Constructing Feature Model by Identifying Variability-aware Modules

Yutian Tang, Hareton Leung
The Hong Kong Polytechnic University
ICPC 2017
May 22-23, BA

Feature Model

domain expert

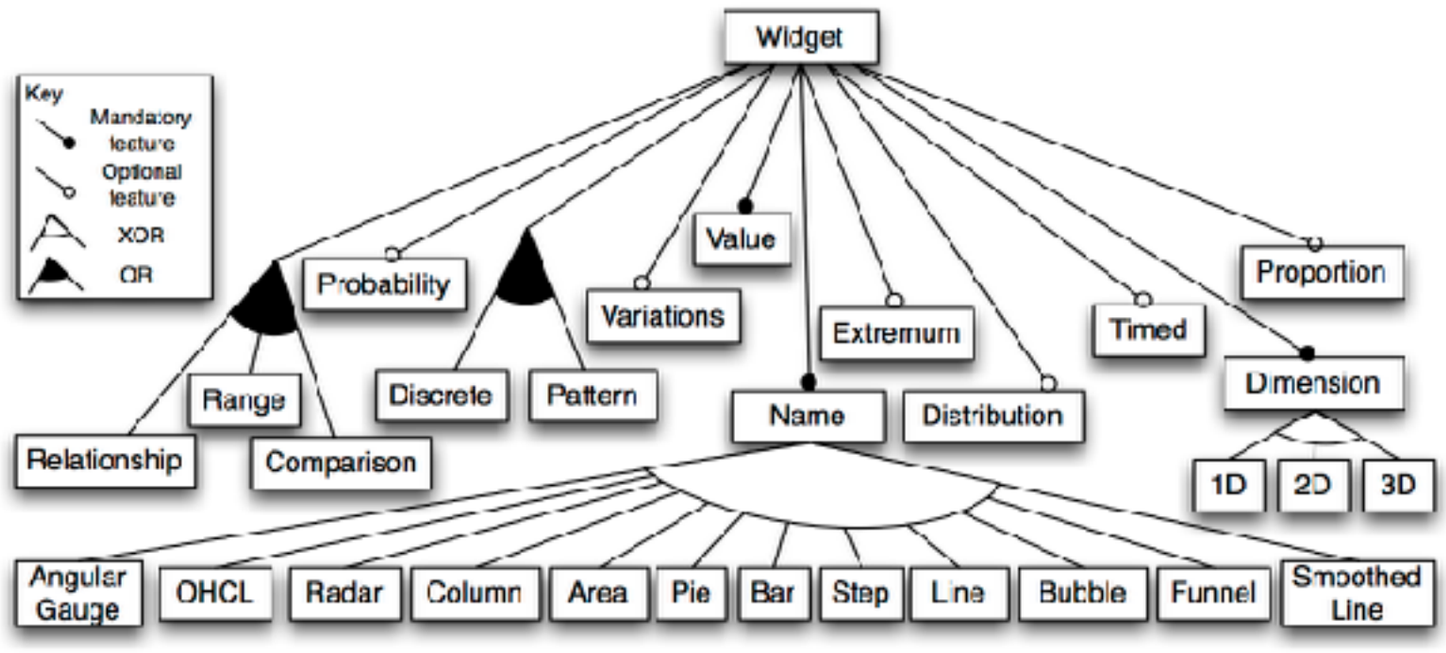


image source: www.i3s.unice.fr

Motivation

Migrate Legacy into Product Line

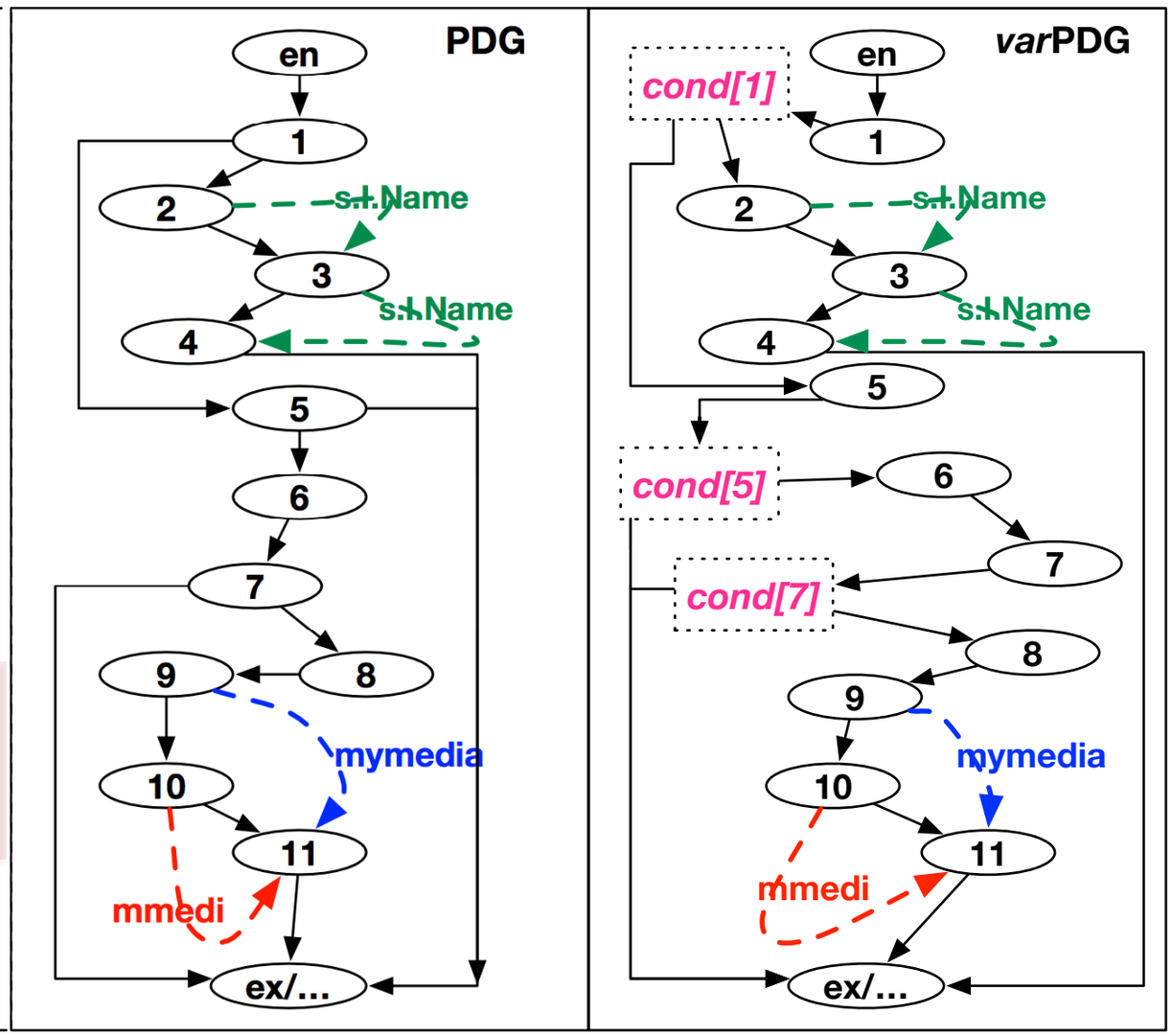
domain expert not available

specification not available

Configurations & Configuration Variables

varPDG (variability-aware PDG)

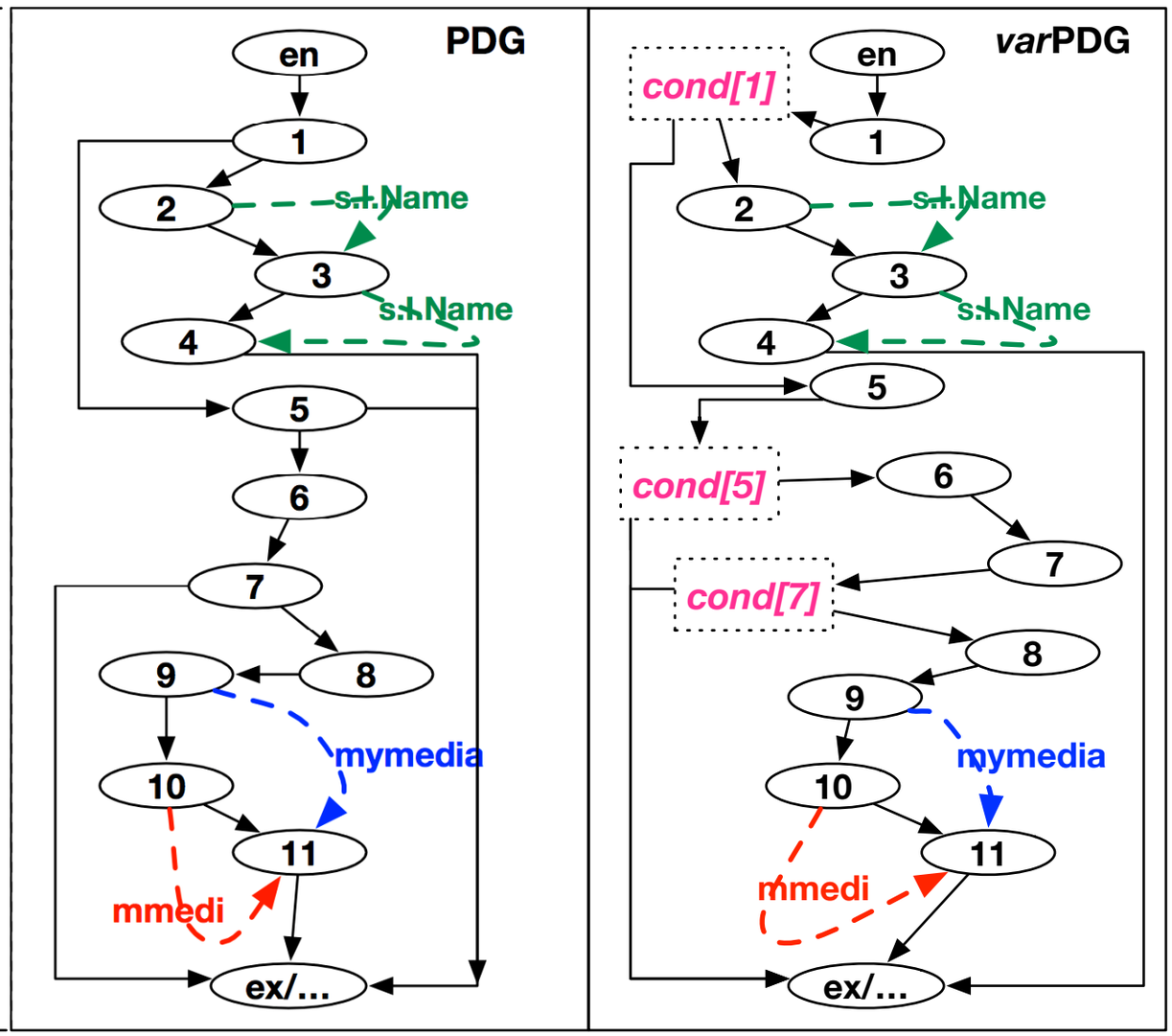
```
// #ifdef includePhotoAlbum
1 if (label.equals("View")) {
2    String selectedImageName = getSelectedMediaName();
3    showImage(selectedImageName);
     // #ifdef includeSorting
4    incrementCountViews(selectedImageName);
     //#endif
     …
}

//#endif
5 if (label.equals("Save Item")) {
6       getAlbumData().addNewMediaToAlbum(…);
        // #ifdef includeMusic
7       if (getAlbumData() instanceof MusicAlbumData){

8         getAlbumData().loadMediaDataFromRMS(getCurrentStoreName());
9         MediaData mymedia = getAlbumData().getMediaInfo(…);
10        MultiMediaData mmedi = new MultiMediaData(…);
11        getAlbumData().updateMediaInfo(mymedia, mmedi);

        }
        //#endif
        …
}
```



Line 8 - 11: cond[5] ^ cond [7]

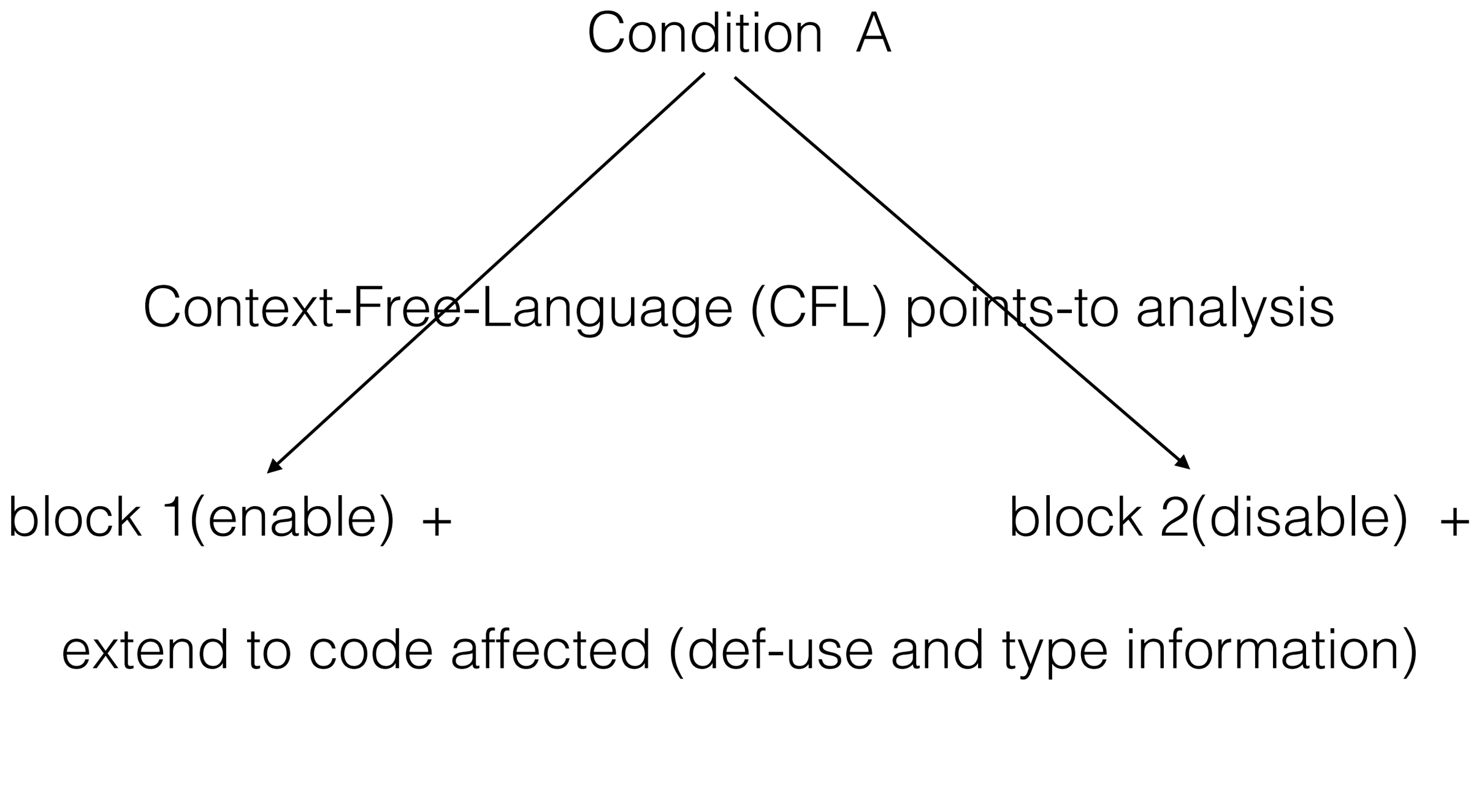
varPDG (variability-aware PDG)

```
// #ifdef includePhotoAlbum
1 if (label.equals("View")) {
2    String selectedImageName = getSelectedMediaName();
3    showImage(selectedImageName);
     // #ifdef includeSorting
4    incrementCountViews(selectedImageName);
     //#endif
     …
}

//#endif
5 if (label.equals("Save Item")) {
6       getAlbumData().addNewMediaToAlbum(…);
        // #ifdef includeMusic
7       if (getAlbumData() instanceof MusicAlbumData){

8         getAlbumData().loadMediaDataFromRMS(getCurrentStoreName());
9         MediaData mymedia = getAlbumData().getMediaInfo(…);
10        MultiMediaData mmedi = new MultiMediaData(…);
11        getAlbumData().updateMediaInfo(mymedia, mmedi);

        }
        //#endif
        …
}
```

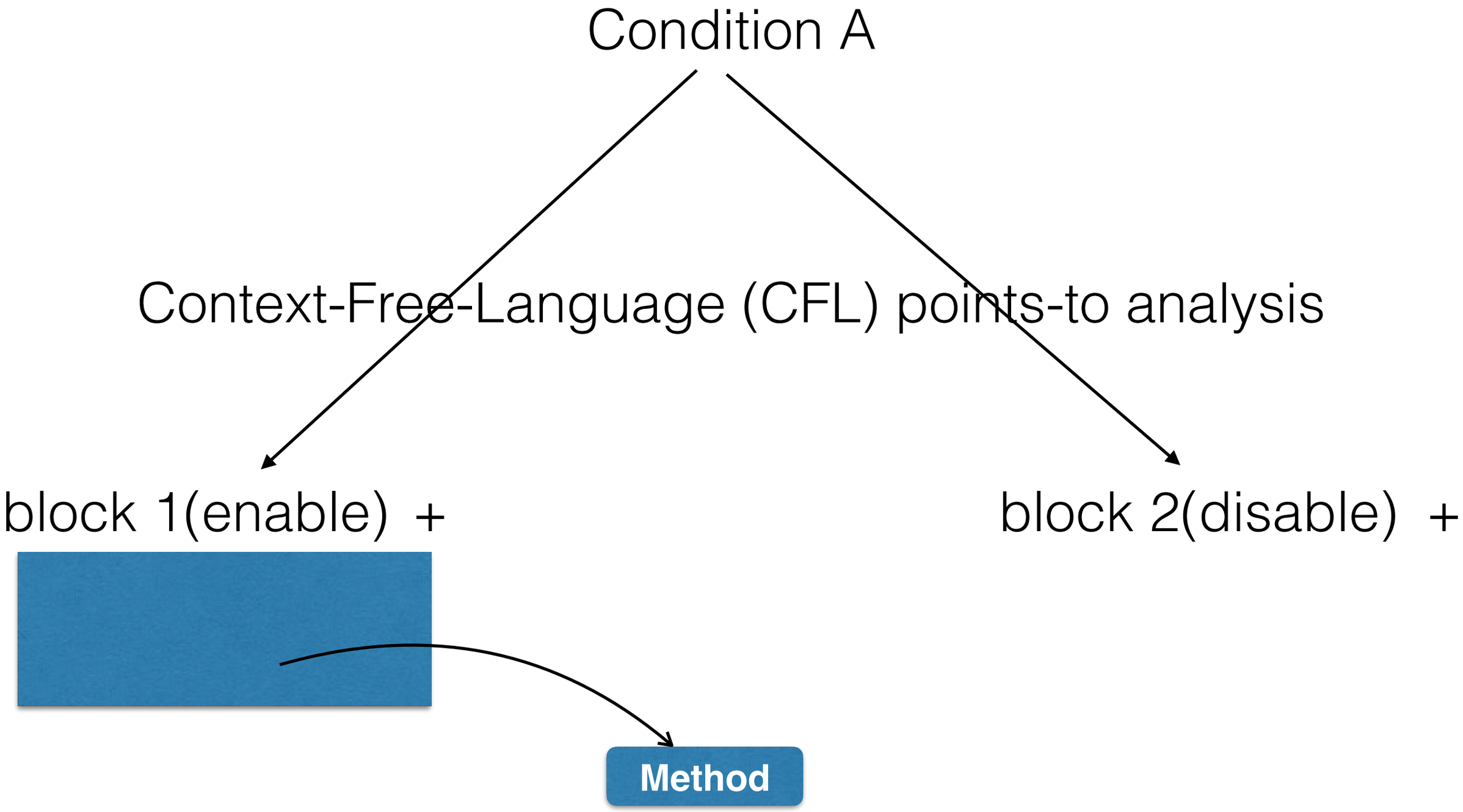


1. Extract all possible configuration variables
2. Link the PDG node with conditions obtained

varPDG (variability-aware PDG)



varPDG (variability-aware PDG)



Modules

Modules

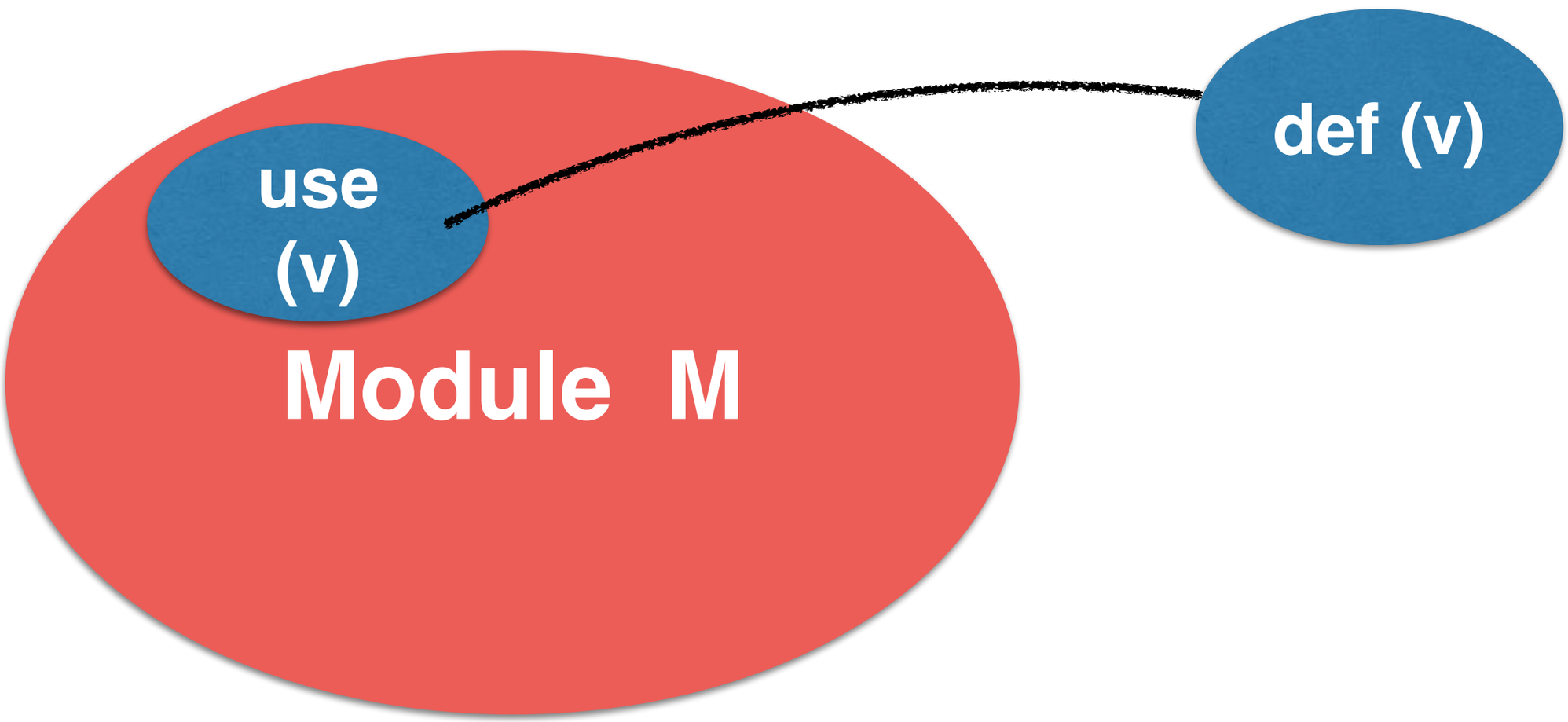
$$m = (v, i, j, \Gamma, \Delta)$$

Notation	Remark
$e \in \mathcal{E}$	expressions
$t \in \mathcal{T}$	types
$x \in \mathcal{X}$	function names
$o \in \mathcal{O}$	configuration options
$c \in \mathcal{C} = 2^{\mathcal{O}}$	configurations
$v \in \mathcal{V} = 2^{\mathcal{C}}$	variability model
$\Gamma \in \mathcal{C} \xrightarrow{p} \mathcal{X} \xrightarrow{p} \mathcal{T}$	import function signature
$\Delta \in \mathcal{C} \xrightarrow{p} \mathcal{X} \xrightarrow{p} \mathcal{T} \times \mathcal{E}$	function defintion
$m = (v, i, j, \Gamma, \Delta) \in \mathcal{M}; i, j \subseteq \mathcal{O}$	module[1]
$\phi_{DUC \wedge TC}(m)$	constraint function

class

Module Constraints

- Def-Use Constraints (DUC):
- - from use to its def



Module Constraints

- Type Constraints (TC):
- - (1) variable/field; (2) method; (3) interface/class. For example, in (1), a variable in a child class can be used without define when it is defined in its parent class; (2) is simply referred as method overriding; and (3) is considered as a case of inheritance.

Module Constraints

$\phi_{DUF \wedge TC}(m)$

DEF-USE constraint is satisfied

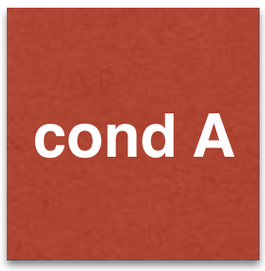
and

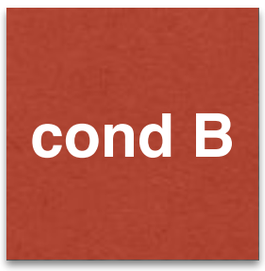
type constraint is satisfied

Module Constraints

$$\phi_{DUF \wedge TC}(m)$$

cond A excludes cond B





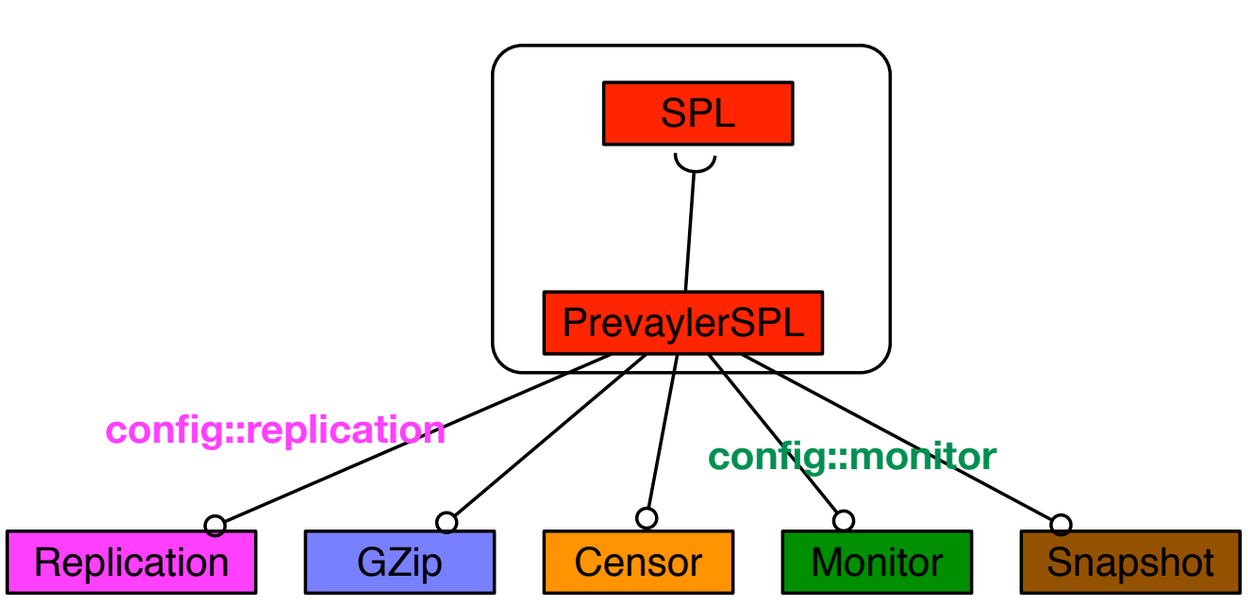
def: v

use: v

Variability-Aware Module Strategy (VMS)

- Core Idea

Step 1: Common Modules



Step 2: Enable options and find modules affected

Step 3: Clustering under each configuration

config::replication:{Replication, SPL, PrevaylerSPL}

config::gzip:{Gzip, SPL, PrevaylerSPL}

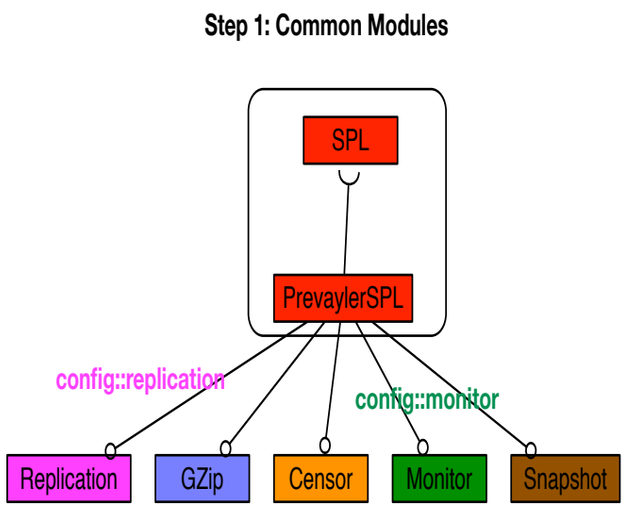
config::replication:{Censor, SPL, PrevaylerSPL}

config::monitor:{Monitor, SPL, PrevaylerSPL}

config::replication:{Replication, SPL, PrevaylerSPL}

Variability-Aware Module Strategy (VMS)

- Core Idea



Step 2: Enable options and find modules affected

Step 3: Clustering under each configuration

config::replication:{Replication, SPL, PrevaylerSPL}

config::gzip:{Gzip, SPL, PrevaylerSPL}

config::replication:{Censor, SPL, PrevaylerSPL}

config::monitor:{Monitor, SPL, PrevaylerSPL}

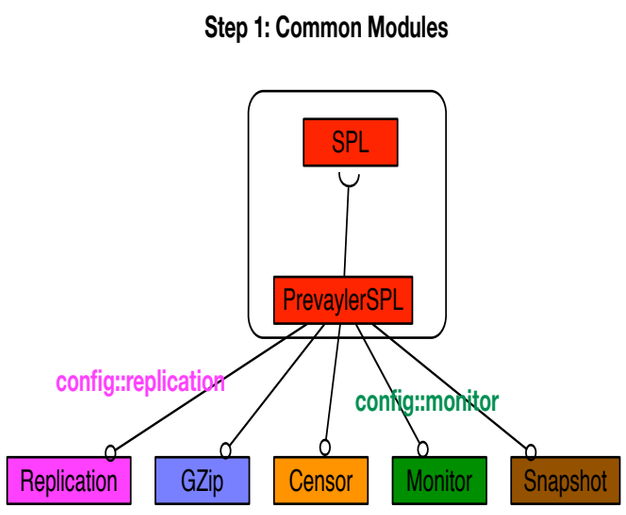
config::replication:{Replication, SPL, PrevaylerSPL}

Step 1.

Extract all common modules from the system

Variability-Aware Module Strategy (VMS)

- Core Idea



Step 2: Enable options and find modules affected

Step 3: Clustering under each configuration

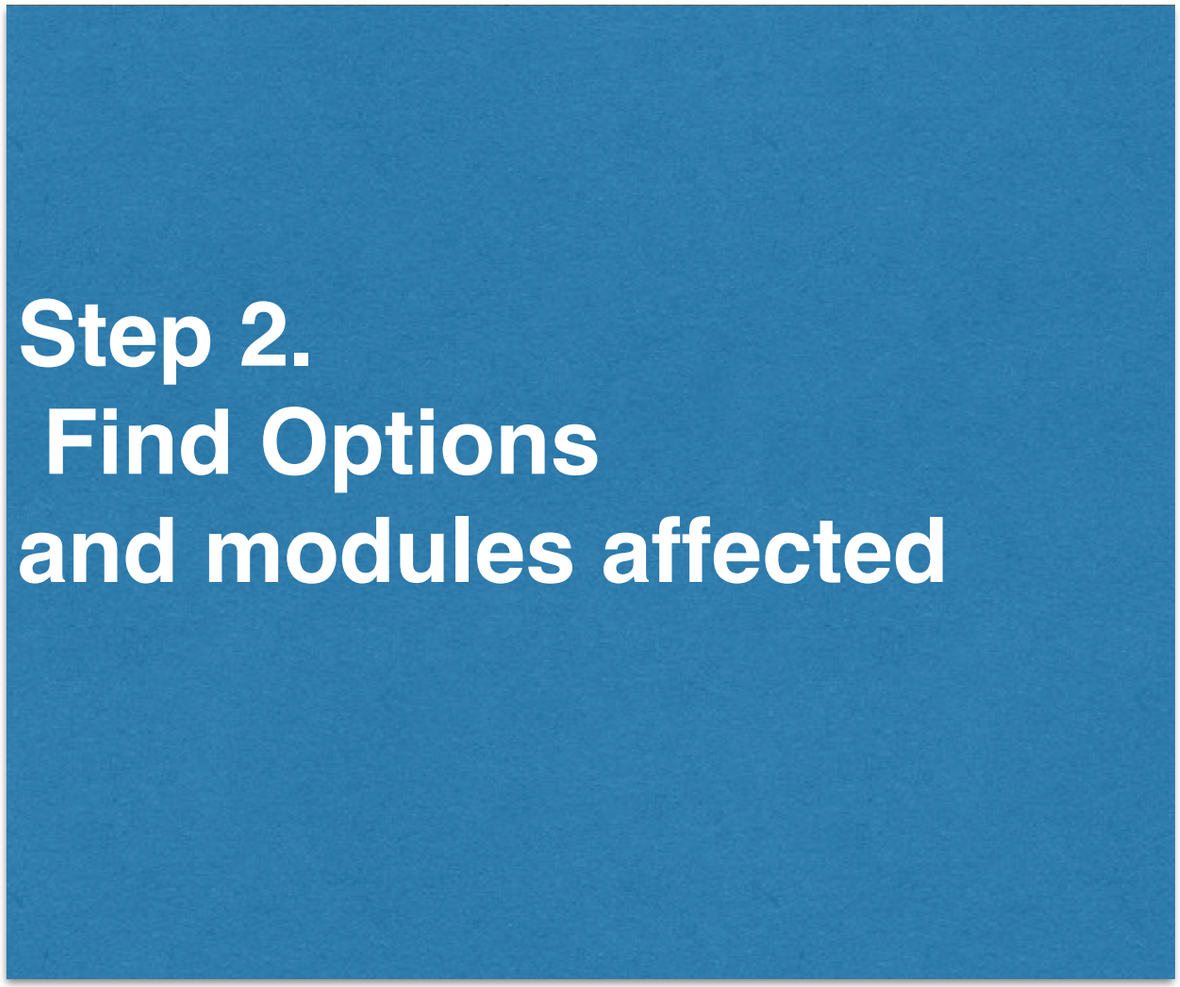
config::replication:{Replication, SPL, PrevaylerSPL}

config::gzip:{Gzip, SPL, PrevaylerSPL}

config::replication:{Censor, SPL, PrevaylerSPL}

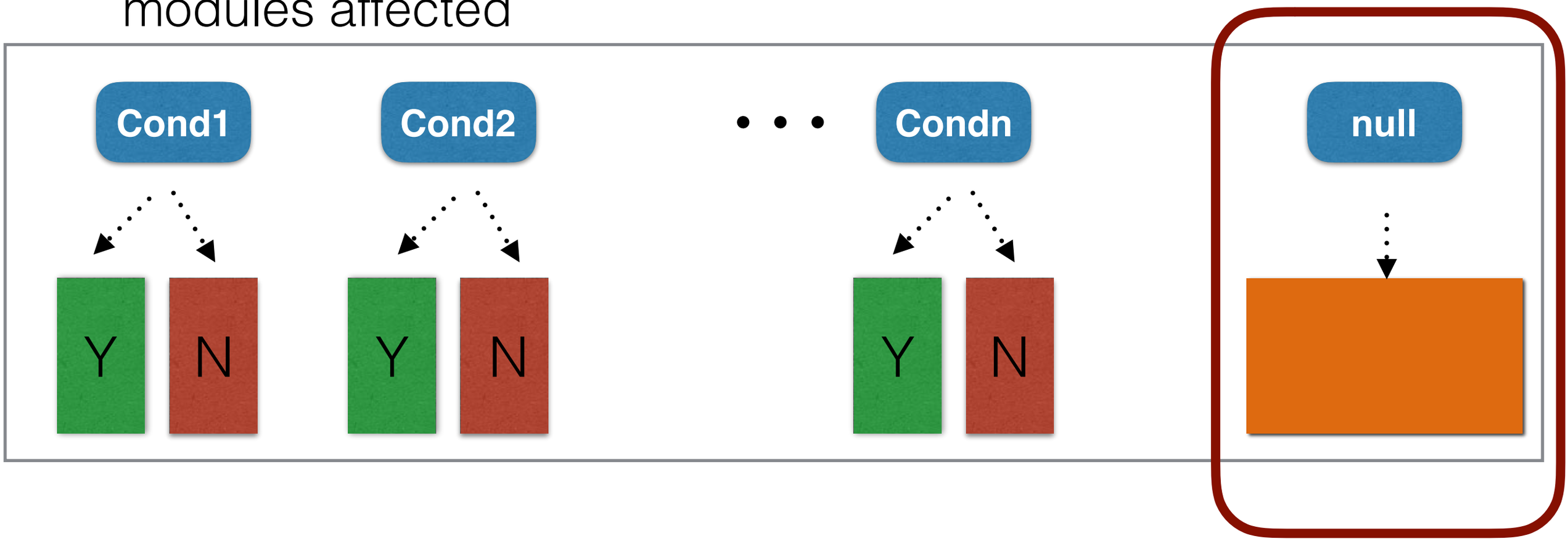
config::monitor:{Monitor, SPL, PrevaylerSPL}

config::replication:{Replication, SPL, PrevaylerSPL}



Variability-Aware Module Strategy (VMS)

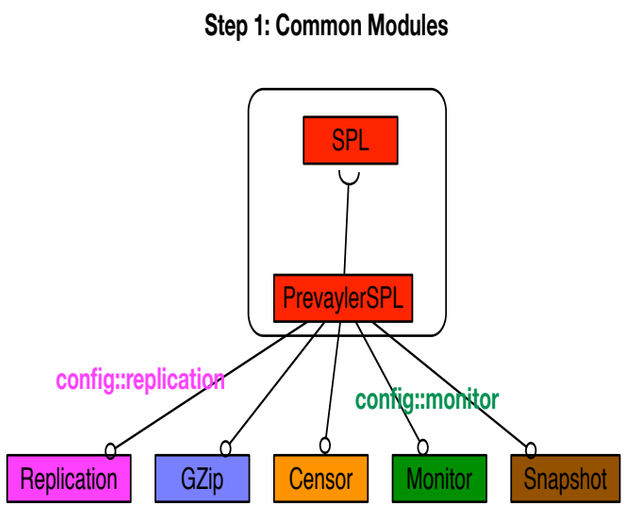
- Step 1 & 2 Extracting common modules; find options and modules affected



re-organize the program to condition-block

Variability-Aware Module Strategy (VMS)

- Core Idea



Step 2: Enable options and find modules affected

Step 3: Clustering under each configuration

config::replication:{Replication, SPL, PrevaylerSPL}

config::gzip:{Gzip, SPL, PrevaylerSPL}

config::replication:{Censor, SPL, PrevaylerSPL}

config::monitor:{Monitor, SPL, PrevaylerSPL}

config::replication:{Replication, SPL, PrevaylerSPL}

Step 3. Clustering modules

Variability-Aware Module Strategy (VMS)

- Step 3 Clustering Strategy

- Strategy 1: Topology based Method Reference[8]

  it computes the similarity using two metrics specificity and reinforcement. Specifically, the **specificity** suggests that if an element A only refers to an other element B should be ranked higher comparing to C refer to many elements including B. And the intuition behind **reinforcement** is that if elements refer to (or referred from) many elements are in one cluster, possibly they should be considered as a part of that cluster.

Variability-Aware Module Strategy (VMS)

- Step 3 Clustering Strategy

- Strategy 2: Type Reference

The underlying idea in type reference is that for each module, it looks up all possible references, such as, method reference - from method invocation to method definition; variable reference - from variable access to its definition; or type reference - from a type reference to its declaration and explore the types referred in all these references.

$$w_{tr}(m,f) = \frac{1}{2n}\left(\sum_{m_i \in f}\left(\begin{array}{l} csd(def(m), ref(m_i)) + \\ csd(ref(m), def(m_i)) \end{array}\right)\right)$$

Variability-Aware Module Strategy (VMS)

- Step 3 Clustering Strategy

- Strategy 3: Documental Topic Similarity

Upon this *corpus*, an information retrieval approach named Latent Dirichlet Allocation (LDA) can be used to extract the topic distribution. Furthermore, a topic z is given based on a multinomial probability distribution upon a set of words ws obtained from a Dirichlet distribution with the shape parameter β.

Variability-Aware Module Strategy (VMS)

- Combine these 3 strategies

$$w_* = w_{tmr} \uplus w_{tr} \uplus w_{dt}$$

$$x \uplus y = x + y - xy$$

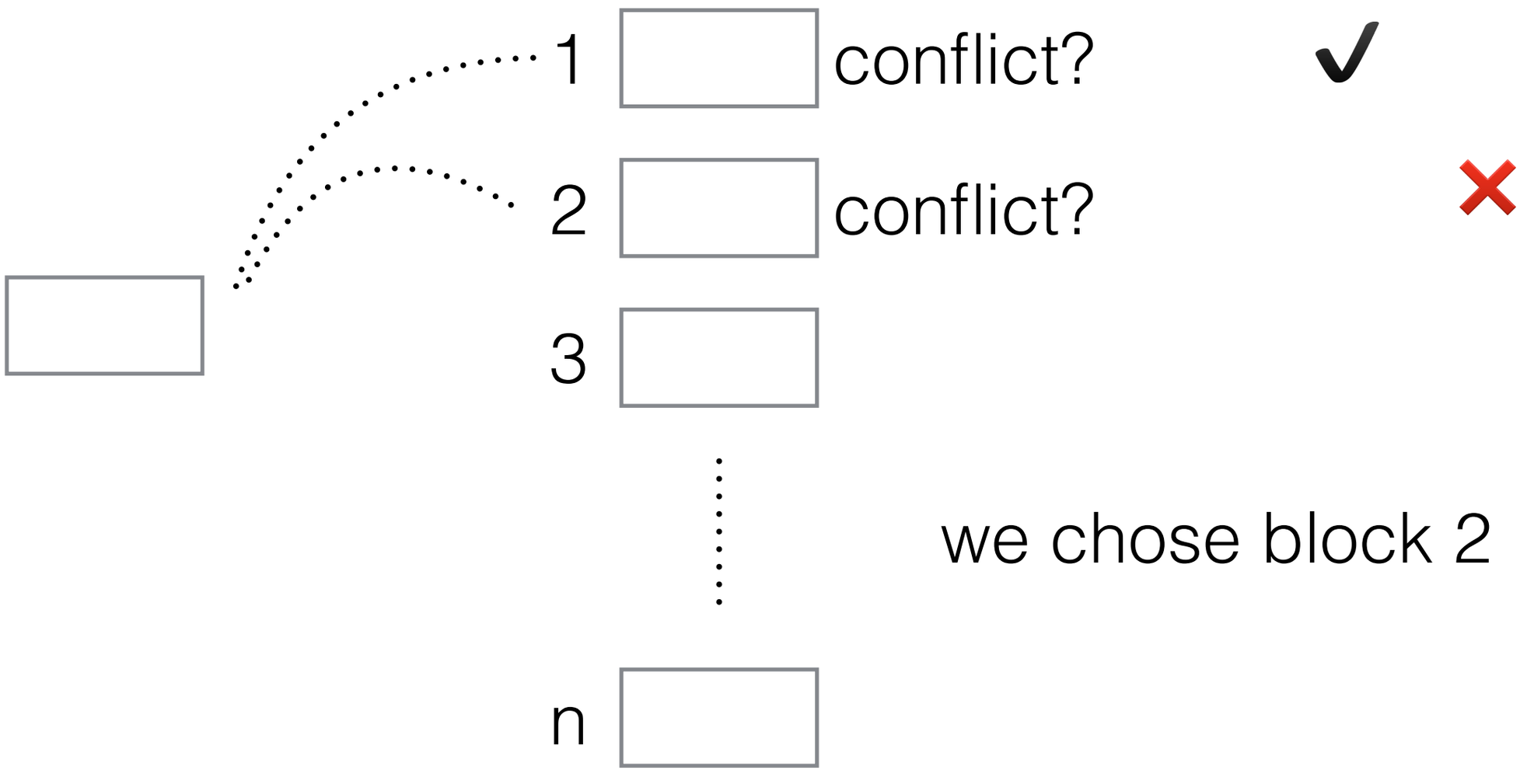
w_{tmr} topology analysis

w_{tr} type reference

w_{dt} document type similarity

Variability-Aware Module Strategy (VMS)

$$w_* = w_{tmr} \uplus w_{tr} \uplus w_{dt}$$



Case Study

- Prevalyer: An open-source object persistence library for Java with 8009 LOC. (5 features)

- MobileMedia: Developed by University of Lancaster with 4653 LOC. (7 features)

Case Study

- ArgoUML: 120KLOC, provides modeling support for UML v1.4 diagrams and supports multiple program- ming languages. (7 features)
- Berkeley DB (in java): 84KLOC, an embedded application to other applications and provides a storage engine. It performs safety transaction and several useful APIs to cope with IO, logging, memory and so forth.(38 features)

Relative Approaches

- ACDC:Algorithm for Comprehension-Driven Clustering (ACDC) recovers the architecture of system by inspecting certain patterns that could exist in systems. Specifically, **ACDC contains *source file pattern*, *body-header pattern*, *leaf collection and support library pattern*, and *ordered and limited subgraph domination***. ACDC identifies clusters by using these patterns with orphan adoption techniques.

Relative Approaches(cond')

- *LIMBO:* scaLable InforMation BOttleneck (LIMBO) optimizes the usage of **information loss** when conducting clustering on a system. It builds on ***Information Bottleneck (IB)*** framework and could collected relevant information during clustering.

Relative Approaches(cond')

- *ARC:* Architecture Recovery using Concern (ARC) uses a generative probabilistic model for text corpora named Latent Dirichlet Allocation(LDA) to retrieve concerns and identify programming elements

Relative Approaches(cond')

- *Bunch:* Bunch regards the recovery task as an **optimization program**. It starts with a random partition and iteratively updates each cluster by optimizing the object function called **Modularization Quality (MQ)** until it cannot find a better solution.

Relative Approaches(cond')

- *W-UE and W-UENM:* Weighted Combined Algorithm (WCA) combines hierarchical clusters into larger sets by **computing the inter distance between two possible variants** using Unbiased Ellenberg (UE) and Unbiased Ellenberg NM(UENM) distance measurement respectively

Metrics- MoJo

- *MoJo Similarity:*

SimilarMoJo metric gives a representation of closeness between two architectures with a percentage. It helps to analyze two different architecture strategies. SimilarMoJo is defined as:

$$SimilarMoJo(A,B) = \left(1 - \frac{MoJo(A,B)}{N}\right) \times 100\%$$

Metrics-MoJo

$$mno(A,B)$$

$$\min\left(mno(A,B), mno(B,A)\right)$$

Number of Join or Move needed to transform from A to B

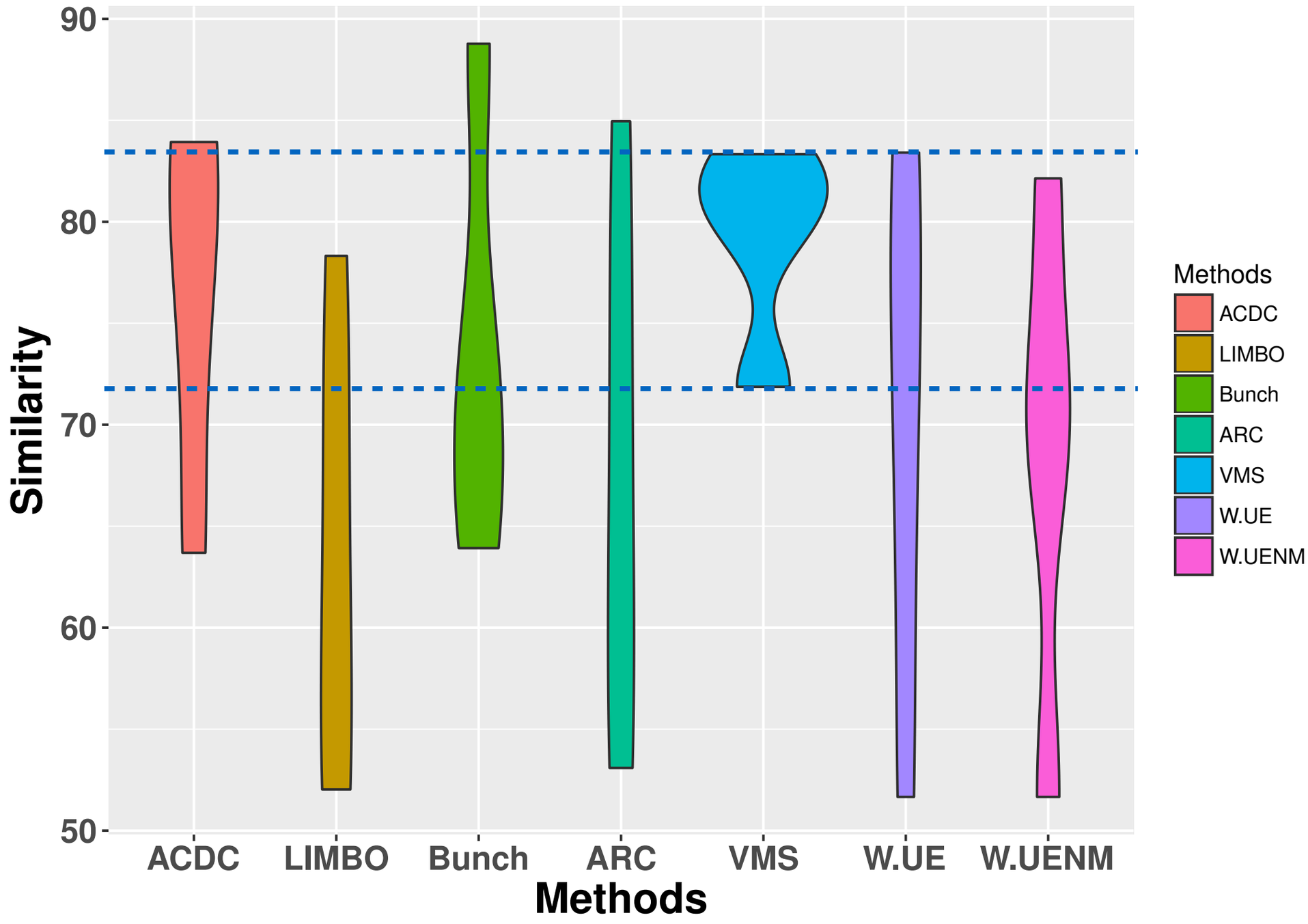
$$SimilarMoJo(A,B) = \left(1 - \frac{MoJo(A,B)}{N}\right) \times 100\%$$

number of units in system

Metrics- MoJo

Algorithm	Prevayler	MobileMedia	ArgoUML	BerkelyDB
ACDC	83.0	75.0	63.69	83.93
LIMBO	55.56	68.75	52.03	78.31
Bunch	74.07	68.42	63.92	88.77
ARC	59.25	73.43	53.09	84.95
VMS	83.33	71.87	79.78	81.62
W-UE	66.67	78.12	51.66	83.41
W-UENM	68.52	71.87	51.66	82.14

Metrics- MoJo



Metrics- A2A

- *Architecture-to-architecture Measurement:*

a2a is developed to overcome the limitation of MoJo to measure the discrepancy of files between recovered result and ground truth.

$$a2a(A,B) = \left(1 - \frac{mto(A_i, A_j)}{aco(A_i) + aco(A_j)}\right) \times 100\%$$

Metrics-A2A

$$mto(A_i, A_j) = remC(A_i, A_j) + addC(A_i, A_j) + remE(A_i, A_j) + addE(A_i, A_j) + mov(A_i, A_j)$$

$$a2a(A, B) = \left(1 - \frac{mto(A_i, A_j)}{aco(A_i) + aco(A_j)}\right) \times 100\%$$

$$aco(A_i) = addC(A, A_i) + addE(A, A_i) + movE(A, A_i)$$

Metrics-A2A

$$mto(A_i, A_j) = remC(A_i, A_j) + addC(A_i, A_j) + remE(A_i, A_j) + addE(A_i, A_j) + mov(A_i, A_j)$$

is the minimum changes from architecture from A_i to A_j

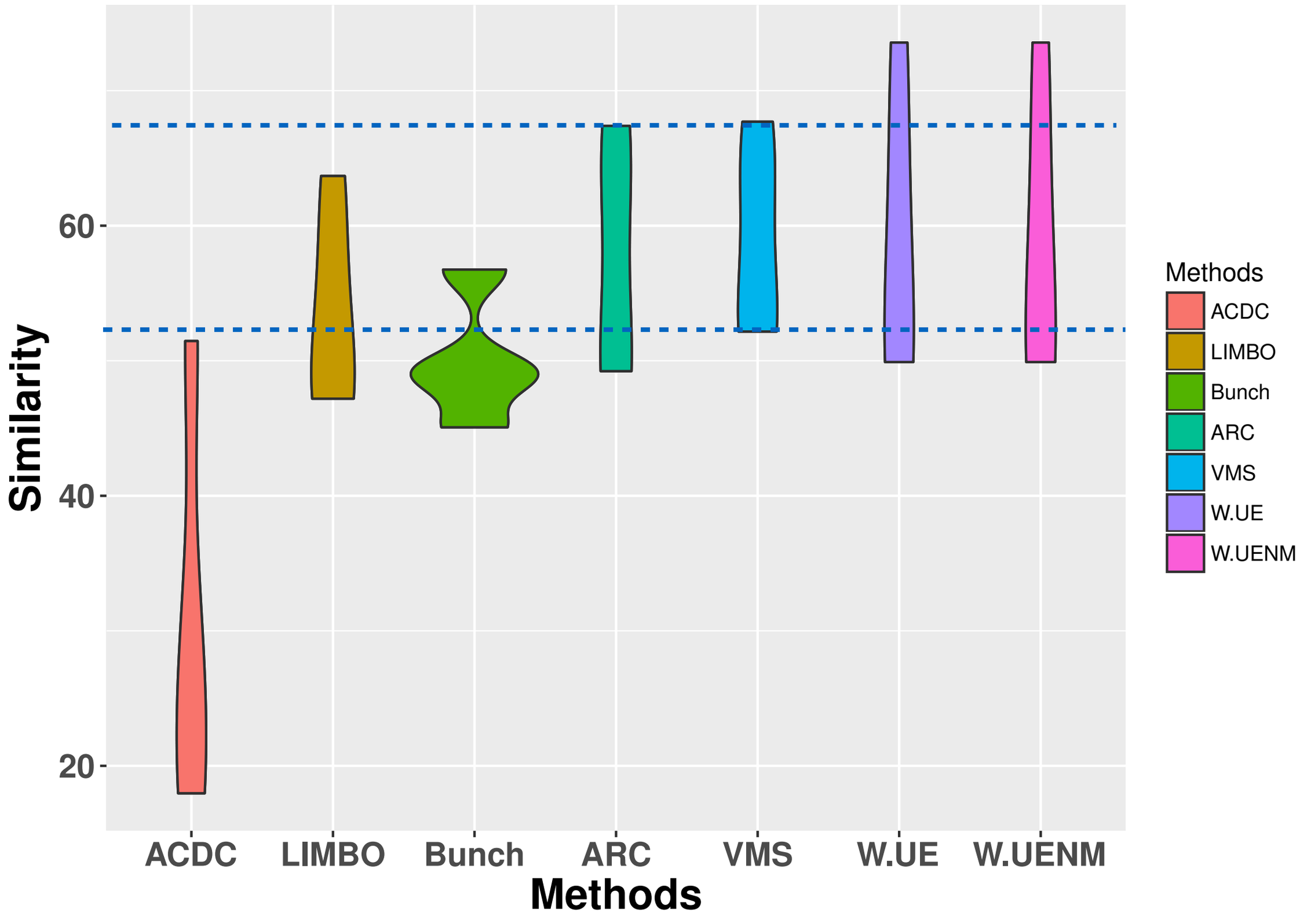
$$aco(A_i) = addC(A, A_i) + addE(A, A_i) + movE(A, A_i)$$

is the minimum changes from architecture from “null” A to A_i

Metrics- A2A

Algorithm	Prevayler	MobileMedia	ArgoUML	BerkelyDB
ACDC	21.18	30.31	51.46	17.97
LIMBO	47.19	63.68	47.67	55.57
Bunch	45.07	56.75	49.06	49.08
ARC	49.23	67.37	49.76	63.48
VMS	52.16	67.70	52.87	62.92
W-UE	50.0	73.56	49.90	62.01
W-UENM	50.0	73.56	49.90	61.64

Metrics- A2A


60
40
20
Similarity
ACDC
LIMBO
Bunch
ARC
VMS
W.UE
W.UENM
Methods
Methods
ACDC
LIMBO
Bunch
ARC
VMS
W.UE
W.UENM

Metrics- C2C

- *Cluster-to-cluster Coverage:*

*c*2c explores component-level accuracy

$$c2c\left(c_i, c_j\right) = \frac{\left|entities\left(c_i\right) \cap entities\left(c_j\right)\right|}{\max\left(\left|entities\left(c_i\right)\right|, \left|entities\left(c_j\right)\right|\right)} \times 100\%$$

entities(c) shows all candidates in the cluster c

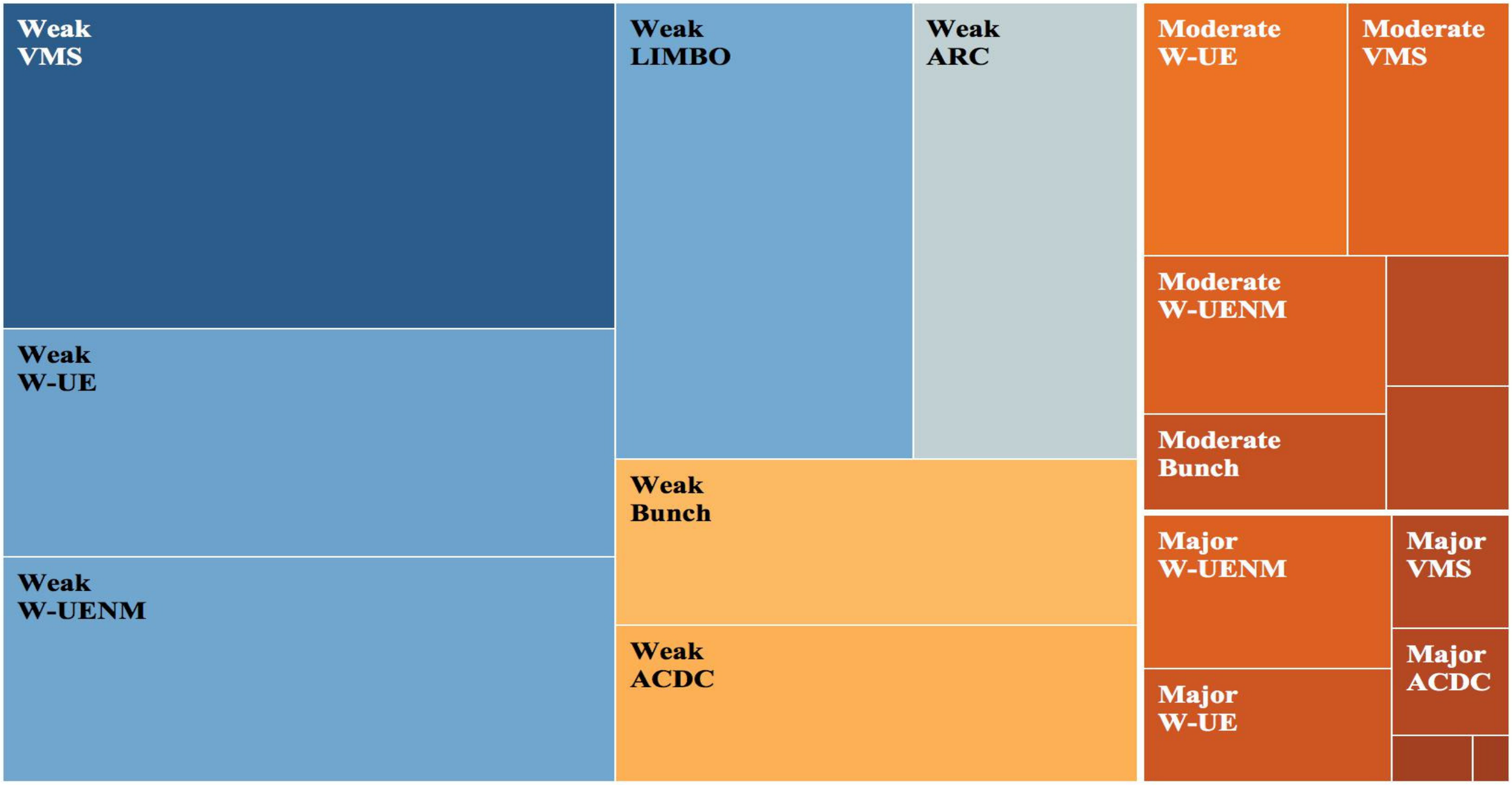
Metrics- C2C

TABLE V

CLUSTER-TO-CLUSTER MEASURING(MAJORITY MATCH(50%), MODERATE MATCH(33%),WEAK MATCH(10%))

Algorithm	Prevayler			MobileMedia			ArgoUML			BereleyDB		
	Major	Mod.	Weak	Major	Mod.	Weak	Major	Mod.	Weak	Major	Mod.	Weak
ACDC	13.64	18.18	54.55	0.00	0.00	30.00	4.55	4.55	22.73	0.00	0.00	7.32
LIMBO	0.00	0.00	80.00	0.00	0.00	28.57	0.00	0.00	66.67	0.00	0.00	14.63
Bunch	5.26	15.79	47.37	0.00	14.29	42.86	0.00	3.03	21.21	0.00	0.00	9.76
ARC	0.00	0.00	20.00	0.00	14.29	57.14	0.00	0.00	22.22	2.38	7.14	45.24
VMS	16.67	16.67	83.33	0.00	14.29	71.43	0.00	22.23	77.78	2.44	4.88	46.34
W-UE	40.00	40.00	60.00	0.00	28.57	71.43	0.00	0.00	11.11	0.00	4.17	54.17
W-UENM	40.00	40.00	60.00	14.29	14.29	71.43	0.00	0.00	11.11	0.00	0.00	50.00

Metrics- C2C


Weak
VMS
Weak
W-UE
Weak
W-UENM
Weak
LIMBO
Weak
ARC
Weak
Bunch
Weak
ACDC
Moderate
W-UE
Moderate
VMS
Moderate
W-UENM
Moderate
Bunch
Major
W-UENM
Major
W-UE
Major
VMS
Major
ACDC
((%)Matching)
0.00
69.72

Metrics-Runtime(millsecond)

In this paper, all algorithms are run on a MacOS 10.12 with Intel i5 2.6GHz, 8G 1600 MHz DDR3, and targeting on Eclipse 4.5 with JRE 7.

Algorithm	Prevayler	MobileMedia	ArgoUML	BerkelyDB
ACDC	665	215	46127	1394
LIMBO	555	165	2629637	1349
Bunch	192	229	25284	258
ARC	2356	3247	139883	6408
VMS	122	313	453278	21847
W-UE	235	67	30309	733
W-UENM	144	50	32054	551

Discussion

*Is **current** architecture recovery technique qualified for constructing feature model?*

*-We can conclude that traditional techniques designed for architecture recovery cannot meet the need of feature model construction. Another apparent limitation for other approaches is that **they cannot ensure all programming elements in a cluster are well-typed which is solved in our approach.** Therefore, our strategy could be a better choice for product line feature model building.*

Discussion(cond')

RQ2: What are the potential limitations for VMS approach?

(1) runtime performance; and

(2) it's still a coarse-granularity approach.

Conclusion

Constructing feature model and modeling variability are promising and worth investigating in product-line oriented re- search. In this paper, we proposed an approach on constructing feature model by investigating variability-aware modules. As our results suggest, traditional methods used in architecture recovery could not reach a stable and competitive performance comparing to our variability-aware approach.

Q & A