

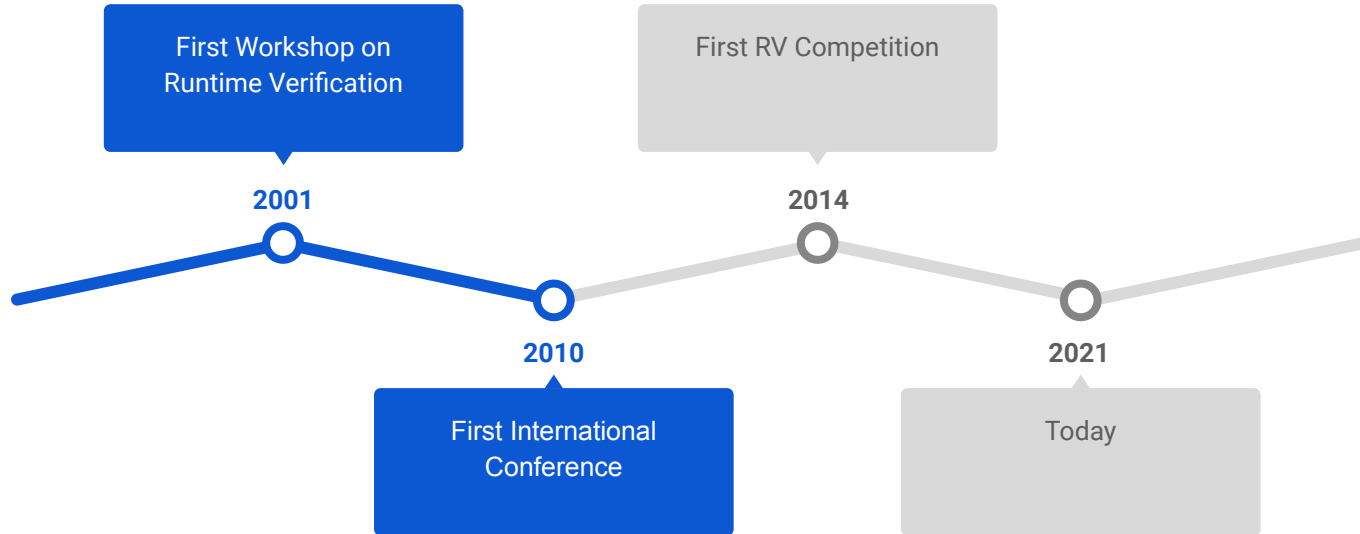
Runtime Verification: Passing on the Baton

Workshop on Formal Methods in Outer Space

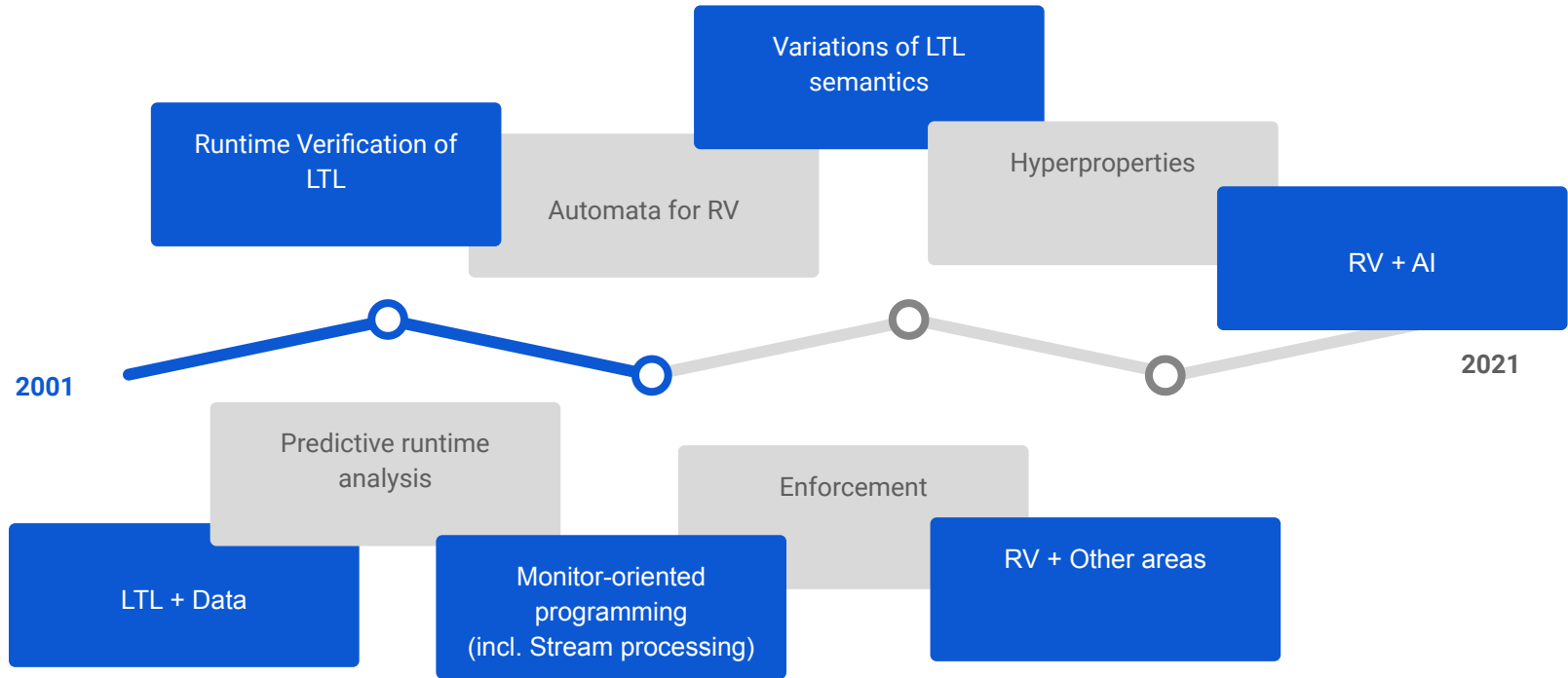
Rhodes - October 2021

Christian Colombo, Gordon J. Pace, and Gerardo Schneider

Milestones of RV



Topics in RV



The Past and the Future

This is what has led us to where we are today

Where do we want to go from here?

How do we pass on the “baton”?

The Past and the Future

This is what has led us to where we are today

Where do we want to go from here?

How do we pass on the “baton”?



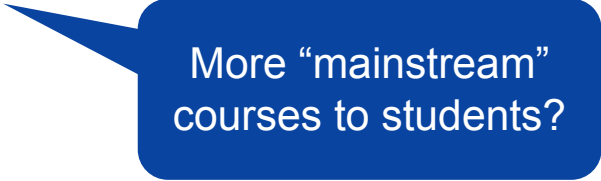
More use in
industry?

The Past and the Future

This is what has led us to where we are today

Where do we want to go from here?

How do we pass on the “baton”?



More “mainstream”
courses to students?

Hands-On Course to RV

Aimed to be practical:

- Develop their own “RV tool”
- Apply RV to a “realistic” system

Not aimed to give the full theory
(LTL comes quite late in the course)

Introduction

What is RV?

An introduction to verification in general, and runtime verification more specifically.

Introduction

What is RV?

Manual
monitoring

An introduction to verification in general, and runtime verification more specifically.

What is RV at its most basic level?

Where does simplicity stop working?

The need for
separation of concerns
+ abstraction

Introduction

What is RV?

An introduction to verification in general, and runtime verification more specifically.

Manual monitoring

What is RV at its most basic level?

Where does simplicity stop working?

The need for **separation of concerns**
+ abstraction

Instrumentation

How to automate event extraction and monitor injection



Introduction

What is RV?

An introduction to verification in general, and runtime verification more specifically.

Manual monitoring

What is RV at its most basic level?

Where does simplicity stop working?

The need for **separation of concerns + abstraction**

Instrumentation

How to automate event extraction and monitor injection

Specification Languages

How do we automate verifier synthesise?

- Guarded command language
- Automata
- Regular expressions
- LTL



Introduction

What is RV?

An introduction to verification in general, and runtime verification more specifically.

Manual monitoring

What is RV at its most basic level?

Where does simplicity stop working?

The need for **separation of concerns + abstraction**

Instrumentation

How to automate event extraction and monitor injection

Specification Languages

How do we automate verifier synthesise?

- Guarded command language
- Automata
- Regular expressions
- LTL

Various Directions

- Real-time
- Offline
- Other topics



1. Introduction to RV

**Software is difficult
to do well**

So many examples of
expensive and deadly
bugs!

1. Introduction to RV

**Software is difficult
to do well**

So many examples of
expensive and deadly
bugs!

+

**Software ever more
central to our lives**

Most of our daily lives
depend on it!

1 >> Introduction to RV

Software is difficult to do well

So many examples of expensive and deadly bugs!



Software ever more central to our lives

Most of our daily lives depend on it!

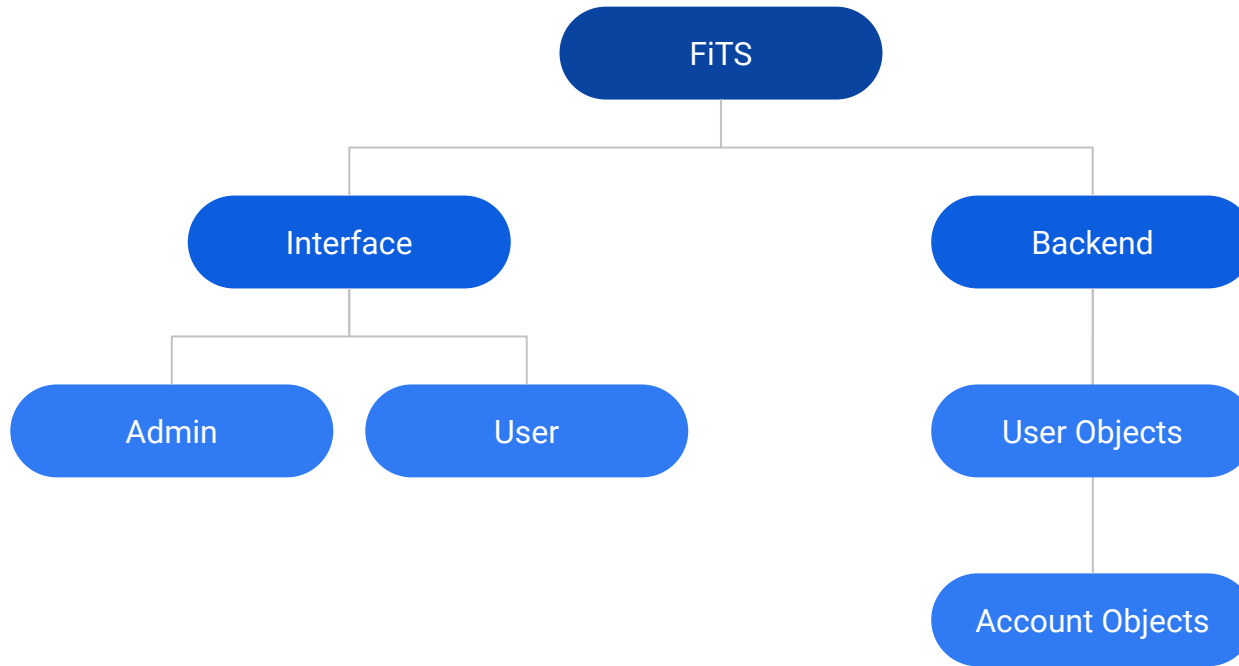


WE NEED TO CHECK!

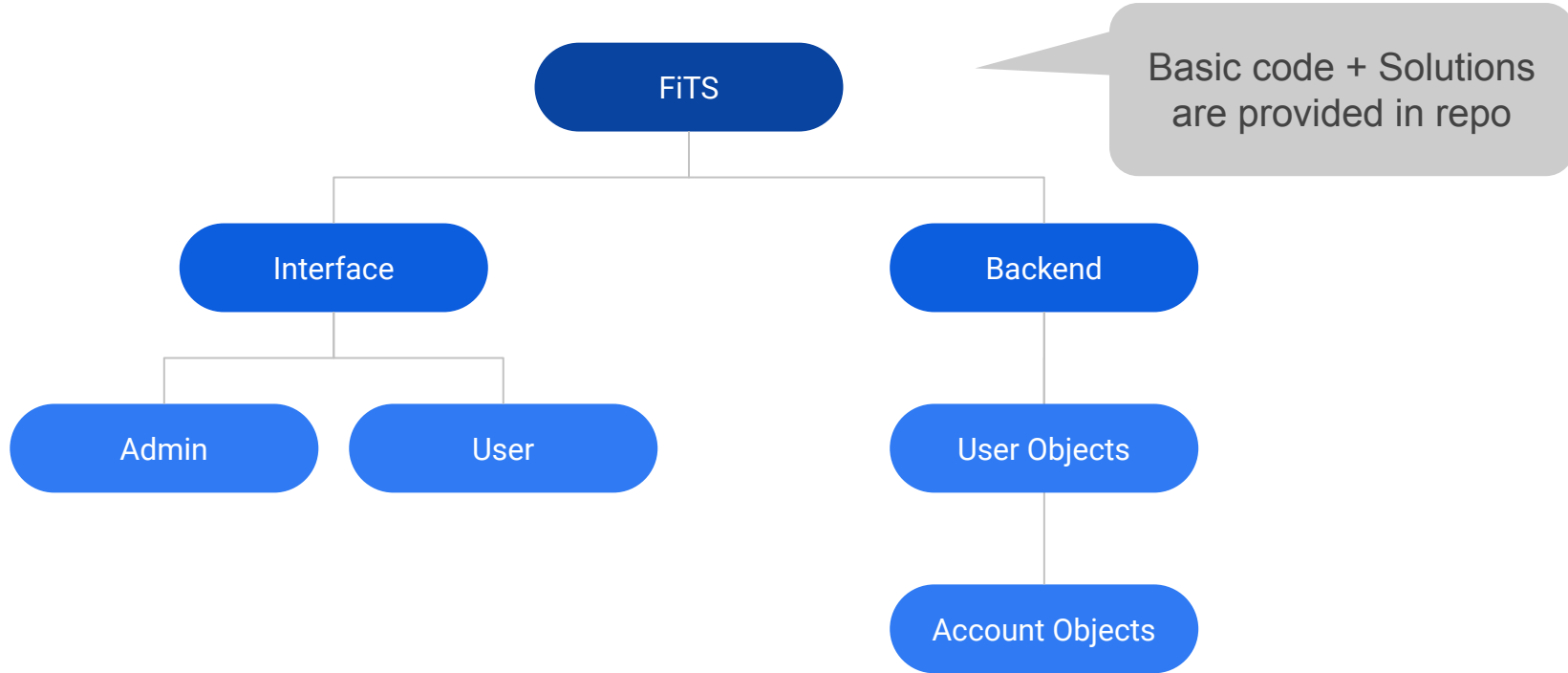
Testing is practical but incomplete
Model checking is complete but impractical

RV as an alternative

2a. FiTS - Financial Transaction System



2a. FiTS - Financial Transaction System



2b. Manual RV - Assertions

“Only users from Argentina can be Gold Users”




Point assertion

2b. Manual RV - Assertions

*“The system should be initialised **before** the first user session is opened”*




Temporal property



Things start to get
messy!

2b. Manual RV - Assertions

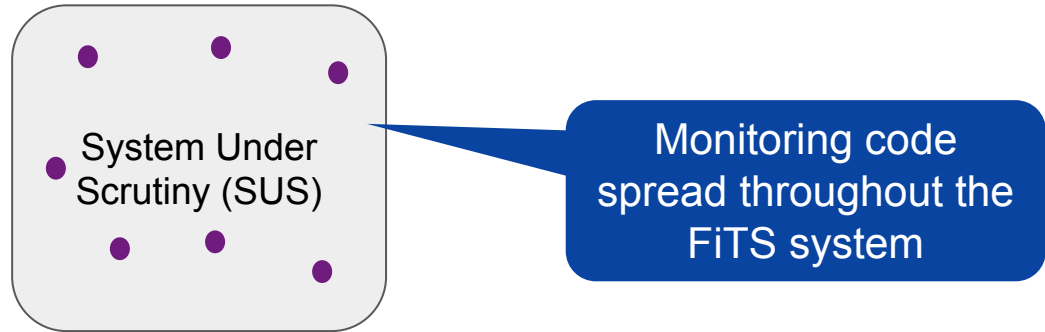
*“The administrator must reconcile accounts **every 1000 attempted external money transfers or whenever an aggregate total of one million dollars in attempted external transfers is reached**”*



This is what we're
talking about!

2b. Manual RV - Assertions

*“The administrator must reconcile accounts **every 1000 attempted external money transfers or whenever an aggregate total of one million dollars in attempted external transfers is reached**”*



2 >> Need for Separation of Concerns

There are two main problems with the manual approach:

- Placement of the monitors - **Non-modular**



AOP

2 >> Need for Separation of Concerns

There are two main problems with the manual approach:

- Placement of the monitors - **Non-modular**
- Programming of the verifier - **Error-prone**

A blue rounded rectangular callout box with a white pointer pointing to the text 'Placement of the monitors - Non-modular'.

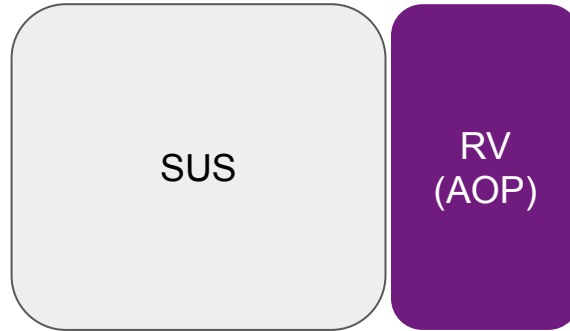
AOP

A dark blue rounded rectangular callout box with a white pointer pointing to the text 'Programming of the verifier - Error-prone'.

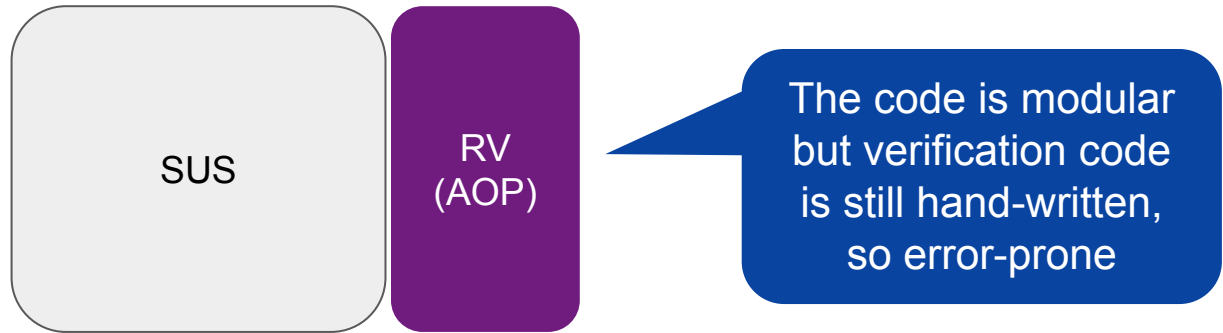
Specification
languages

3. Instrumenting Monitors

Student use AOP code to write monitoring code in a single file



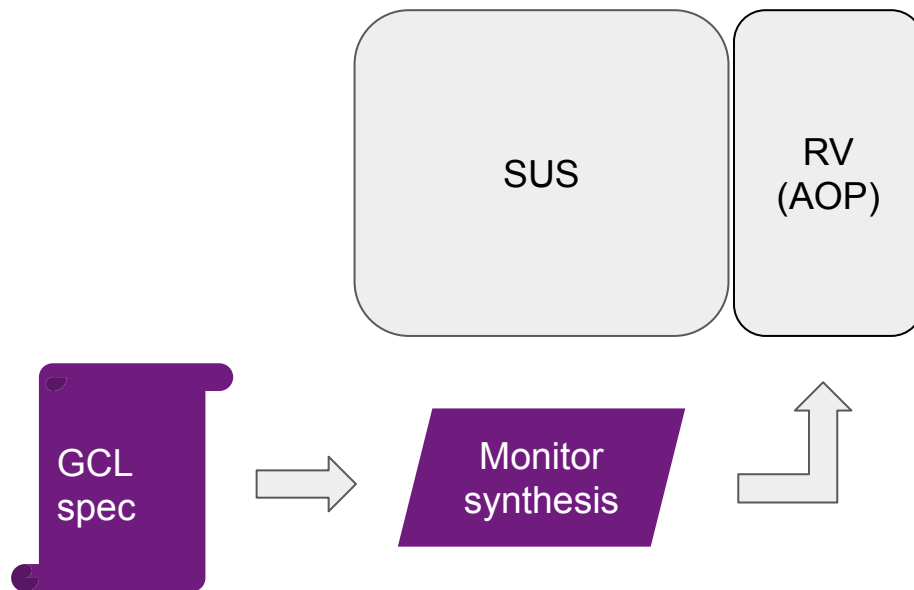
3 >> Instrumenting Monitors



4a. Specification Languages: Guarded-Commands

Students are introduced to Guarded Commands (event | condition \rightarrow action)

- They write the spec and
- The compiler



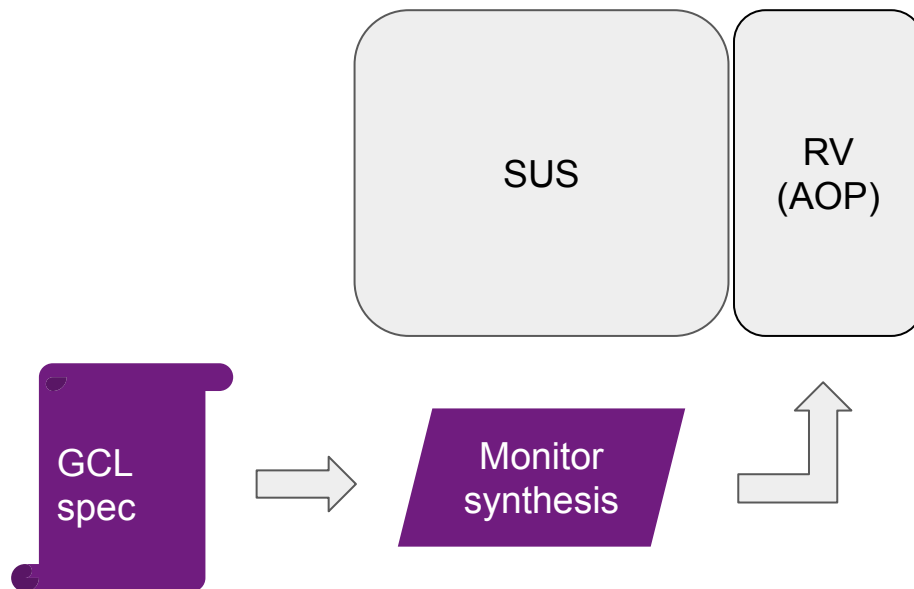
4a. Specification Languages: Guarded-Commands

Students are introduced to Guarded Commands (event | condition \rightarrow action)

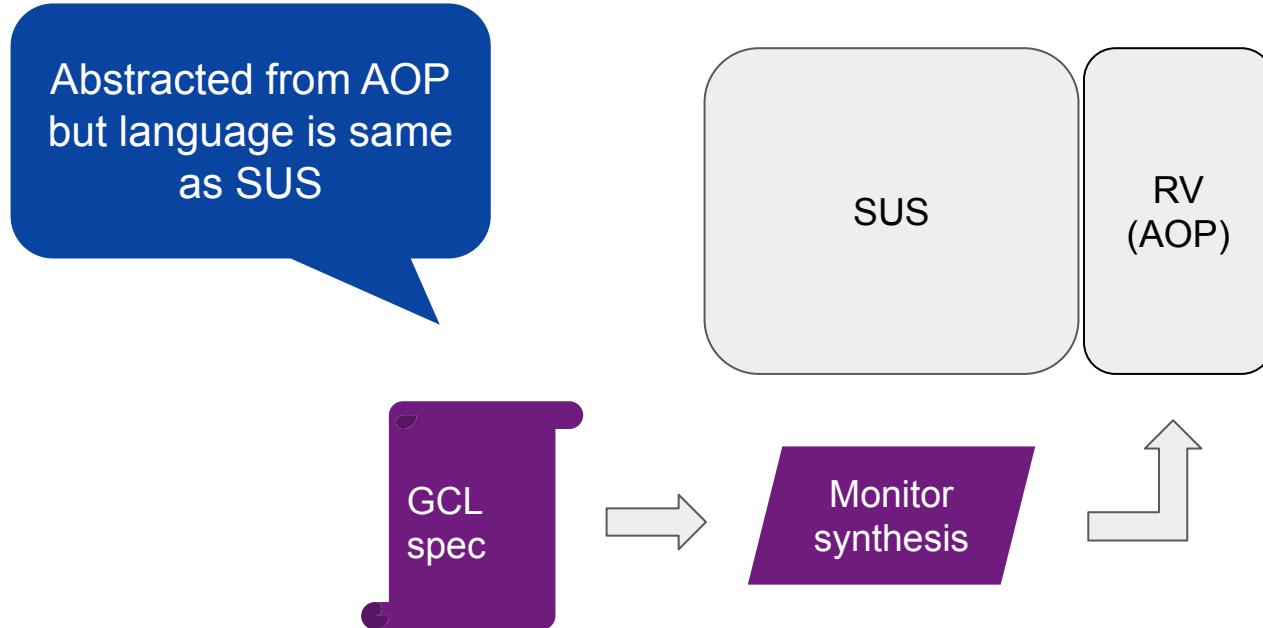
- They write the spec and
- The compiler

Introduction to

- Monitor state
- Parametrised properties



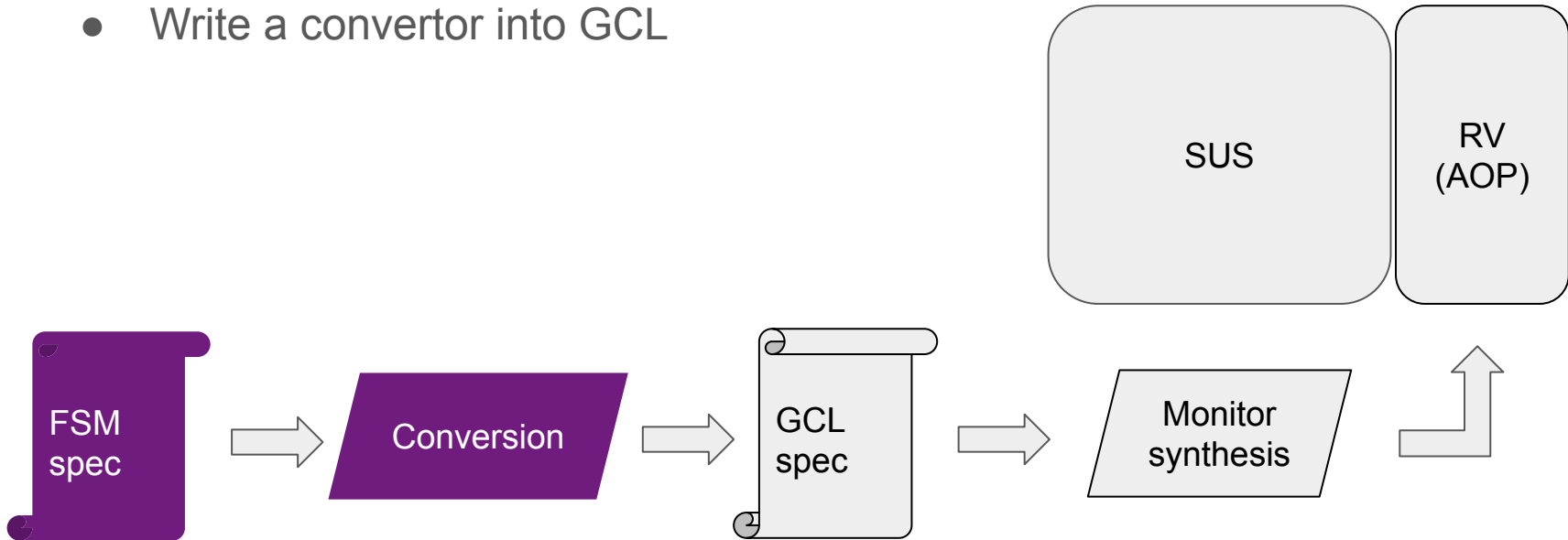
4a >> Specification Languages: Guarded-Commands



4b. Specification Languages: Finite State Automata

Students are introduced to automata

- They write the properties in automata
- Write a convertor into GCL



4b >> Specification Languages: Finite State Automata

Cool but FSMs lack structure

FSM spec

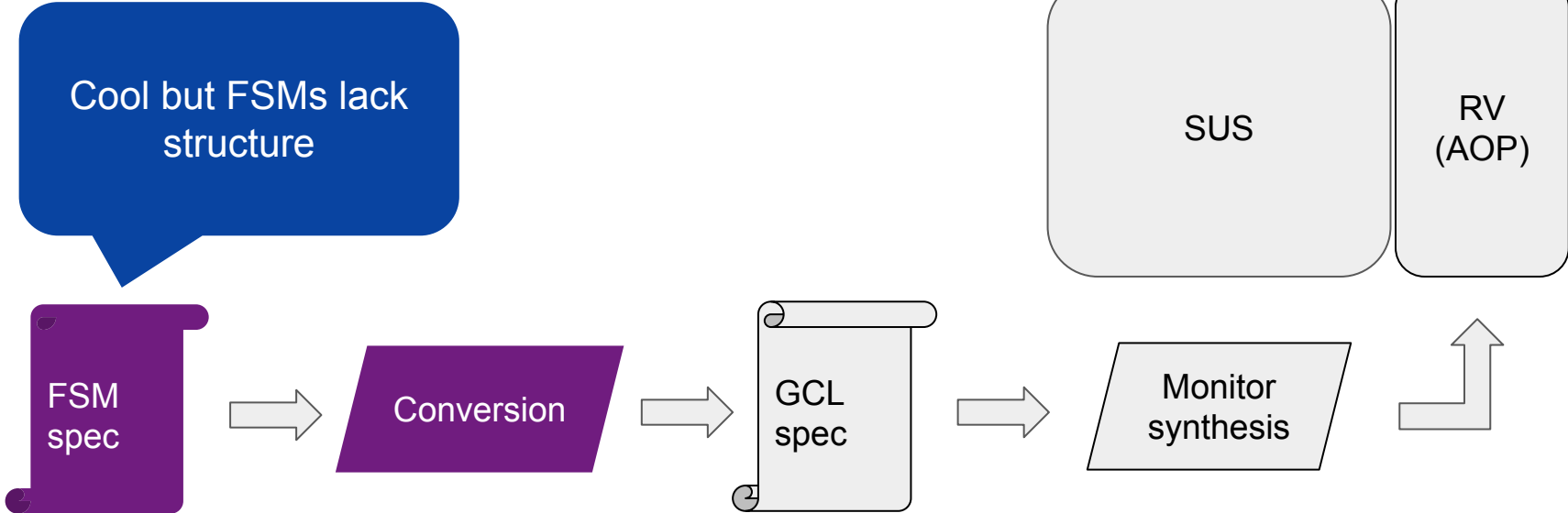
Conversion

GCL spec

Monitor synthesis

SUS

RV (AOP)



4c. Specification Languages: Regular Expressions

Students are introduced to REs

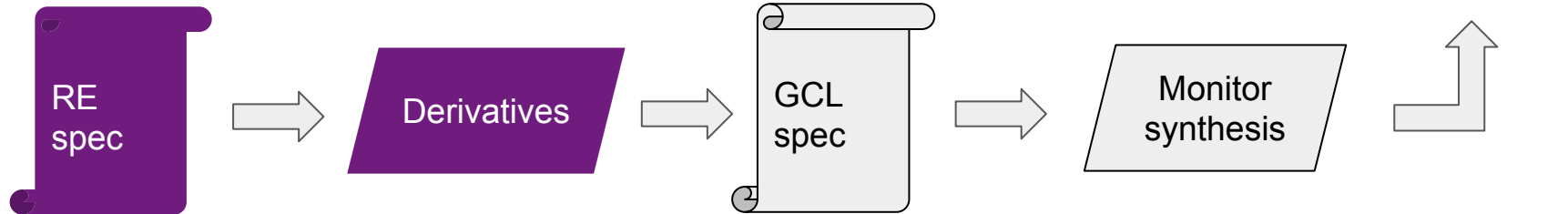
- Understand semantics
- How to monitor using derivatives

4c. Specification Languages: Regular Expressions

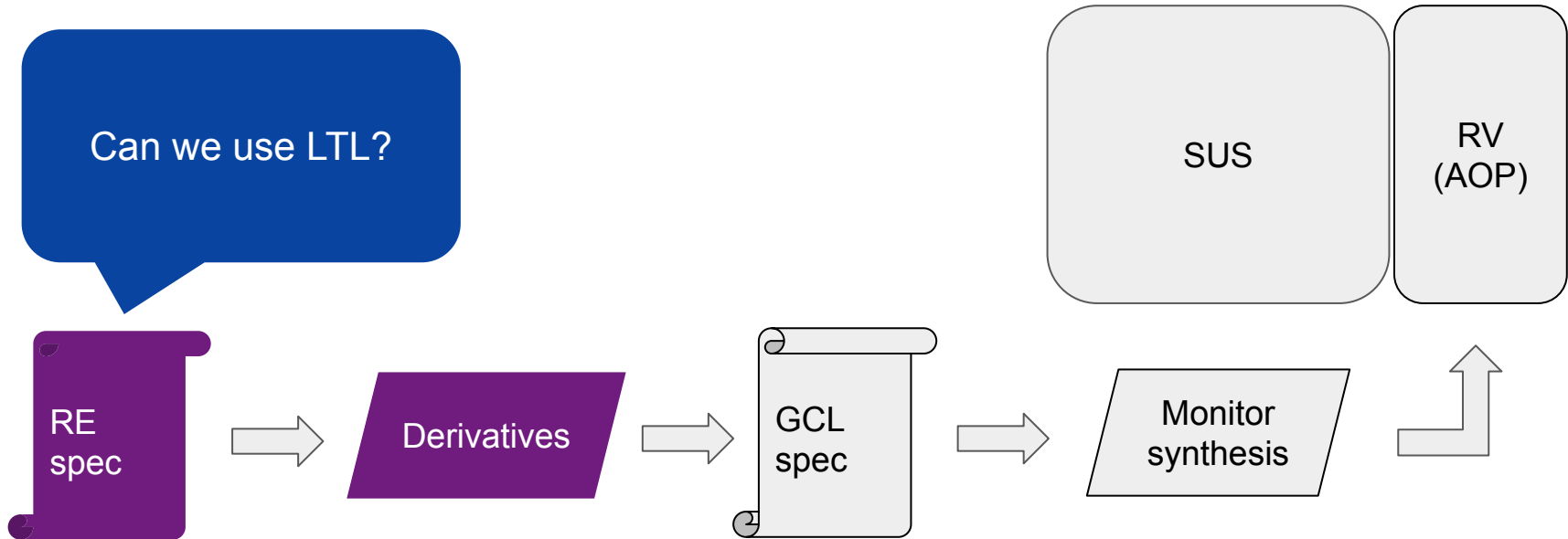
Students are introduced to REs

- Understand semantics
- How to monitor using derivatives

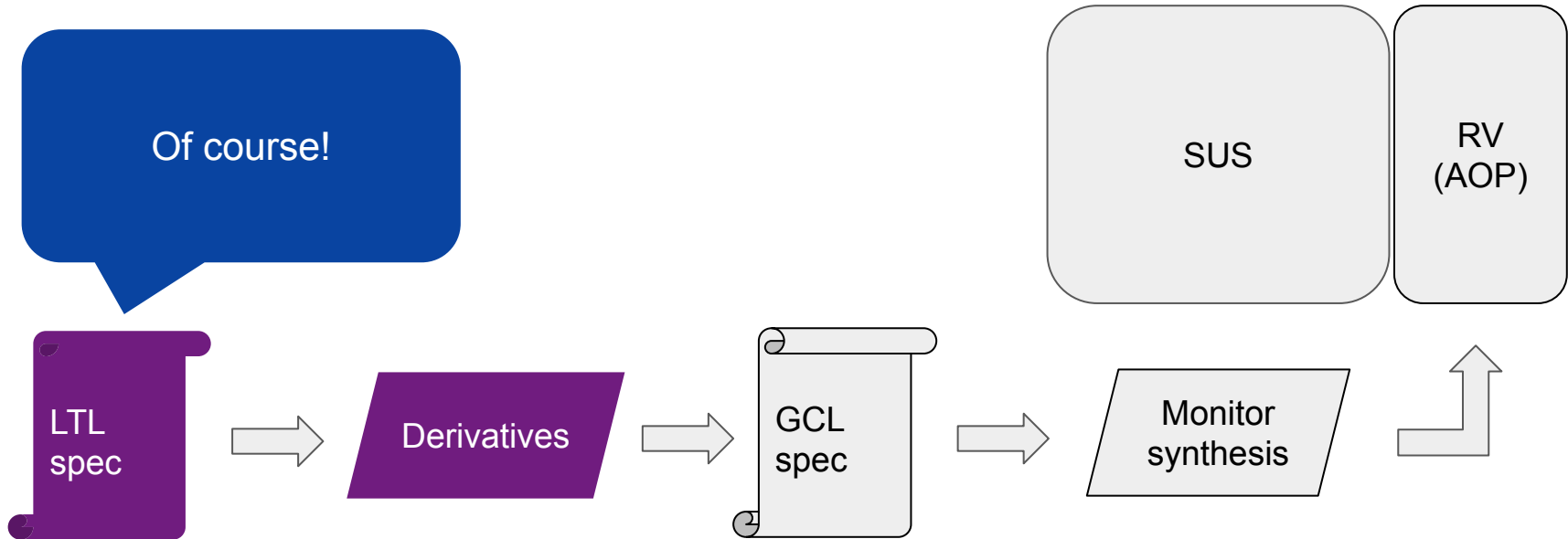
They express properties as REs
Generate the code as GCL



4c >> Specification Languages: Regular Expressions



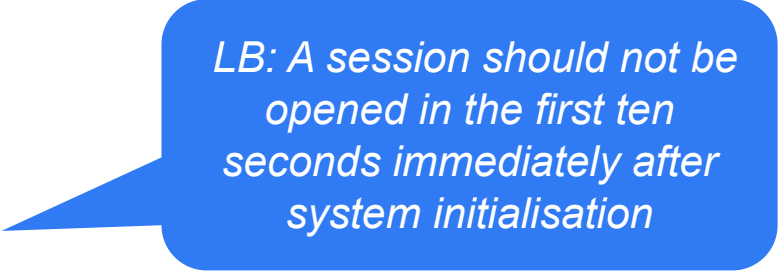
4d >> Specification Languages: LTL



5. Real-Time Properties

Students should understand:

- Lowerbound vs Upperbound properties
- LB can be monitored with timestamps



LB: A session should not be opened in the first ten seconds immediately after system initialisation

5. Real-Time Properties

Students should understand:

- Lowerbound vs Upperbound properties
- LB can be monitored with timestamps
- UB require events to detect as-early-as-possible

LB: A session should not be opened in the first ten seconds immediately after system initialisation

UB: A new account must be approved or rejected by an administrator within 24 hours of its creation

5 >> Real-Time Properties

Students should understand:

- Lowerbound vs Upperbound properties
- LB can be monitored with timestamps
- UB require events to detect as-early-as-possible

- The effect of slowdown on real-time properties

LB: A session should not be opened in the first ten seconds immediately after system initialisation

UB: A new account must be approved or rejected by an administrator within 24 hours of its creation

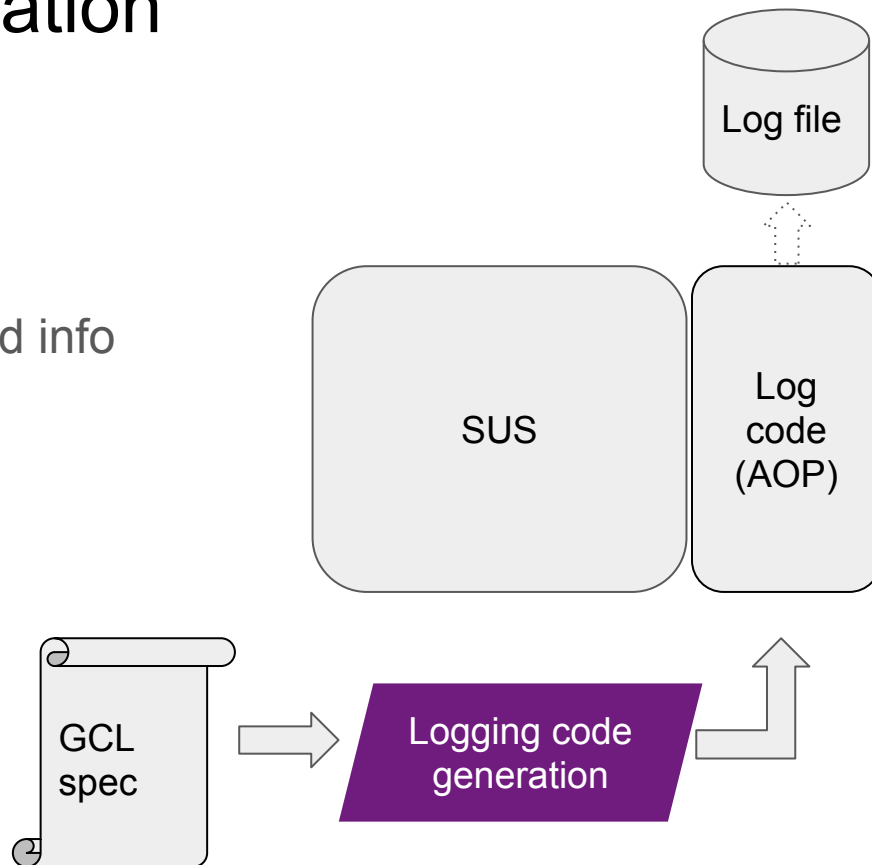
*Applies for any aspect of the system which is reflexiv
E.g. System reasons on its own memory consumption*

6a. Offline Runtime Verification

The need to be non-intrusive

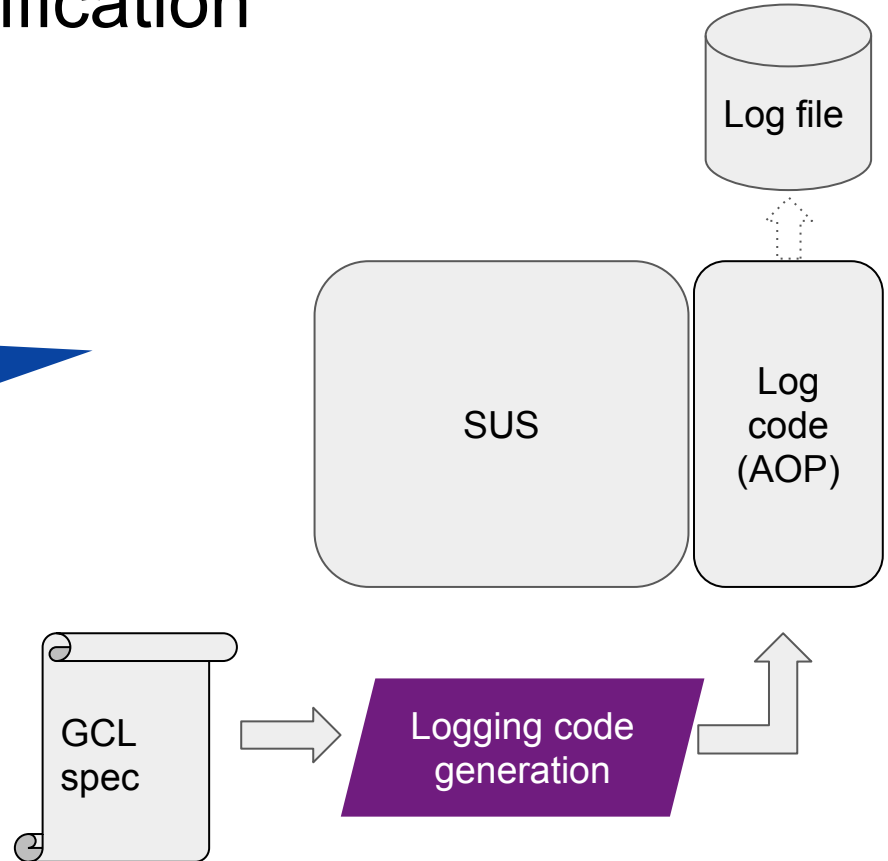
- RV reduced to logging at runtime

Students generate code to record needed info



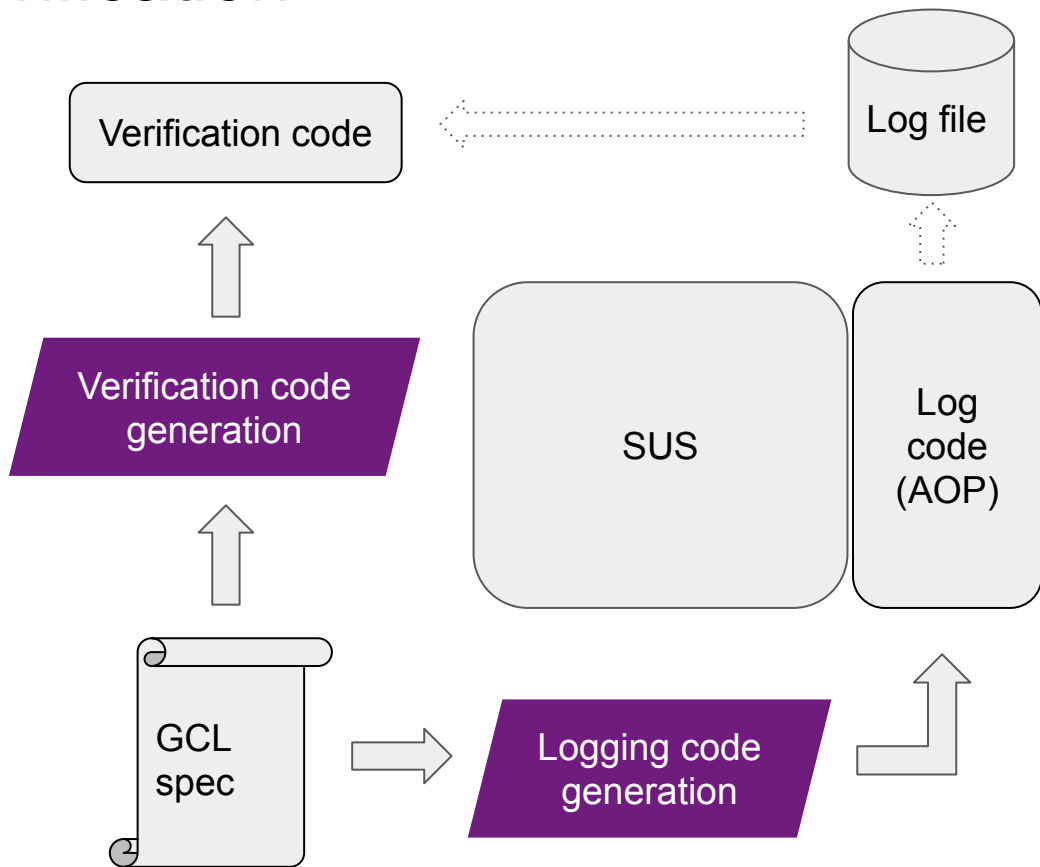
6a >> Offline Runtime Verification

Whatever isn't recorded is lost



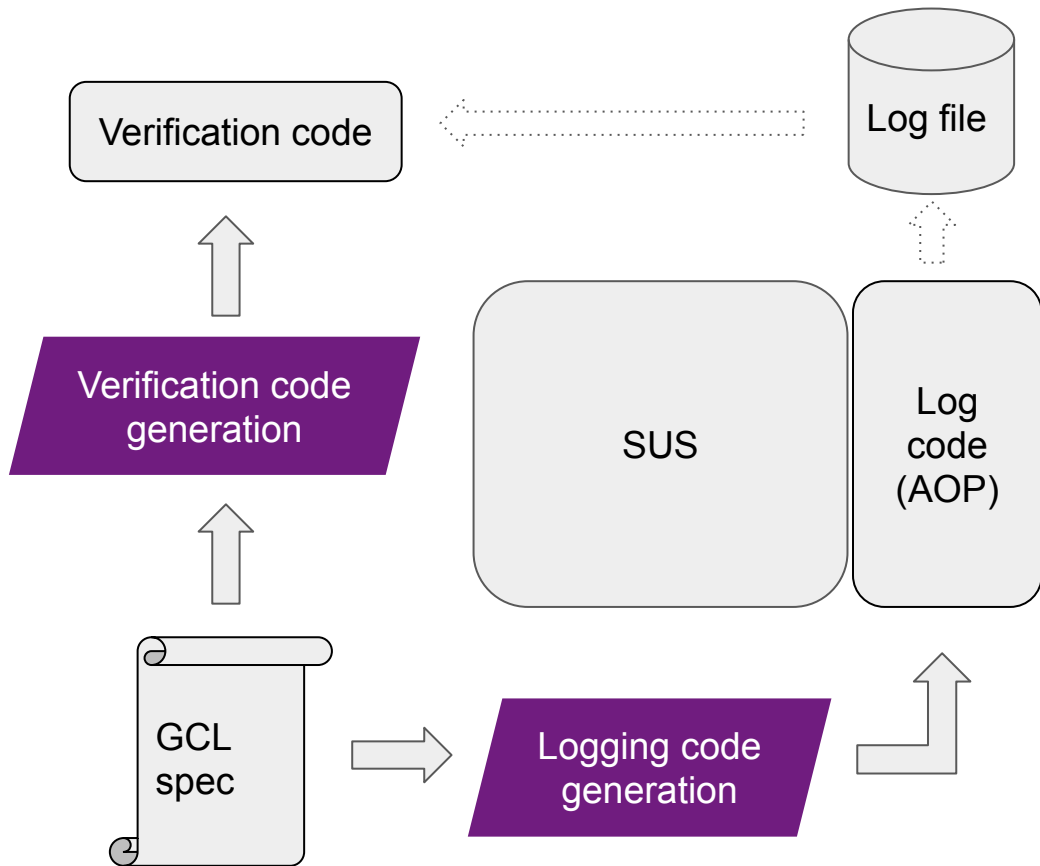
6b. Offline Runtime Verification

Verification code is updated to consume events from the log file



6b >> Offline Runtime Verification

If a violation is detected,
it is too late to do
anything about it!



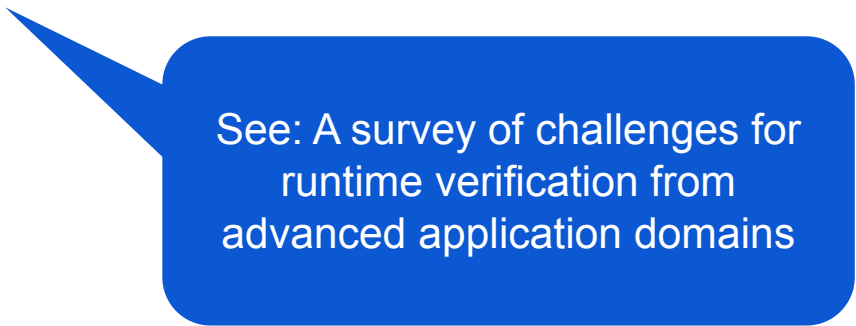
7a. Advanced Topics to Consider

- More theory
- Reactivity: Enforcement / Reparation and Control / Compensation
- Efficiency: Profiling / Optimisation
- Persistence of Monitors
- DSLs

7b. Advanced Topics - Links to Applications

Link to students' area of specialisation

- Distributed systems
- Hardware
- Hybrid and embedded systems
- Security/Privacy
- Contracts/Policies
- Transactional information systems
- Huge, unreliable or approximated domains



See: A survey of challenges for runtime verification from advanced application domains

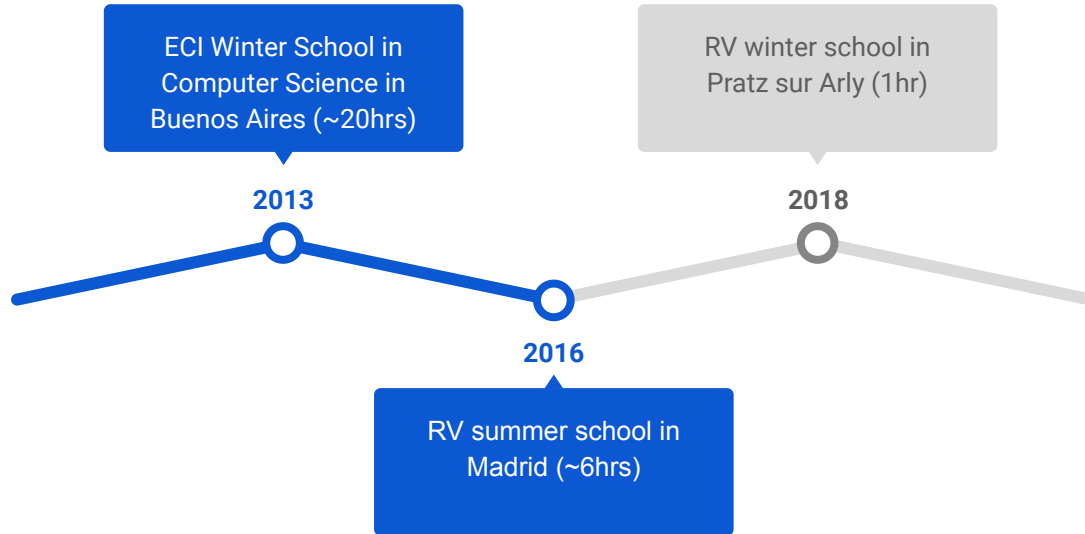
7c. Advanced Topics - Links to Other Areas

Link to students' area of specialisation

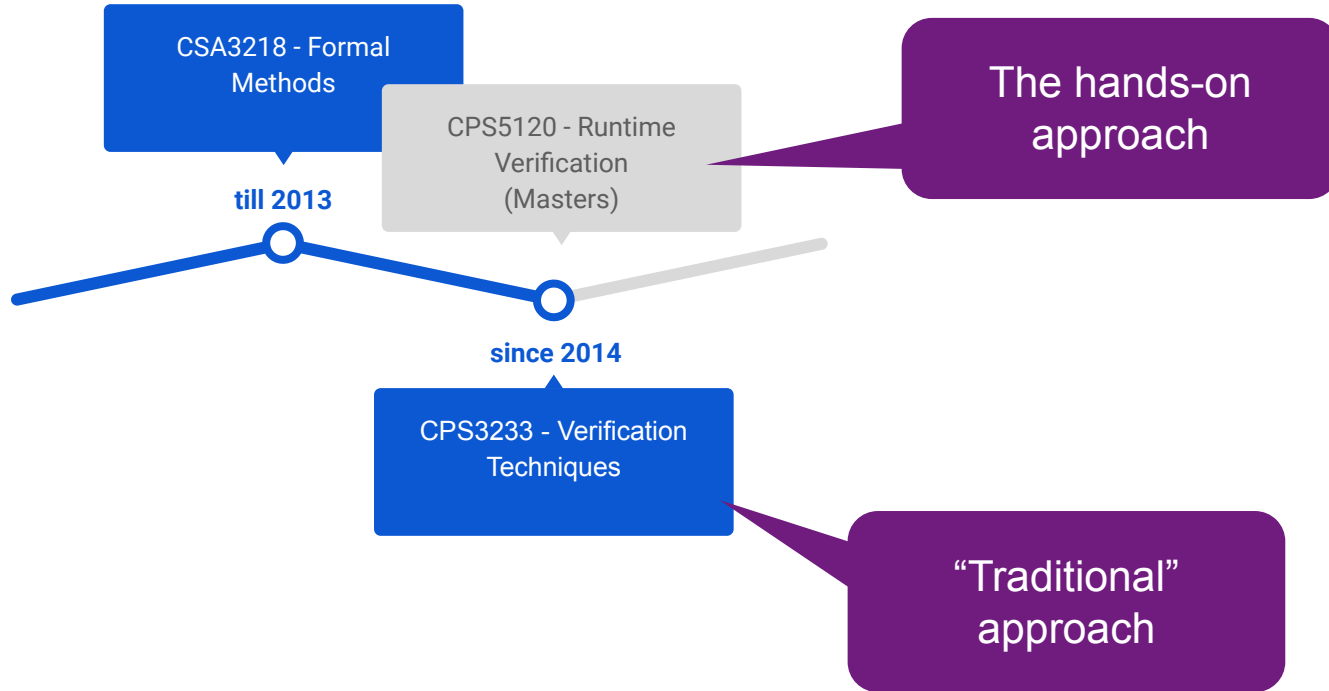
- Static Techniques
- Model Learning
- Testing
- Runtime Assertion Checking

See: COST Action IC 1402 ArVI:
Runtime Verification Beyond Monitoring
Activity Report of Working Group 1

Experiences Teaching the Course



Experiences Teaching the Course at University



Variations of the Course

Length

- Cutdown versions (summer/winter schools)
- 5-credit version at University
- Longer version in the book

Technology

- Java + AspectJ

Case study

- FiTS



Other options would work just as well

Conclusion

A lot to celebrate in our past

Time to see more RV take up in industry

Passing on the baton through education

Ideas can be adapted according to context

Easy to connect to many other areas