TOOL-BASED INTERACTIVE SOFTWARE PARALLELIZATION: A CASE STUDY

A. WILHELM, F. CAKARIC, T. SCHÜLE, M. GERNDT, ICSE-SEIP 2018

SIEMENS

Ingenuity for life

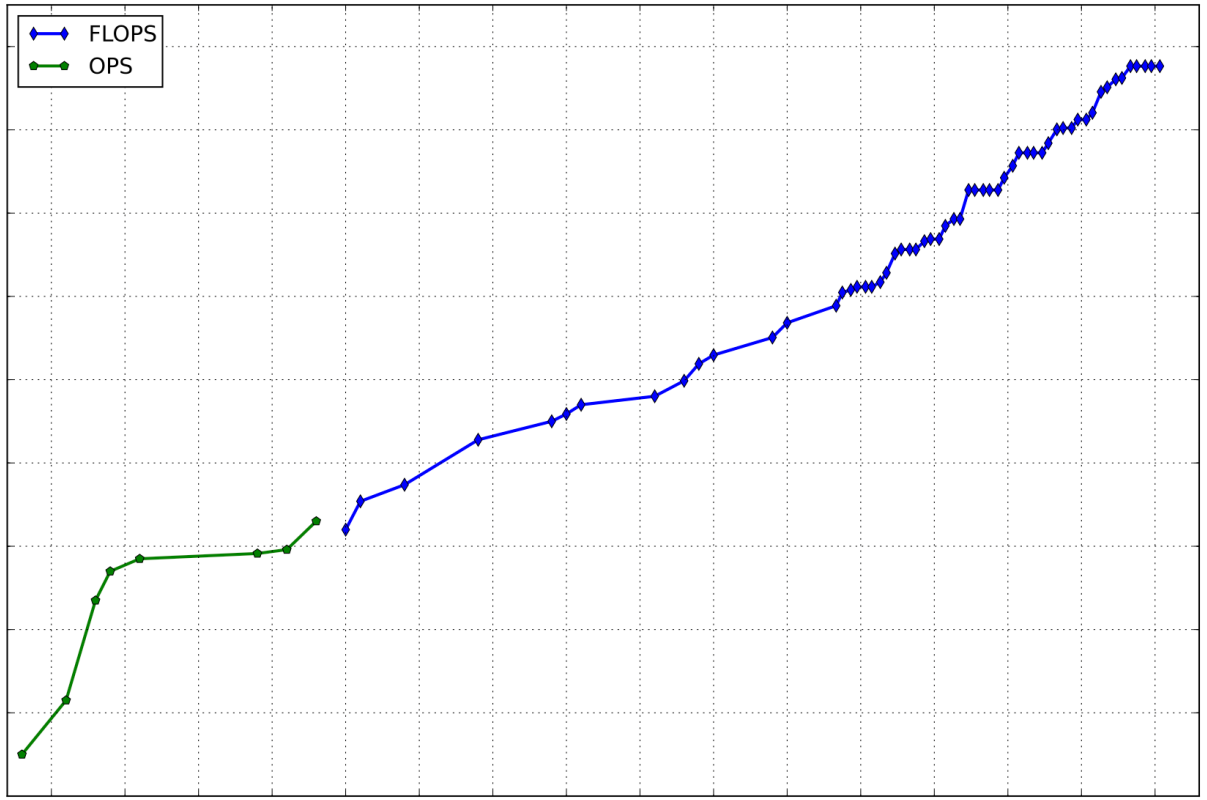
These slides are available at
https://github.com/wilhelma/icse18-presentation

50 YEARS OF SOFTWARE ENGINEERING

A REAL SUCCESS STORY

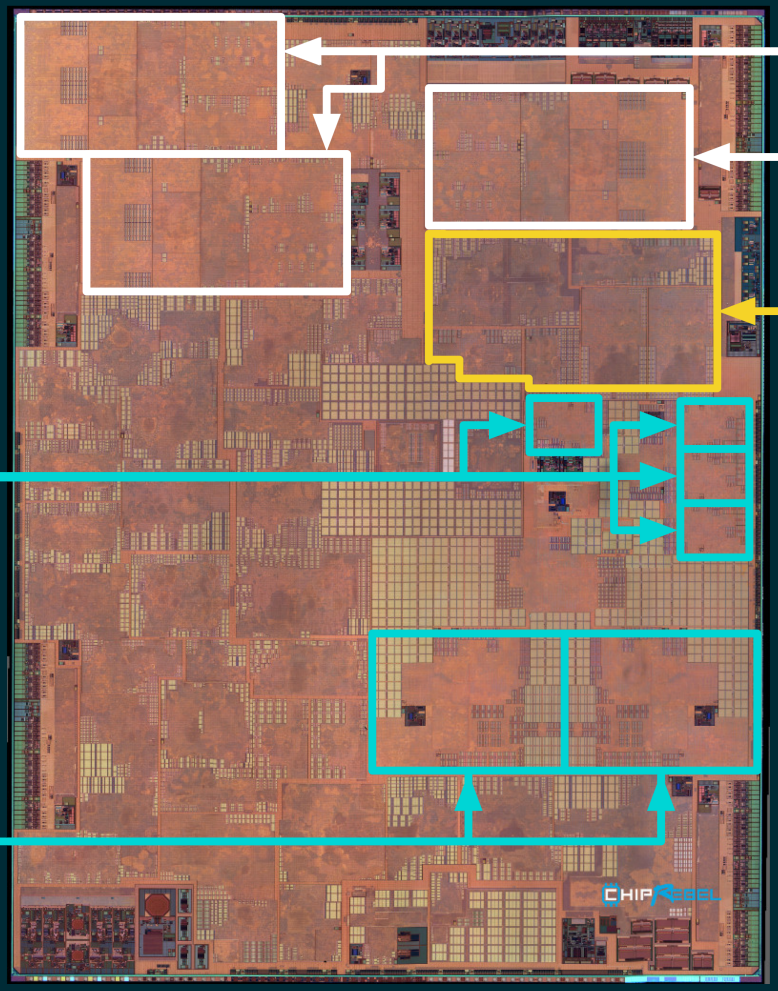
RIDING ON MOORE'S LAW

MOORE'S LAW IS DEAD, WHAT'S NEXT?



https://en.wikipedia.org/wiki/Supercomputer

THE FUTURE IS WILD



https://www.chiprebel.com/apple-a11-bionic

KNOW YOUR DEPENDENCIES

LISTEN TO JOHN L. HENNESEY, ET.AL, NOT ME

https://www.acm.org/turing-award-50/video/moores-law-is-really-dead

LEGACY CODE MEETS MULTI-CORE

https://www.siemens.com

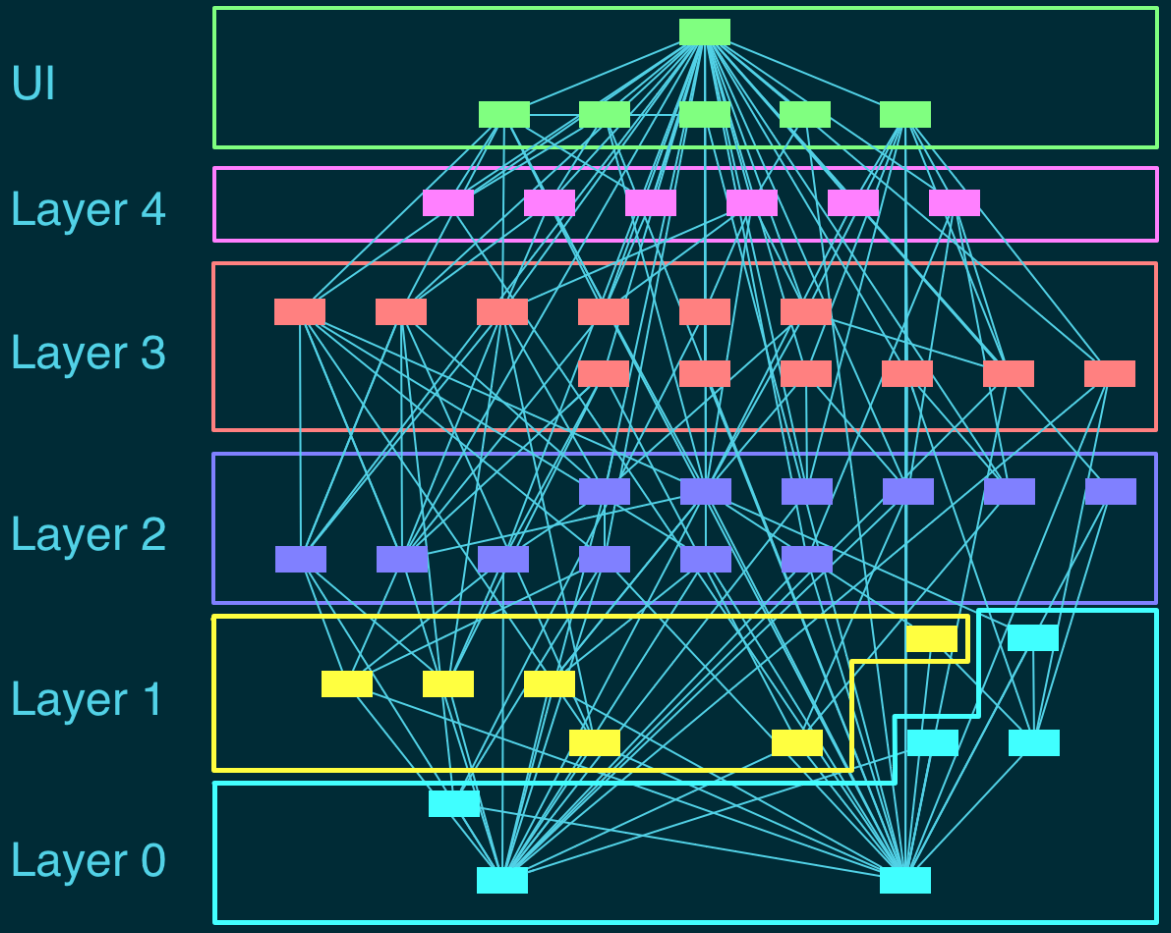
Systems are asynchronous, distributed and event driven in nature.
This should be reflected in the language to define them and the tools to build them [...]

MARGARET HAMILTON, ICSE 2018

OUR MODUS OPERANDI

1. Identify concurrency
2. Redesign the software architecture
3. Parallelize the code
4. Validate and verify
5. Analyze the runtime

HIGH-LEVEL PROBLEM



LOW-LEVEL PROBLEM

Titus Winters
@TitusWinters

Folgen

Overheard at #icse18 - "Or maybe it's legacy code and you don't want to touch it, maybe it'll break."

If you're afraid of your code, you've lost. No haunted graveyards. 😠

Tweet übersetzen

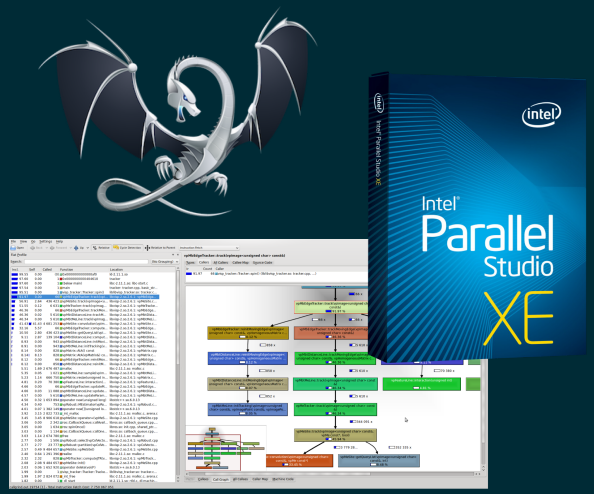
01:00 - 29. Mai 2018

SOLUTION A: HIRE NINJAS

https://github.com/codewarz-ninja

SOLUTION B: USE TOOLS

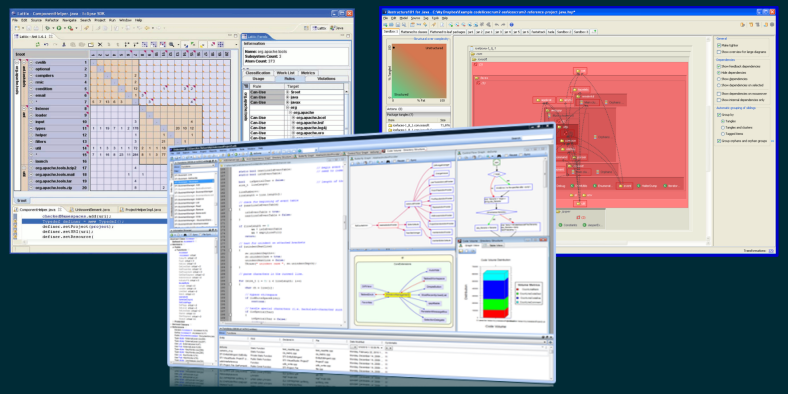
PARALLELIZATION TOOLS



✓ Very useful for *regular* code

✗ Operate on low abstraction levels

ARCHITECTURE ANALYSIS TOOLS



✓ Help to understand and evaluate

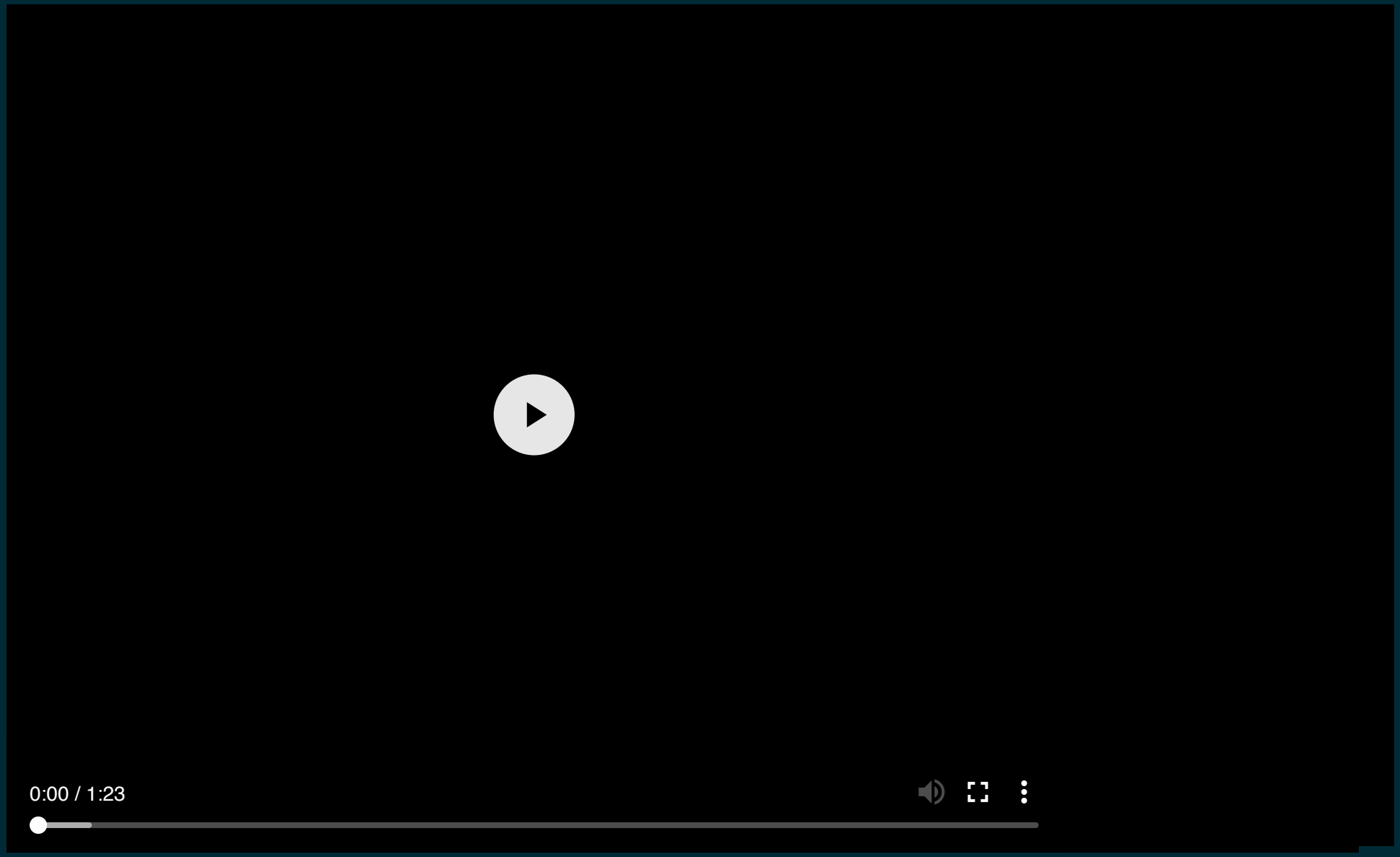
✗ Neglect dynamic aspects

PARALLELIZATION REQUIRES COMPREHENSION

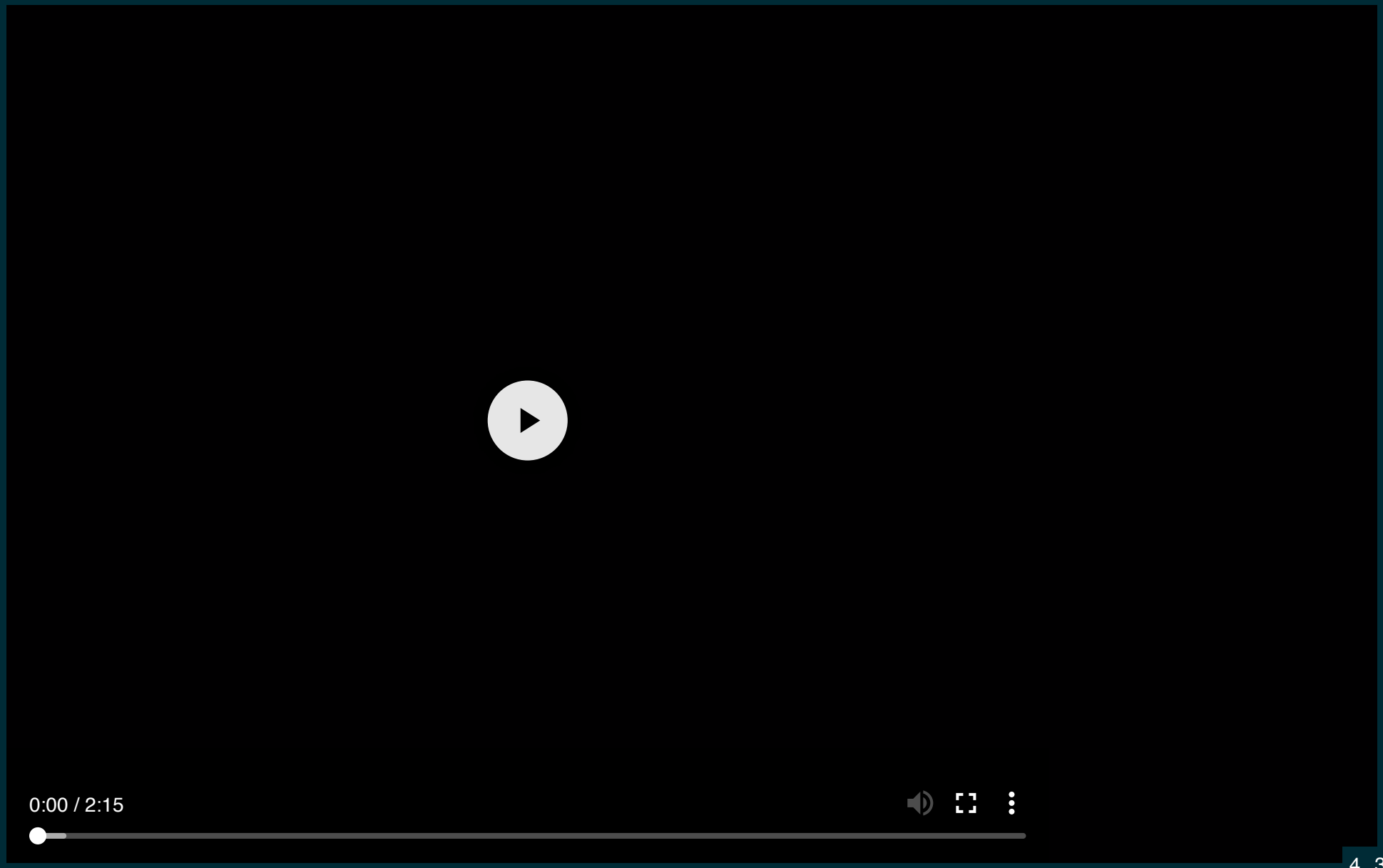
(OF THE STRUCTURE AND THE BEHAVIOR)


PARCEIVE

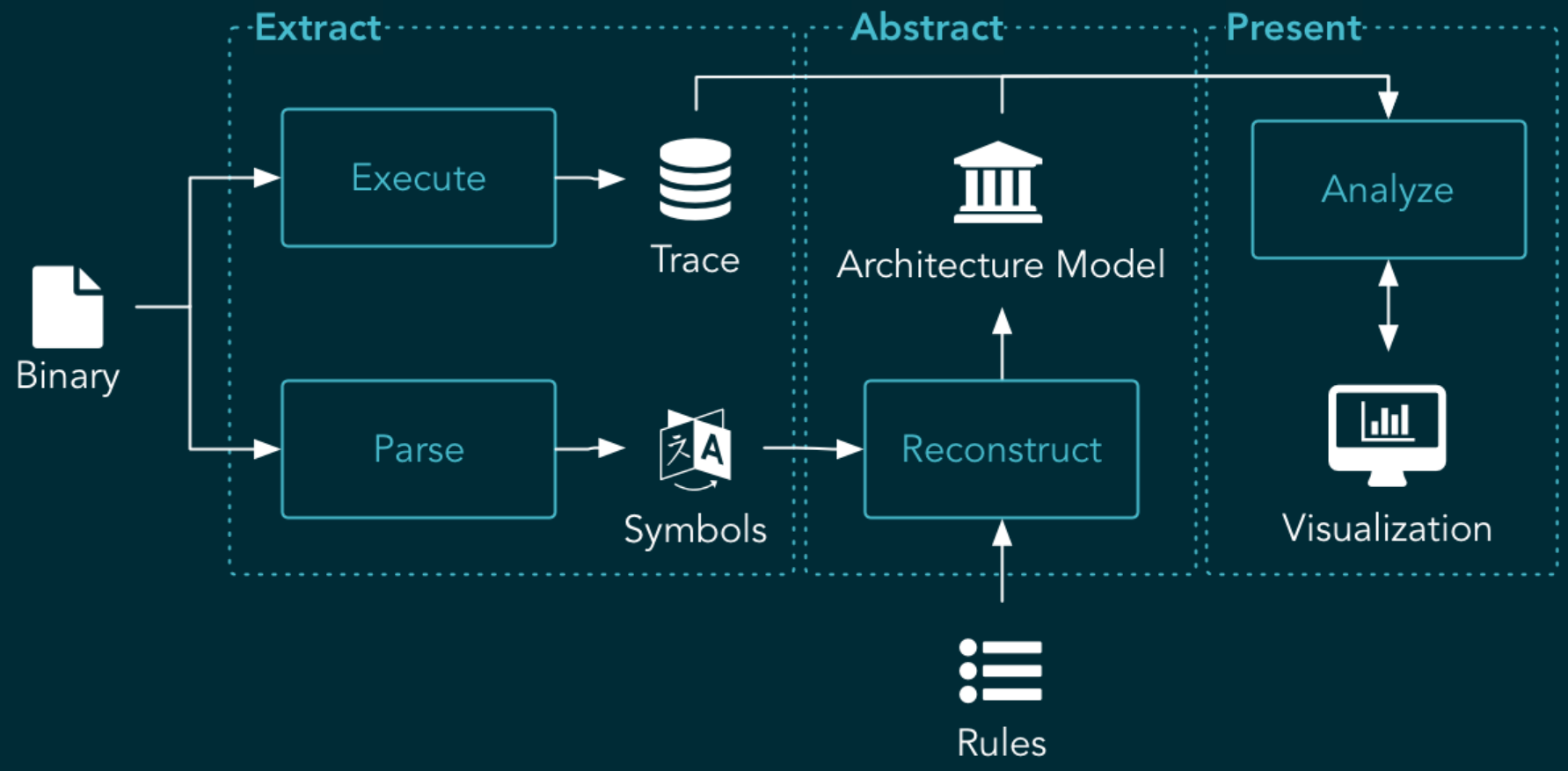
KEY FEATURE 1: DEPENDENCY ANALYSIS



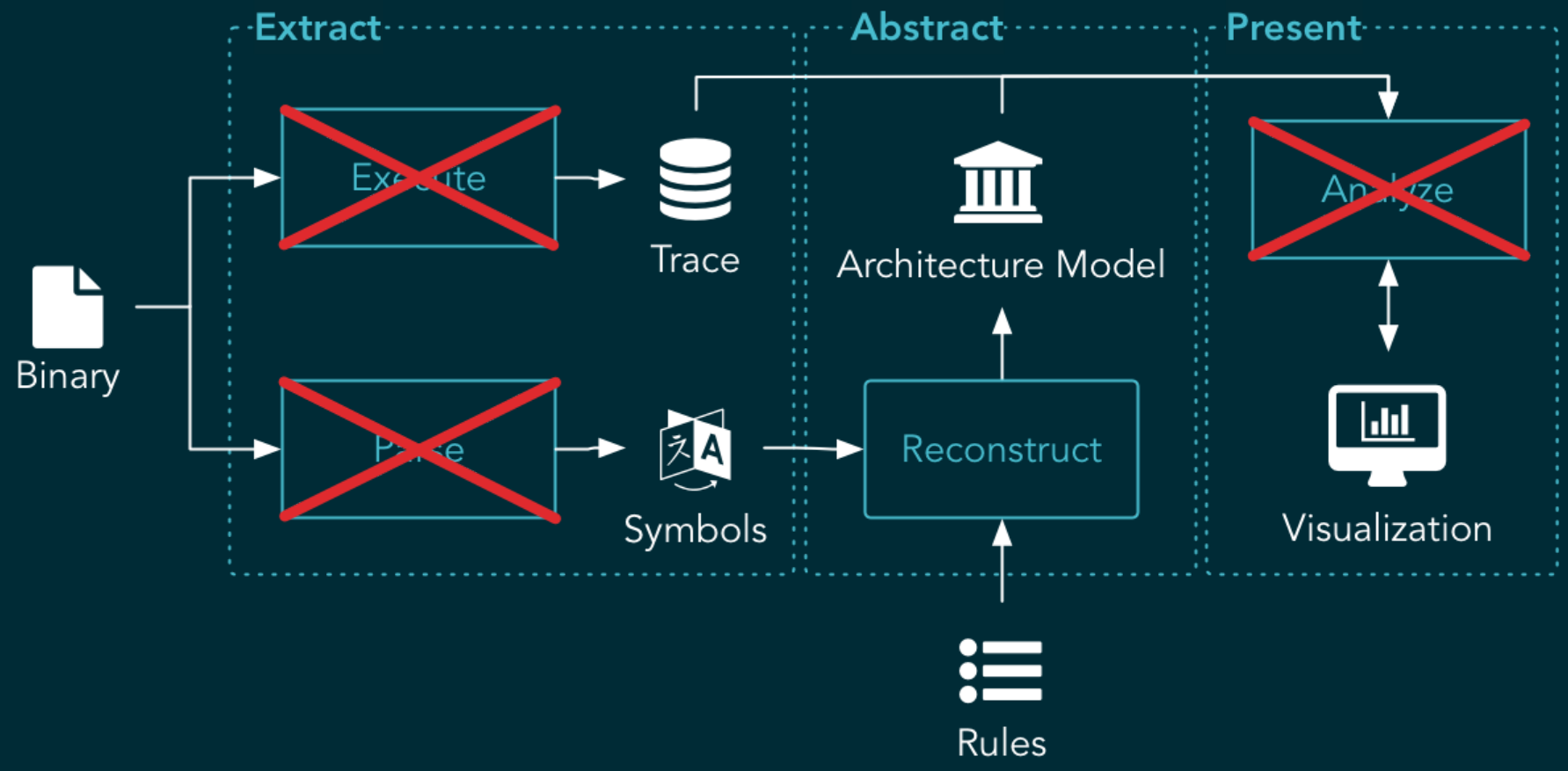
KEY FEATURE 2: ARCHITECTURE ANALYSIS



WORKFLOW


Extract
Abstract
Present
Execute
Trace
Architecture Model
Analyze
Binary
Parse
Symbols
Reconstruct
Visualization
Rules

I WILL NOT TALK ABOUT



SOFTWARE ARCHITECTURE RECONSTRUCTION, ANOTHER TRY?

SO, YOU USE DIRECTORIES?

```
controller := inFile("/src/controller/.**")
```

OR NAMESPACES?

```
logic := namespace("log")
```

SHARED LIBRARIES ANYBODY?

```
libs := image("/3dparty/.*.so")
```

ALL TOGETHER?

```
operations := libs || function("operate") && !logic
```

PARCEIVE HELPS TO...

1. Identify concurrency +++
2. Redesign the software architecture ++
3. Parallelize the code -
4. Validate and verify +
5. Analyze the runtime -+

SOME FINAL THOUGHTS

WHY BINARY ANALYSIS?

- ✓ No source code needed
- ✓ Optimizations and generics
- ✓ Language independent
- ✗ Slowdown
- ✗ Complex deployments

THE FACTS

Biggest software applied: >10M loc
Supported plattforms: Linux, Windows
Supported languages: C/C++
Current state: Working prototype

WE STRIVE FOR COLLABORATIONS

I brought some business cards

QEUSTIOSN?