IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Quarkslab
Securing every bit of your data

Traceability of the compilation process

CLAP-HiFi-LVP 2023

Bruno MATEU

21st of March

Introduction

1 • Context

Quarkslab

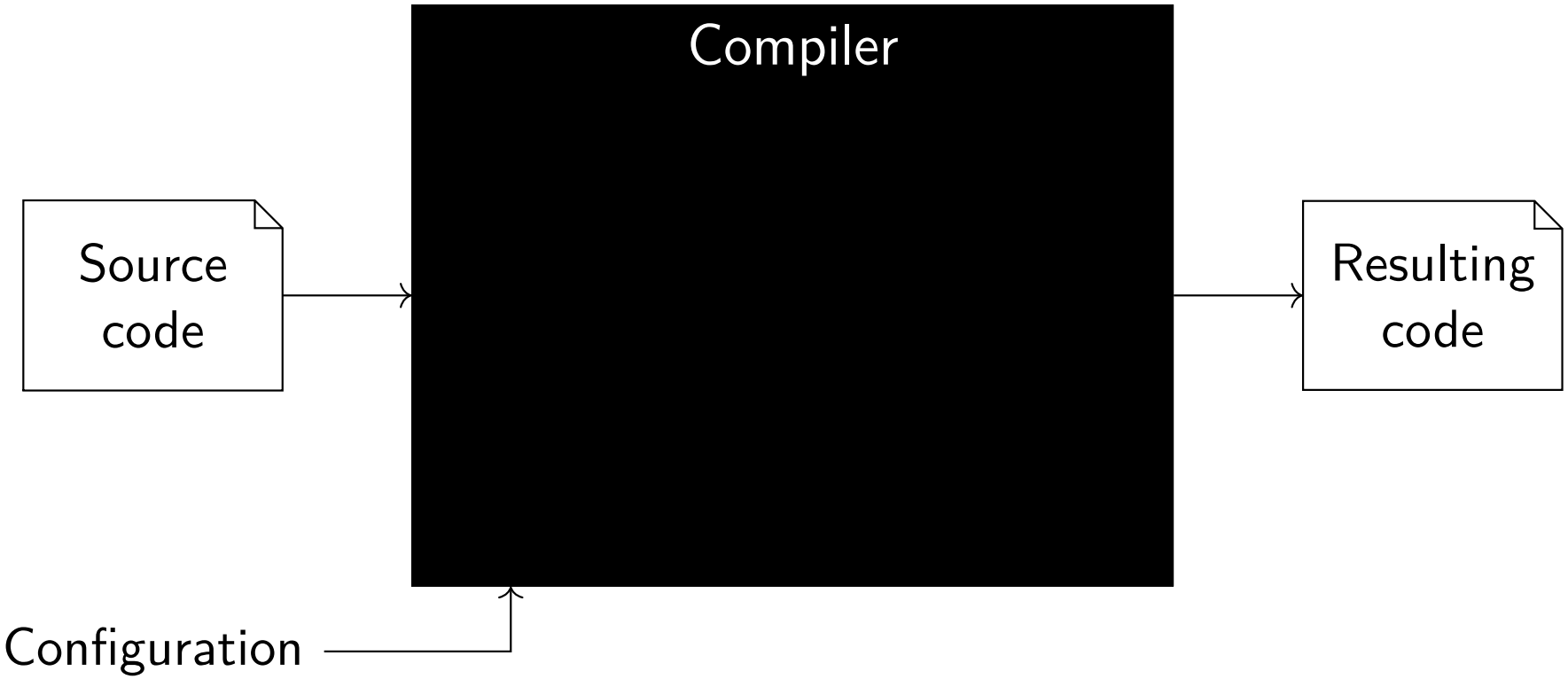
Securing every bit of your data

IMT Atlantique

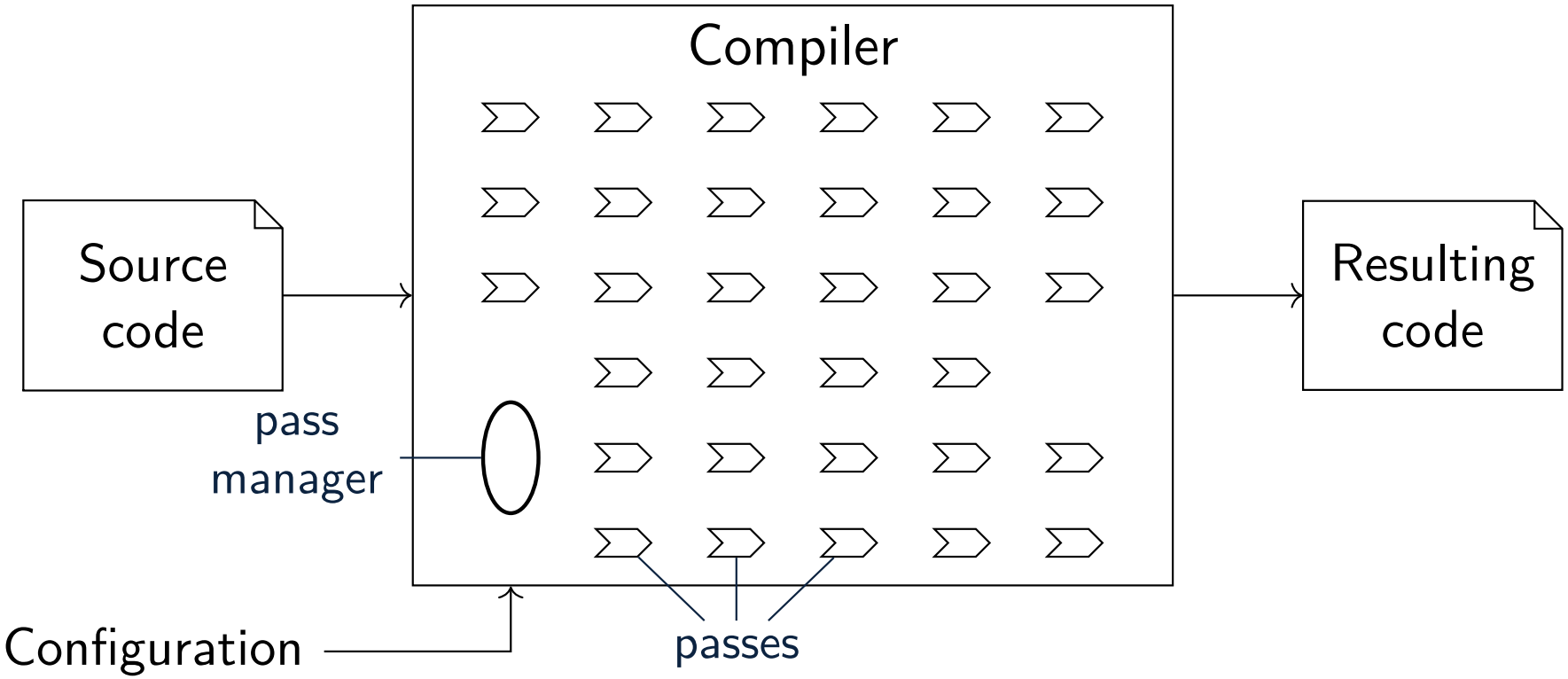
Bretagne-Pays de la Loire

École Mines-Télécom

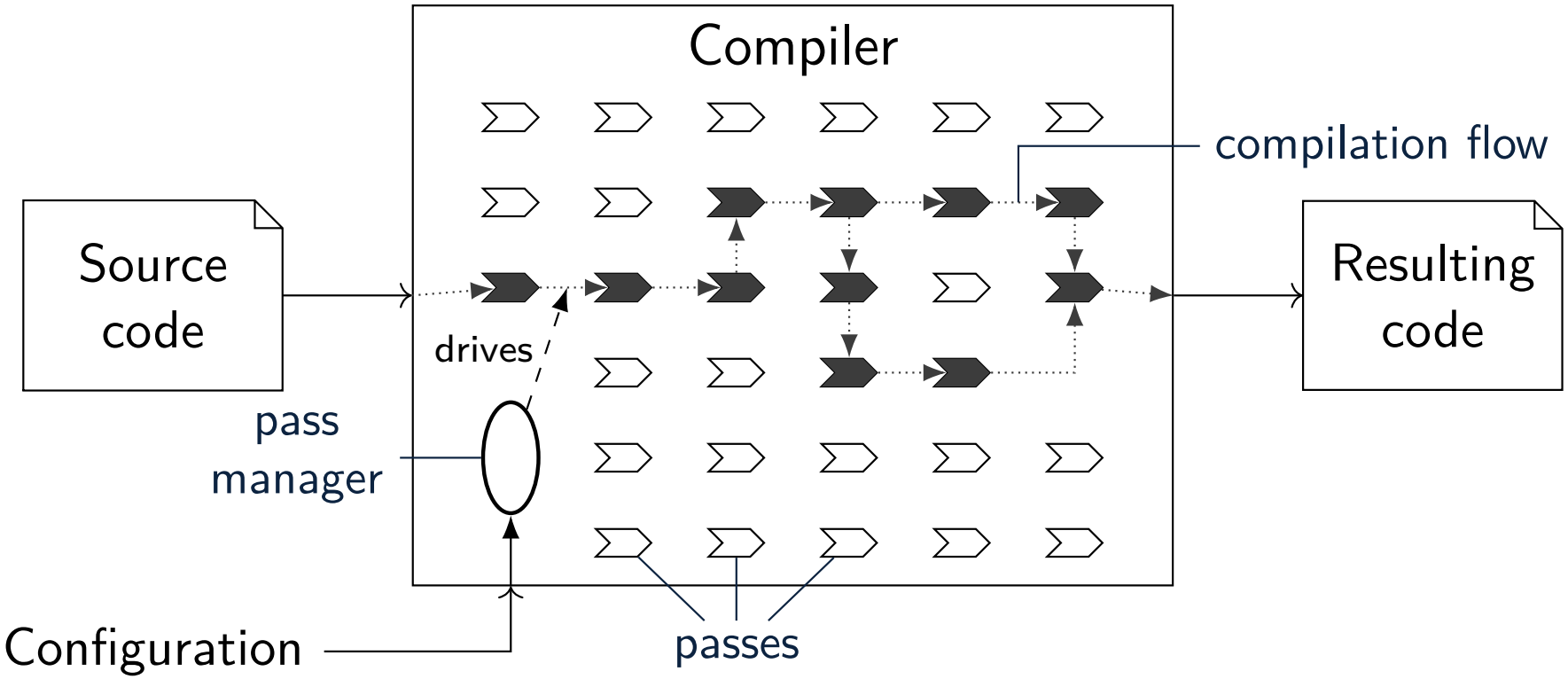
Context
Overview



Context
LLVM Compilation and passes

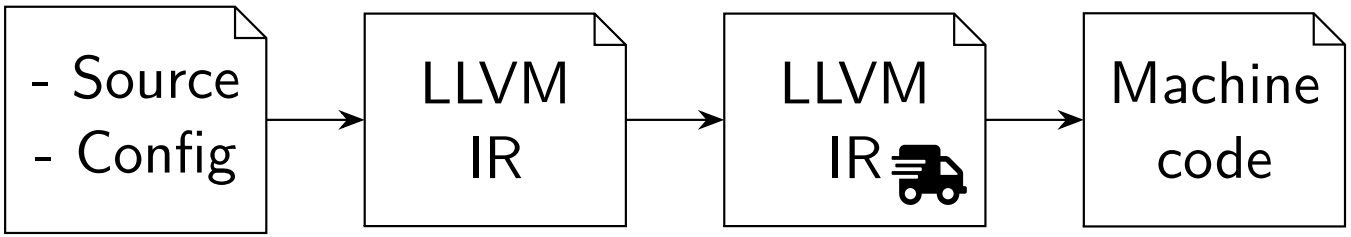


Context
LLVM Compilation and passes



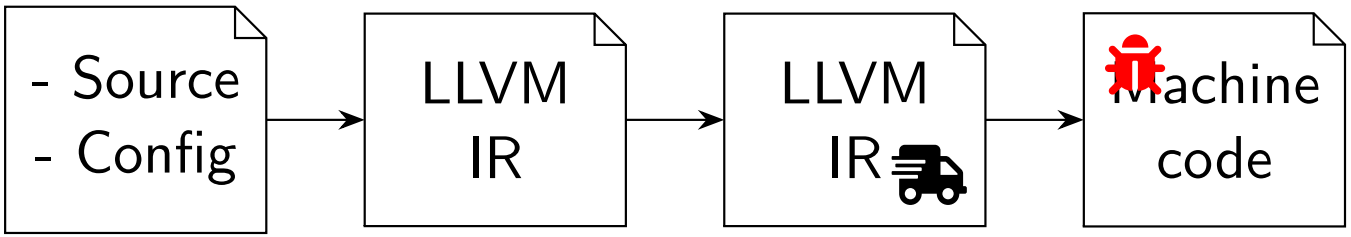
Motivations

Use-case: Debug



Motivations

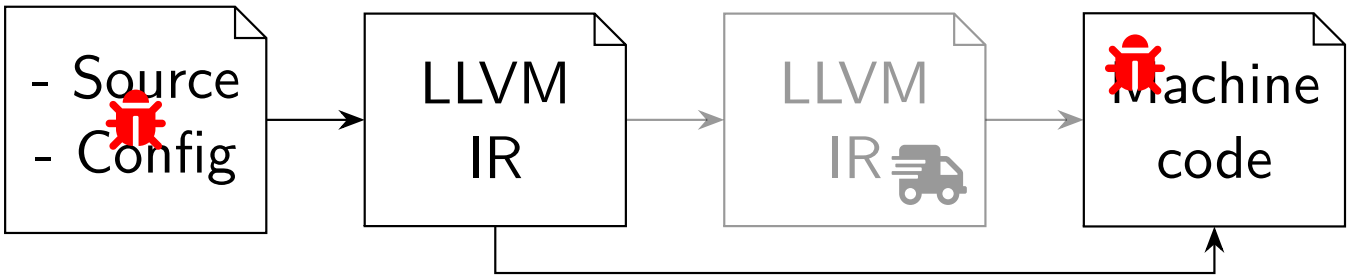
Use-case: Debug



Where is the bug ?

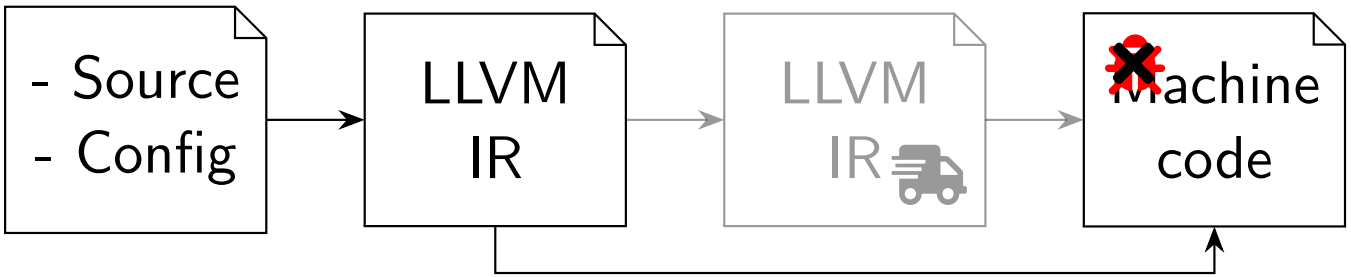
Motivations

Use-case: Debug



Motivations

Use-case: Debug



Where is the bug ?

Backward traceability : the story of instructions, from the produced binary to the source code

Motivations
Traceability

Backward traceability : the story of instructions, from the produced binary to the source code

Forward traceability : the story of instructions, from the source code to the produced binary

Traceability inside compilers

- Create a traceability framework
- Not dedicated to a specific usage
- Implemented in LLVM, but designed with a global approach

Traceability inside compilers

- Create a traceability framework
- Not dedicated to a specific usage
- Implemented in LLVM, but designed with a global approach

Trace is optional

- Enable and disable it on demand
- Partial traces are still useful
- No need to implement trace features in every compiler pass to produce useful data

2 • Existing work about traceability

Quarkslab
Securing every bit of your data

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Definitions of Traceability

In software engineering

The degree to which a relationship can be established between two or more products of the development process [...][1]

[1] Ravensteijn, "Visual Traceability across Dynamic Ordered Hierarchies".
[2] Ibid.

Definitions of Traceability

In software engineering

The degree to which a relationship can be established between two or more products of the development process [...][1]

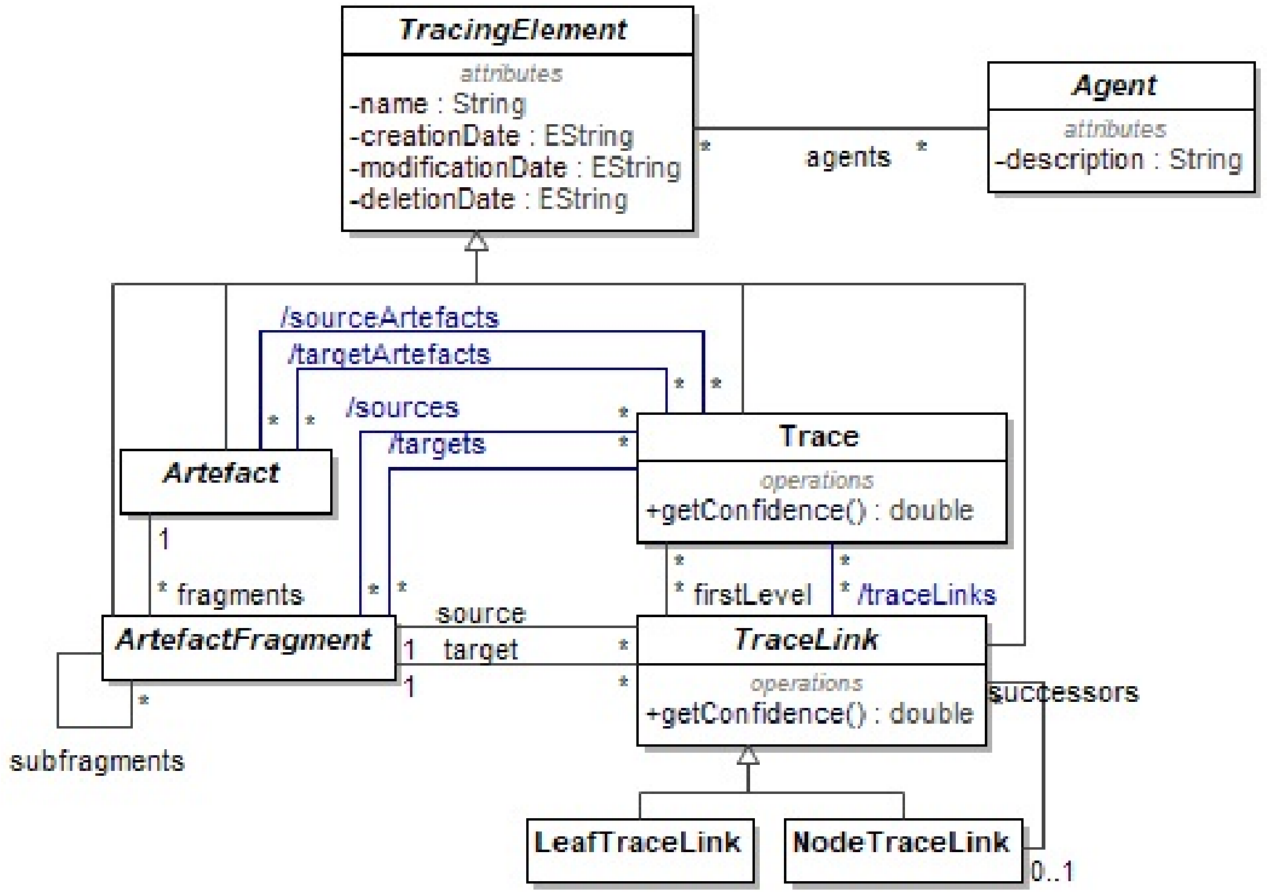
Outside of software engineering

The ability to verify the history, location, or application of an item by means of documented recorded identification. [...][2]

[1] Ravensteijn, "Visual Traceability across Dynamic Ordered Hierarchies".
[2] Ibid.

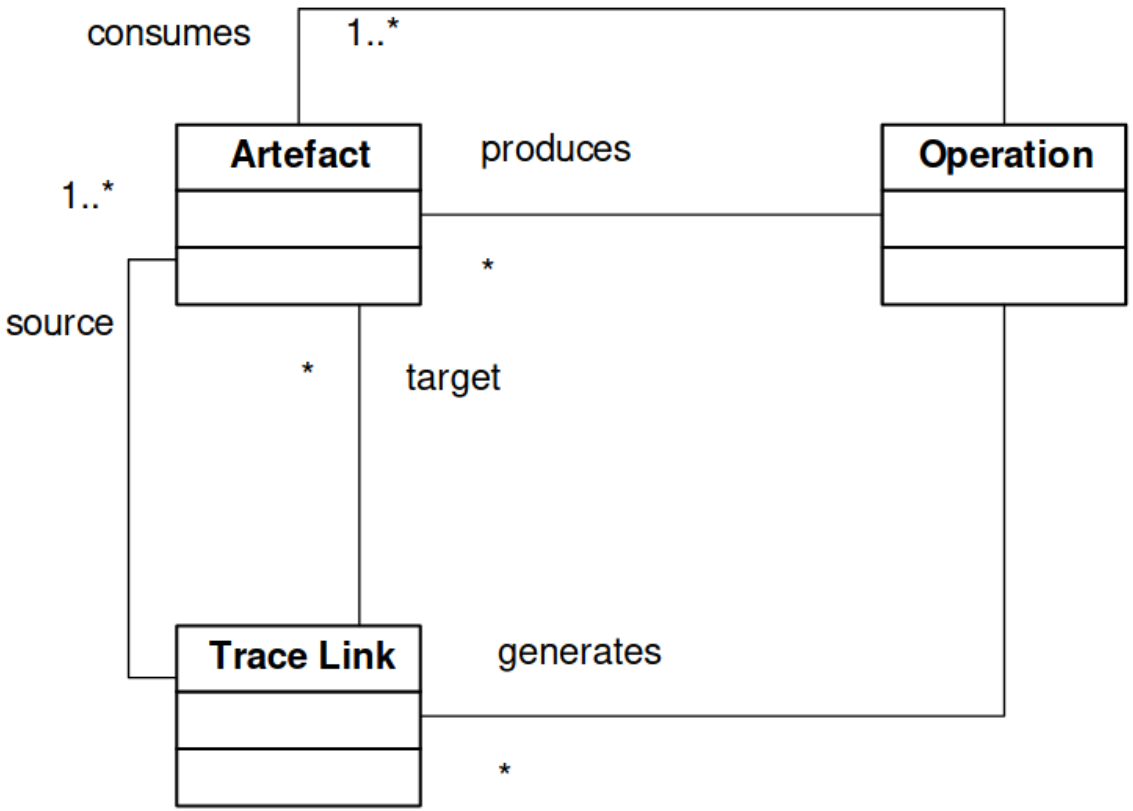
Existing trace model

Traceα[3]



[3]Batot, Cabot, and Gérard, "(Not) Yet Another Metamodel For Traceability".

Existing trace model
TEAP[4]



[4] Paige et al., "Building model-driven engineering traceability classifications".

Trace specification

Existing concepts

- Artefacts: The IR at a given stage of the compilation process
- Trace links: Called events in my case

Trace specification

Existing concepts

- Artefacts: The IR at a given stage of the compilation process
- Trace links: Called events in my case

Instant: Timeline information

- Has a start and an end
- Describes a time window of the compilation process
- Can be nested

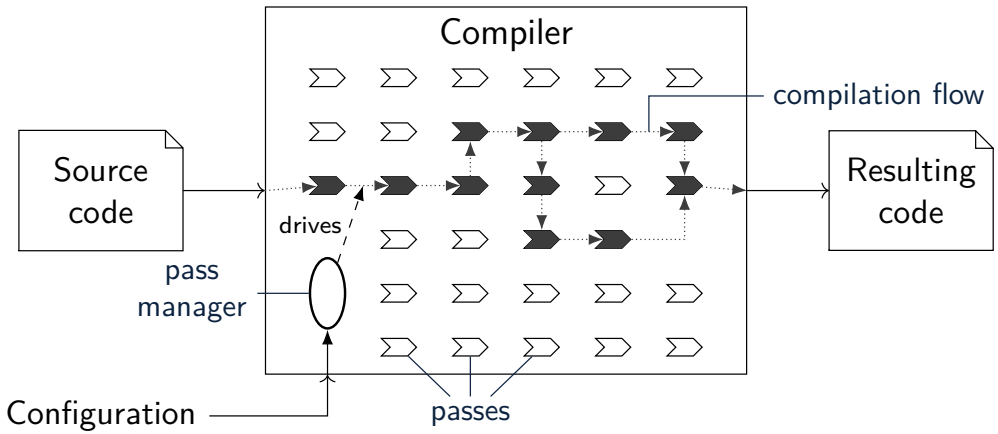
3 • Implementation

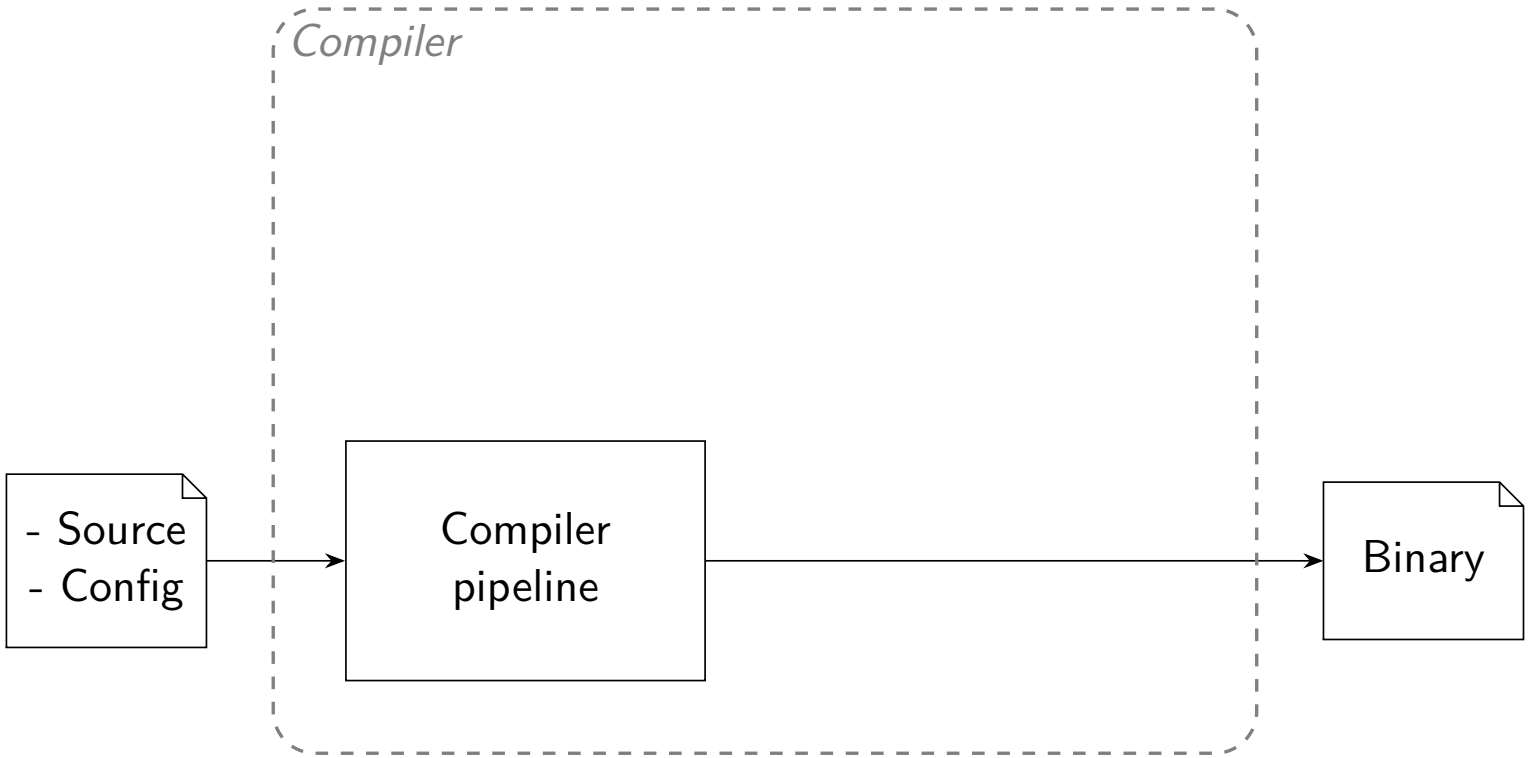
Quarkslab

Securing every bit of your data

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

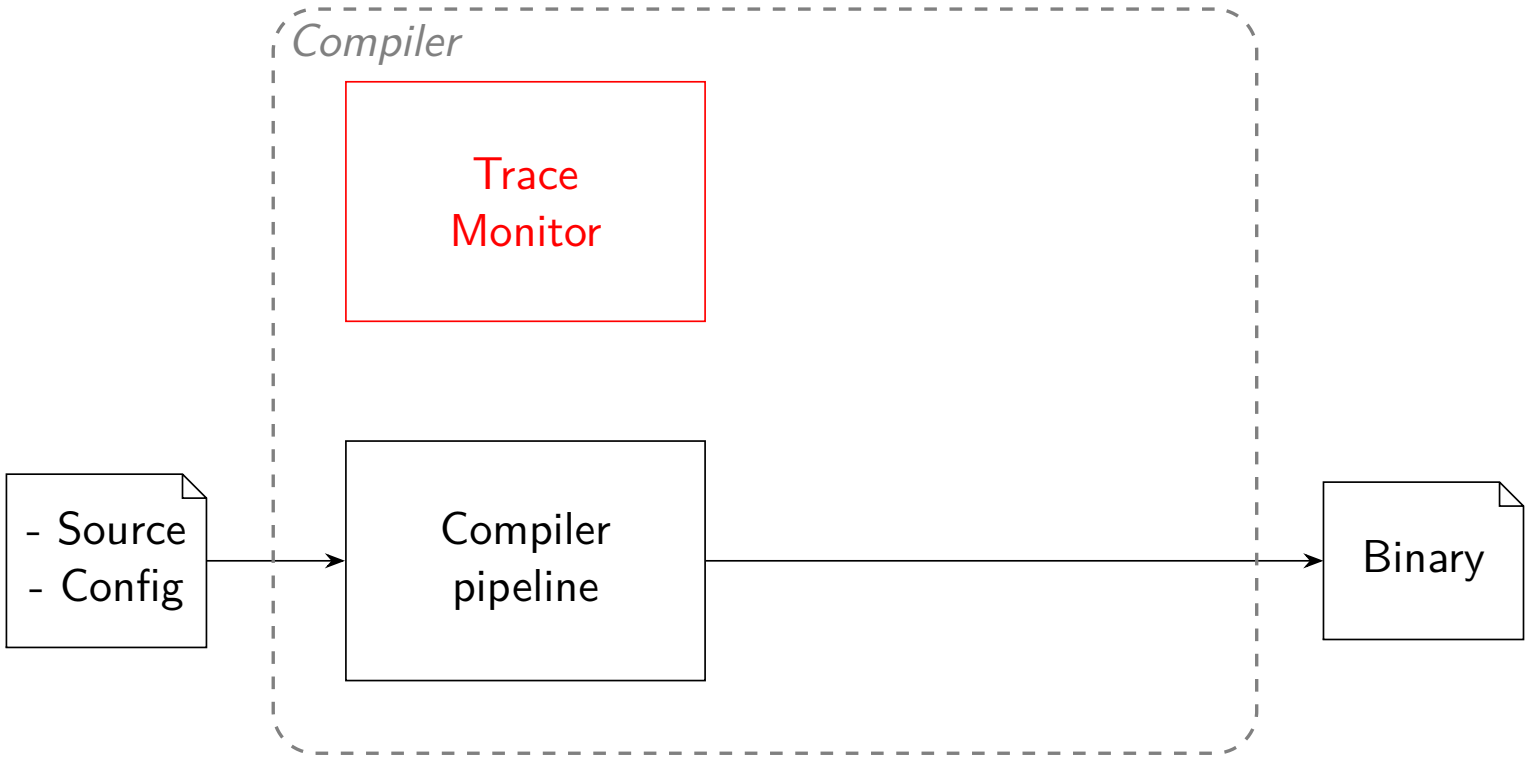
Trace monitor
The trace API





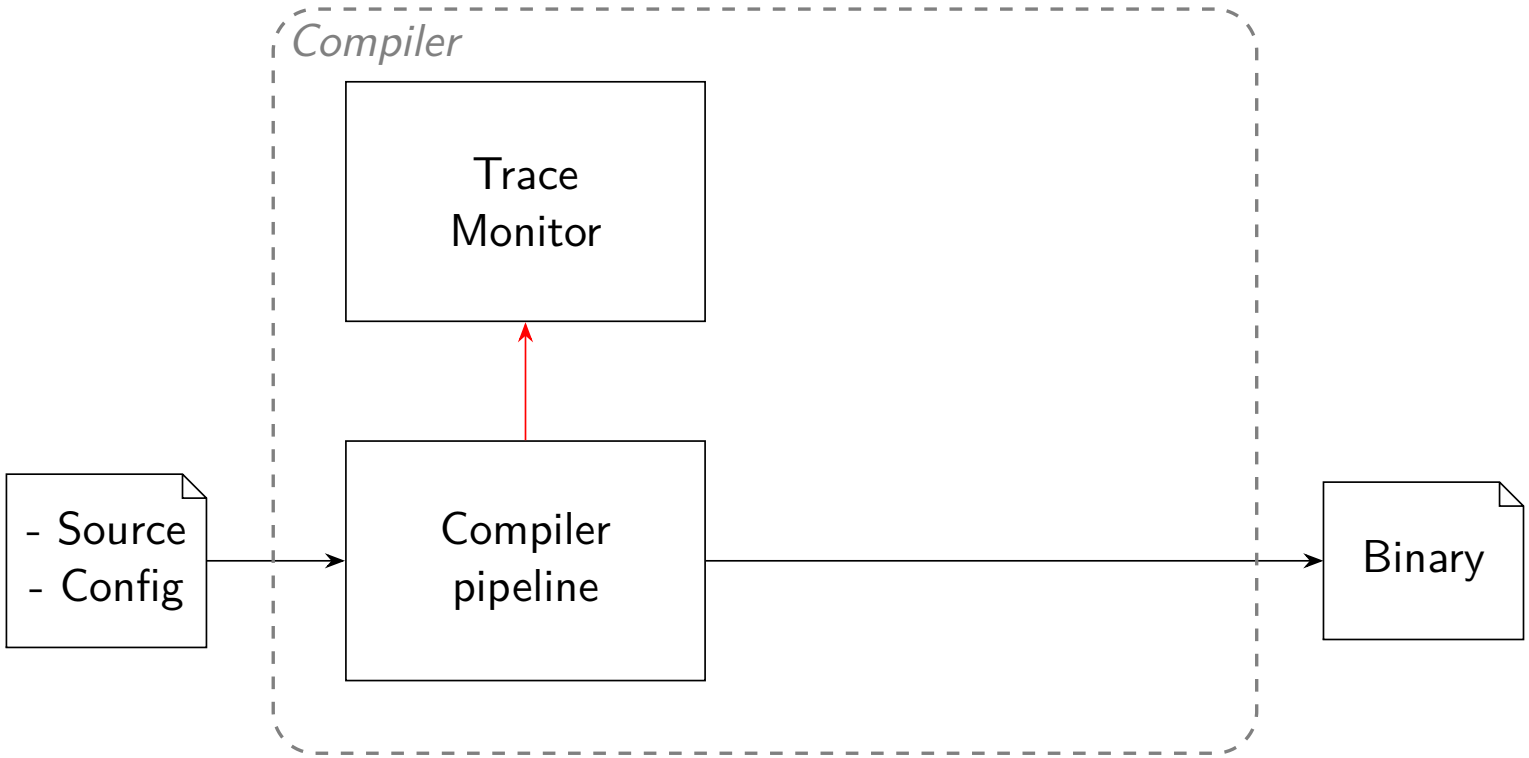
Trace Monitor

- Inside LLVMCore
- API to register Instants and Events
- Accessible from anywhere



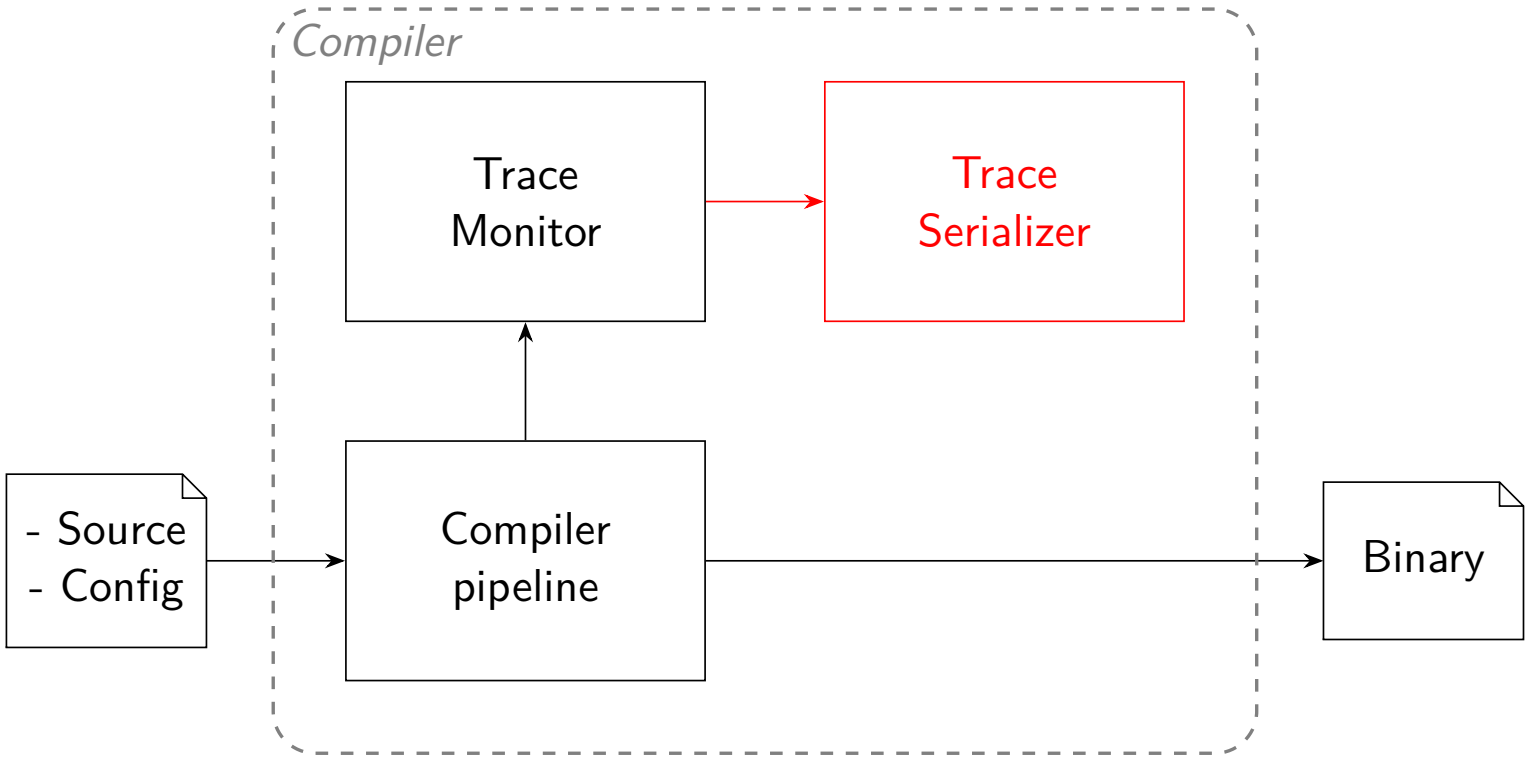
Integration with LLVM codebase

- Modifications to LLVM APIs to use trace instants and events
- New events and instants types can be created to enrich the trace



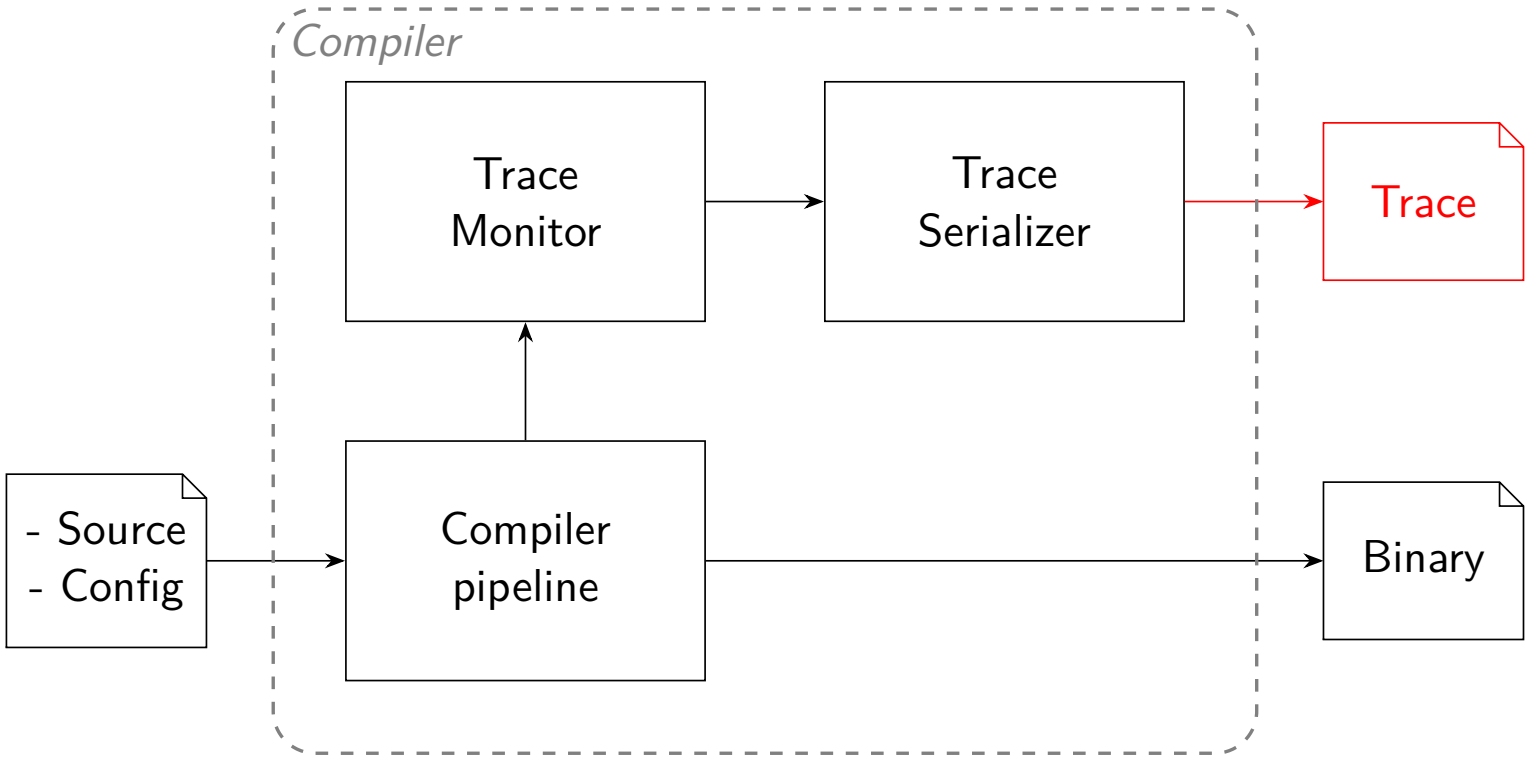
Serializer

- No pre-analysis is done by the serializer
- Easily parseable by external tools



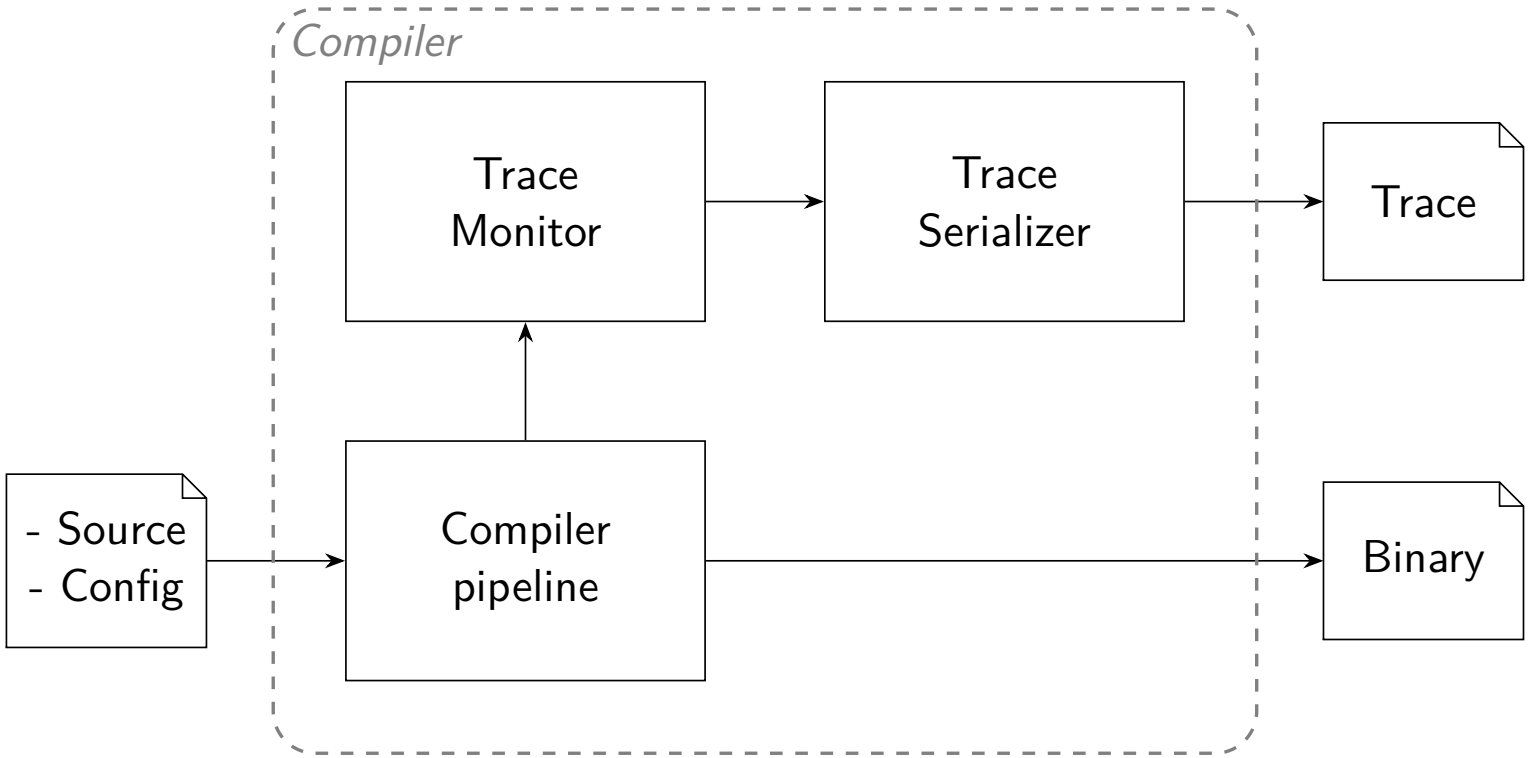
Serializer

- No pre-analysis is done by the serializer
- Easily parseable by external tools

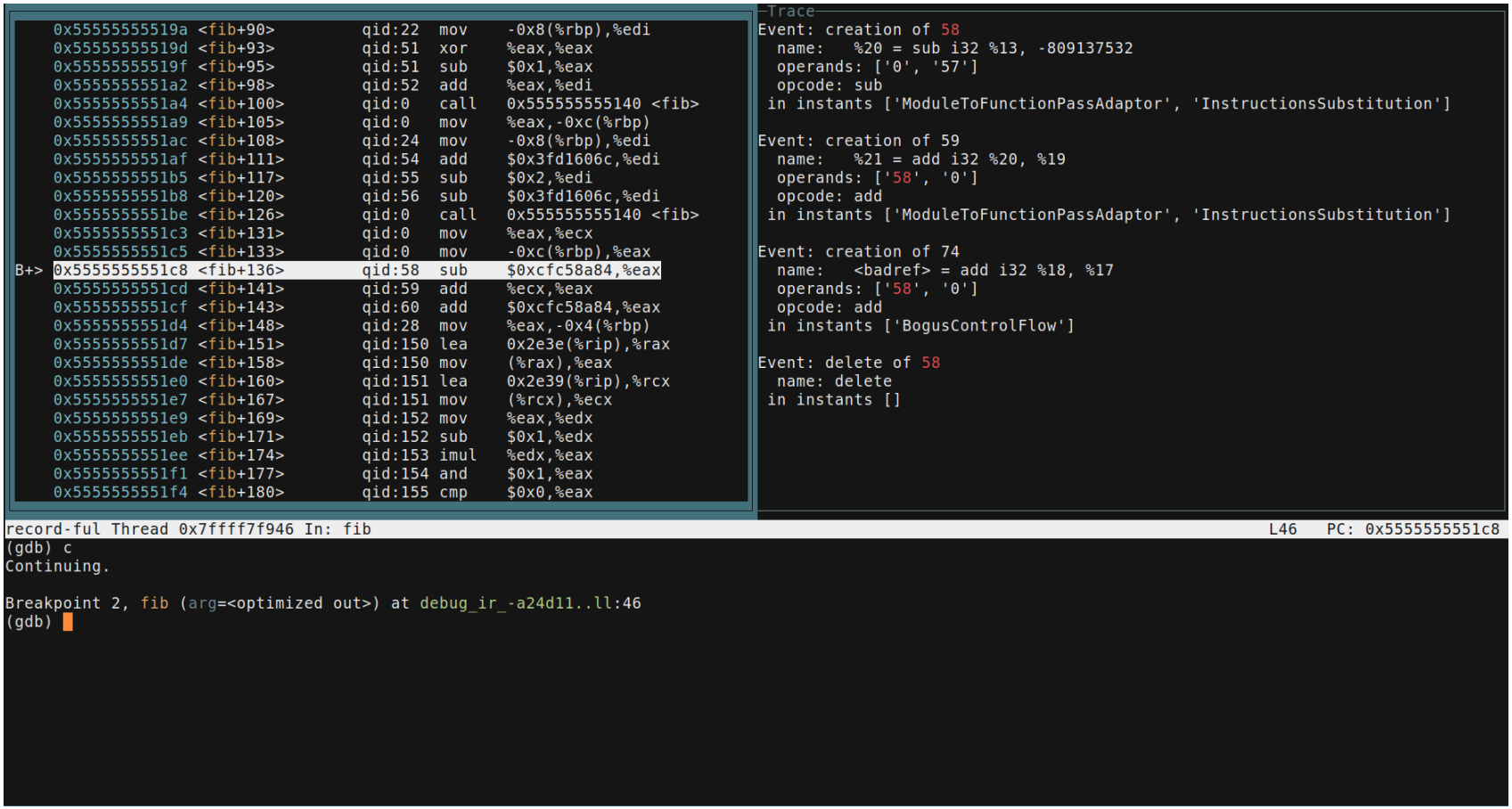


Link with the binary

- Binary and Trace are separate artifacts
- Each *Value* is uniquely identified in the trace



Using the trace



Summary

Trace Model

- Flexible and on-going process

Summary

Trace Model

- Flexible and on-going process

Implementation

- Limited to the middle-end for now

Summary

Trace Model

- Flexible and on-going process

Implementation

- Limited to the middle-end for now

Using traces

- GDB integration
- Future automated tools

Bibliography

- Ravensteijn, W. J. P. van. "Visual Traceability across Dynamic Ordered Hierarchies". MA thesis. Eindhoven University of Technology, 2011. URL: https://research.tue.nl/en/studentTheses/visual-traceability-across-dynamic-ordered-hierachies.
- Batot, Edouard R., Jordi Cabot, and Sébastien Gérard. "(Not) Yet Another Metamodel For Traceability". In: *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. Oct. 2021, pp. 787–796. DOI: 10.1109/MODELS-C53483.2021.00125.
- Paige, R. et al. "Building model-driven engineering traceability classifications". In: *ECMDA Traceability Workshop (ECMDA-TW)*. Sintef. 2008, p. 49. URL: https://www.semanticscholar.org/paper/4d83fdf48055ee609ea7bfff0e467e6eae45e0ff.

Costs

Lua		886kB source, 300kB compiled			
Options		-O0	-O1	-O2	-O3
time (s)	Clang-14	1.16	3.81	4.24	4.42
	Clang-15 patched	17.00	22.36	23.44	23.70
Trace size (MB)		5.80	43.51	48.53	50.04

keepassxc		9.3MB source, 6.9MB compiled			
Options		-O0	-O1	-O2	-O3
time (s)	Clang-14	327.55	371.96	377.75	384.157
	Clang-15 patched	6302.95		7790.97	7721.80
Trace size (MB)		38		250	

LLVM Metamodel