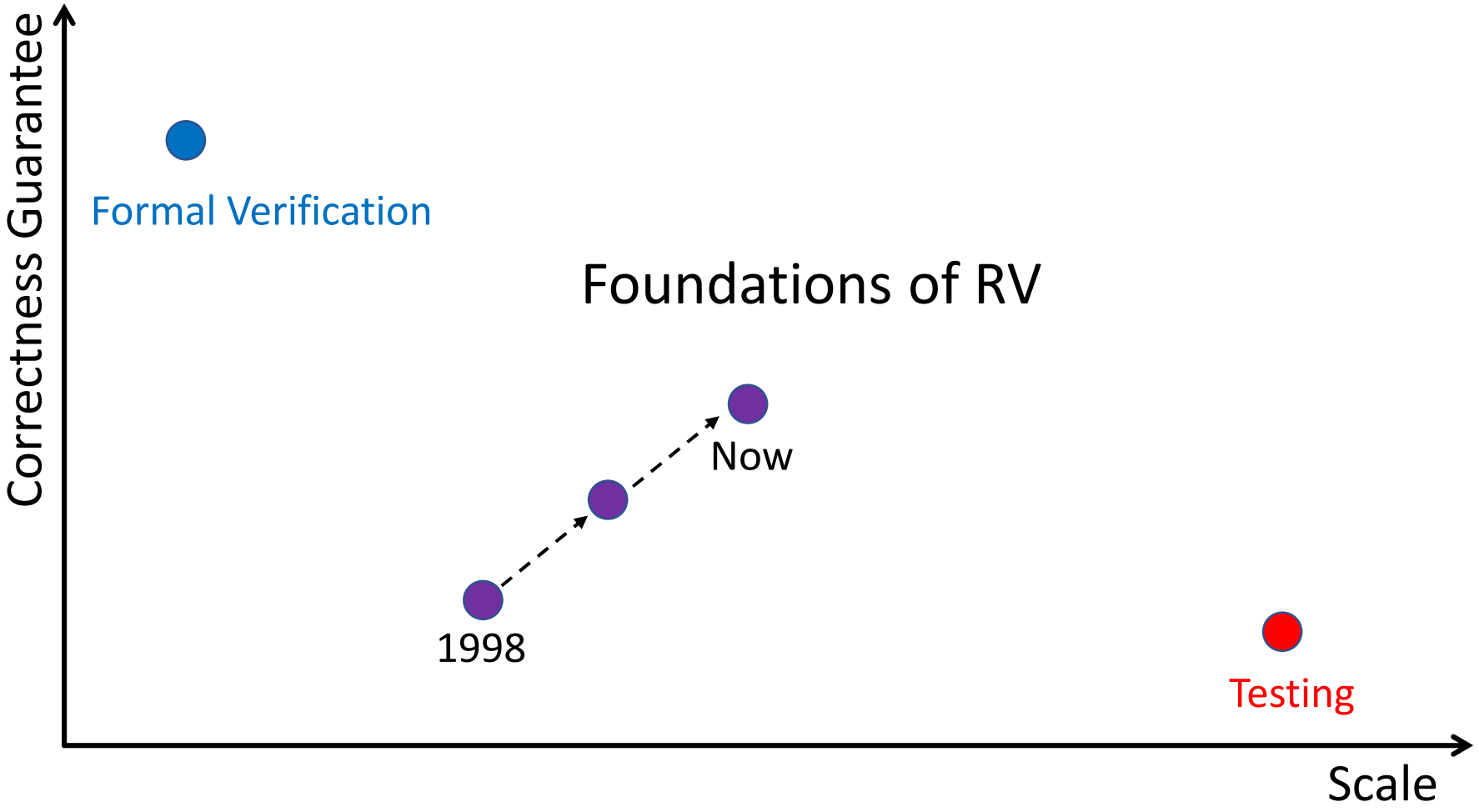
One reason why RV is appealing



About me

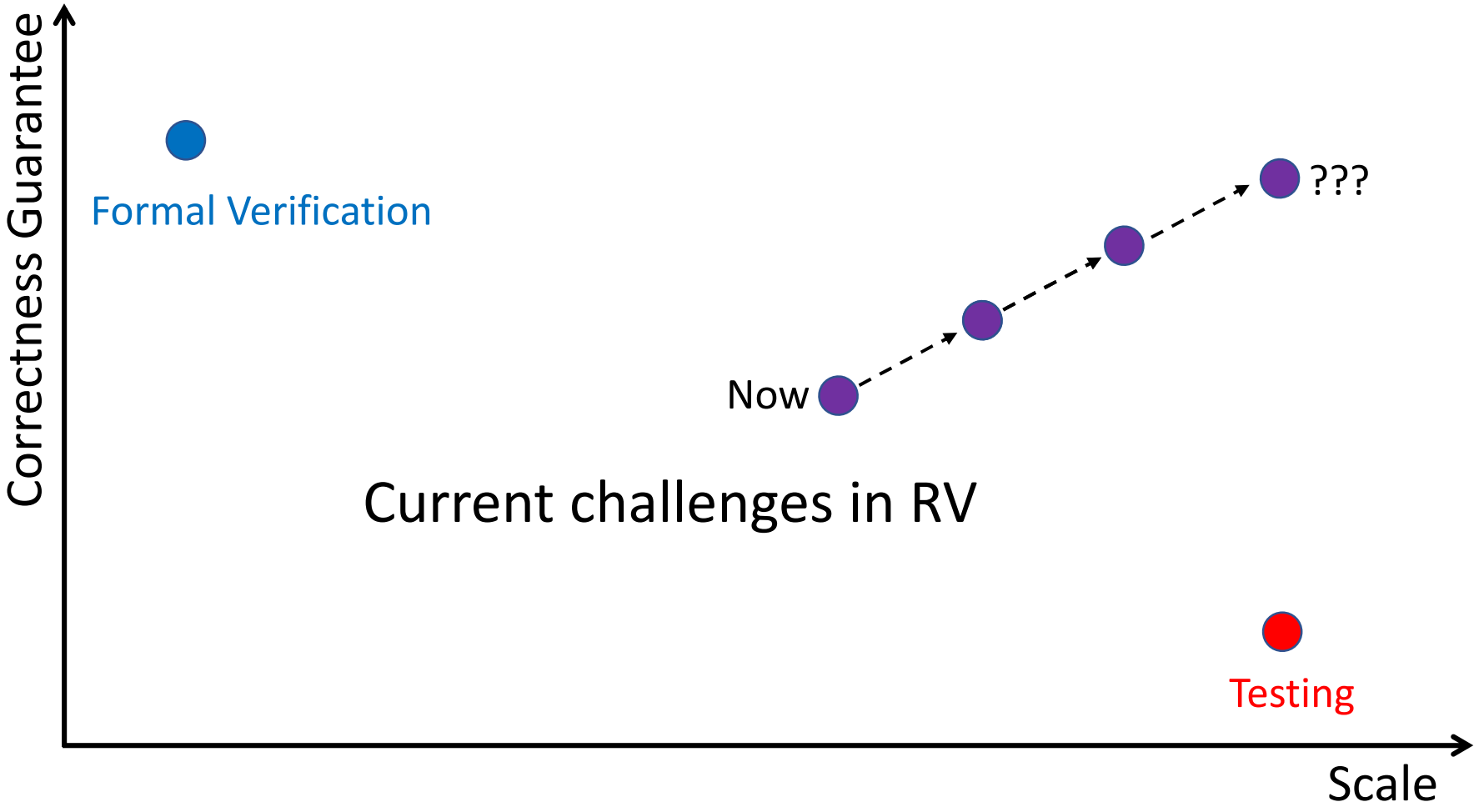
- I work on software testing and applied formal methods like RV
- I received my PhD from UIUC in 2019
  - thesis: incremental RV during software testing
- I found my thesis topic while trying to streamline work with my two co-advisors

What this course is about (1)



How does RV work? How to scale RV to large software?

What this course is about (2)



Can RV scale like testing and have guarantees of verification?

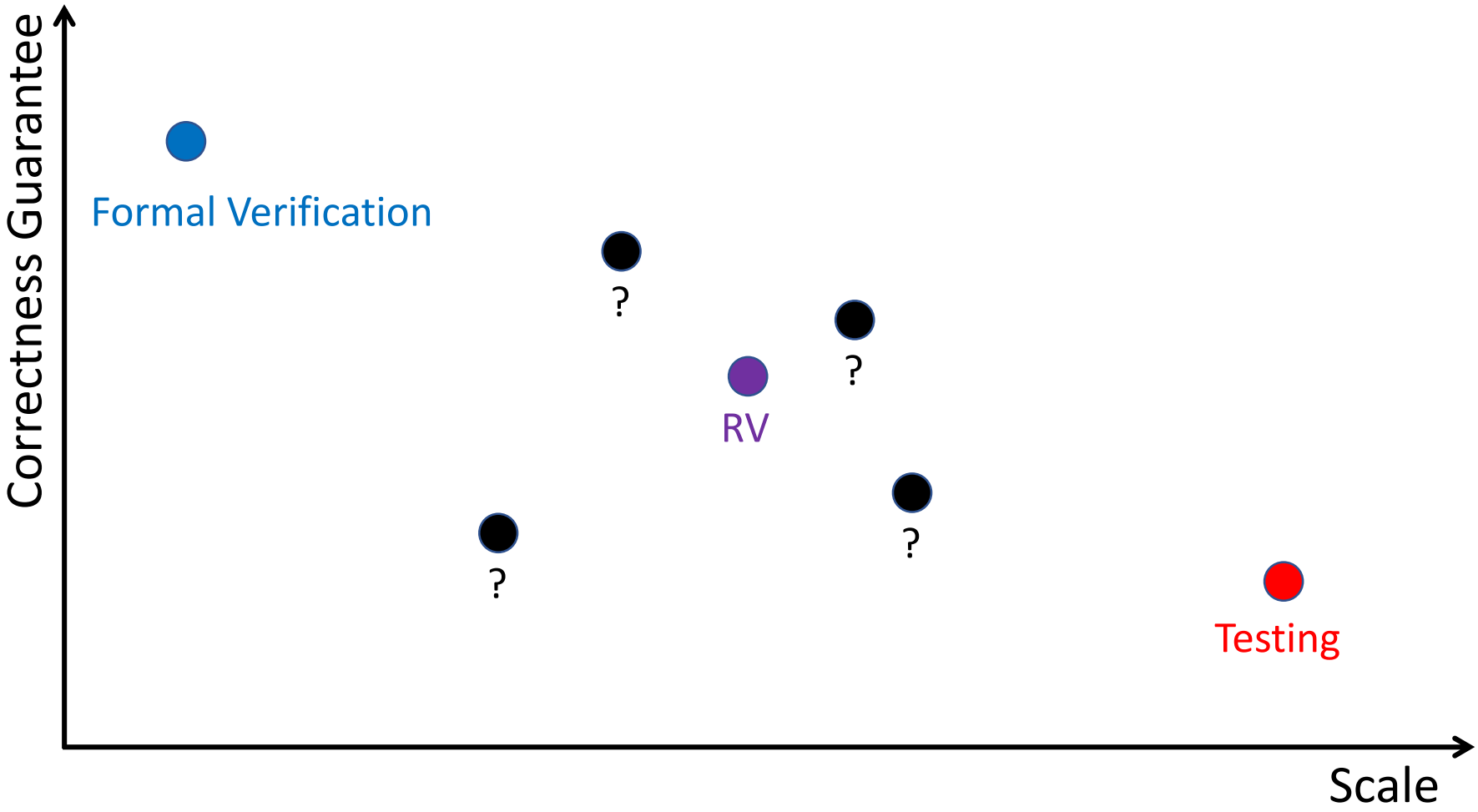
What this course is about (3)

- Hands-on exposure to RV
  - Learn how to use at least one RV tool
  - Apply RV to open-source software
  - Project: do research on RV or apply RV in your research
  - Figure out if RV is an area of (research) interest for you

What this course is not about

- Formal verification, proof methodology, etc.
- Learning about logic (but we will use some logics)
- Basic software engineering knowledge and skills
  - Take CS5150 or CS{?} in Spring 2020 if that's your goal

Your turn: other QA approaches?



Small group discussion (10 mins)

- Introduce yourself to people in your group
- What other QA approaches have you used or heard about?
  - What are the advantages and disadvantages of each?
- Share the results of your group discussion

What did your group discuss?

QA	Pros	Cons
Design by Contract	more assurance than testing	• writing contracts is hard • slows the runs
Bounded Model checking	needs no proof	less guarantees than verification
Random / fuzz testing	easy scales	no guarantees

What did your group discuss?

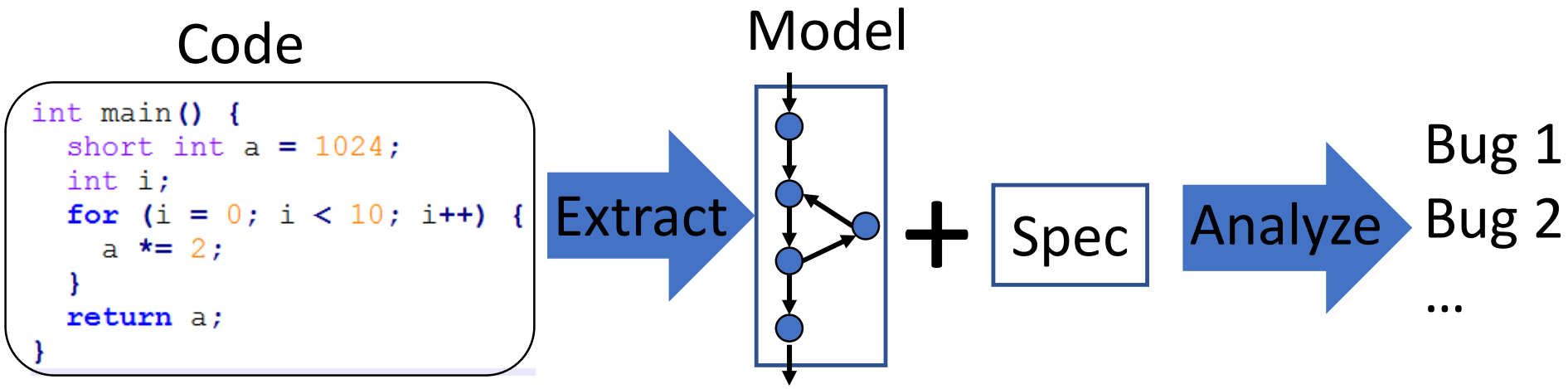
QA	Pros	Cons
Fuzzing	tests robustness	no notion of expected behavior
Code review code audit	scales helps onboarding	little guarantee
Property-based testing	developers don't have to write tests	writing generators is hard

Now that we broke the ice…

- Feel free to unmute yourself and ask questions
  - This is a small-enough discussion-based class
- Or you can post your question in the zoom chat
- At the very least, feel free to use zoom “raise hand”

Formal (static) verification

- E.g., model checking, static analysis



Pros	Cons
Good code coverage	Errors in modeling
Applied early in development	False positives
Mature and well studied	Does not scale

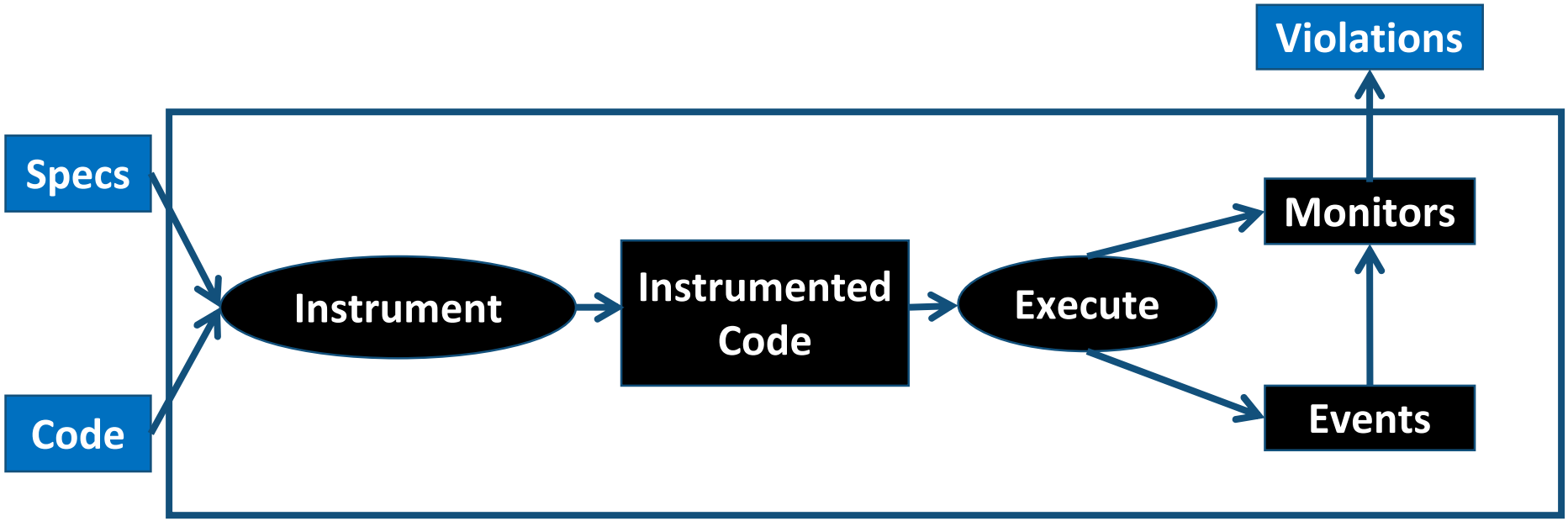
Software testing

Pros	Cons
Easier for most developers	Low code coverage
Scales well in practice	Oracle generation is hard
Leverages developer insights	High maintenance costs, e.g., obsolete tests

Runtime verification

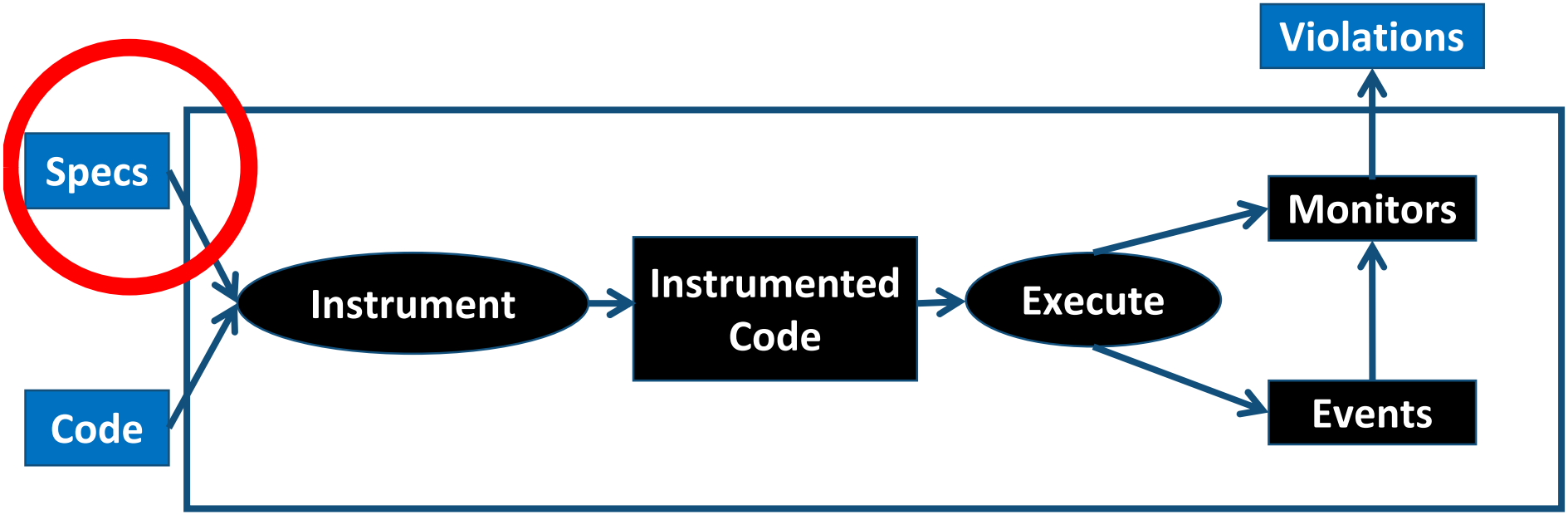
Pros	Cons
No false positives	Limited to observed executions
Scales better than formal verification	May require training in formal methods
More rigorous than software testing	Incurs runtime + developer overhead

How runtime verification works



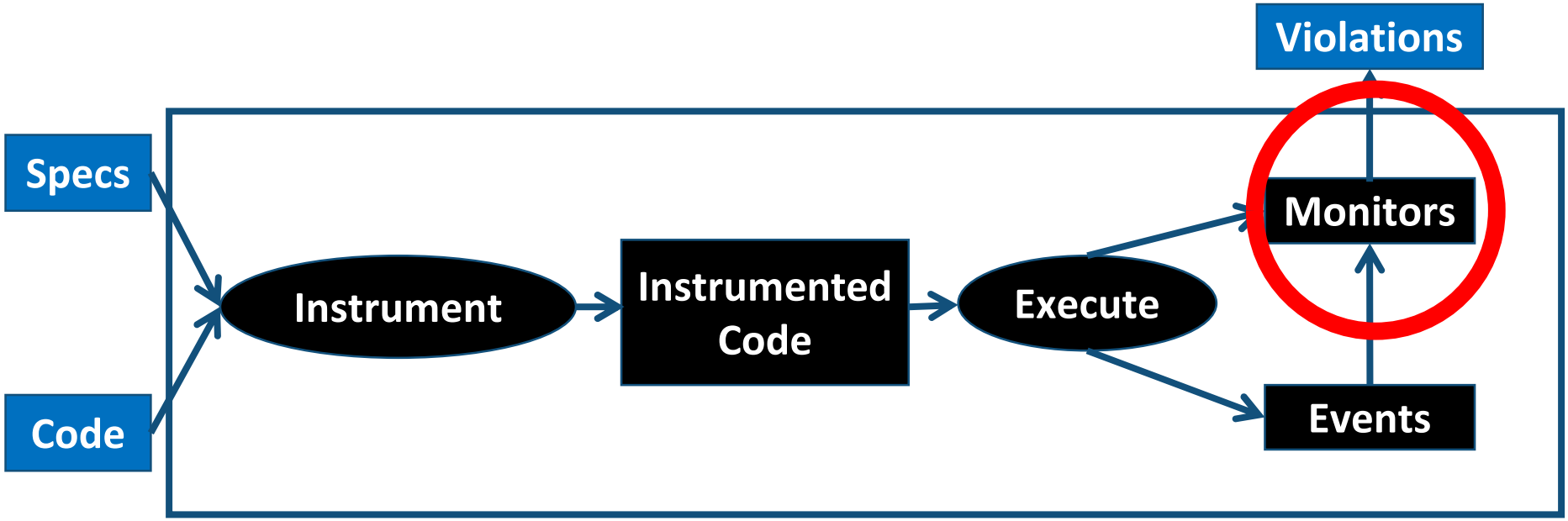
- Many (but not all) RV techniques follow this model
- CS 6156 is (mostly) organized around this model

What you'll learn (specifications)



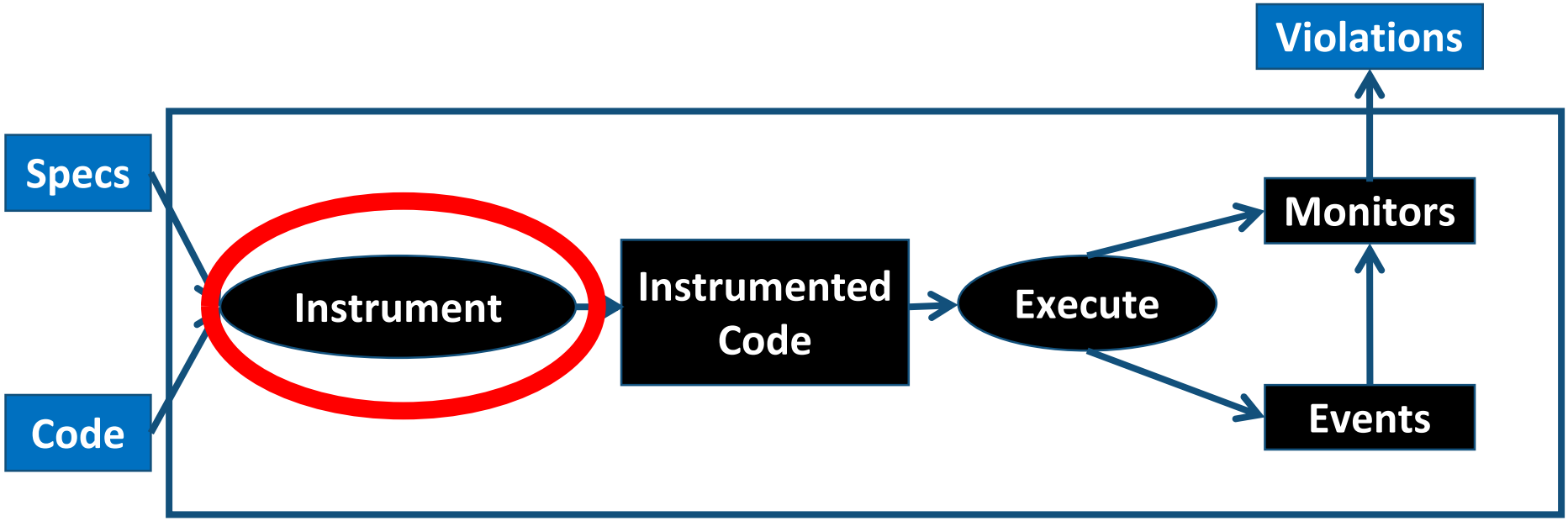
- What kinds of properties can RV check?
- What are languages for specifying properties in RV?
  - LTL, ERE, CFG, and other logical formalisms
- Where do properties come from? (You'll write some)

What you'll learn (monitors)



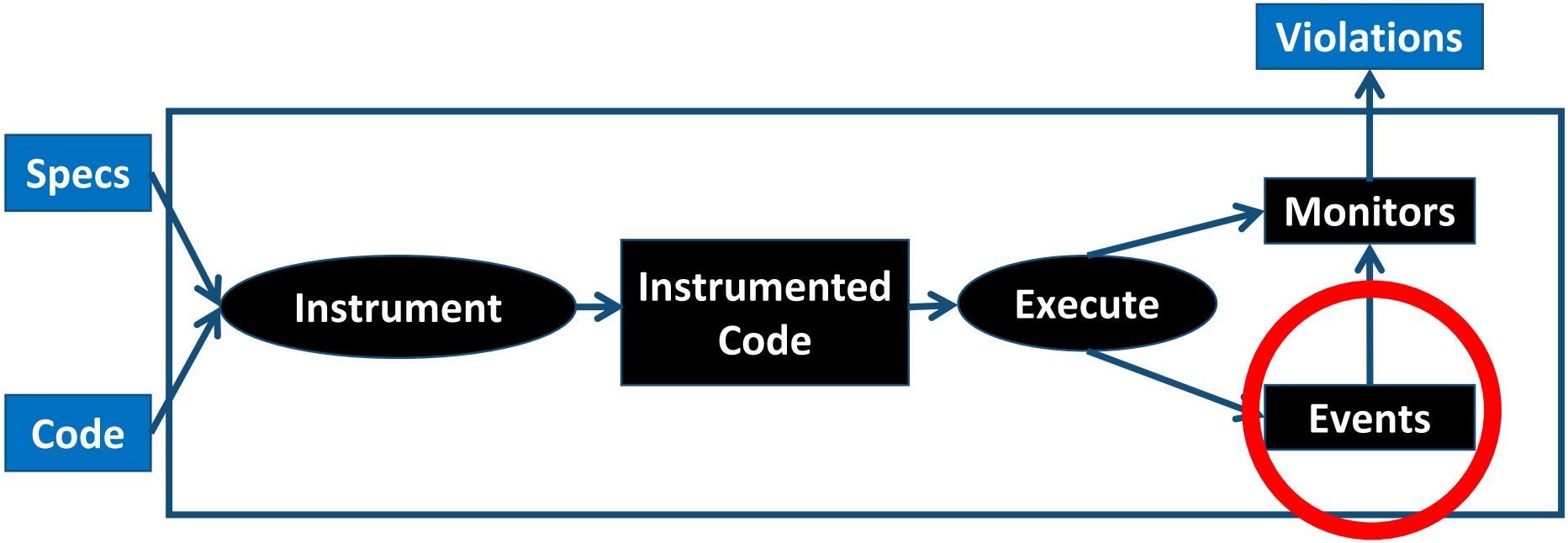
- Monitor synthesis (translating specs to monitors)
- Monitoring algorithms (how monitors get and check events)
- Monitor indexing and garbage collection
  - Small-sized programs often generate tens of millions of monitors

What you'll learn (instrumentation)



- How to instrument code to obtain runtime events?
- Compile-time vs. runtime instrumentation
- Problems and challenges of instrumentation

What you'll learn (events)



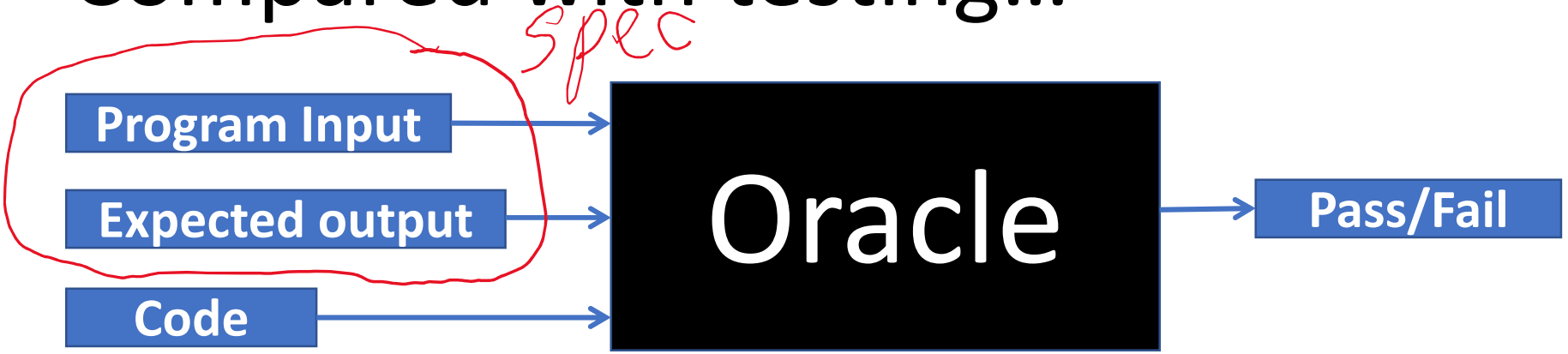
- A formal view of events, traces, and properties
- Program events (e.g., method calls, field access, etc)
- Event dispatch (e.g., which monitors to send events to?)

What you'll learn (other topics)

- How to reduce RV overhead?
  - Combine with static analysis
  - Hardware-assisted RV
  - Sampling the events to check
- How to increase RV coverage?
  - Use RV during software testing
  - Incremental RV
- RV in other domains (depending on your interests)
  - Hardware monitoring
  - Security policy monitoring and enforcement

Discuss: Why is RV a "verification"?

Compared with testing…



Is there any QA approach that can't be shown as above?

Why RV is “verification”

- RV can be done as a system runs in production
- RV allows the system to recover from violations
  - But this seems under-explored in the literature
- So, RV can be used to ensure that a system never goes wrong with respect to a specification
  - In theory, RV can force the system to always be correct

Logistics

Runtime Verification

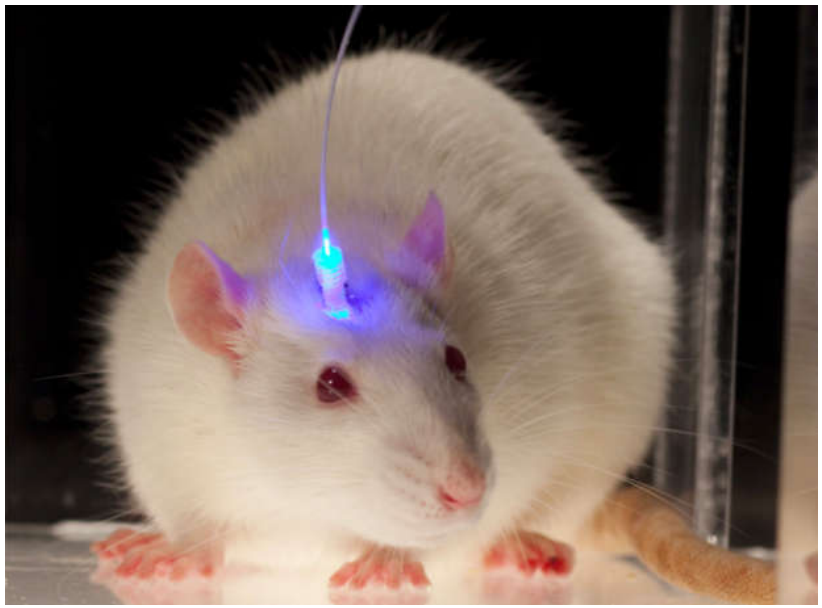
Fall 2020

Runtime Verification is a lightweight formal method for checking program executions against specifications. Foundations, algorithms, and tools for major approaches to runtime verification will be covered, including monitor synthesis, specification languages, parametric monitoring, monitorability, instrumentation, and static analysis for reducing runtime verification overhead. Students will become familiar with recent research results and challenges in runtime verification, gain experience with runtime verification tools, and conduct a research project.

Prerequisites. Graduate standing (Ph.D, MS, or MEng) in CS or CS majors with CS 3110 grade of B+ or better. Experience with Java programming will be helpful for programming assignments.

This course is in Beta. CS 6156 is a brand new course. Everything might change. Nothing is certain.

Logistics: CS 6156 in β is not this



You

Me

Logistics: CS 6156 in β should be

https://freelearners.files.wordpress.com/2015/02/exploring_learning_together4-2arl9wb.jpg

CS6156 information

- Owolabi Legunsen
  - Web: https://www.cs.cornell.edu/~legunsen
  - Email: legunsen@cornell.edu
  - Office Hours: Right after class on Zoom

- Course web page (with in-progress schedule)
  - https://www.cs.cornell.edu/courses/cs6156/2020fa
  - Take some time to go through the web page this week
  - Check the news section frequently for announcements

You are expected to…

- Read assigned texts before each class
  - Reading for Lecture is already assigned
- Complete 4 – 6 homework assignments
- Conduct a research project related to RV
- Lead discussion of 1 paper and present your project

Your grade will be based on…

Readings	**20%**
Homework assignments	**20%**
Presentation and discussion lead	**10%**
Course project	**50%**

Readings

- Assigned text will complement class discussions
  - You may feel lost in class if you don’t read
- Summary due 11:59pm AOE the day before class
- Summary: ≤ 500-word response to a prompt

Homework assignments

- 4 – 6 homework throughout the semester

- Two goals
  - Assess your understanding of reading and lectures
  - Give you opportunity to practice different aspects of RV

Presentation and discussion lead

- Each student will lead discussion of a paper in class
- Each student will also present their final project
  - (more on that in a later slide)

Projects

- Individually or in self-assigned pairs
  - Pairs will do 2x more work than individuals

- The goal is to gain deeper RV knowledge and expertise than we can cover in class + homework

Some possible project directions

- Apply your research solve some problem in RV
  - Or apply RV to your own research
- Come up with an idea and explore it
  - Owolabi is happy to help!
- Reproduce and extend results from RV papers
- Extend RV tools, components, and systems
- Compare other RV models with the one in CS6156
- Survey the literature on some aspect(s) of RV
  - A good option for undergraduates or MEng students

Tentative project timeline

Milestone	When
Discuss some concrete project topics in class	By 9/17
Meet Owolabi to discuss your project proposal*	Before 10/5
Project proposal is due (up to 1 page)	10/6
Meet Owolabi to discuss project progress*	Before 10/26
Project progress report 1 is due (up to 2 pages)	10/27
Meet Owolabi to discuss project progress*	Before 11/18
Project progress report is due (up to 2 pages)	11/19
Present final project in class	TBD
Final project report is due	12/17

* These meetings are mandatory

Questions on content or logistics?

?

Recall: high-level view of RV

Now: concrete examples of RV tool, inputs, and outputs

- One RV tool that we will use in this class is JavaMOP
  - https://github.com/runtimeverification/javamop

Example spec: Collection_SynchronizedCollection (CSC)

https://docs.oracle.com/javase/7/docs/api/java/util/Collections.html#synchronizedCollection(java.util.Collection)

synchronizedCollection

```
public static <T> Collection<T> synchronizedCollection(Collection<T> c)
```

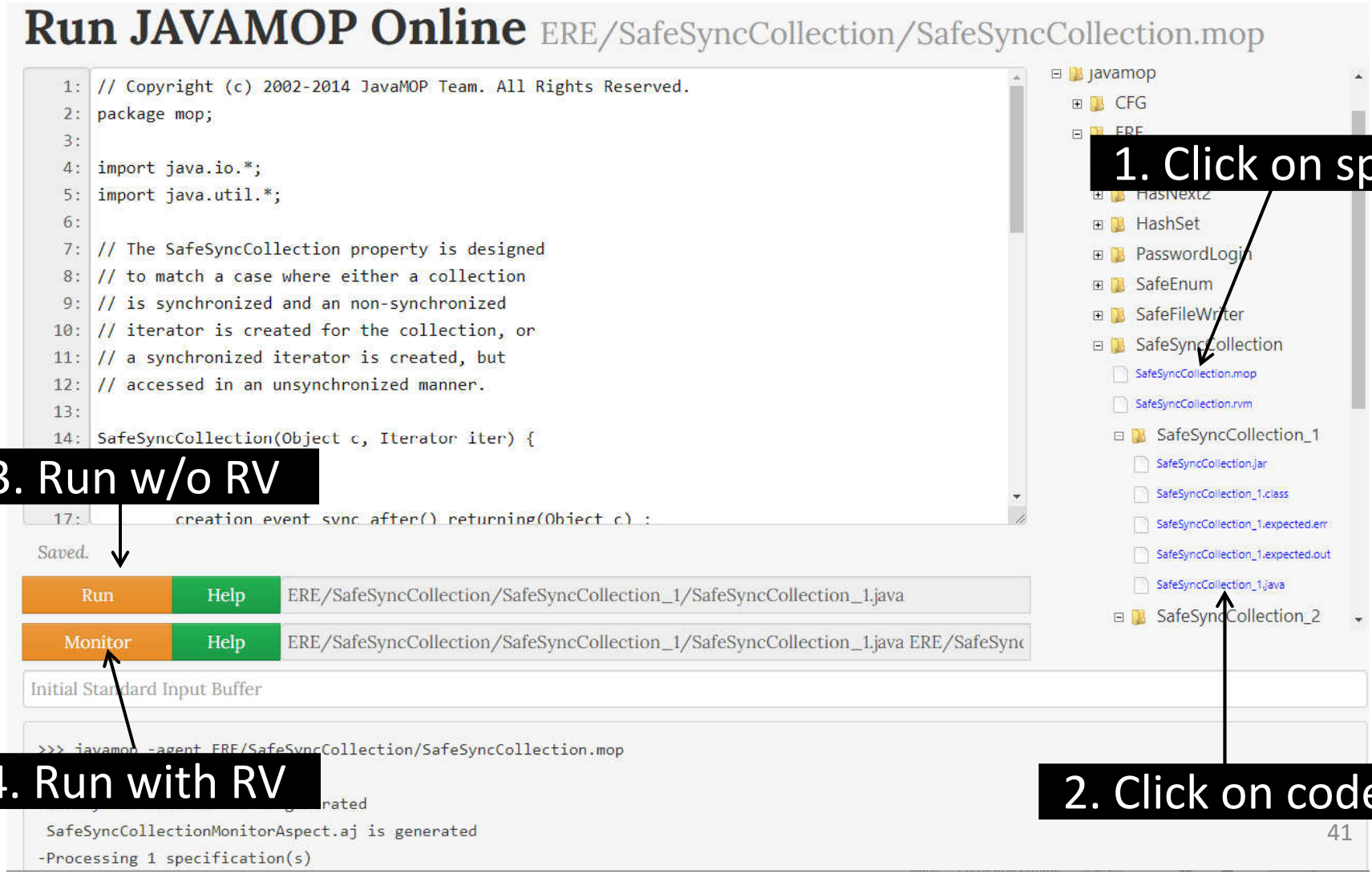
It is imperative that the user manually synchronize on the returned collection when iterating over it:

```
Collection c = Collections.synchronizedCollection(myCollection);
   ...
synchronized (c) {
    Iterator i = c.iterator(); // Must be in the synchronized block
    while (i.hasNext())
       foo(i.next());
}
```

Failure to follow this advice may result in non-deterministic behavior.

Live demo: RV of CSC on toy code

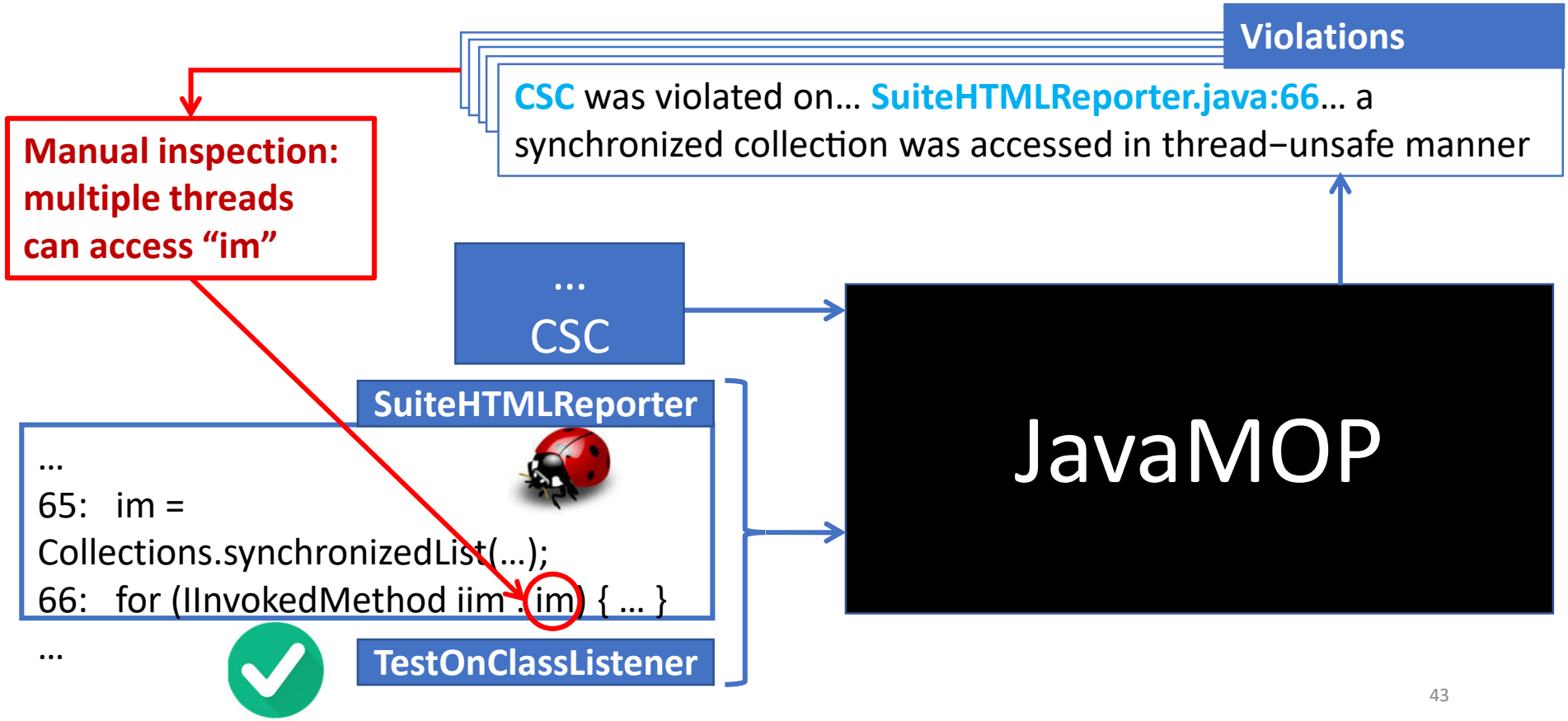
http://www.kframework.org/tool/run/javamop



What we saw during the demo

- A spec (in ERE formalism)
- JavaMOP output
- JavaMOP finds a violation in code that runs “correctly”
  - is the violation a bug, though?
- An online environment for using JavaMOP

The “RV process” (also used in demo)



RV in my (RV + testing) research

- Monitored the tests in 229 open source software
  - some of them have over 200K lines of code
- RV found hundreds of bugs that testing missed
  - many have been confirmed
- But there are still many challenges
  - You'll discover some of them in this class

Before next class (pre-homework)

- Introduce yourself on Piazza
  - What you'd like us to call you
  - Your degree (PhD, MS, MEng, BS)
- Read the course webpage
  - https://www.cs.cornell.edu/courses/cs6156/2020fa
- If you are not a PhD student, send me an email answering these questions:
  - What is CS 6156 about?
  - Why did you decide to take this course?

Next class…

- Start with the basics: events, traces, properties

- Reading is assigned
  - Due by 11:59pm Sunday 9/6/2020 AOE

A review of today's class

- A comparison of RV with other QA approaches
- A whirlwind tour of RV
- Learning outcomes, course content, and logistics
- Online demo of an RV tool (JavaMOP)