– P2P2P –

“PlusCal to PlantUML to PDF”

Making Formal Methods Explainable

Juan José Serrano (presenter), Alejandro Benito, Jackson Belzer and Christian Lamb

TLA+ Community Event @ ETAPS 2026

Table of contents

01

Problem

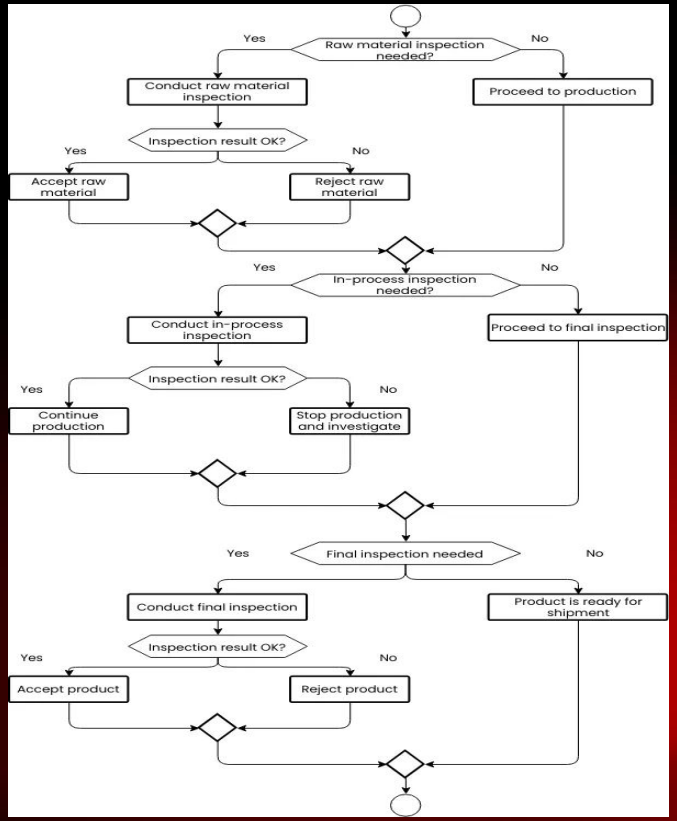
The Problem

Formal methods are powerful...
but hard to communicate

- Engineers -> code
- Stakeholders -> Visuals
- Formal models -> in between

Diagrams Are the Bottleneck

- Manual & Slow
- Quickly Outdated
- Not Trustworthy



SanDisk Current Workflow

English Specs

UML Diagrams

Reviews

Verification

PDF

Manual Reviews Risk

Manual Diagrams ≠ Verified Logic

- Hand-crafted diagrams
- Verification not enforced
- Risk of hidden errors

What if documentation came directly from verified models?

02

Solution

P2P2P Pipeline Overview

PlusCal

PlantUML

PDF

Your Team/Client :D

What It Does

VSCode Integration

Automatic Diagram Generation

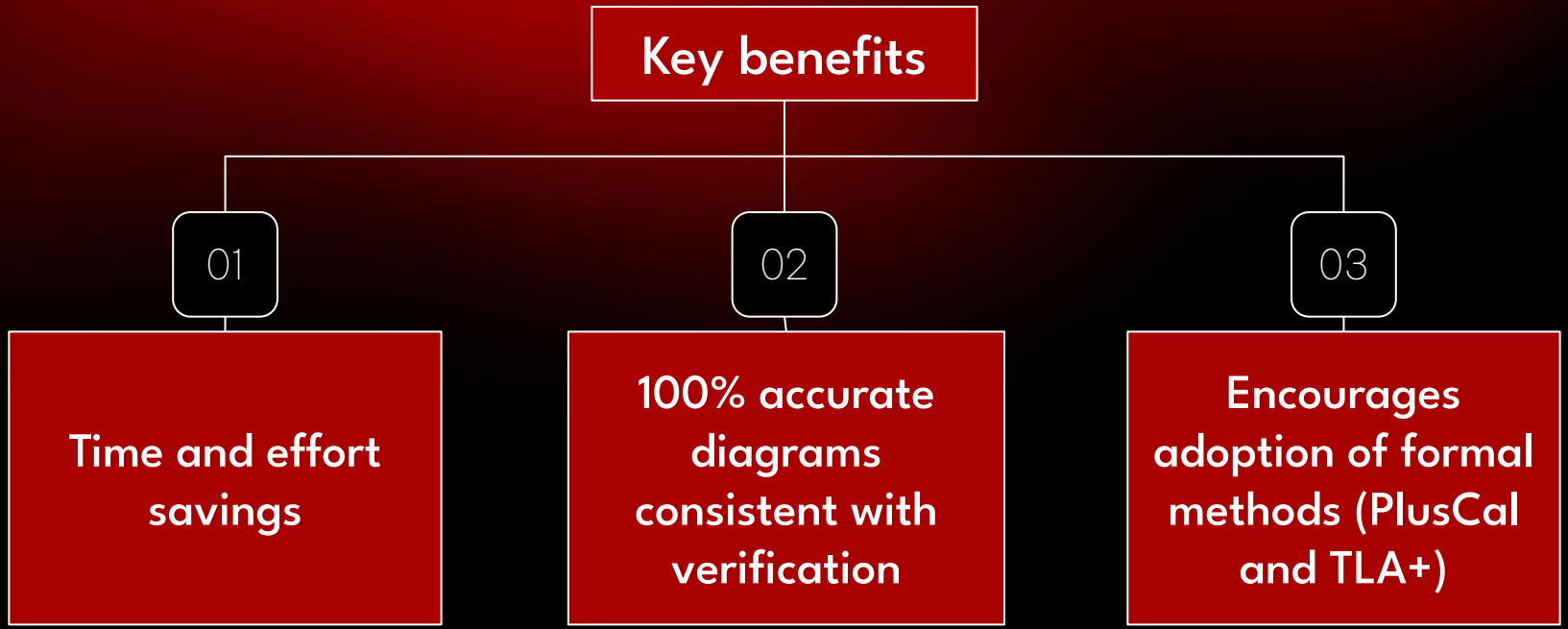
Professional PDFs

Key Idea

Making formal methods social

Math  Visual  Shared understanding

Value of the Solution



Core Principle

If it's not verified,

it doesn't get documented

03

Transformation

PlusCal Model

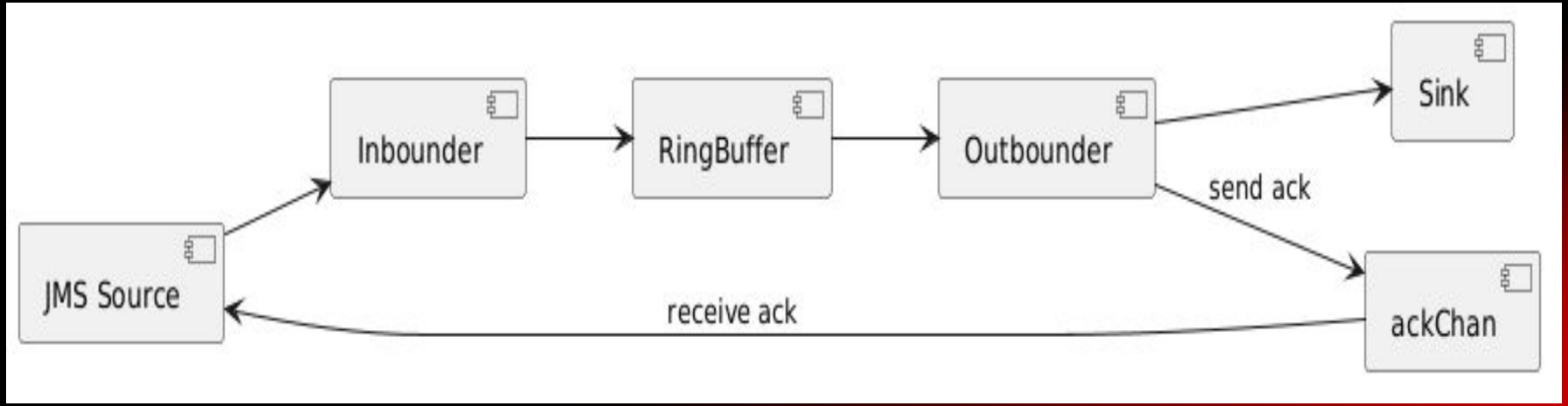
3 processes + shared queues

```
process JMSSource:
  source := Append(source, uid);

process Inbounder:
  DuplexTransferOne(source, sentUnacked, ringBuffer);

process Outbounder:
  TransferOne(ringBuffer, sink);
  ackChan := Append(ackChan, event);
```

System Structure



Key Semantics

- Message ordering
- Bounded buffer (RingBufSz)
- Acknowledgment correctness

Verification (TLC)

Config

Model Checking:

- RingBufSz = 16
- State constraint:
  - Len(source) < 11
  - Len(sink) < 13

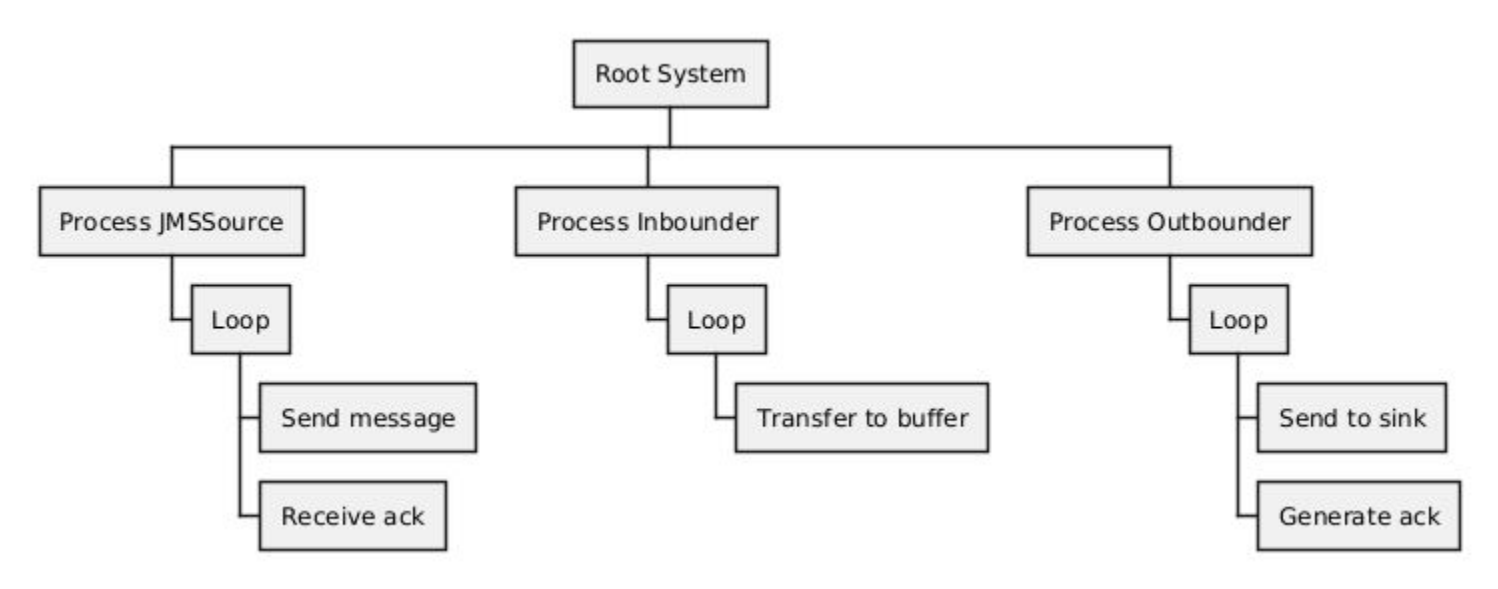
Properties

Assertions:

- ack = inmsg
- No duplicate acks
- sink[i] = i (ordering)

Model Structure (AST)

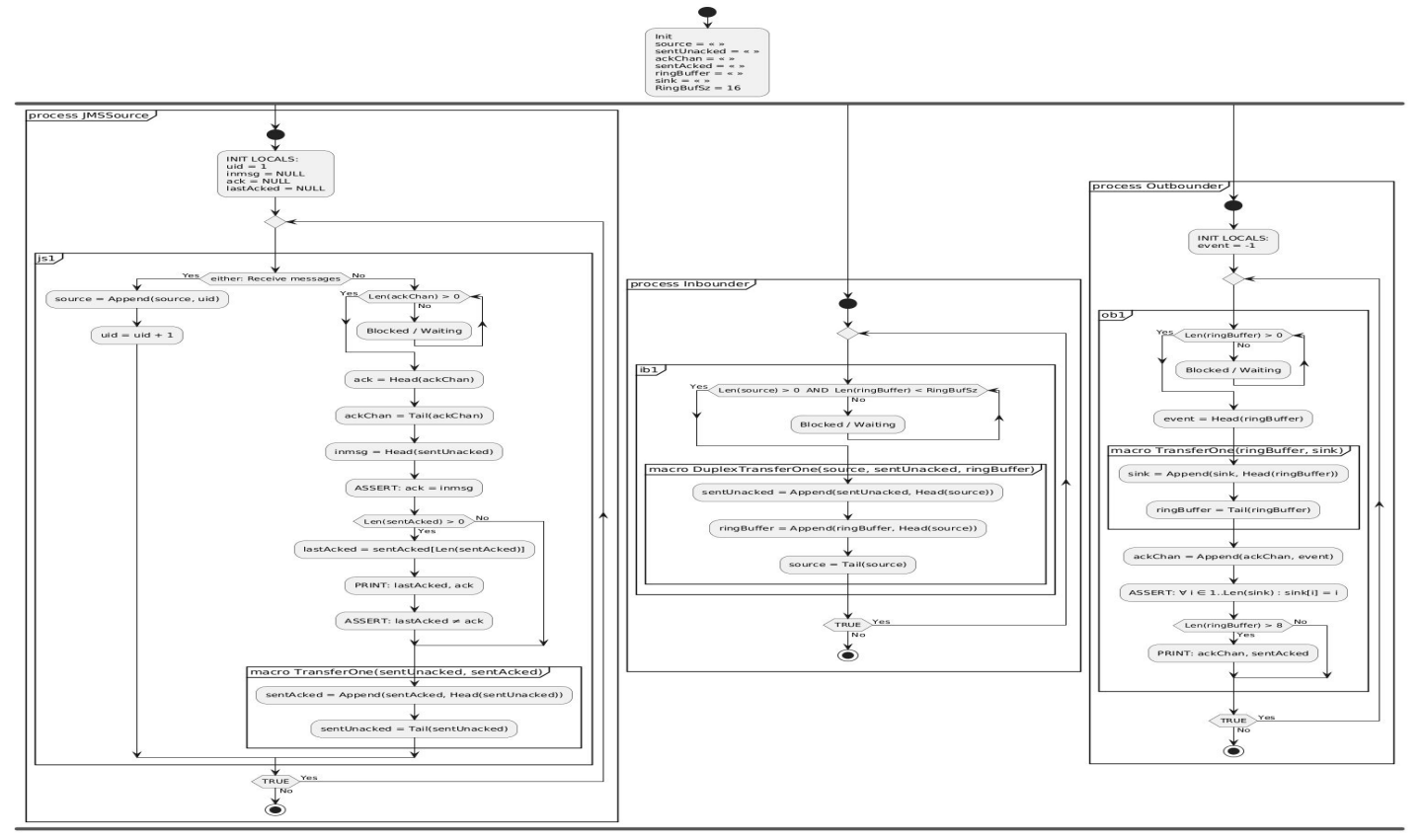
Extracted from TLA+ model



Mapping to UML

PlusCal	Diagram
process	swimlane
while	loop
await	guard

Generated Diagram



What you see

=

What was verified

PDF Output

DONE RIGHT (→)
DESIGNED RIGHT

SANDISK™

Challenging the Status Quo By Introducing Cheap and Practical Formal Methods

A More Precise and Scientific Way to Achieve Quality

DIAGRAM EXPORT | FEB.2026

HASH KEY: e610a5e1cc4bbd6e6ec0befe3daa908d73f5cf716ed371daa7dd3e8d25f744fa

VERIFICATION:

COMMIT: 8d7b620f36e33ce9041eb49caa5812d59a814564

TLA+: (notavailable)

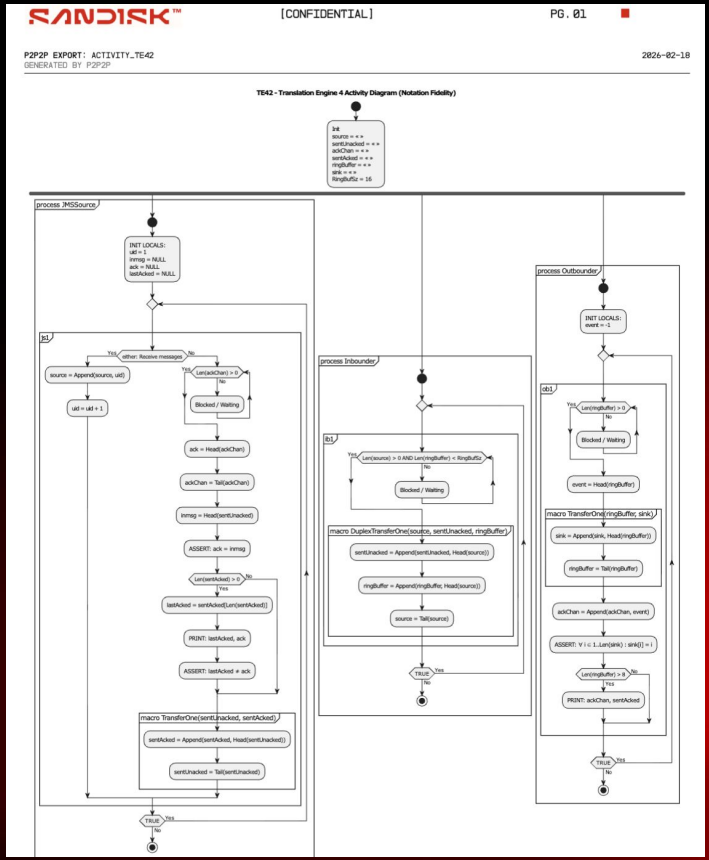
TLC OUTPUTS: (notcomputed) {RESULT+STATS+COVERAGE+LOG}

TLC STATUS: UNKNOWN

PLML: bd28a6c7086974286ddf7772183b099fe57be3c6ad6712392fe6b123c60942ba

MANIFEST: ACTIVITY_TE42_ACTIVITY.JSON

P2P2P-TEAM "SANDISK CONFIDENTIAL" FEB.2026



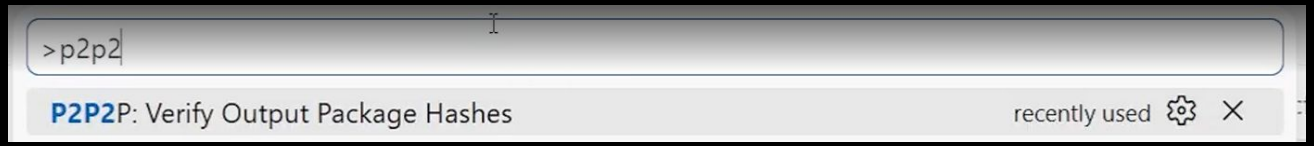
Supported Diagrams

Activity

Sequence

Predicate-Action

Hashing




```
[PASS] PUML File (TwoPhaseCommit.puml): Hash matched
(4bf2eb2fba6ae95b02ed579a762d9c20fb3112f67184d9688f2ea48974a08aae)
[PASS] TLA File (TwoPhaseCommit.tla): Hash matched
(4f265e26e61a17b4ebde3845b3b9a0b96d314fafab69d83cecaf80c4538ab086)
[PASS] Embedded Diagram (diagram.png): Hash matched
(abe4db703f369eb71f56a584ebeedcaacc4c3e5c53b3baca1daa4c0f2f7eccda)
[INFO] Final PDF exists in package (TwoPhaseCommit_state.pdf).

✅ SUCCESS: All hashes verified successfully.
---- P2P2P VERIFY PACKAGE END ----
```

ⓘ Package Verification Passed! All files matched the manifest hashes.

```
(4bf2eb2fba6ae95b02ed579a762d9c20fb3112f67184d9688f2ea48974a08aae)
[FAIL] TLA File (TwoPhaseCommit.tla): Hash mismatch!
  Expected: 4f265e26e61a17b4ebde3845b3b9a0b96d314fafab69d83cecaf80c4538ab086
  Actual:   b75b0782e9c003dc1994d490750f038495835500cd69c6d5c0c62b42069a7922
[PASS] Embedded Diagram (diagram.png): Hash matched
(abe4db703f369eb71f56a584ebeedcaacc4c3e5c53b3baca1daa4c0f2f7eccda)
[INFO] Final PDF exists in package (TwoPhaseCommit_state.pdf).

❌ FAILURE: Hash mismatch detected. Files may ha
---- P2P2P VERIFY PACKAGE END ----
```

ⓧ Package Verification Failed. See P2P2P output for details.

Living Documentation

- Always updated
- Always verified
- Always consistent

Live Demo


DONE RIGHT (→)
DESIGNED RIGHT
SANDISK™
Challenging the
Status Quo
By Introducing
Cheap and Practical
Formal Methods

04

Impact

Team Impact

Faster
Reviews

Less
Manual Work

Fewer
inconsistencies

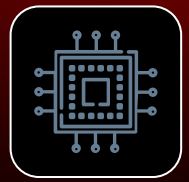
Methods Impact

- Lower barrier to TLA+
- Better communication

Guarantees

- *Less than 60 seconds per model*
- *Deterministic outputs*
- *100% accuracy with verified PlusCal*

Credibility



Unit Testing

Functions in Rust modules (parser, AST, codegen)

Integration Testing

Full Pipeline (from TLA+ to PDF)

Usability Testing

Visual Studio Code workflow

Tests run continuously during development (CI + pre-demo validation)

05

Architecture

High-Level Architecture

01

VS Code Extension

As an user interface

02

Rust Compiler Pipeline

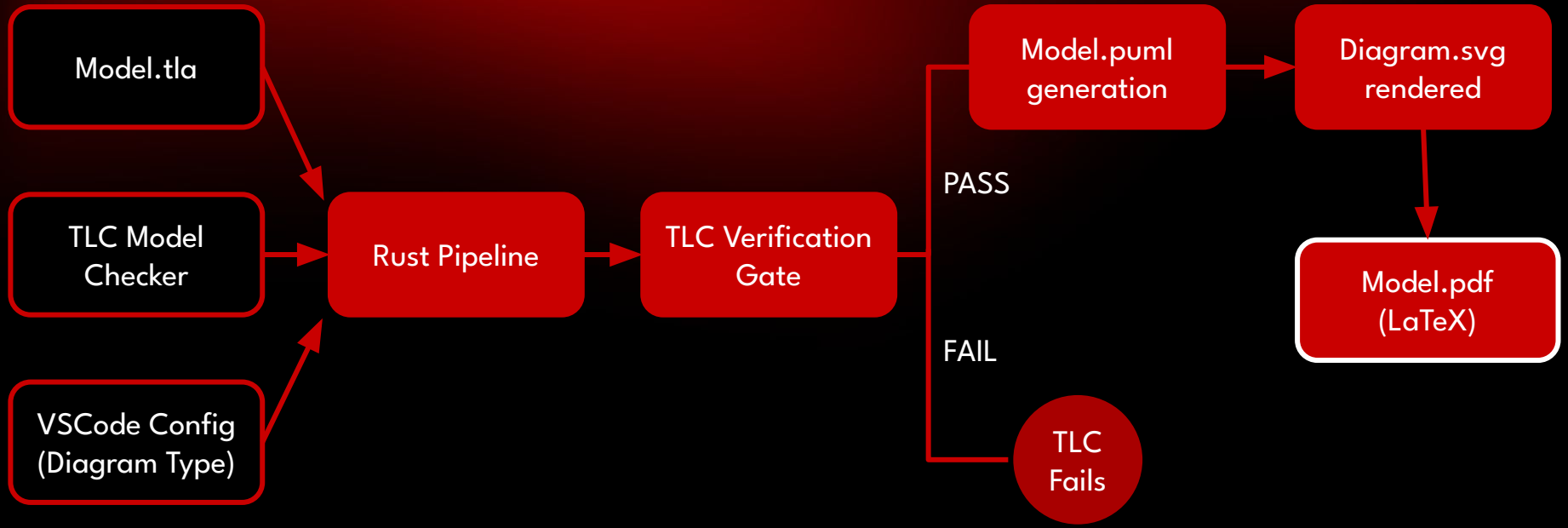
Orchestrates the compilation of the project and ensures correctness

03

LaTeX PDF

With AI insertions to document

Data Flow


Model.tla
TLC Model Checker
VSCode Config (Diagram Type)
Rust Pipeline
TLC Verification Gate
PASS
FAIL
Model.puml generation
Diagram.svg rendered
Model.pdf (LaTeX)
TLC Fails

Architectural Decisions

Modular Monolith

Modular monolith over microservices

Distributed scaling would add latency, complexity and dependances

Pipeline Dependency Model

Each stage depends only on artifacts from the previous stage

This ensures determinism and traceability

Rust Backend

Memory safety
High performance
Strong type system

Critical for large AST parsing

06

Closing

Formal methods are only useful

if people understand them

Future Work

More diagram types

Better customization

Wider adoption

TLA+ proves correctness

P2P2P proves it to the people

Thanks!

Do you have any questions?