Benchmark Generation with VEVOS

A Coverage Analysis of Evolution Scenarios in Variant-Rich Systems

Paul Bittner

25.01.2023

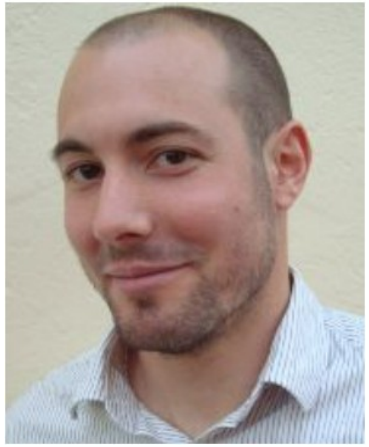
HUMBOLDT-UNIVERSITÄT ZU BERLIN

ulm university universität uulm

Alexander Schultheiß

Paul Bittner

Sandra Greiner



Timo Kehrer

UNIVERSITÄT BERN

UNIVERSITÄT BERN

Facing the Truth: Benchmarking the Techniques for the Evolution of Variant-Rich Systems

Daniel Strüber[1], Mukelabai Mukelabai[1], Jacob Krüger[2], Stefan Fischer[3], Lukas Linsbauer[3], Jabier Martinez[4], Thorsten Berger[1]
[1]Chalmers | University of Gothenburg, Sweden, [2]University of Magdeburg, Germany, [3]JKU Linz, Austria, [4]Tecnalia, Spain

ABSTRACT

The evolution of variant-rich systems is a challenging task. To support developers, the research community has proposed a range of different techniques over the last decades. However, many techniques have not been adopted in practice so far. To advance such techniques and to support their adoption, it is crucial to evaluate them against realistic baselines, ideally in the form of generally

1 INTRODUCTION

Evolving a variant-rich software system is a challenging task. Based on feature additions, bugfixes, and customizations, a variant-rich system evolves in two dimensions: (1) in its variability when new variants are added over time, and (2) in each individual variant, as variants are continuously modified. From these dimensions, various evolution scenarios arise. For example, variability may be

Scenario coverage of various benchmarks

Benchmark	Original Context	VS	VI	FIL	CE	FMS	AR	TR	FT	ANF	VZ	CPS

Scenario coverage of various benchmarks

Benchmark	Original Context	VS	VI	FIL	CE	FMS	AR	TR	FT	ANF	VZ	CPS
ArgoUML-SPL FLBench	Feature location	○	○	◓	○	○	○	○	○	○	○	○
Drupal	Bug detection	○	○	○	○	○	○	○	●	○	◓	○
Eclipse FLBench	Feature location	○	○	◓	◓	◓	○	○	○	○	○	○
LinuxKernel FLBench	Feature location	○	○	◓	◓	◓	○	○	○	○	○	○
Marlin & BCWallet	Feature location	○	○	◓	○	○	○	○	○	○	◓	○
ClaferWebTools	Traceability	○	○	◓	○	◓	○	○	○	○	◓	○
DoSC	Change discovery	○	◓	◓	○	◓	○	◓	○	○	◓	○
SystemsSwVarModels	FM synthesis	○	◓	○	●	●	○	○	○	○	◓	○
TraceLab CoEST	Traceability	○	○	◓	○	○	○	○	○	○	○	○
Variability bug database	Bug detection	○	○	◓	○	○	○	○	●	○	◓	○

Scenario coverage of various benchmarks

Benchmark	Original Context	VS	VI	FIL	CE	FMS	AR	TR	FT	ANF	VZ	CPS
ArgoUML-SPL FLBench	Feature location	○	○	◓	○	○	○	○	○	○	○	○
Drupal	Bug detection	○	○	○	○	○	○	○	●	○	◓	○
Eclipse FLBench	Feature location	○	○	◓	◓	◓	○	○	○	○	○	○
LinuxKernel FLBench	Feature location	○	○	◓	◓	◓	○	○	○	○	○	○
Marlin & BCWallet	Feature location	○	○	◓	○	○	○	○	○	○	◓	○
ClaferWebTools	Traceability	○	○	◓	○	◓	○	○	○	○	◓	○
DoSC	Change discovery	○	◓	◓	○	◓	○	◓	○	○	◓	○
SystemsSwVarModels	FM synthesis	○	◓	○	●	●	○	○	○	○	◓	○
TraceLab CoEST	Traceability	○	○	◓	○	○	○	○	○	○	○	○
Variability bug database	Bug detection	○	○	◓	○	○	○	○	●	○	◓	○

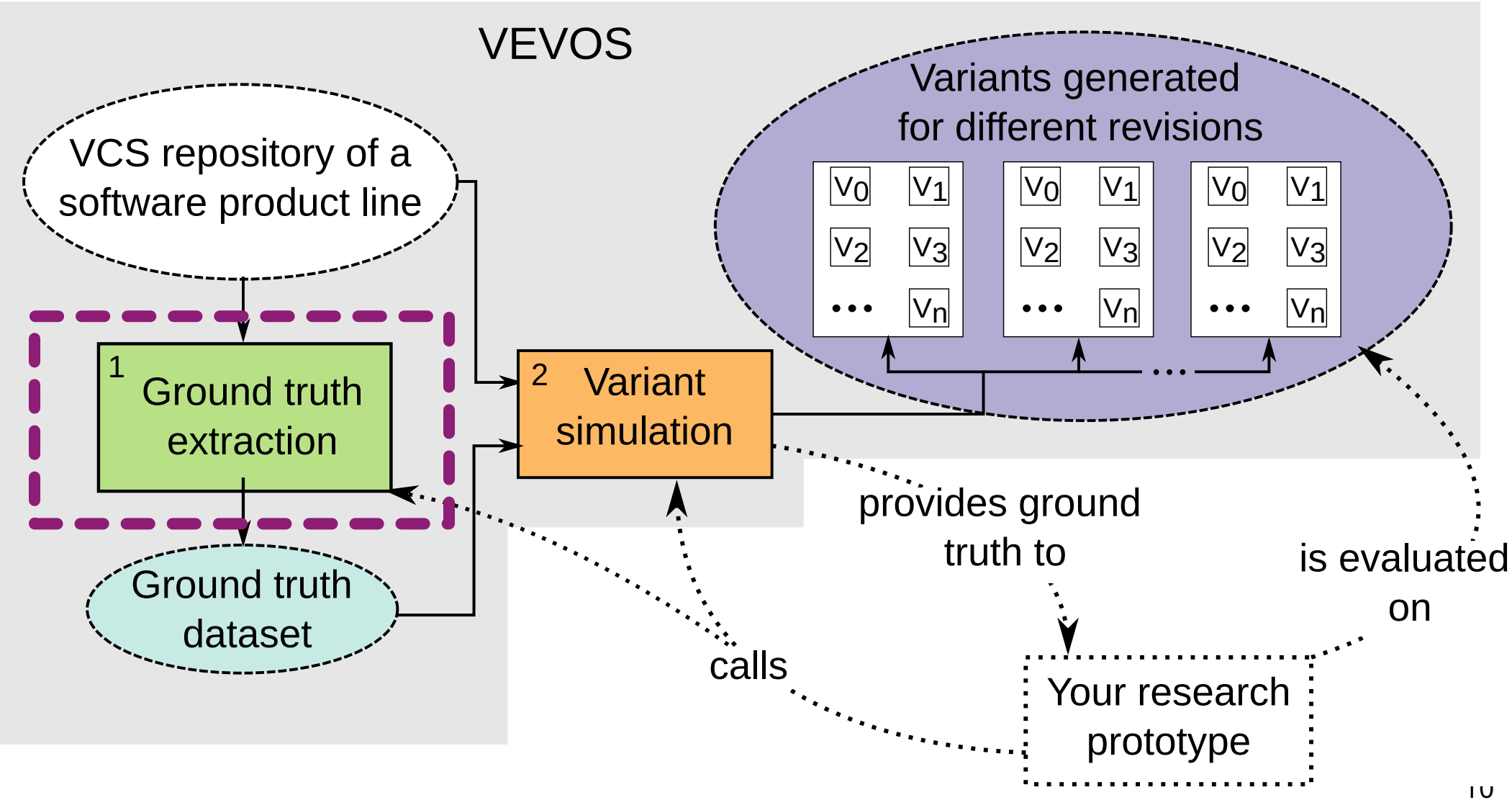
Scenario coverage of various benchmarks

Benchmark	Original Context	VS	VI	FIL	CE	FMS	AR	TR	FT	ANF	VZ	CPS
ArgoUML-SPL FLBench	Feature location	○	○	◓	○	○	○	○	○	○	○	○
Drupal	Bug detection	○	○	○	○	○	○	○	●	○	◓	○
Eclipse FLBench	Feature location	○	○	◓	◓	◓	○	○	○	○	○	○
LinuxKernel FLBench	Feature location	○	○	◓	◓	◓	○	○	○	○	○	○
Marlin & BCWallet	Feature location	○	○	◓	○	○	○	○	○	○	◓	○
ClaferWebTools	Traceability	○	○	◓	○	◓	○	○	○	○	◓	○
DoSC	Change discovery	○	◓	◓	○	◓	○	◓	○	○	◓	○
SystemsSwVarModels	FM synthesis	○	◓	○	●	●	○	○	○	○	◓	○
TraceLab CoEST	Traceability	○	○	◓	○	○	○	○	○	○	○	○
Variability bug database	Bug detection	○	○	◓	○	○	○	○	●	○	◓	○

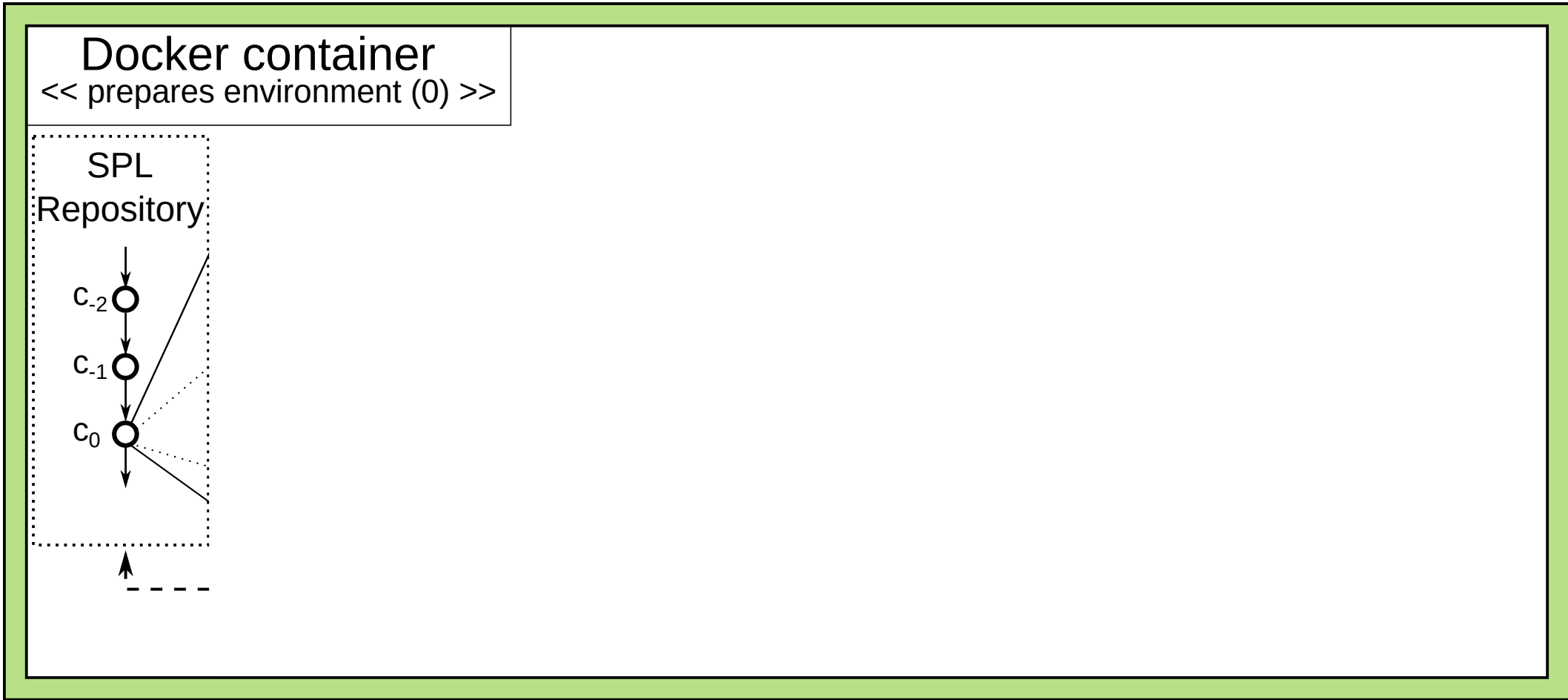
Scenario coverage of various benchmarks

Benchmark	Original Context	VS	VI	FIL	CE	FMS	AR	TR	FT	ANF	VZ	CPS
ArgoUML-SPL FLBench	Feature location	○	○	◓	○	○	○	○	○	○	○	○
Drupal	Bug detection	○	○	○	○	○	○	○	●	○	◓	○
Eclipse FLBench	Feature location	○	○	◓	◓	◓	○	○	○	○	○	○
LinuxKernel FLBench	Feature location	○	○	◓	◓	◓	○	○	○	○	○	○
Marlin & BCWallet	Feature location	○	○	◓	○	○	○	○	○	○	◓	○
ClaferWebTools	Traceability	○	○	◓	○	◓	○	○	○	○	◓	○
DoSC	Change discovery	○	◓	◓	○	◓	○	◓	○	○	◓	○
SystemsSwVarModels	FM synthesis	○	◓	○	●	●	○	○	○	○	◓	○
TraceLab CoEST	Traceability	○	○	◓	○	○	○	○	○	○	○	○
Variability bug database	Bug detection	○	○	◓	○	○	○	○	●	○	◓	○
VEVOS	?	?	?	?	?	?	?	?	?	?	?	?

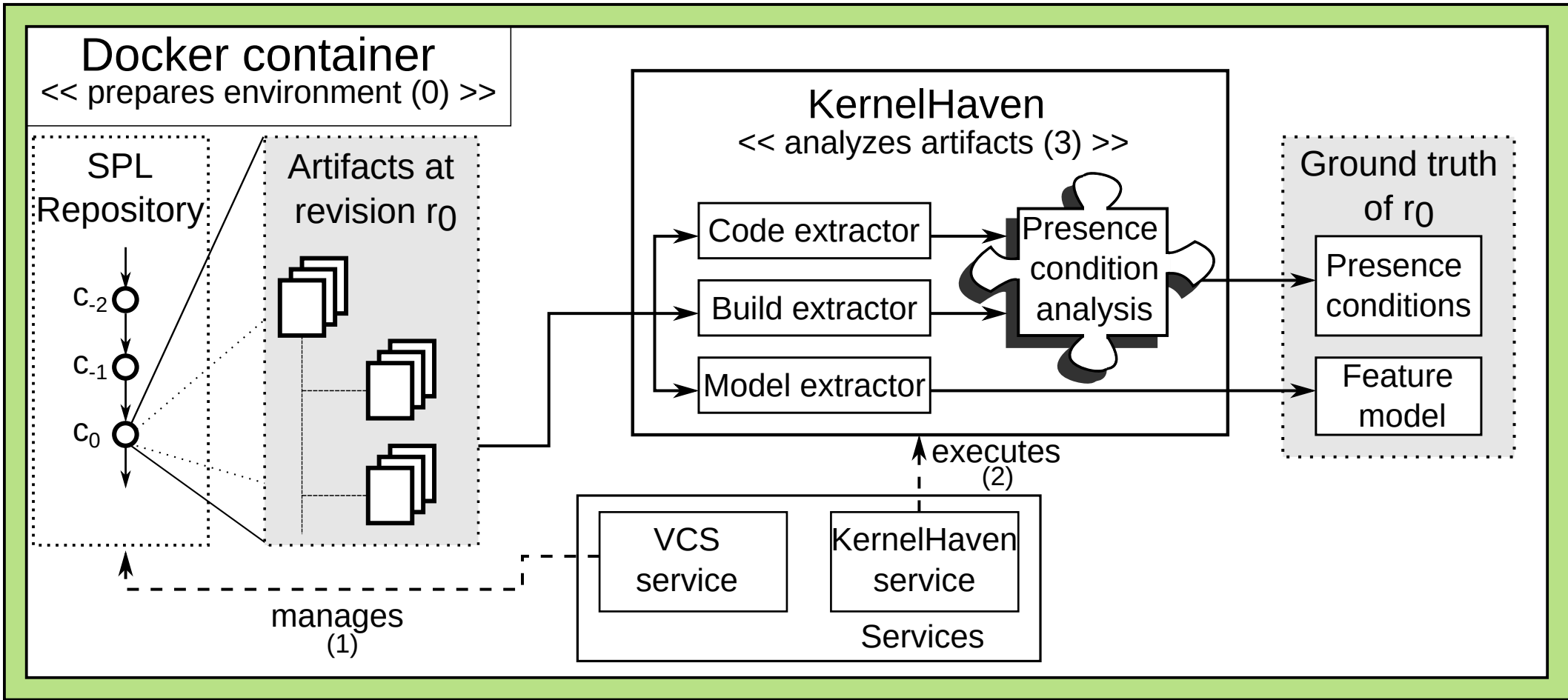
Variant **Evo**lution **S**imulation Framework (VEVOS)

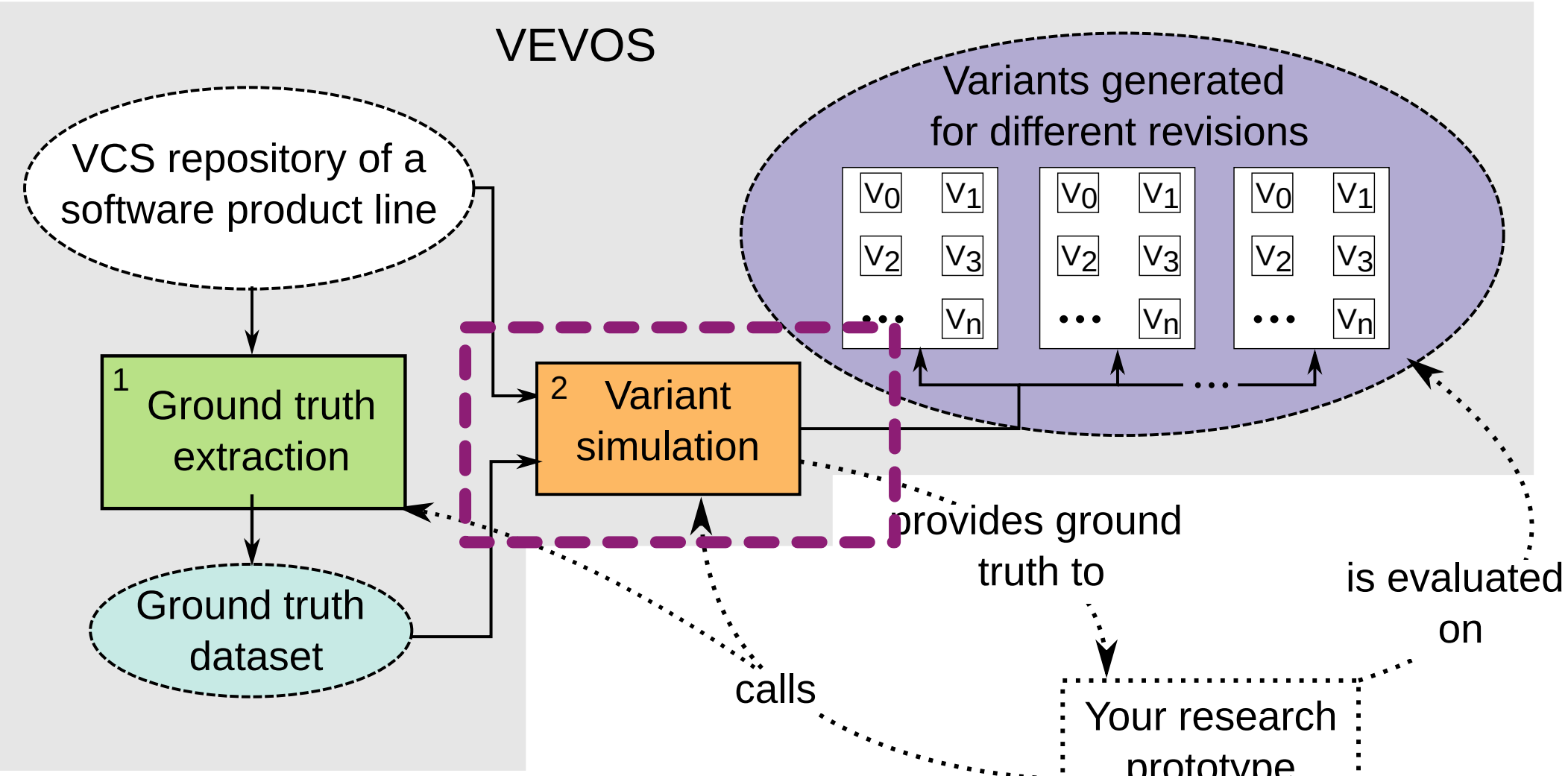

VEVOS
VCS repository of a software product line
1 Ground truth extraction
Ground truth dataset
2 Variant simulation
Variants generated for different revisions
V0 V1 V2 V3 ... Vn
V0 V1 V2 V3 ... Vn
V0 V1 V2 V3 ... Vn
...
provides ground truth to
is evaluated on
calls
Your research prototype

Overview of the ground truth extraction



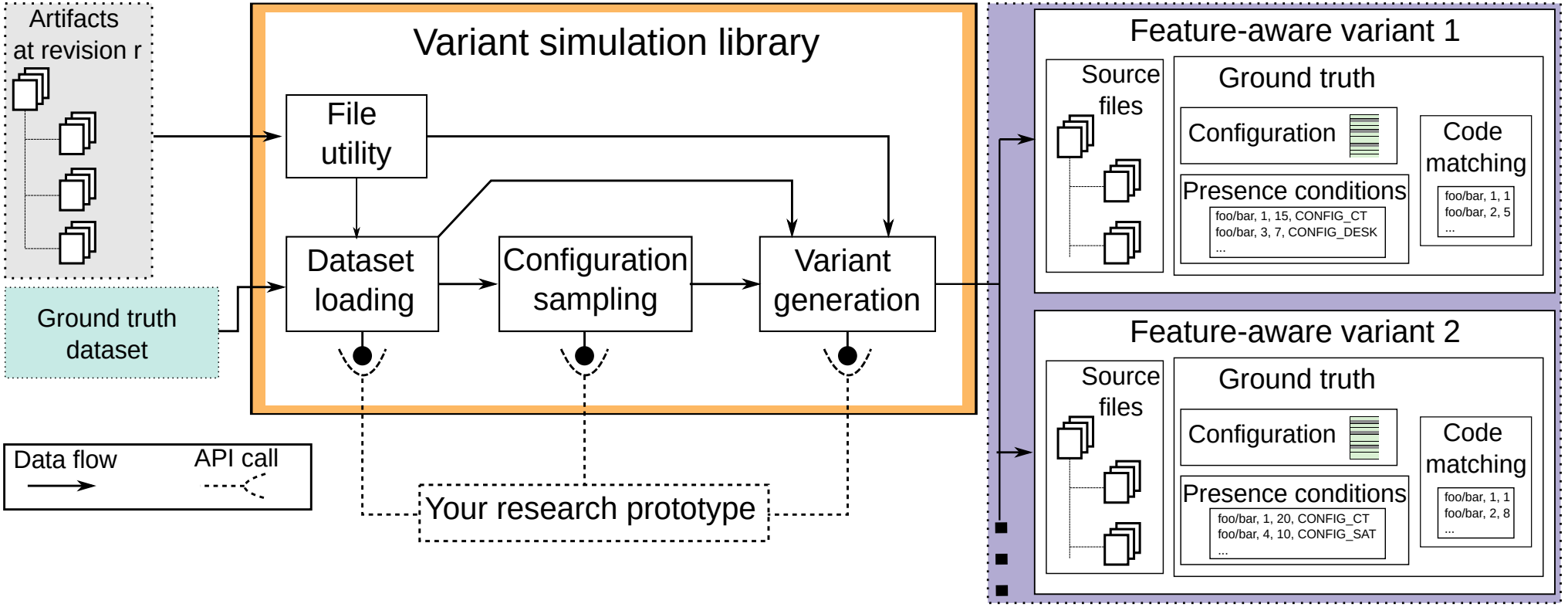
Overview of the ground truth extraction



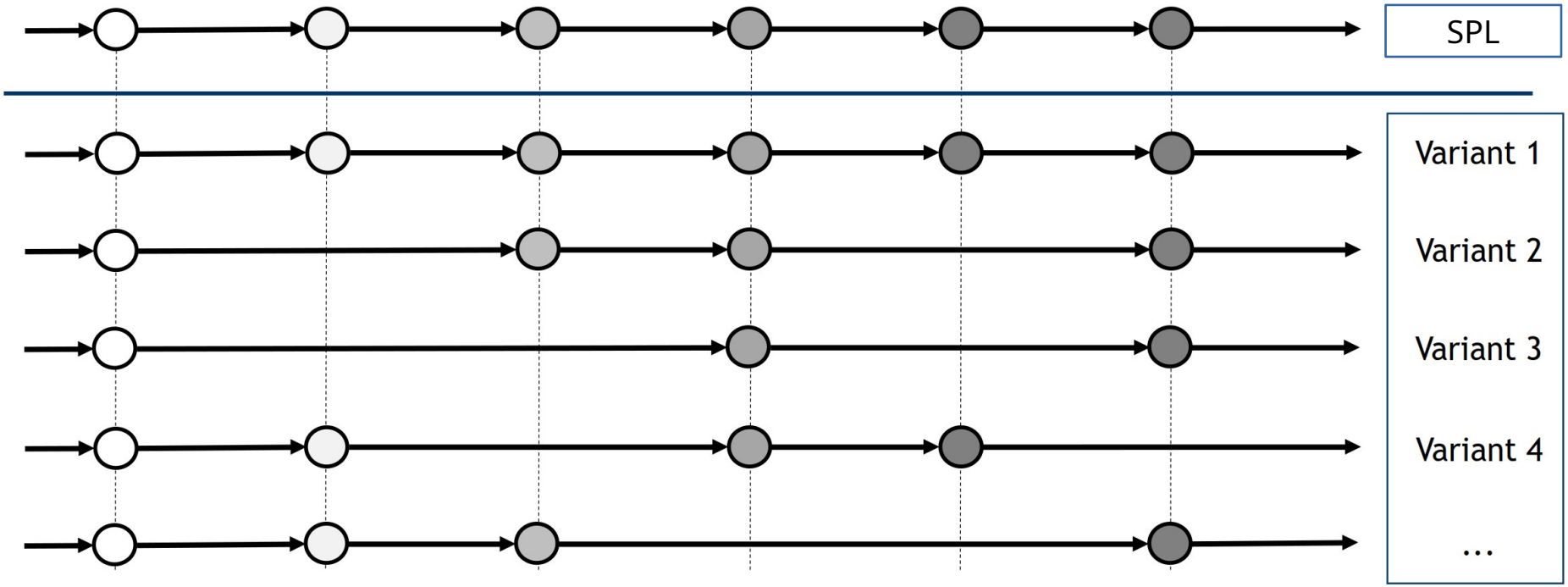

VEVOS
VCS repository of a software product line
1 Ground truth extraction
Ground truth dataset
2 Variant simulation
Variants generated for different revisions
V0 V1 V2 V3 ... Vn
V0 V1 V2 V3 ... Vn
V0 V1 V2 V3 ... Vn
...
provides ground truth to
is evaluated on
calls
Your research prototype

Overview of the variant simulation

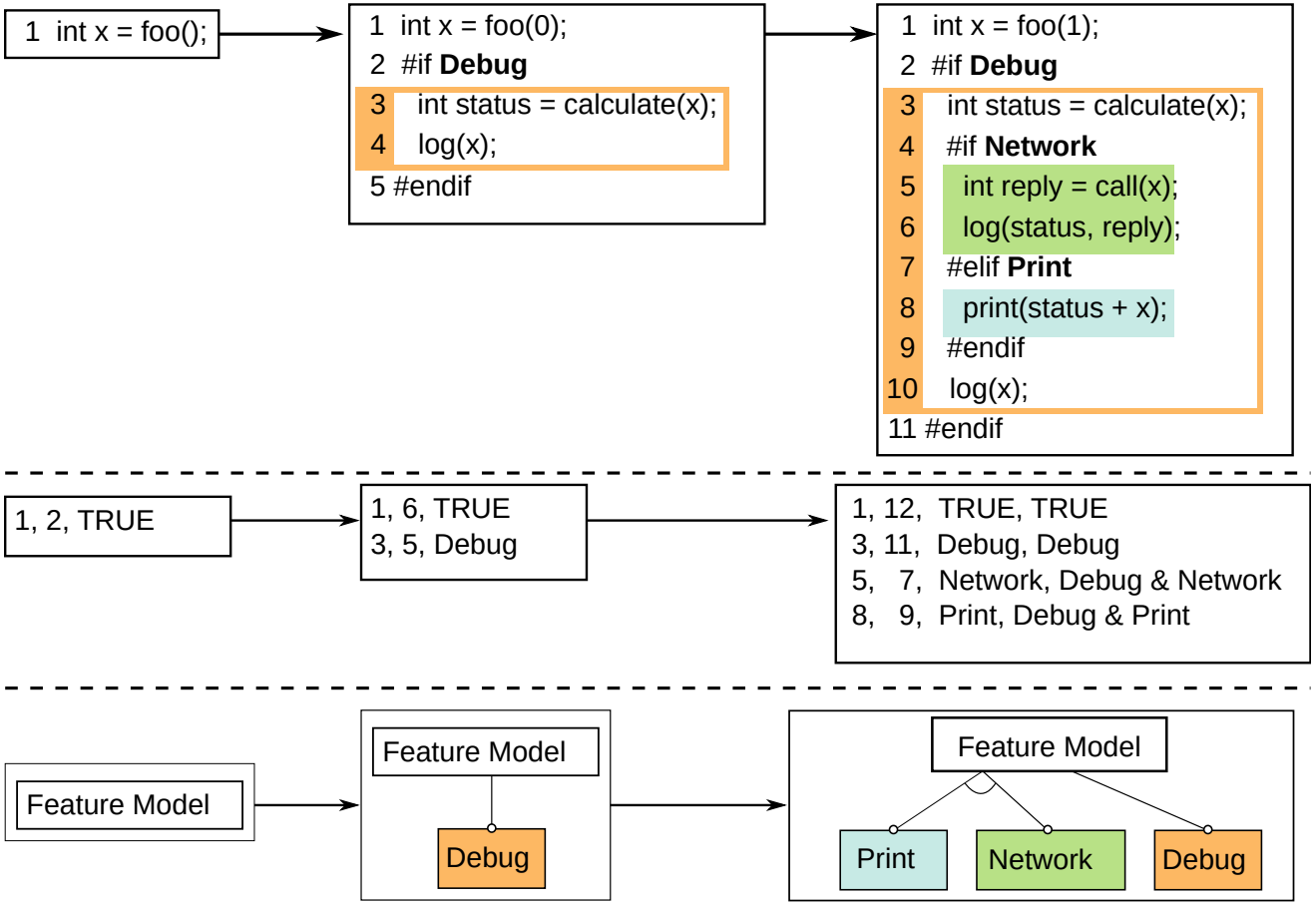
Overview of the variant simulation



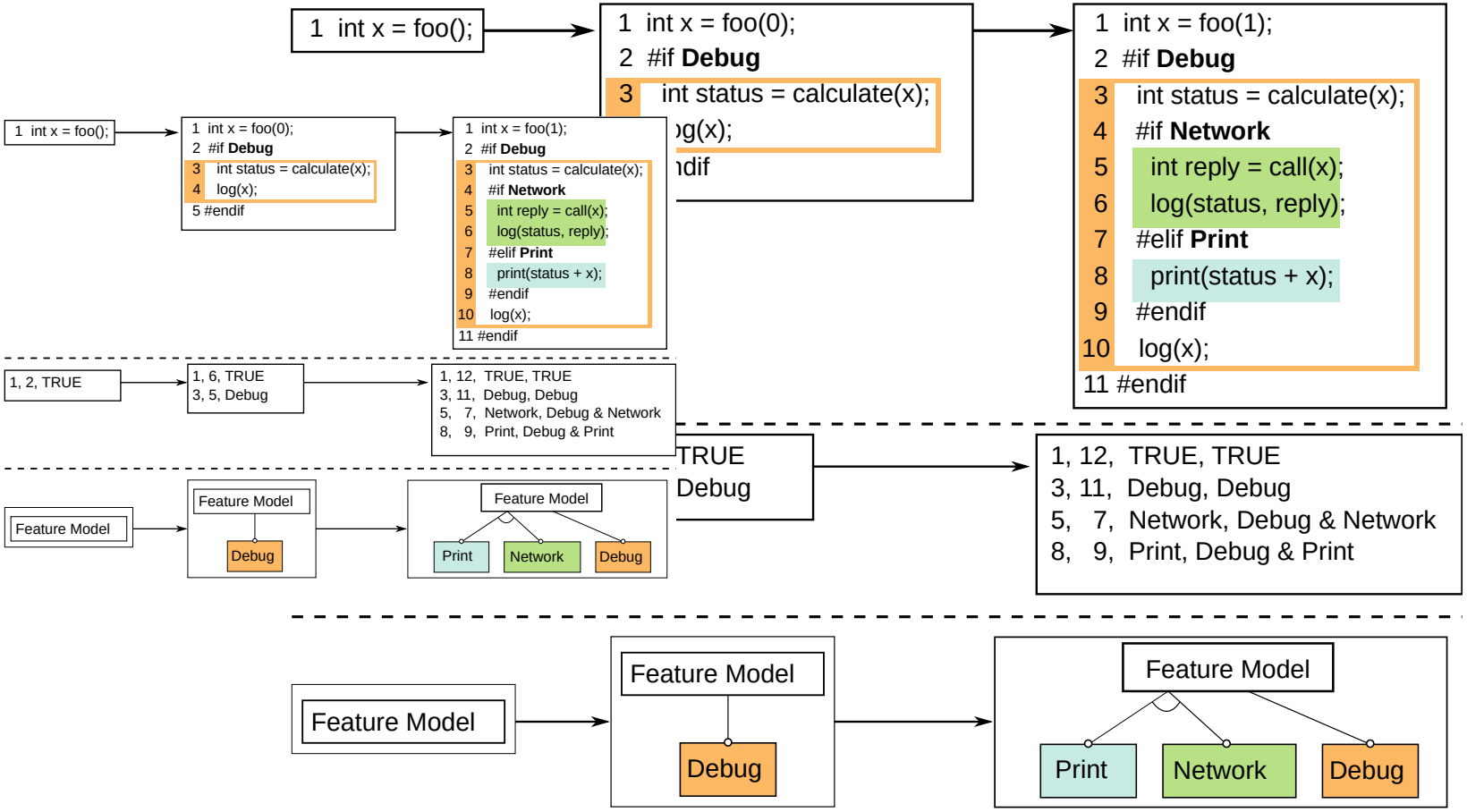
VEVOS simulates the evolution of variants

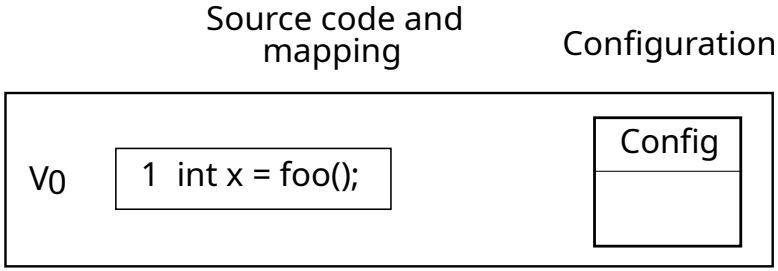


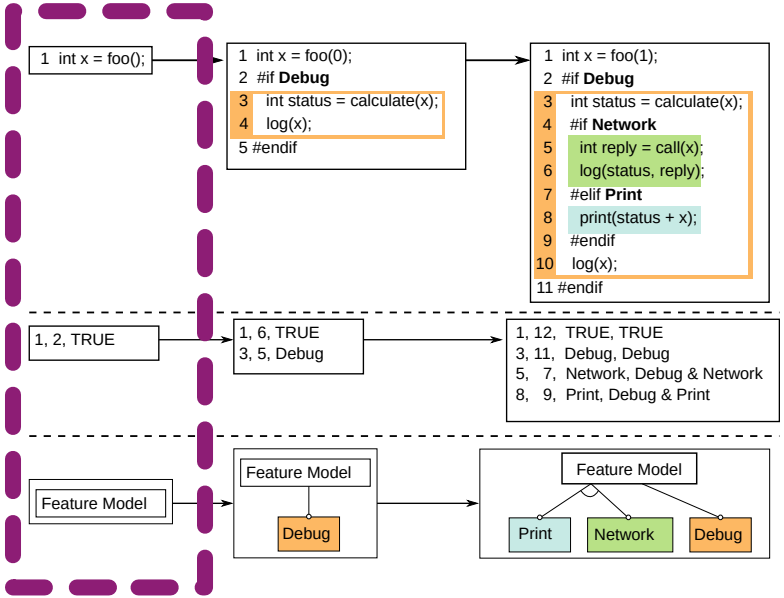
Example of an extracted ground truth

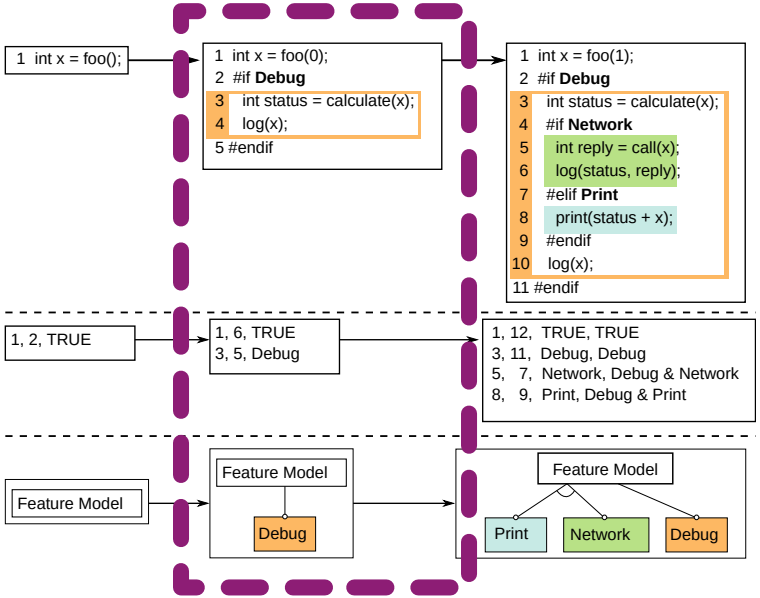


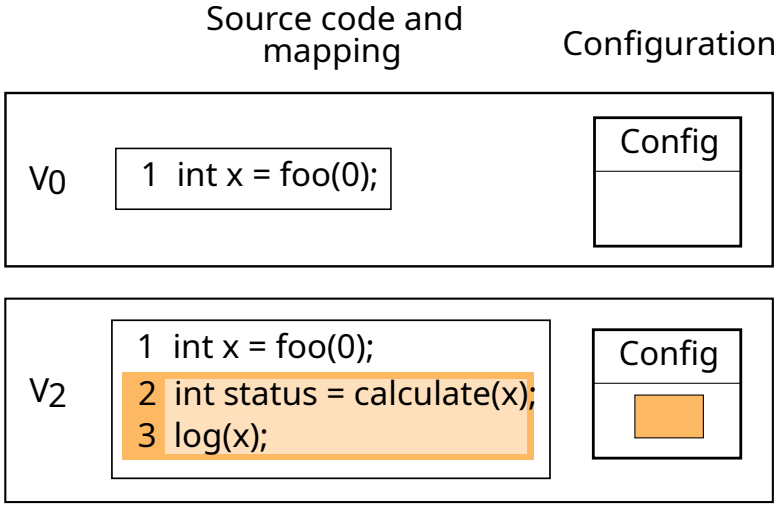
Example of an extracted ground truth

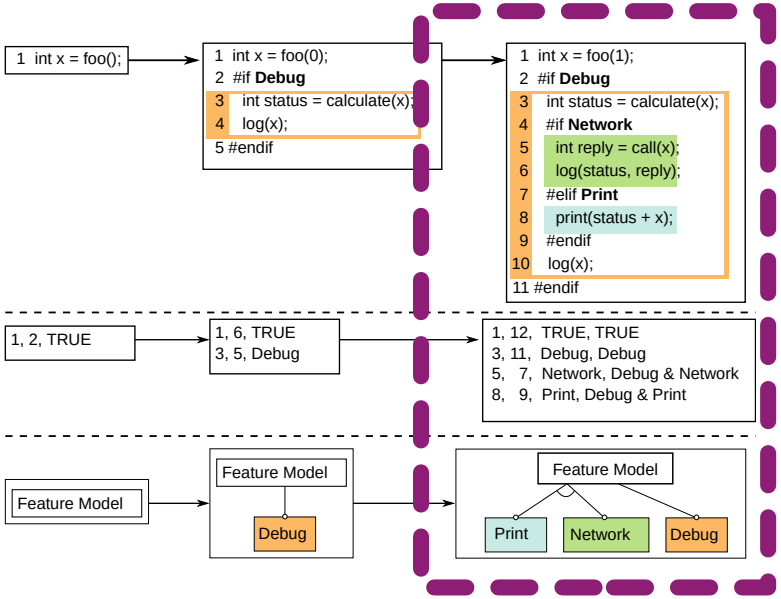



Source code and mapping
Configuration
V0
1 int x = foo();
Config


1 int x = foo();
1 int x = foo(0);
2 #if Debug
3 int status = calculate(x);
4 log(x);
5 #endif
1 int x = foo(1);
2 #if Debug
3 int status = calculate(x);
4 #if Network
5 int reply = call(x);
6 log(status, reply);
7 #elif Print
8 print(status + x);
9 #endif
10 log(x);
11 #endif
1, 2, TRUE
1, 6, TRUE
3, 5, Debug
1, 12, TRUE, TRUE
3, 11, Debug, Debug
5, 7, Network, Debug & Network
8, 9, Print, Debug & Print
Feature Model
Feature Model
Debug
Feature Model
Print
Network
Debug


1 int x = foo();
1 int x = foo(0);
2 #if Debug
3 int status = calculate(x);
4 log(x);
5 #endif
1 int x = foo(1);
2 #if Debug
3 int status = calculate(x);
4 #if Network
5 int reply = call(x);
6 log(status, reply);
7 #elif Print
8 print(status + x);
9 #endif
10 log(x);
11 #endif
1, 2, TRUE
1, 6, TRUE
3, 5, Debug
1, 12, TRUE, TRUE
3, 11, Debug, Debug
5, 7, Network, Debug & Network
8, 9, Print, Debug & Print
Feature Model
Feature Model
Debug
Feature Model
Print
Network
Debug


Source code and mapping
Configuration
V0
1 int x = foo(0);
Config
V2
1 int x = foo(0);
2 int status = calculate(x);
3 log(x);
Config



Source code and mapping

Configuration

V0

```
1 int x = foo(1);
```

Config

V2

```
1 int x = foo(1);
2 int status = calculate(x);
3 log(x);
```

Config

V1

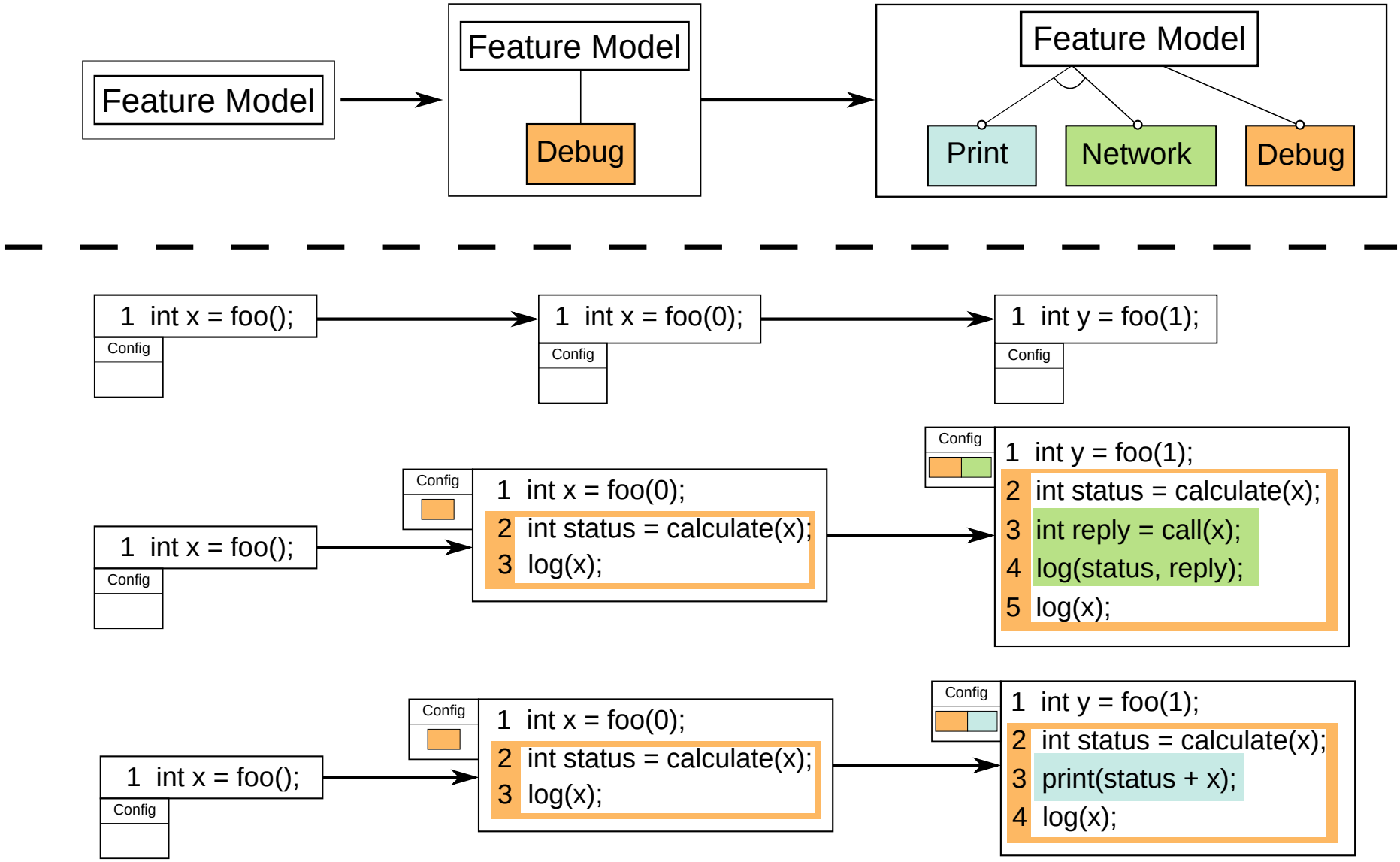
```
1 int x = foo(1);
2 int status = calculate(x);
3 int reply = call(x);
4 log(status, reply);
5 log(x);
```

Config

V2

```
1 int x = foo(1);
2 int status = calculate(x);
3 print(status + x);
4 log(x);
```

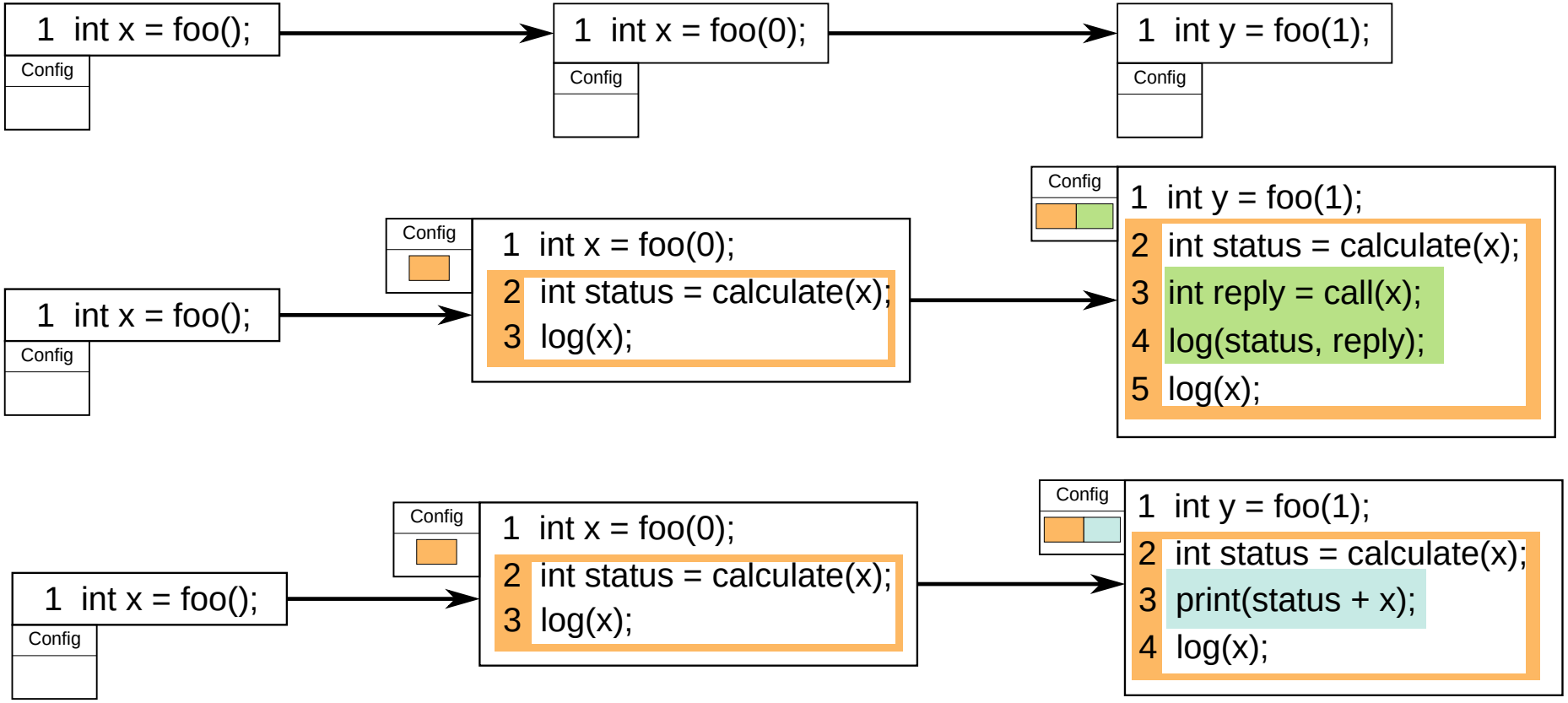
Config


Feature Model
Feature Model
Debug
Feature Model
Print
Network
Debug
1 int x = foo();
Config
1 int x = foo(0);
Config
1 int y = foo(1);
Config
1 int x = foo();
Config
Config
1 int x = foo(0);
2 int status = calculate(x);
3 log(x);
Config
1 int y = foo(1);
2 int status = calculate(x);
3 int reply = call(x);
4 log(status, reply);
5 log(x);
1 int x = foo();
Config
Config
1 int x = foo(0);
2 int status = calculate(x);
3 log(x);
Config
1 int y = foo(1);
2 int status = calculate(x);
3 print(status + x);
4 log(x);

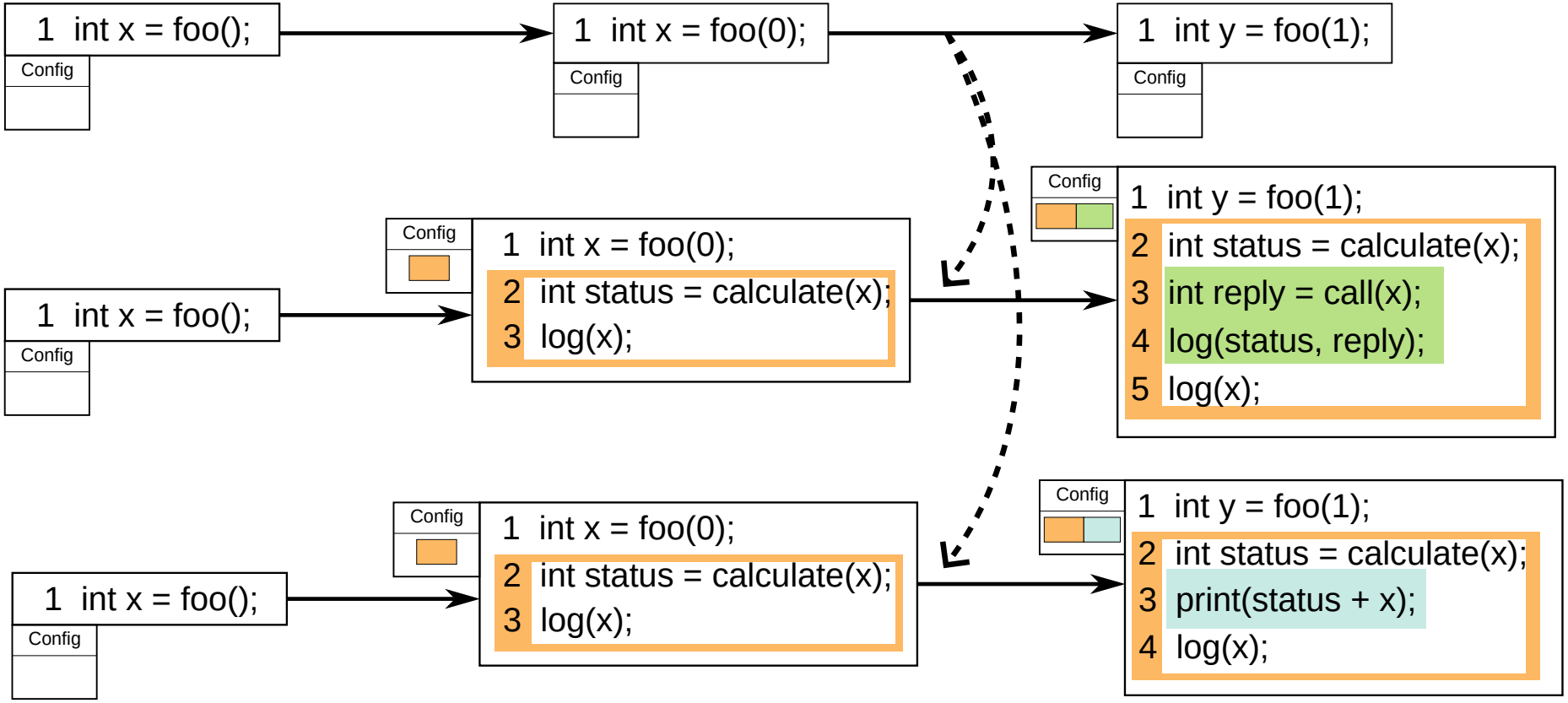
Coverage of evolution scenarios

Main scenario:
Variant synchronization

Variant synchronization (VS)

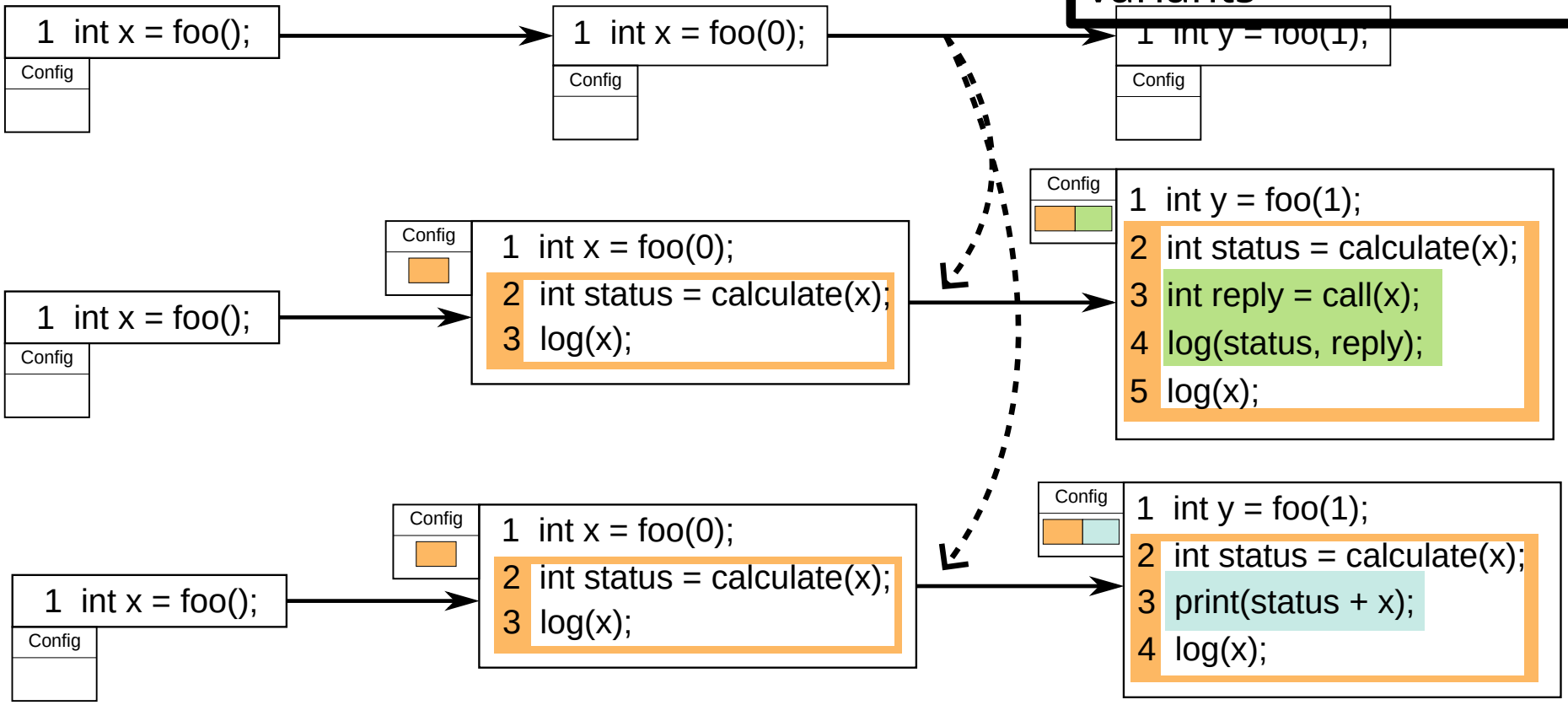


Variant synchronization (VS)



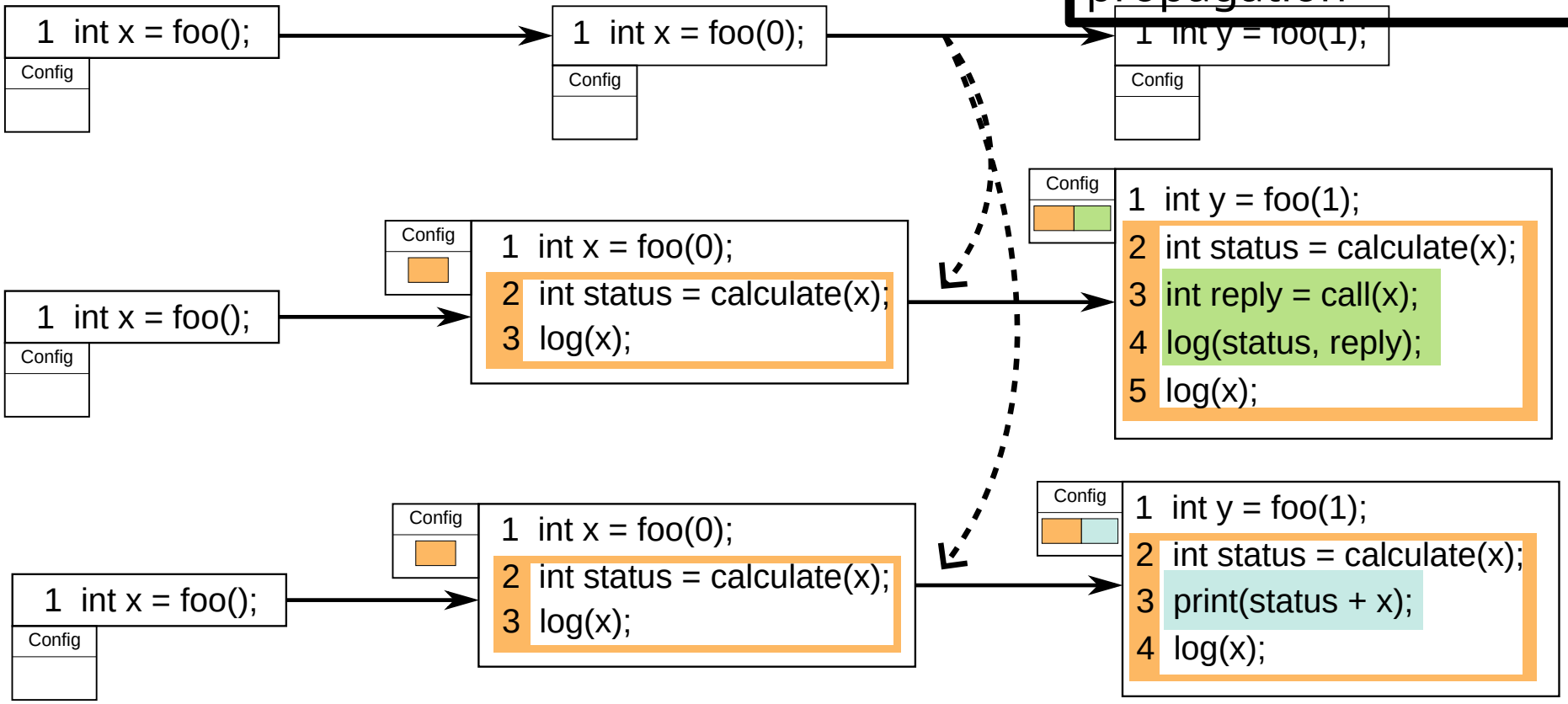
Variant synchronization (VS)

1. Requirement:
Code of at least two variants



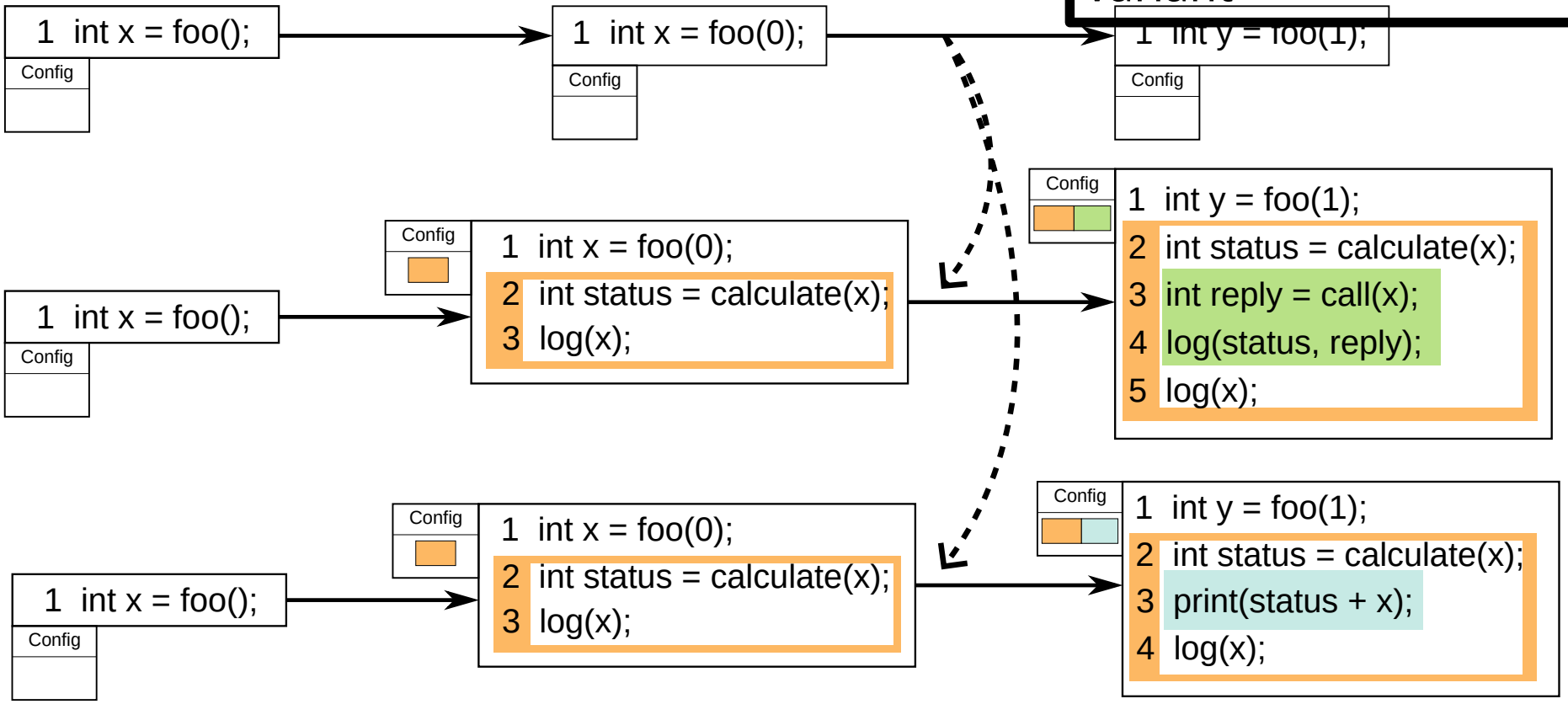
Variant synchronization (VS)

2. Requirement:
Correct code after propagation



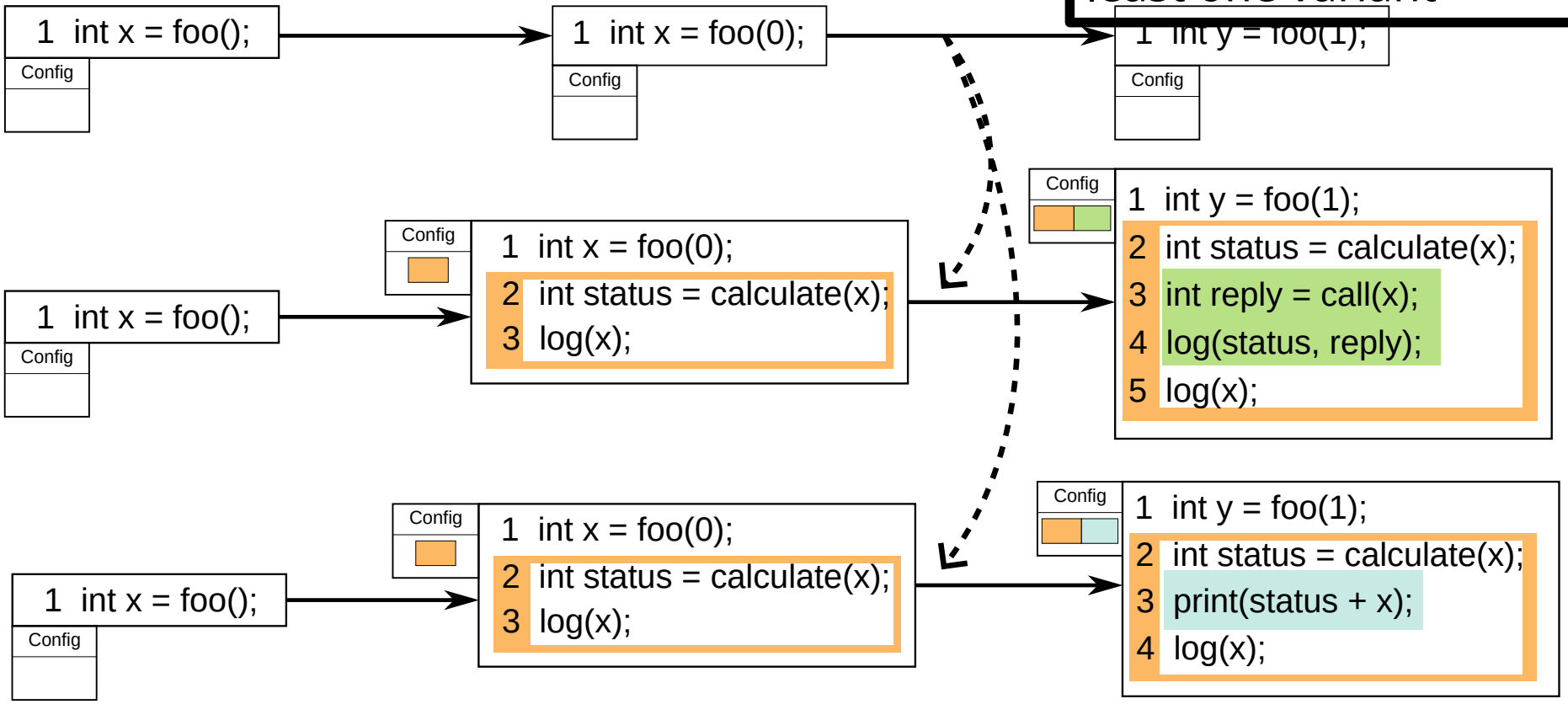
Variant synchronization (VS)

3. Requirement:
Changes of at least one variant



Variant synchronization (VS)

4. Requirement:
Feature locations of at least one variant



Coverage of other scenarios

Variant integration (VI)

2. Variant Integration (VI)

Source code and mapping | Configuration

V_0

```
1 int x = foo();
```

Config

V_2

```
1 int x = foo();
2 int status = calculate(x);
3 log(x);
```

Config

V_1

```
1 int x = foo();
2 int status = calculate(x);
3 int reply = call(x);
4 log(status, reply);
5 log(x);
```

Config

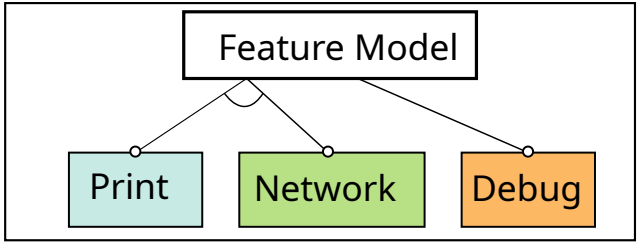
V_2

```
1 int x = foo();
2 int status = calculate(x);
3 print(status + x);
4 log(x);
```

Config

Integration

```
1  int x = foo(1);
2  #if  Debug
3    int status = calculate(x);
4    #if  Network
5      int reply = call(x);
6      log(status, reply);
7    #elif  Print
8      print(status + x);
9    #endif
10   log(x);
11 #endif
```



2. Variant Integration (VI)

1. Requirement:
Set of individual variants

Source code and mapping | Configuration

V0

```
1 int x = foo();
```

Config

V2

```
1 int x = foo();
2 int status = calculate(x);
3 log(x);
```

Config

V1

```
1 int x = foo();
2 int status = calculate(x);
3 int reply = call(x);
4 log(status, reply);
5 log(x);
```

Config

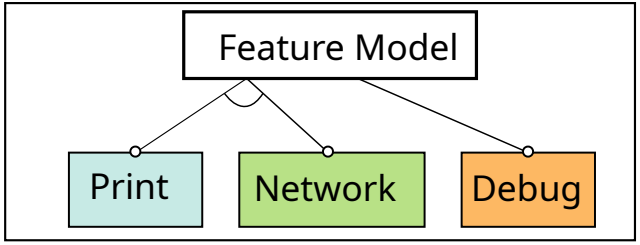
V2

```
1 int x = foo();
2 int status = calculate(x);
3 print(status + x);
4 log(x);
```

Config

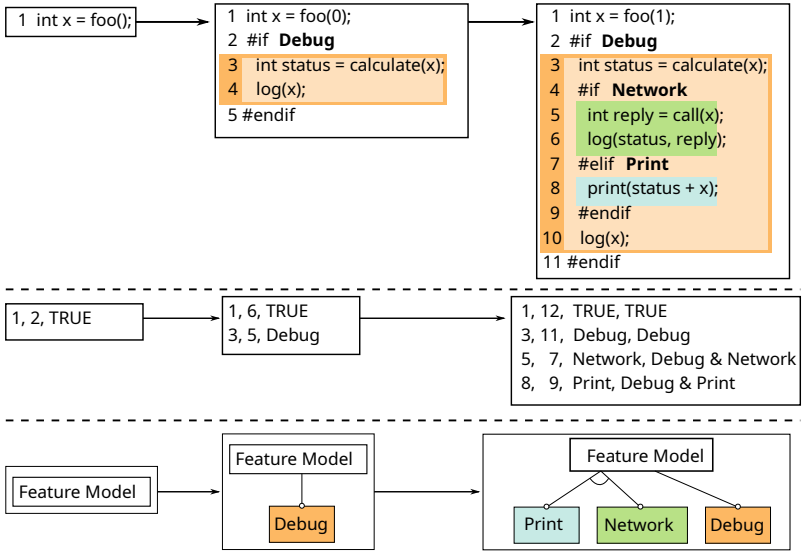
Integration

```
1  int x = foo(1);
2  #if  Debug
3    int status = calculate(x);
4    #if  Network
5      int reply = call(x);
6      log(status, reply);
7    #elif  Print
8      print(status + x);
9    #endif
10   log(x);
11 #endif
```

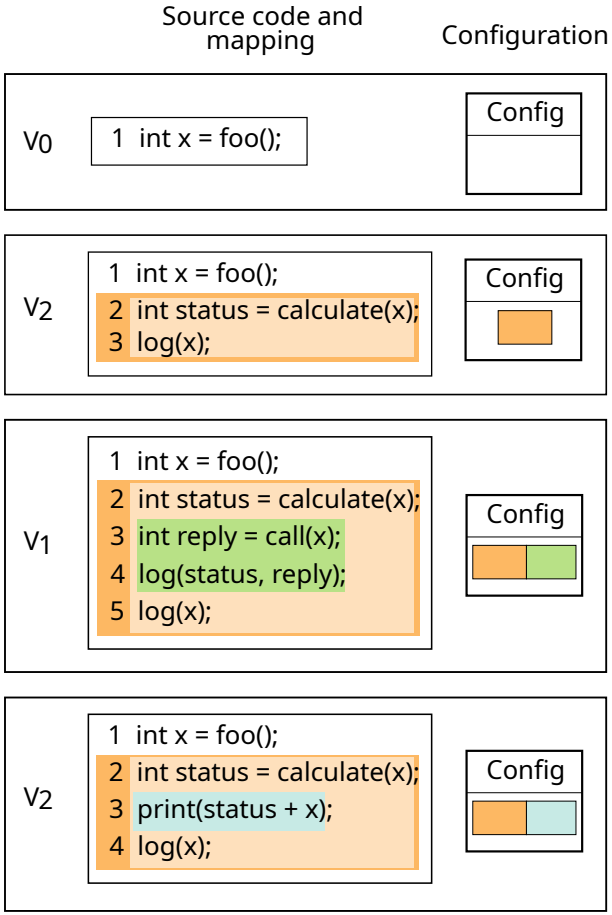


2. Variant Integration (VI)

2. Requirement:
Revisions of a product line

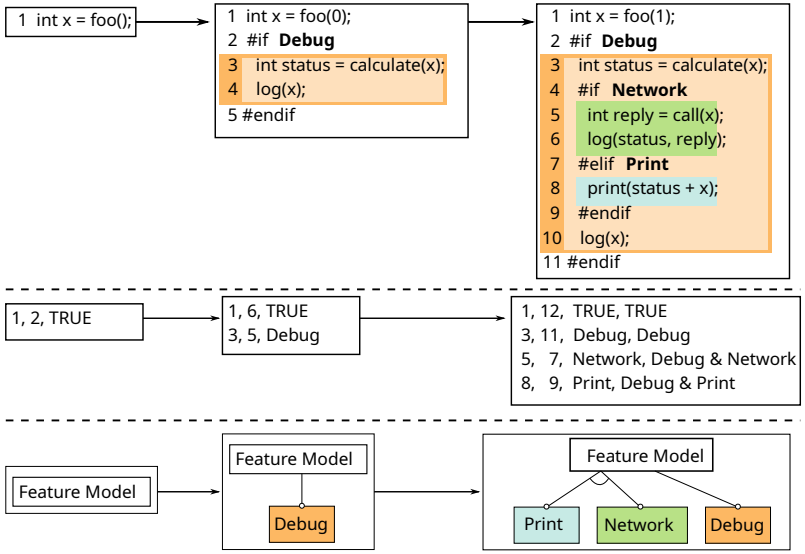


VEVOS

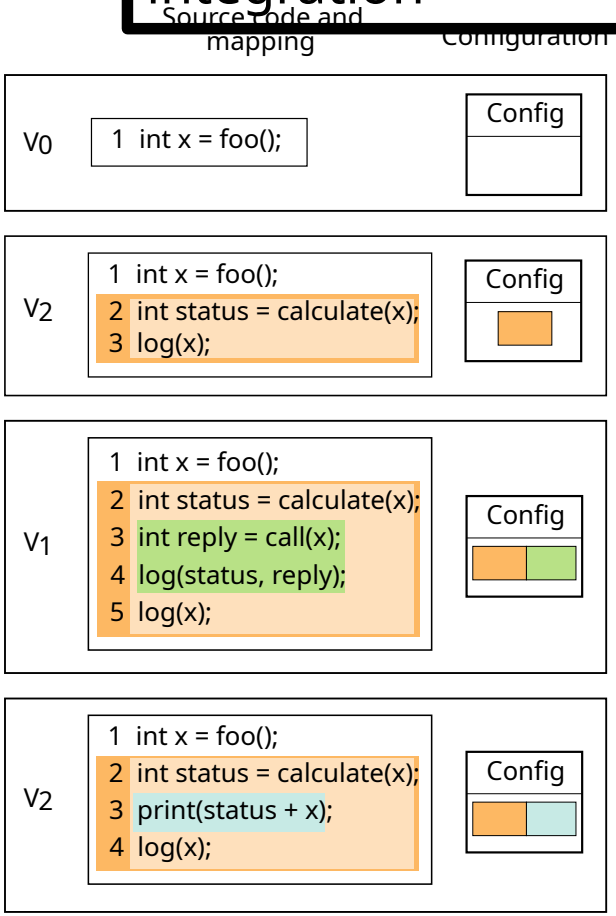


2. Variant Integration (VI)

3. Requirement:
Correct product line after integration

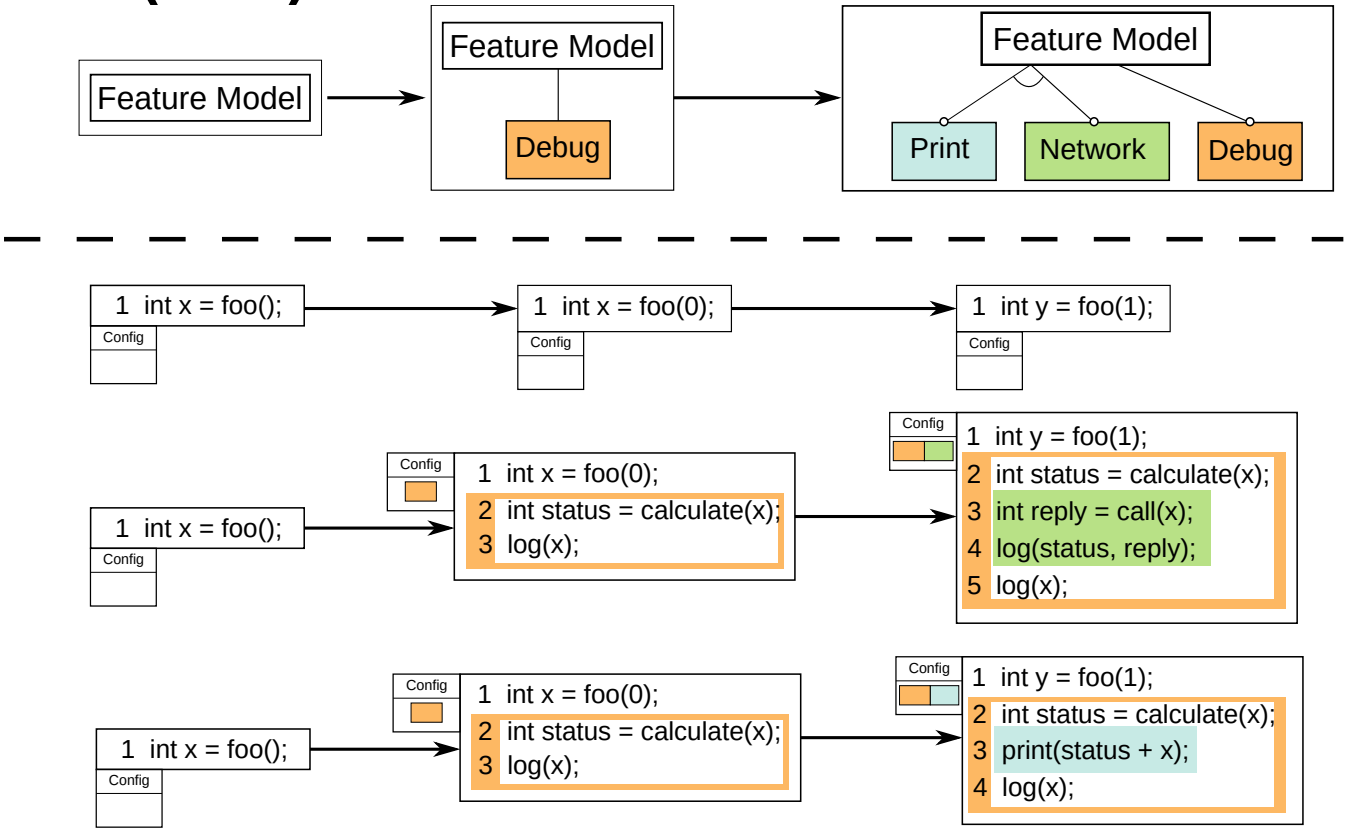


VEVOS



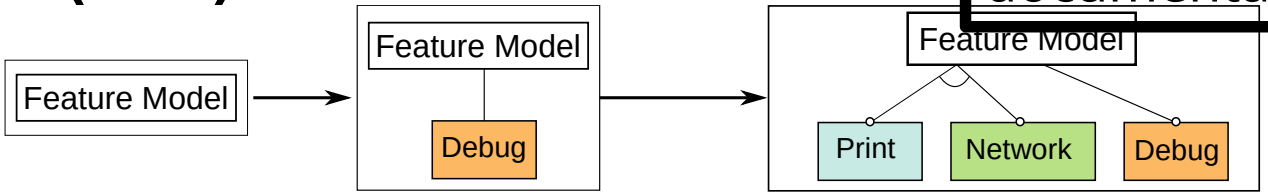
Feature identification and location (FIL)

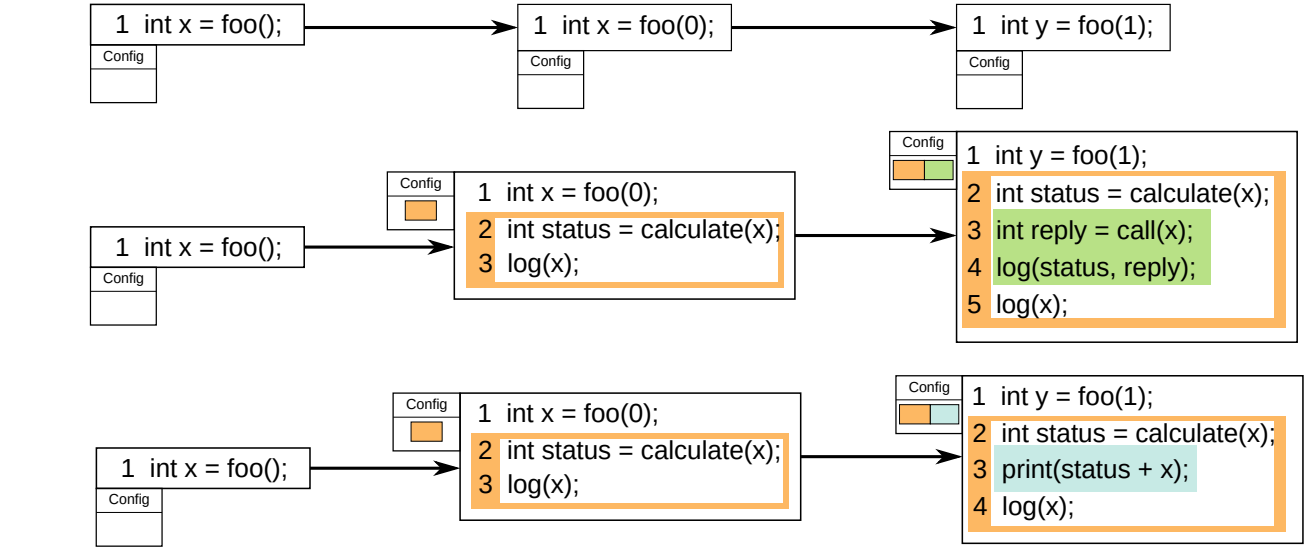
3. Feature Identification and Location (FIL)



3. Feature Identification and Location (FIL)

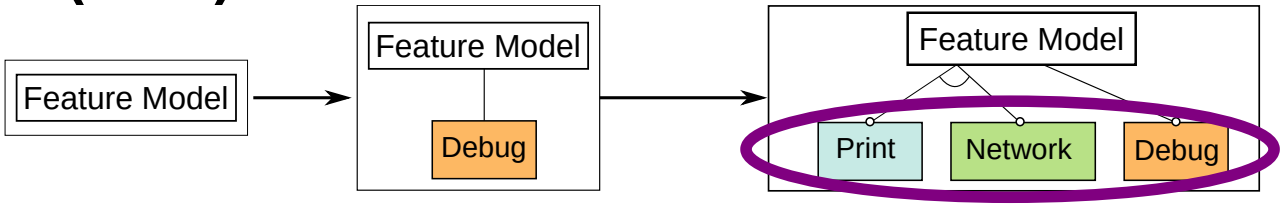
1. Requirement:
Variant Artifacts (e.g. code, documentation, history)

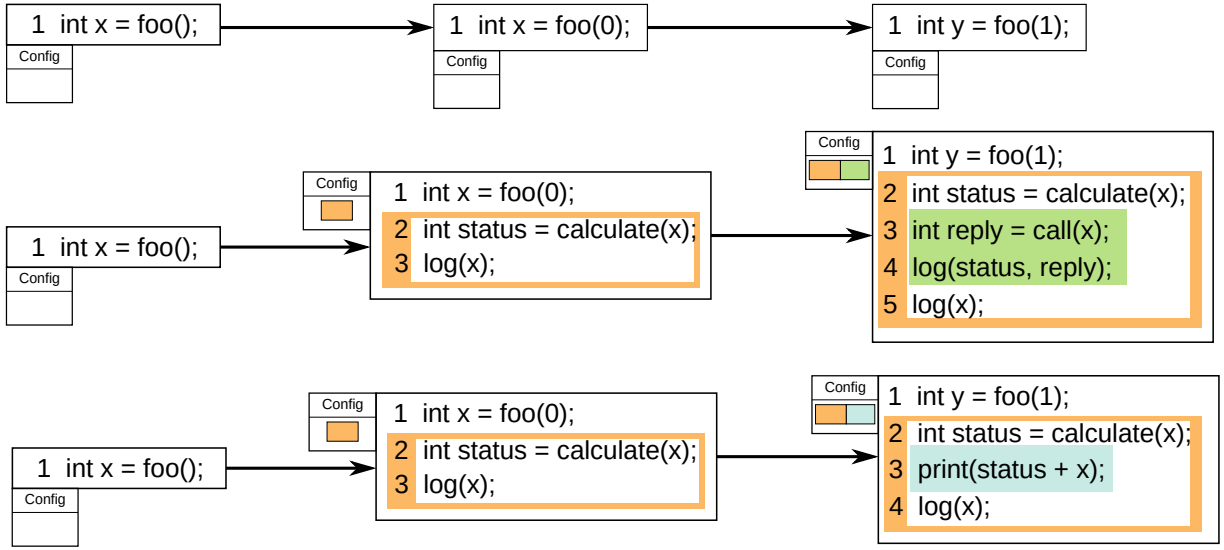




3. Feature Identification and Location (FIL)

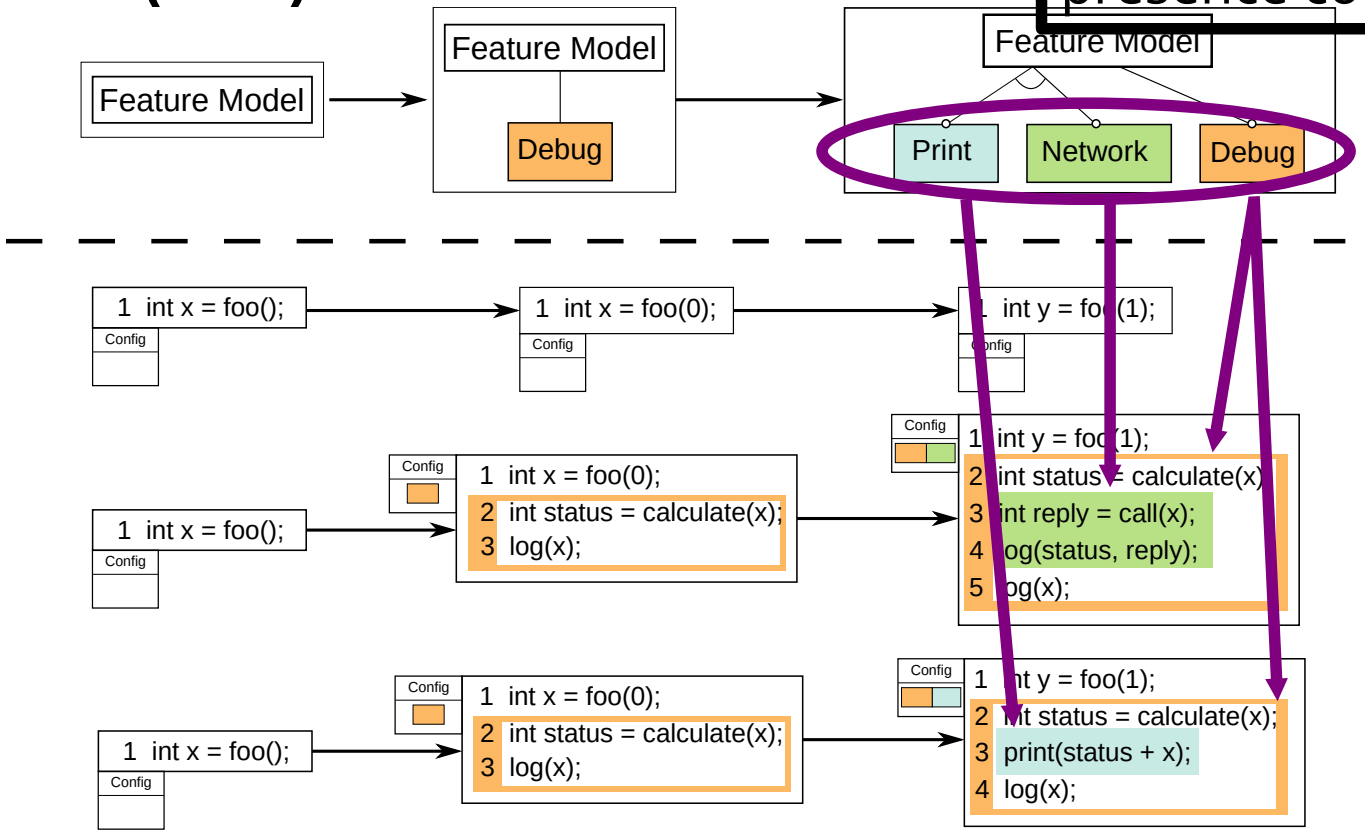
2. Requirement:
List of existing features





3. Feature Identification and Location (FIL)

3. Requirement: Feature mappings or presence conditions



Scenario coverage of VEVOS

Benchmark	Original Context	VS	VI	FIL	CE	FMS	AR	TR	FT	ANF	VZ	CPS
ArgoUML-SPL FLBench	Feature location	○	○	◓	○	○	○	○	○	○	○	○
Drupal	Bug detection	○	○	○	○	○	○	○	●	○	◓	○
Eclipse FLBench	Feature location	○	○	◓	◓	◓	○	○	○	○	○	○
LinuxKernel FLBench	Feature location	○	○	◓	◓	◓	○	○	○	○	○	○
Marlin & BCWallet	Feature location	○	○	◓	○	○	○	○	○	○	◓	○
ClaferWebTools	Traceability	○	○	◓	○	◓	○	○	○	○	◓	○
DoSC	Change discovery	○	◓	◓	○	◓	○	◓	○	○	◓	○
SystemsSwVarModels	FM synthesis	○	◓	○	●	●	○	○	○	○	◓	○
TraceLab CoEST	Traceability	○	○	◓	○	○	○	○	○	○	○	○
Variability bug database	Bug detection	○	○	◓	○	○	○	○	●	○	◓	○
VEVOS	Clone-and-Own	?	?	?	?	?	?	?	?	?	?	?

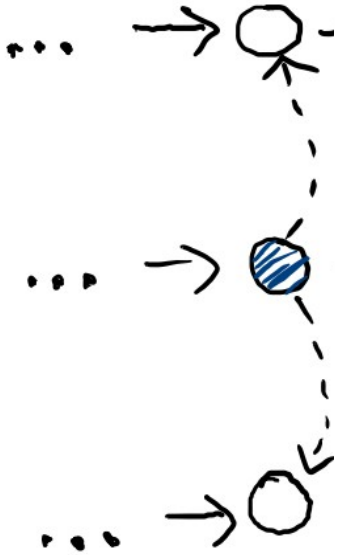
Scenario coverage of VEVOS

Benchmark	Original Context	VS	VI	FIL	CE	FMS	AR	TR	FT	ANF	VZ	CPS
ArgoUML-SPL FLBench	Feature location	○	○	◓	○	○	○	○	○	○	○	○
Drupal	Bug detection	○	○	○	○	○	○	○	●	○	◓	○
Eclipse FLBench	Feature location	○	○	◓	◓	◓	○	○	○	○	○	○
LinuxKernel FLBench	Feature location	○	○	◓	◓	◓	○	○	○	○	○	○
Marlin & BCWallet	Feature location	○	○	◓	○	○	○	○	○	○	◓	○
ClaferWebTools	Traceability	○	○	◓	○	◓	○	○	○	○	◓	○
DoSC	Change discovery	○	◓	◓	○	◓	○	◓	○	○	◓	○
SystemsSwVarModels	FM synthesis	○	◓	○	●	●	○	○	○	○	◓	○
TraceLab CoEST	Traceability	○	○	◓	○	○	○	○	○	○	○	○
Variability bug database	Bug detection	○	○	◓	○	○	○	○	●	○	◓	○
VEVOS	Clone-and-Own	●	●	●	?	?	?	?	?	?	?	?

Experiences of applying VEVOS in various research studies

S1: Automating variant synchronization

(VS)



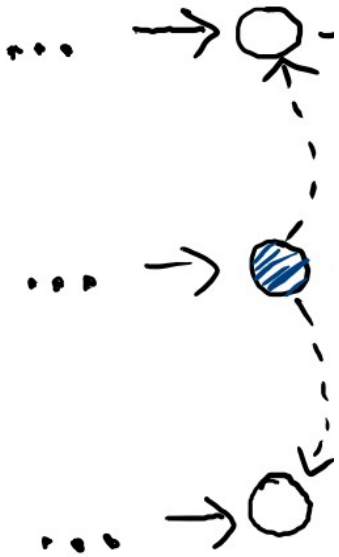
Quantifying the Potential to Automate the Synchronization of Variants in Clone-and-Own,
Schultheiß, Bittner, Thüm, Kehrer, ICSME’2022

Manual development

Automated Synchronization

S1: Automating variant synchronization

(VS)



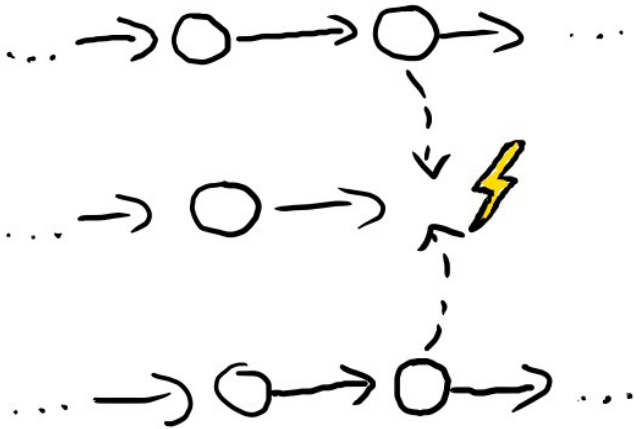
Quantifying the Potential to Automate the Synchronization of Variants in Clone-and-Own, Schultheiß, Bittner, Thüm, Kehrer, ICSME'2022

Manual development

Automated Synchronization

S2: Conflict scenarios during variant synchronization

(VS)



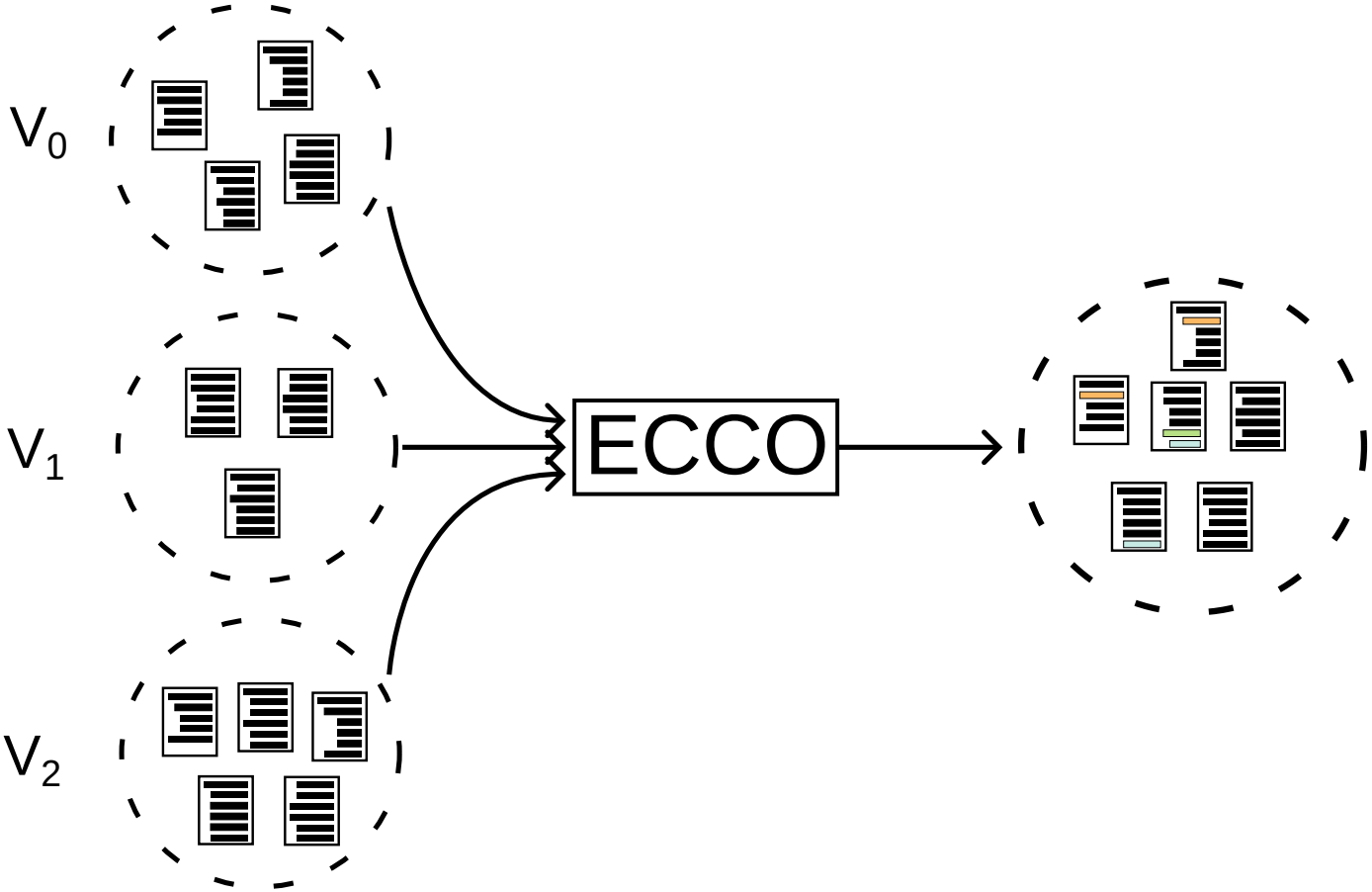
- What types of conflicts exist?

- What are possible conflict resolutions?

- How many conflicts occur?

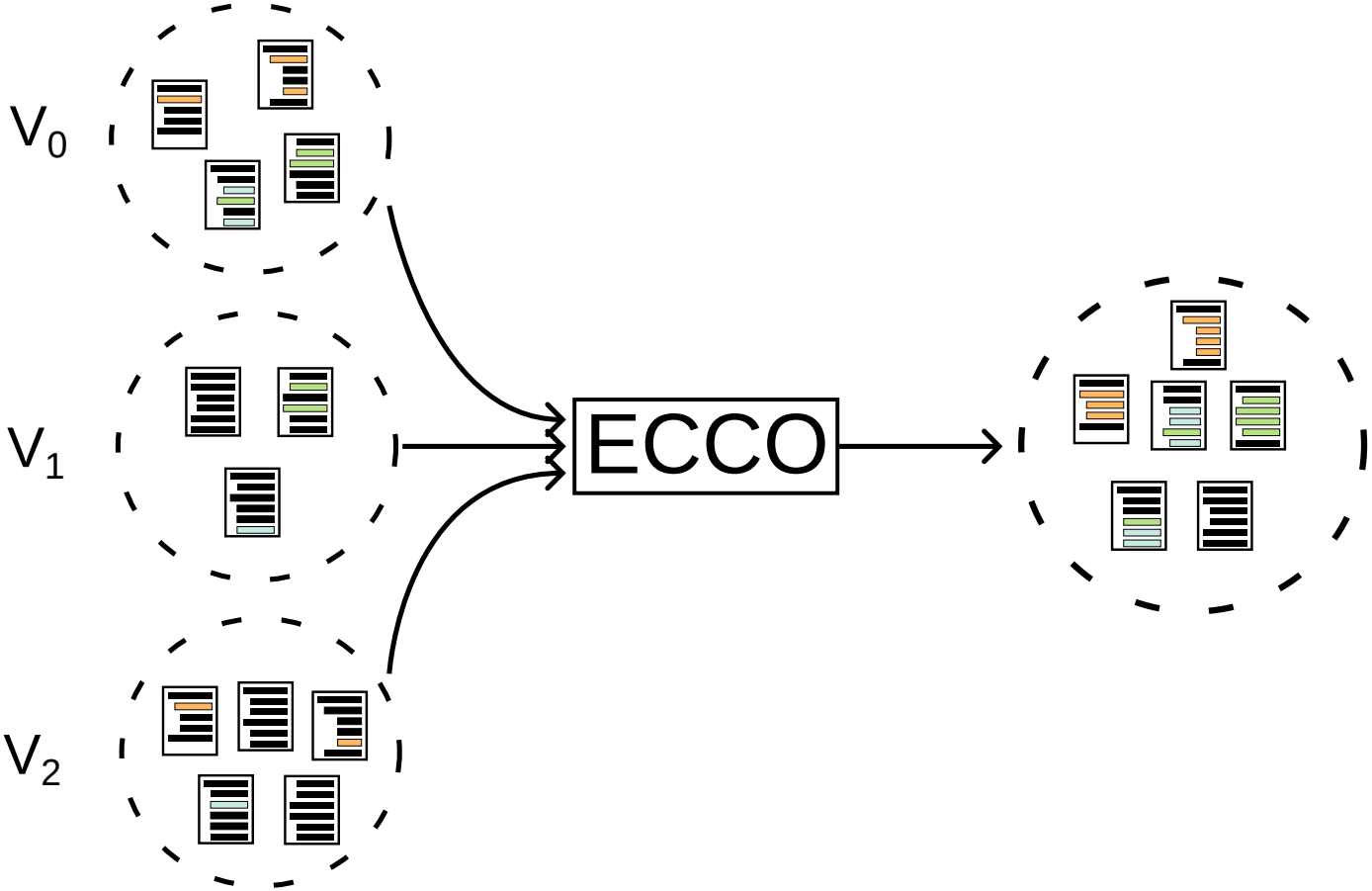
S3: Utilizing partial feature mappings for feature location

(FIL)



S3: Utilizing partial feature mappings for feature location

(FIL)



S3: Utilizing partial feature mappings for feature location

(FIL)

Corresponding evolution scenarios

	S1: Sync. automation	S2: Sync. conflicts	S3: Feature location
Scenario(s)			
VEVOS' Applicability			

Corresponding evolution scenarios

Simulated variant evolution lacks parallel devlopment

	S2: Sync. conflicts	S3: Feature location
	Variant synchronization (VS)	
Applicability		

Corresponding evolution scenarios

	S1: Sync. automation	S2: Sync. conflicts	S3: Feature location
Scenario(s)	Variant synchronization (VS)	Variant synchronization (VS)	
VEVOS' Applicability			

Corresponding evolution scenarios

	S1: Sync. automation	S2: Sync. conflicts	S3: Feature location	S4	S5	S6
Scenario(s)	Variant synchronization (VS)	Variant synchronization (VS)	Feature identification and location (FIL)	?	?	?
VEVOS' Applicability				?	?	?

General experiences

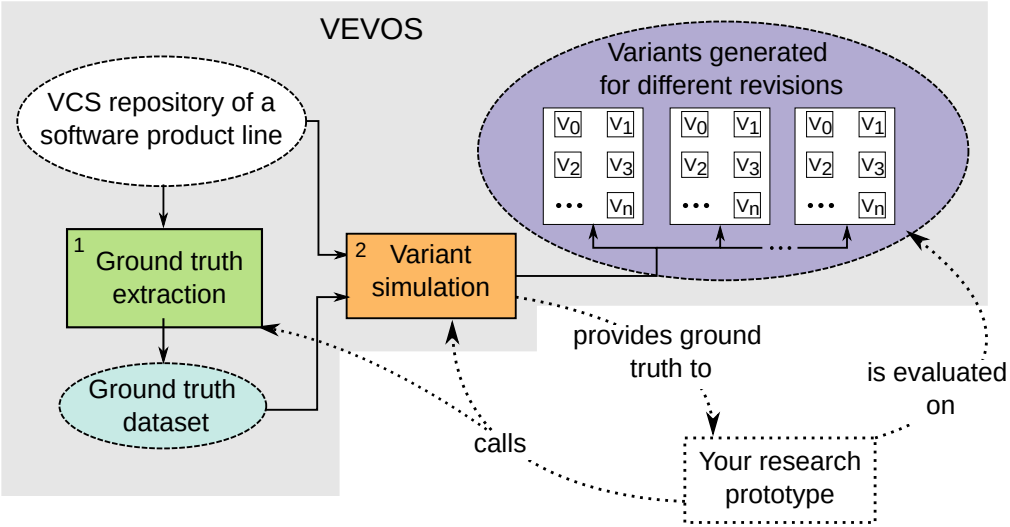
Calling VEVOS requires a JVM-related language

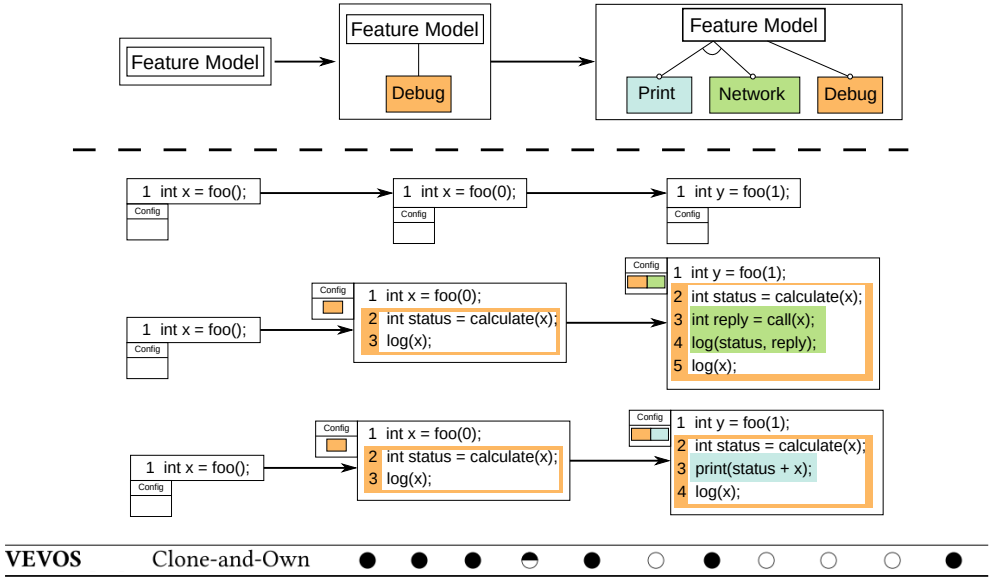
Ground truth extraction has certain restrictions

→ Build model analysis is not robust

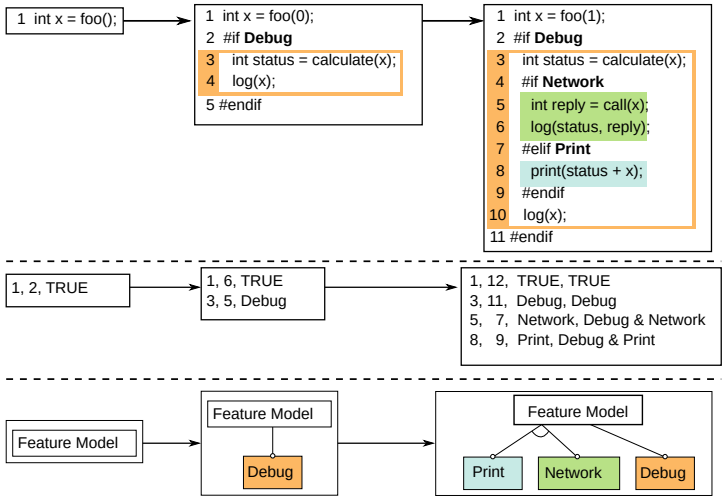
→ But, extraction can be extended

Memory and runtime issues for very large datasets (i.e. entire history of the Linux kernel)





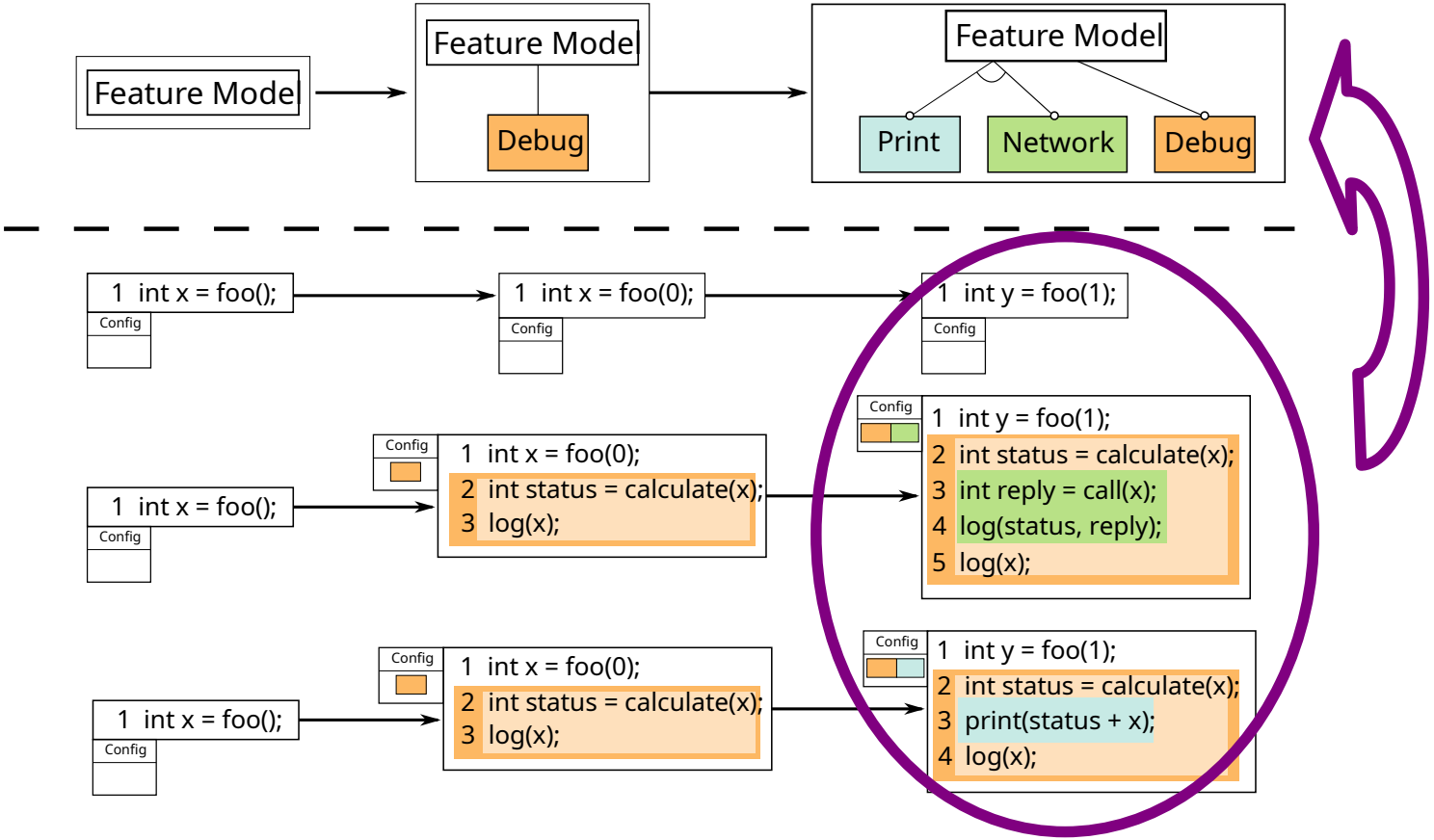
	S1: Sync. automation	S2: Sync. conflicts	S3: Feature location
Scenario(s)	Variant synchronization (VS)	Variant synchronization (VS)	Feature identification and location (FIL)
VEVOS' Applicability			



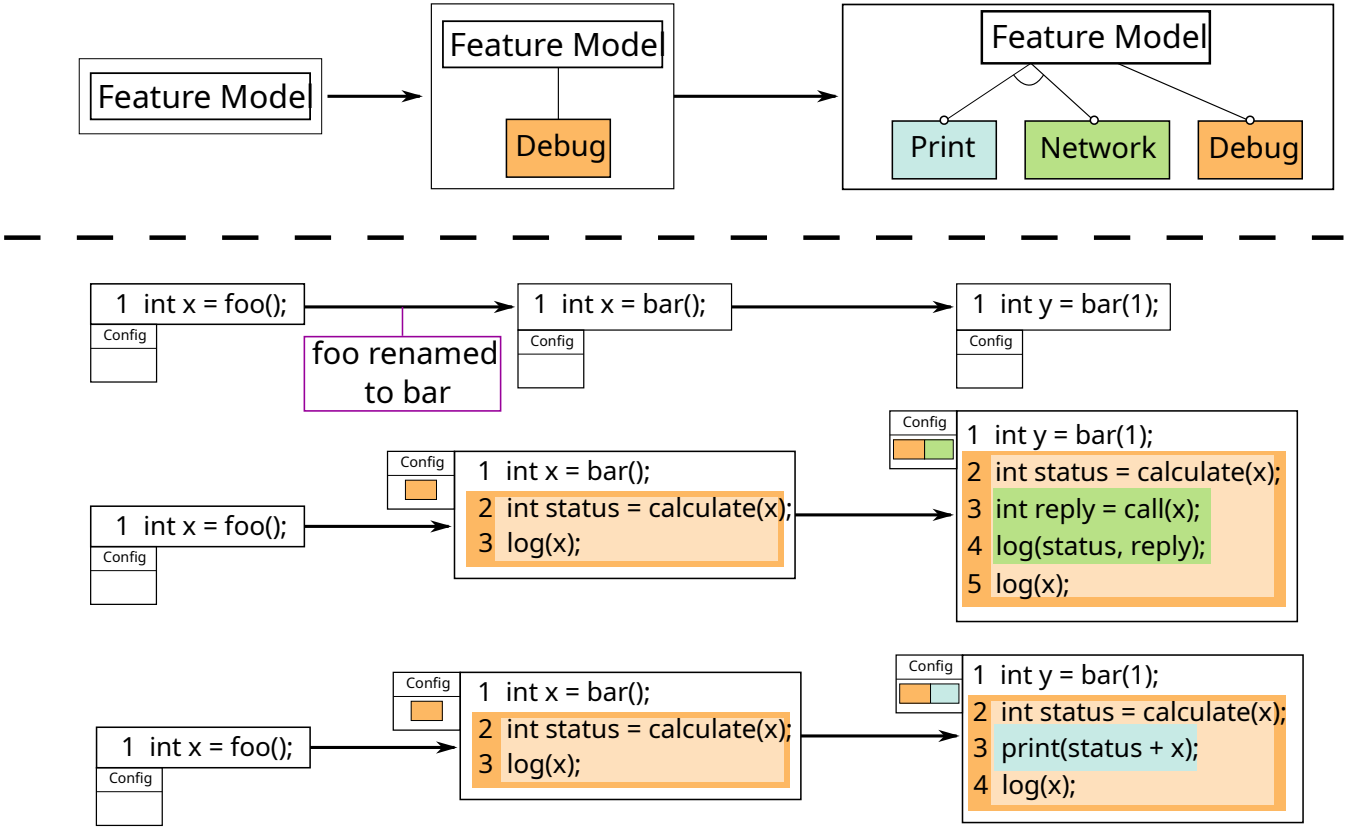
Appendix

Scenario coverage

Feature Model Synthesis (FMS)



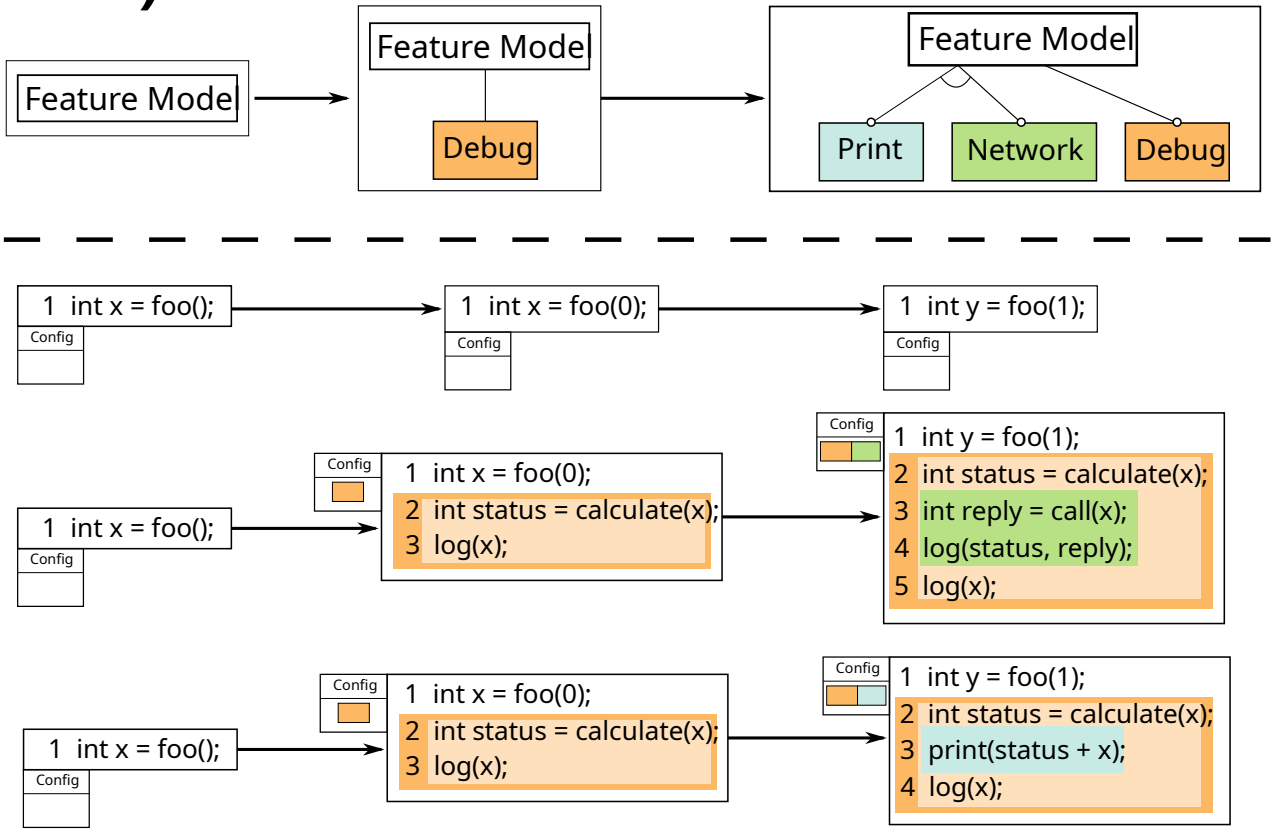
Transformations (TR)



Co-Evolution of Problem Space and Solution Space (CPS)

Problem Space

Solution Space



Research studies

S4: Match-based patch context resolution

- Can we improve patching by searching matching code
- Account for variability

S5: Effect of variant drift on change propagation

- VEVOS does not account for unintentional variant drift
- Can we simulate variant drift?
- Problem: Variant drift invalidates ground truth

S6: SPL analysis tools for clone-and-own variants

- Can SPL analysis tools be applied to cloned variants?
- Problem: Simulated variants depend on product line build files