Reducing Time and Efforts When Verifying Large Software Systems with Klever

Ilia Zakharov

Ivannikov Institute For System Programming of RAS
ilja.zakharov@ispras.ru

Klever Verification Framework

Intended for finding bugs in large software systems using existing software verification tools

- C programs
  - GNU C, Microsoft C
  - x86, ARM
  - > 200 KLOC
- Requirements supported by software verification tools used as backends
  - Memory safety
  - Data races
  - API usage
- Deliver both results and a program-adapted tool
  - Coverage reports, error traces and results of bugs triage
  - A tool to check regressions or reproduce results

Workflow

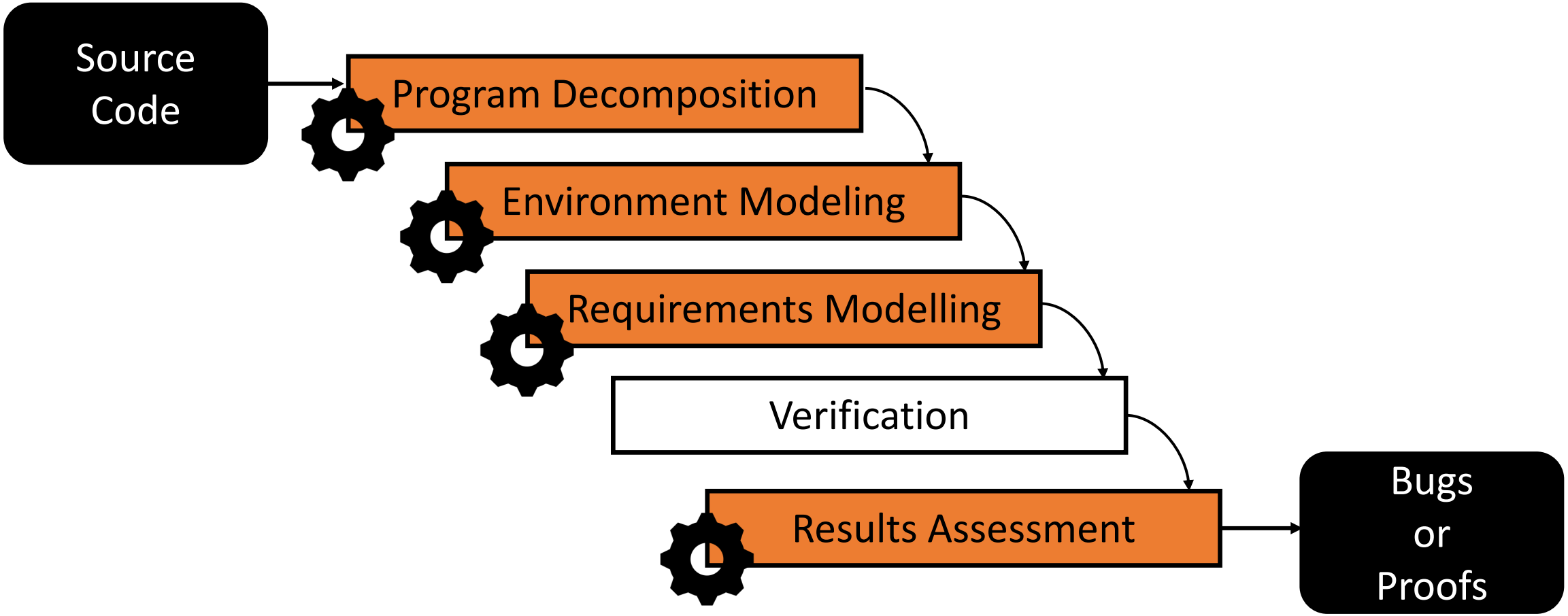
Development

- Implement program-specific parts in the framework
- Prepare models and specifications

Verification

- Find bug
- Deliver the tool and results

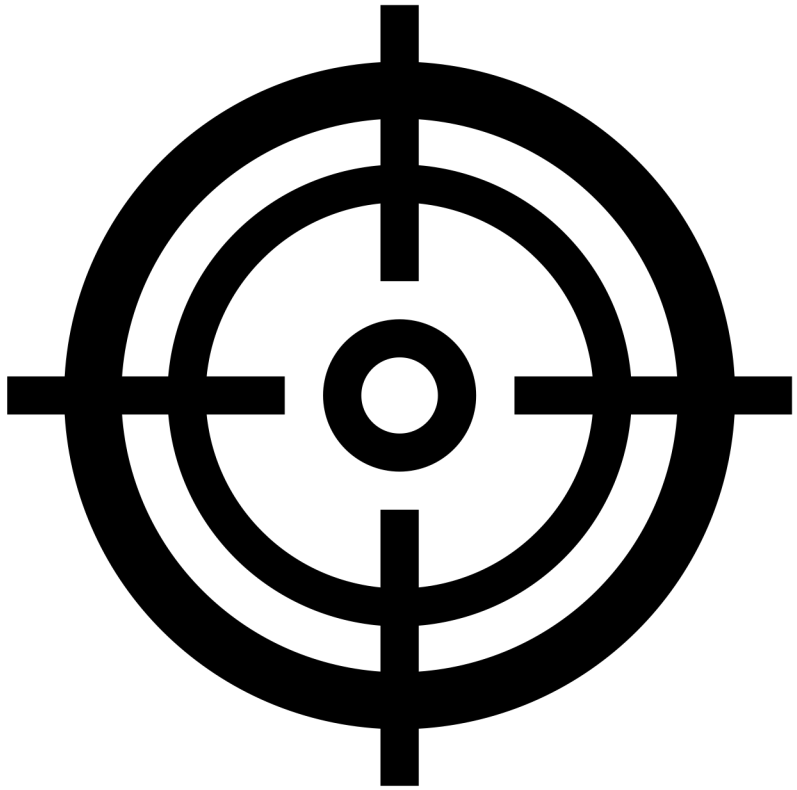
Klever Verification Framework



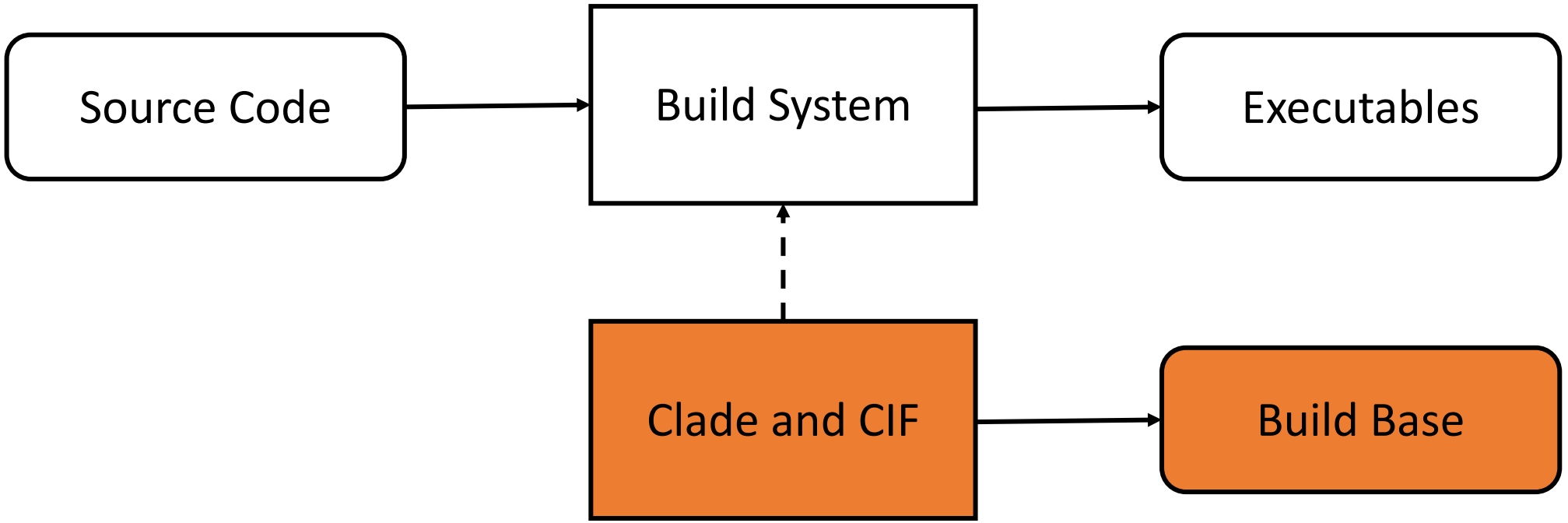
Source Code Preparation

Goals

- Limit scope to specific components, source code versions, architectures
- Prepare requirements
- Plan development stages
- Find bugs or prove correctness
- Deliver results or a tool



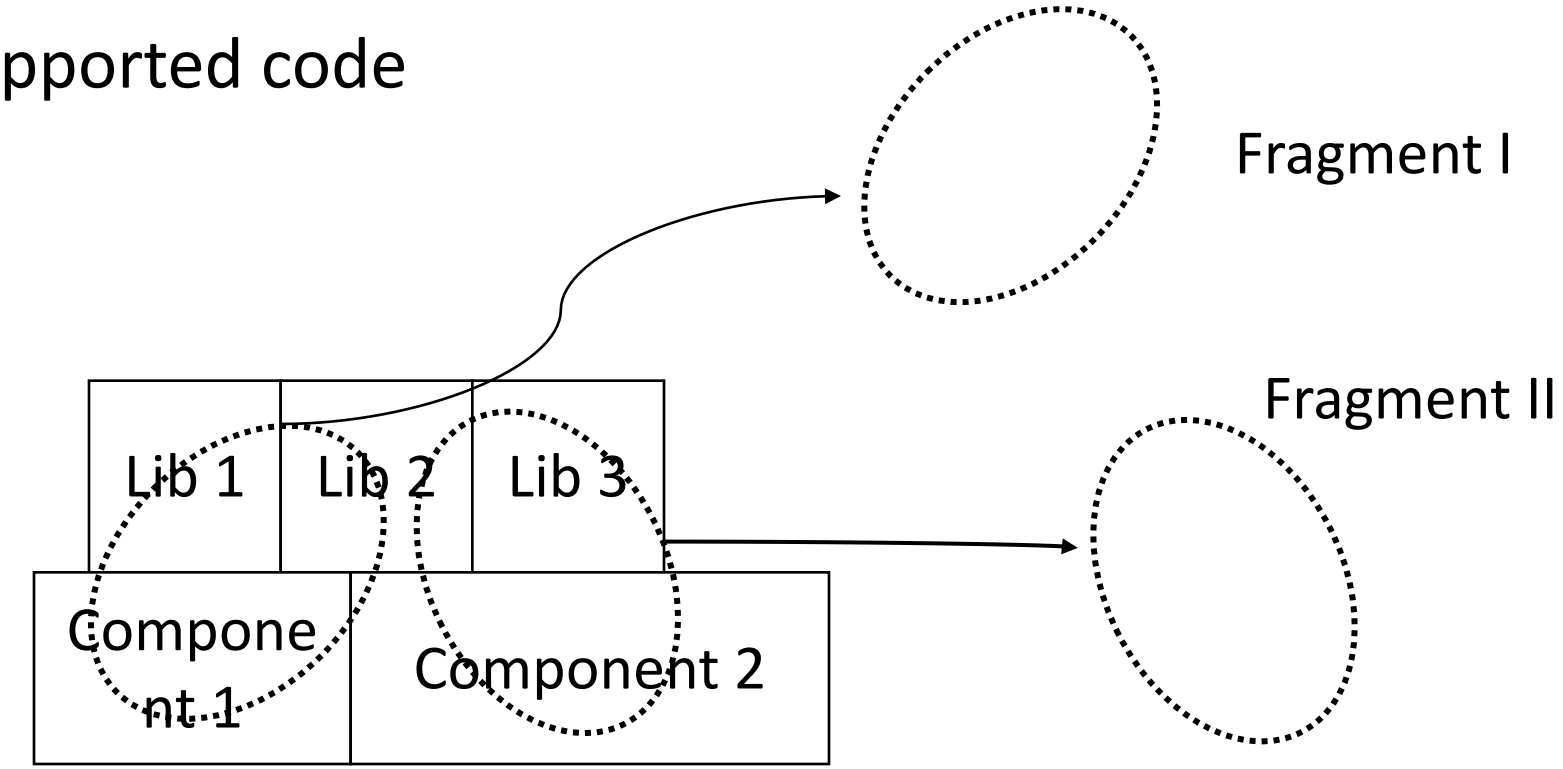
Prepare Build Base



Program Decomposition

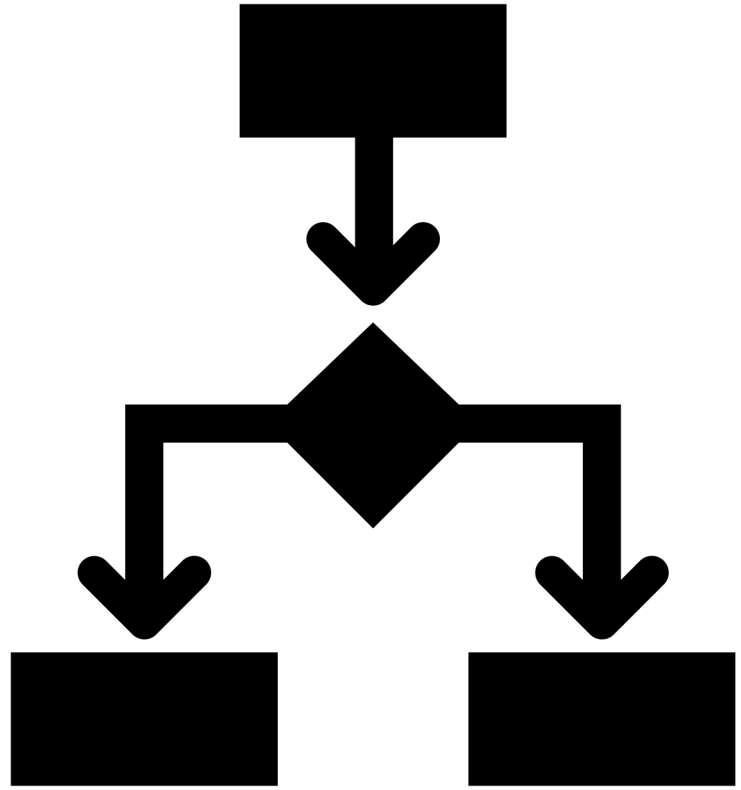
Program Decomposition

- Reduce environment modelling efforts
- Prevent timeouts
- Get rid of unsupported code



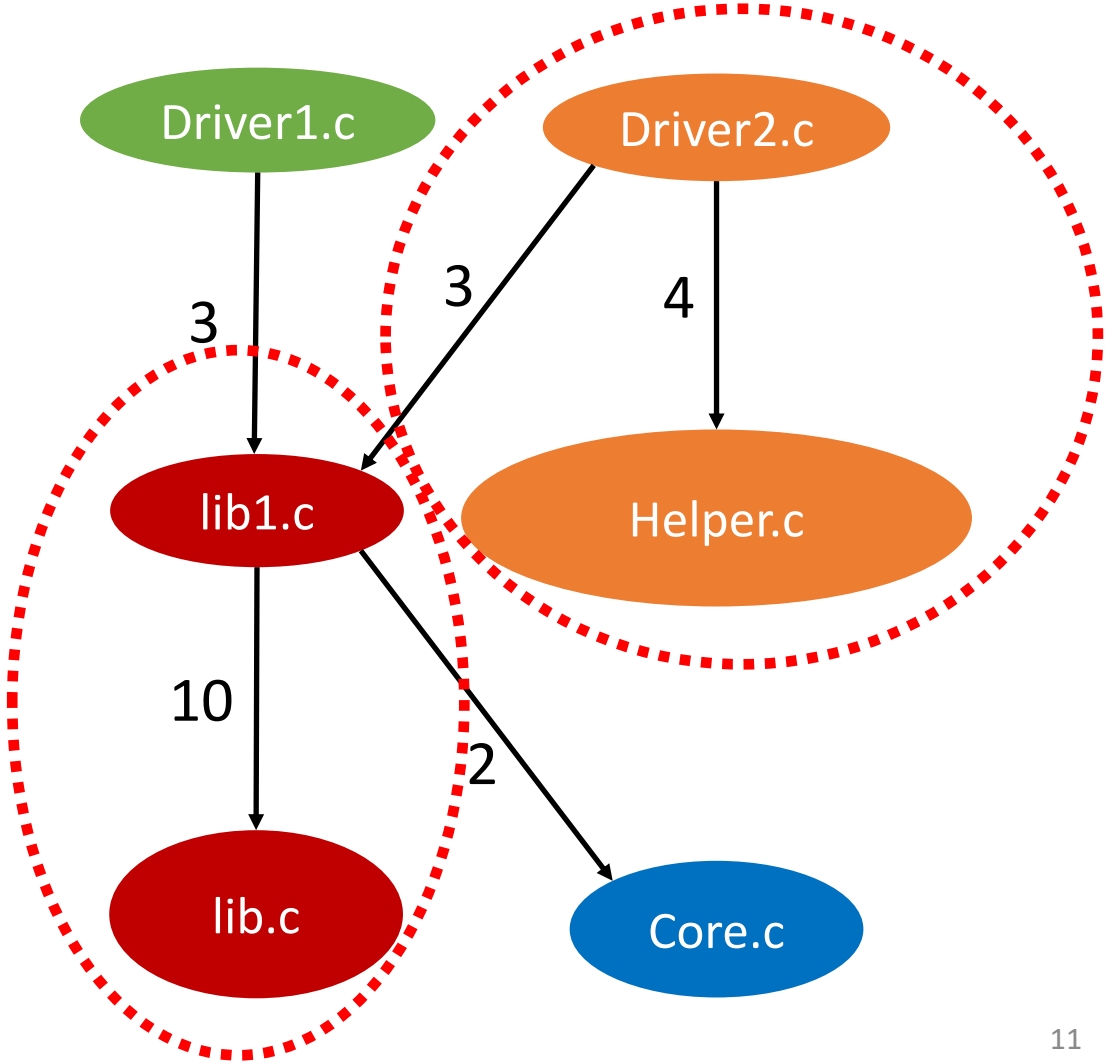
Extract Components as Fragments

1. Determine logical components
   - With unique interface
   - With common interface
2. Separate libraries
3. Remove auxiliary or irrelevant parts
   - Debug
   - Tests



Manual or Automatic Decomposition

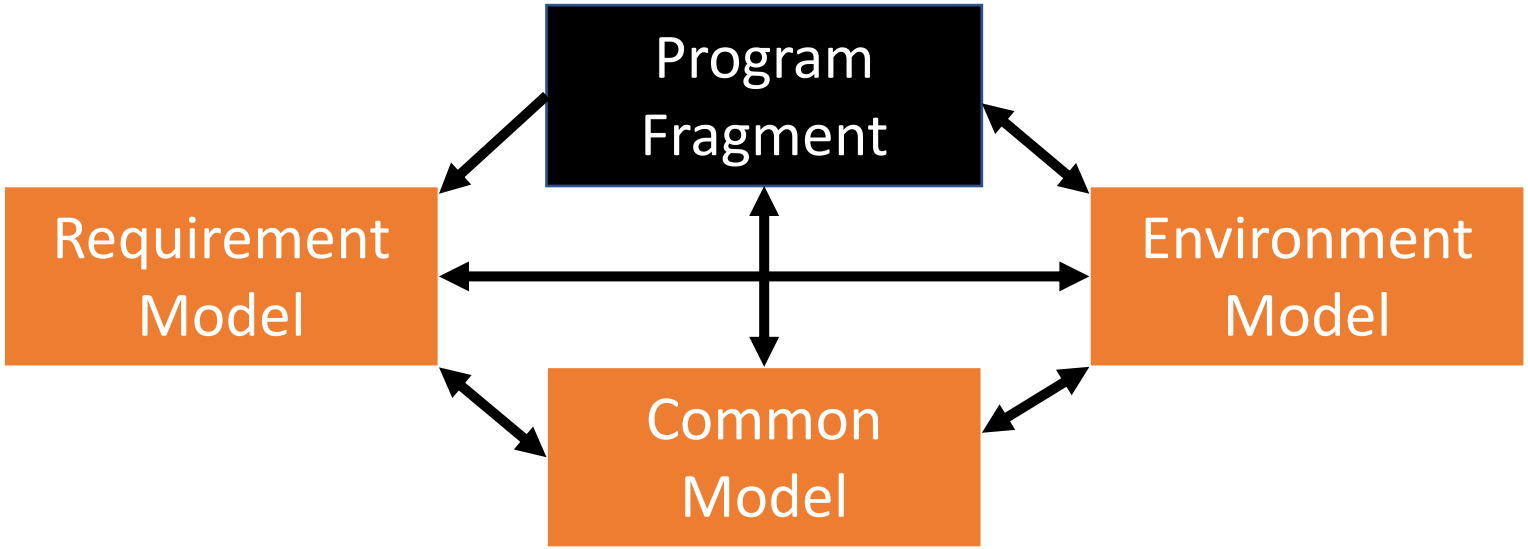
- Define fragments explicitly in advance
- Develop algorithms to decompose the program using its build base



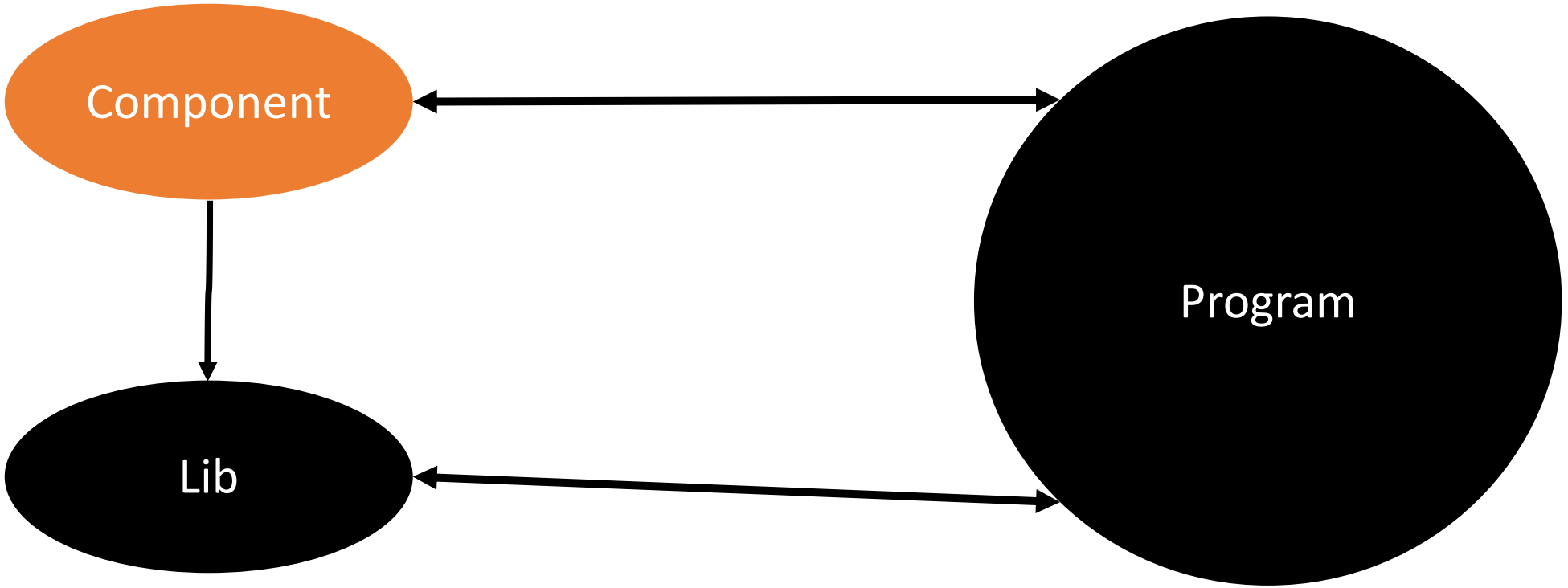
Provide Environment Models

Kinds of Models

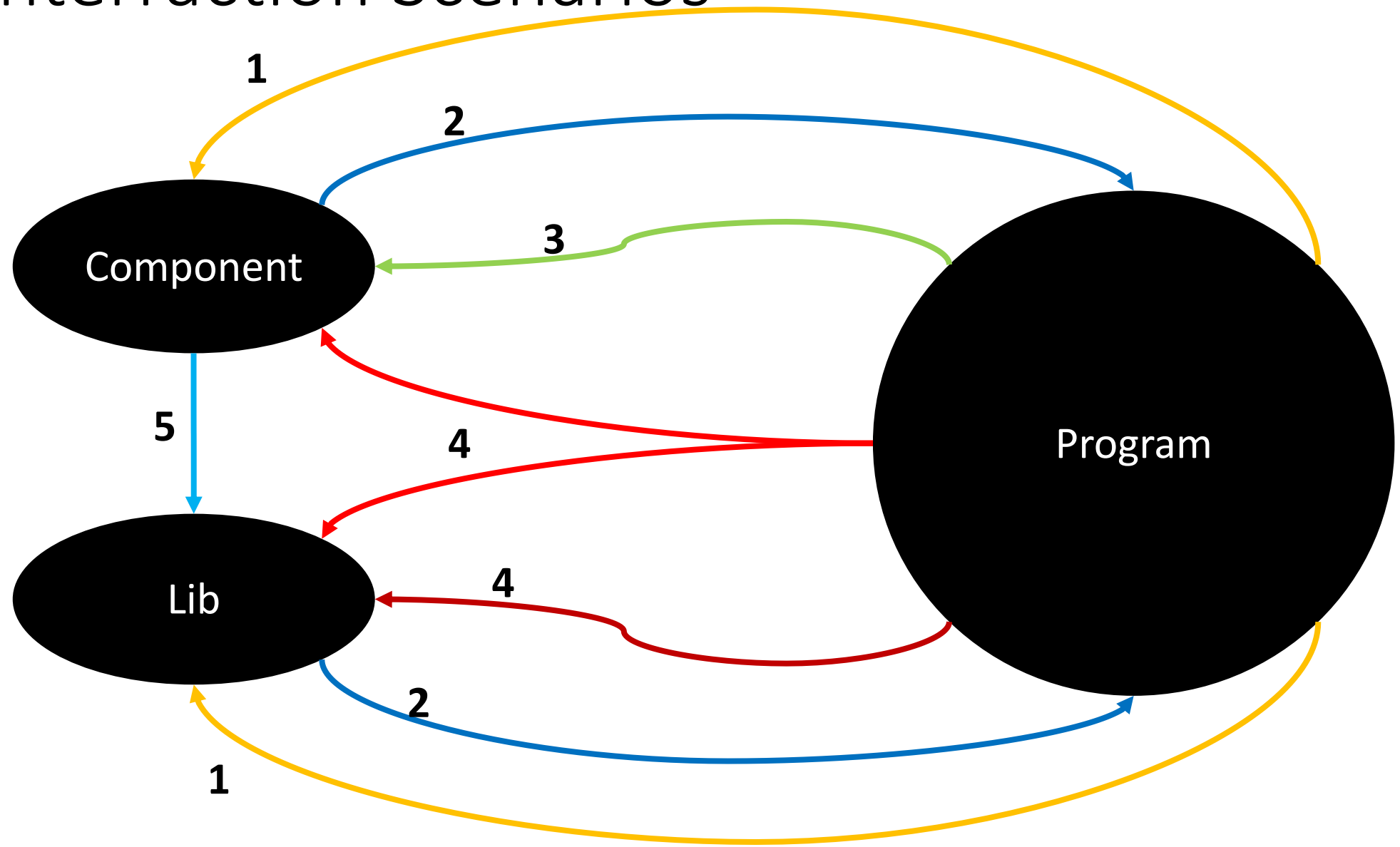
Kind of a Model	External Functions
Common Model	Undefined functions that init/uninit resources and influence the control flow
Requirement Model	Specific API that should be properly used by the fragment
Environment Model	Functions that call entry points and influence these calls



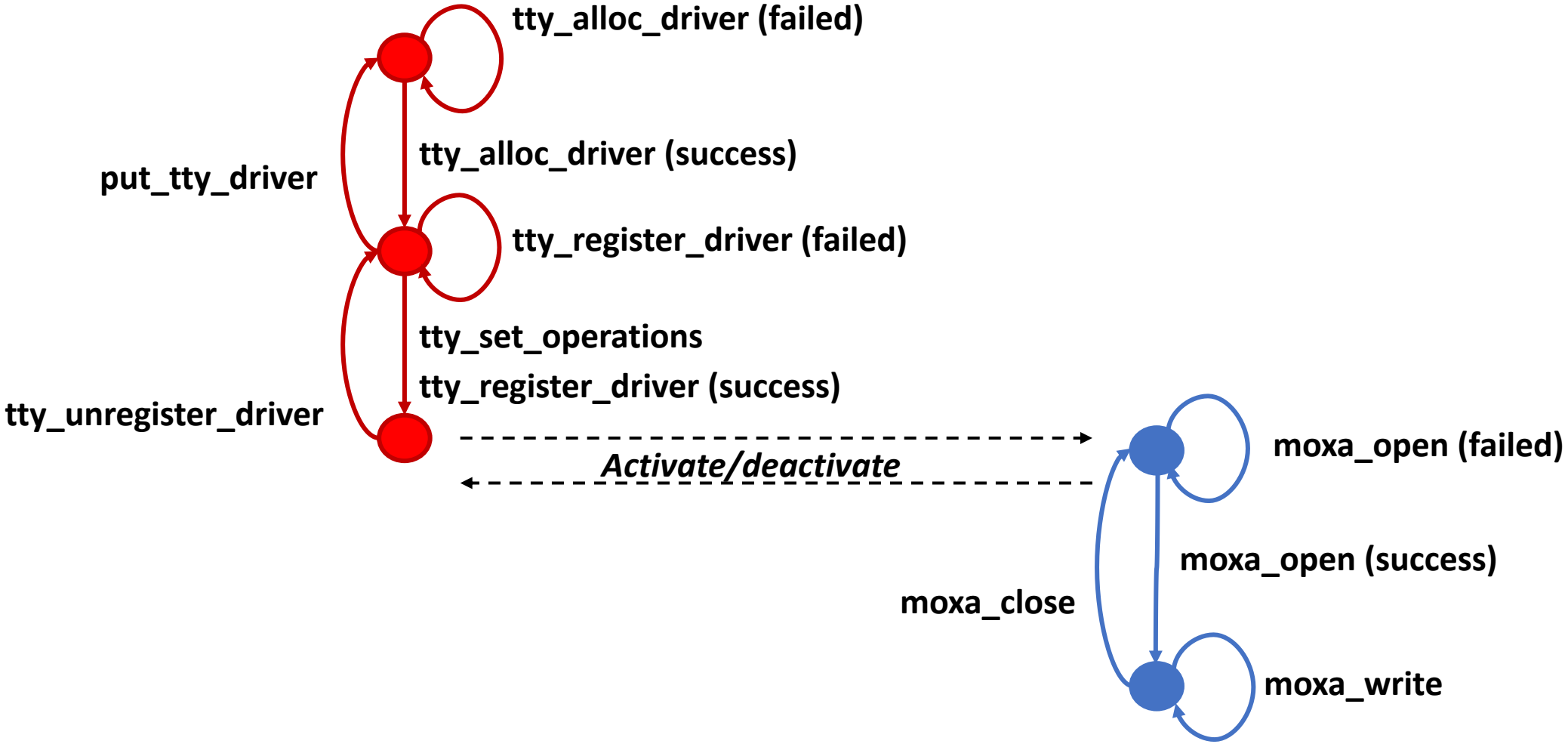
Environment Model



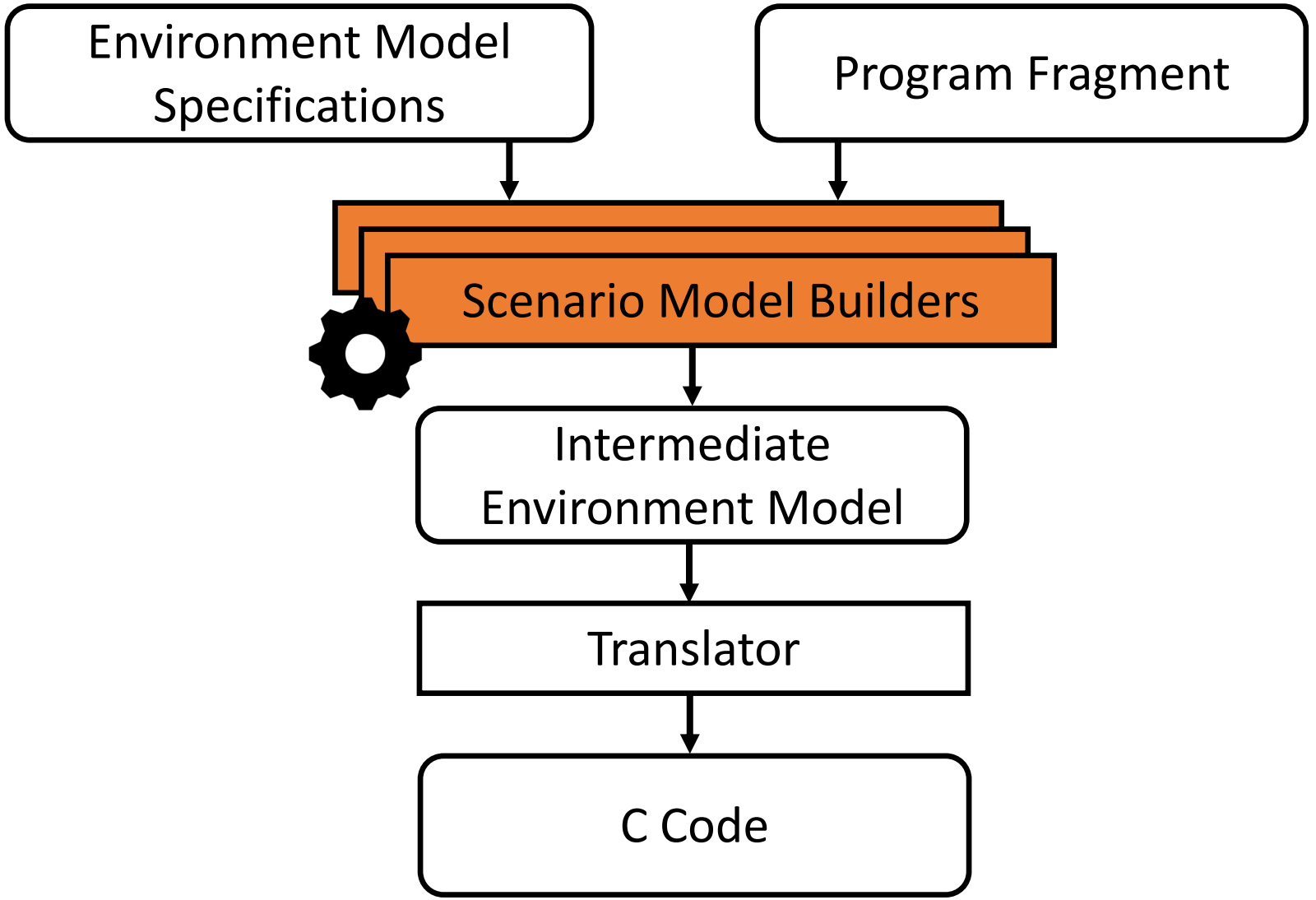
Interraction Scenarios



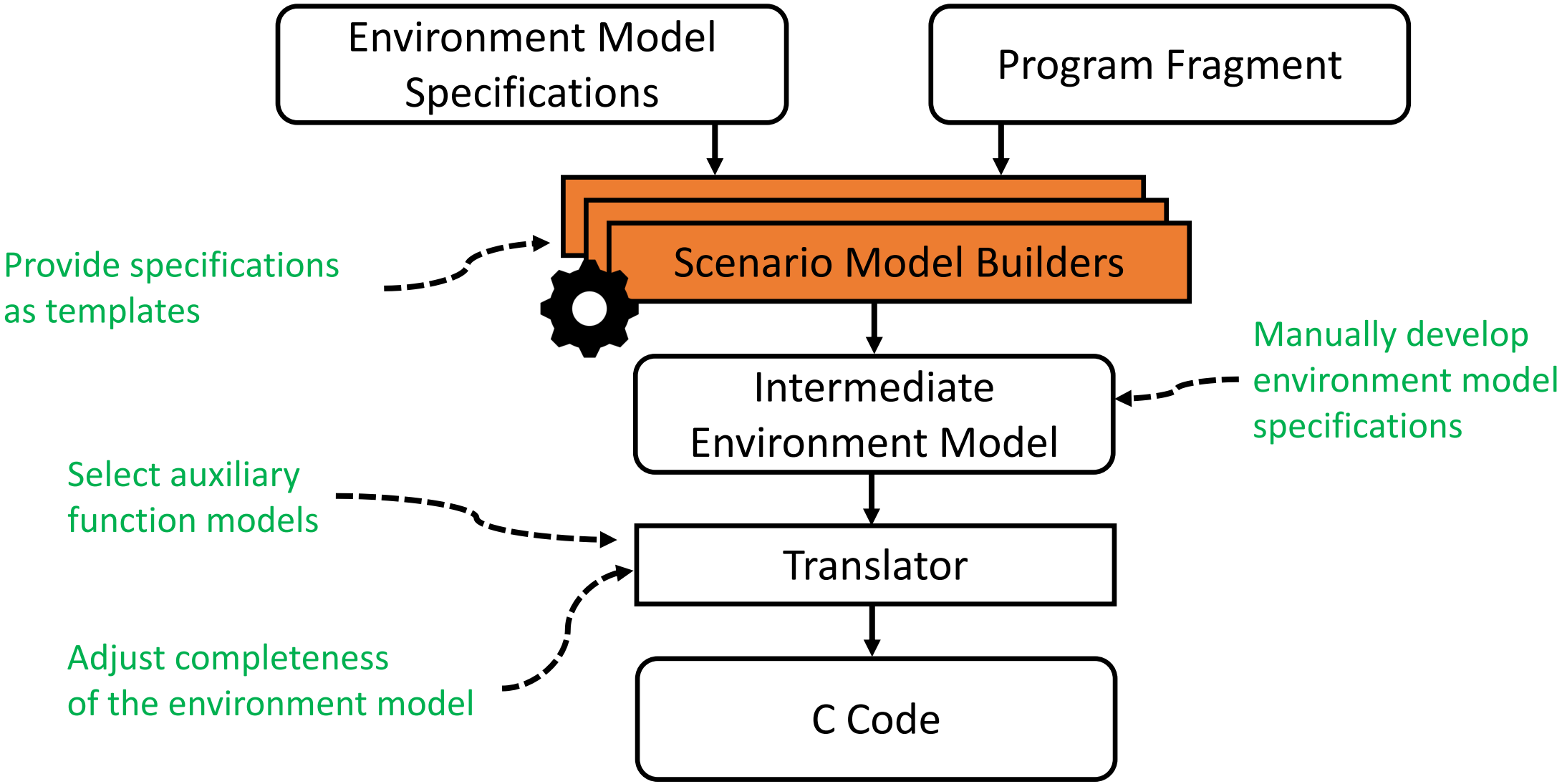
Intermediate Environment Model



Environment Model Generator



Environment Model Generator



Provide Requirement Specifications

Requirement Specification

```
int cnt = 0;

int try_module_get(struct module * m) {
    ret = ldv_random_neg_int();
    if (!ret)
        cnt++;
    return ret;
}

void module_put(struct module * m) {
    cnt--;
}

void ldv_check_final_state(void) {
    ldv_assert(cnt==0);
}
```

Requirement Specifications Development

1. Support an empty requirement to measure the coverage
2. Support memory safety and data race safety requirement specifications
3. Implement other requirement specifications
4. Develop tests for requirement models

Analyse Results

Verification Results

- Error traces (witnesses)
- Coverage reports
- Logs
- Resource consumption statistics

Use cases

Development

- Uncovered entry points
- Complicated code
- Classify fails

Refinement

- Marks and tags for errors and false positives
- Regression tests

Verification

- Find bugs
- Prepare final marks and tags

Evaluation

Manual Effort at Verification of Linux Device Drivers and Subsystems

Stage	Serial device drivers (20KLOC)	All device drivers(4MLOC)	Subsystems (1MLOC)	Total
Development of decomposition algorithms	0,25 man-months (100 LOC Python)	-	0,25 man-months (100 LOC Python)	0,5 man-months (200 LOC Python)
Development of environment model builders	3 man-months (3 KLOC Python)	-	0,5 man-months (500 LOC Python)	3,5 man-months (3,5 KLOC Python)
Development of environment model specifications	4,5 man-months (7 KLOC DSL)	5,5 man-months (10 KLOC DSL)	-	10 man-months (17 KLOC DSL)
Development of requirement specifications	6 man-months (550 LOC DSL)	9 man-months (950 LOC DSL)	0,25 man-months (200 LOC DSL)	15,25 man-months (1500 LOC DSL)
Total	13,75 man-months	14,5 man-months	1 man-month	29,25 man-months

BusyBox Applets Verification

Stage	Efforts
Development of decomposition algorithms	0,25 man-months (100 LOC Python)
Development of environment model builders	0 man-months -
Development of environment model specifications	0,25 man-months (200 LOC DSL)
Development of requirement specifications	0,5 man-months (300 LOC DSL)
Total	1 man-month

Why do you care

- Another point of view
- Train your verification tool
- Get new verification tasks

Thank You!

https://github.com/ldv-klever/klever - Mirror
https://forge.ispras.ru/projects/klever - Issue tracker
https://github.com/17451k/cif - CIF
https://github.com/17451k/clade - Clade
http://linuxtesting.org/kernel - Other links and verification projects

Build Base

- Various information about the program
  - Source code
  - Build command graph
  - File dependencies graph
  - Callgraph
  - …
- Easy to access
  - Movable archive with all sources and data
  - Python API to access the data

Summary

	Decomposition	Environment Modeling	Requirement Specifications Development	Analyzing results
Development	Manually prepared decomposition specification	Coarse model without restrictions	Empty requirement	Code coverage
	Decomposition algorithms	Scenario model builders	Requirement specifications, common model	Code coverage, marks, tags
Refinement	Algorithms to verify libraries	Environment model specifications, tests	Requirement specifications, common model, tests	Code coverage, marks, tags
Verification	-	-	Common model	Code coverage, marks, tags

Verification Time

Verification Job	2 physical cores	4 physical cores	30 * 4 physical cores
Serial device drivers (30KLOC)	5h	2.7h	0.5h
All device drivers (3MLOC)	600h	195h	11h