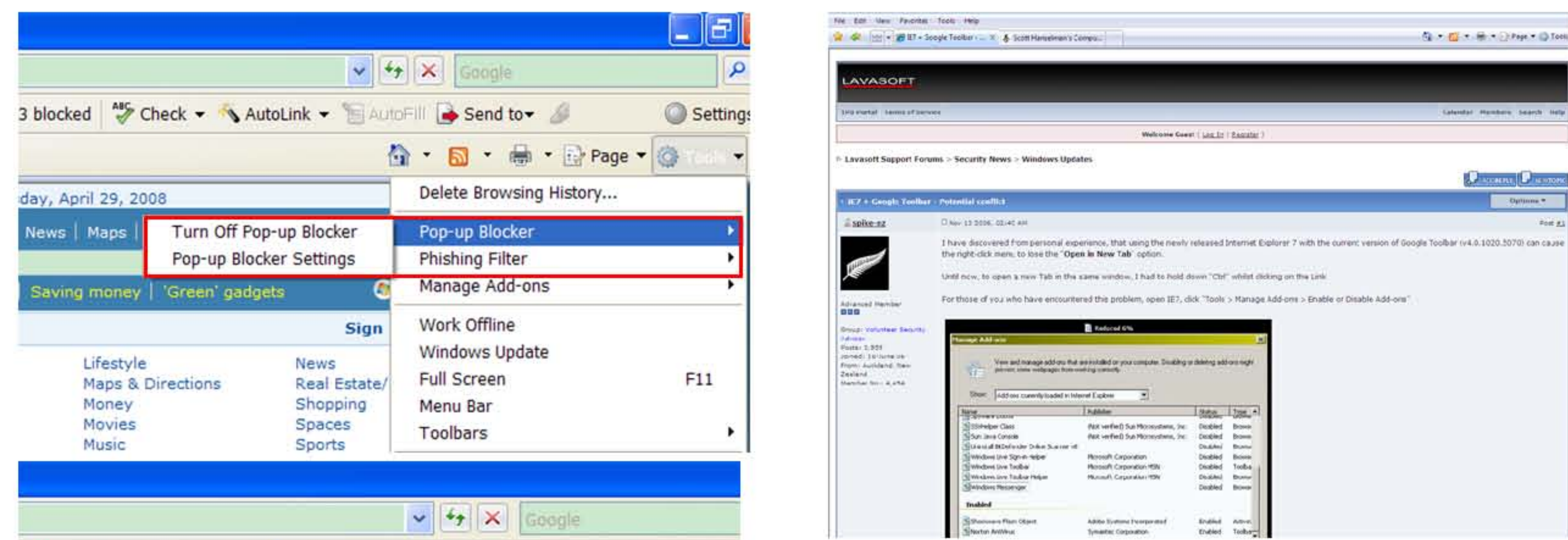


Configuration Aware Prioritization Techniques in Regression Testing

Xiao Qu Advisor: Myra B. Cohen

Computer Science & Engineering • University of Nebraska-Lincoln • {xqu, myra}@cse.unl.edu

Configuration Aware Testing



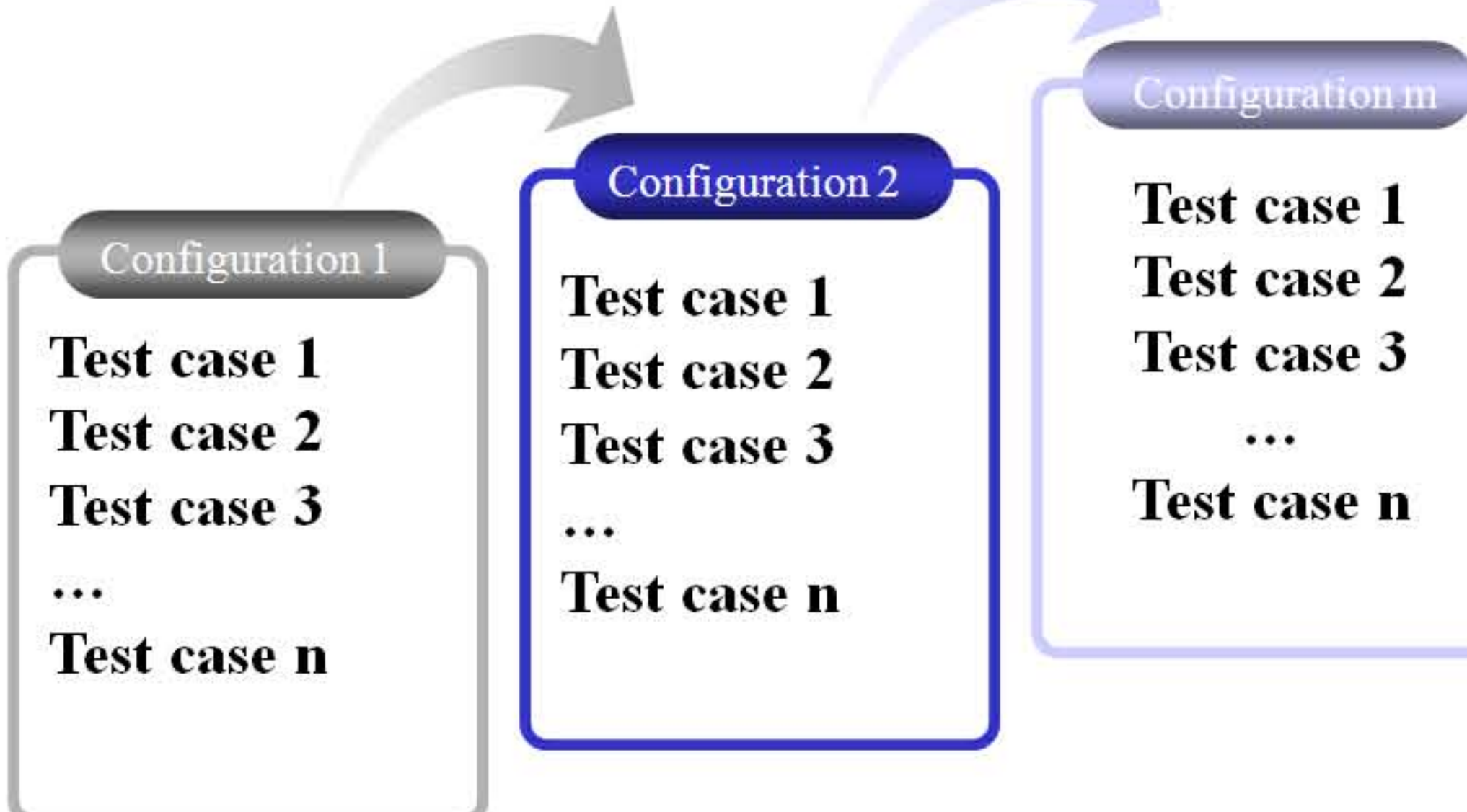
Problem

- Different configurations detect different faults
- May miss faults if we test "wrong" configurations

Solution

- Configuration aware testing
 - Configuration layer
 - ❖ Use **Sampling**
 - Test case layer

Number of configurations: m
Test suite execution time: t
Total time: $m * t$



Configuration Sampling

Sampling Methods

- Impossible to run all configurations because the full configuration space is huge (e.g. gcc optimizer 10^{61})
- Single, Random, Combinatorial Interaction Testing (CIT)

Combinatorial Interaction Testing

Interaction testing is a method of testing configurable systems that provides a natural mechanism for testing systems to be deployed on a variety of hardware and software configurations. It requires that for each t -way combination of factors of a system configuration, every combination of valid values of these factors is covered in at least one configuration. This approach is motivated by the observation that in many applications a significant number of failures are caused by interactions of a small number of factors. It aims at **minimizing the number of configurations** while yielding a high fault/code coverage.

Internet Explorer Configuration

Factors	Filter Level	TIF	Pop-Up	Open Link
Values	High	VPage	IED	NWin
	Medium	VIE	NWin	NTab
	Low	Auto	NTab	CTab

Covers every pair-wise combination of values of each factor

Pair-wise Configuration Samplings

Filter Level	TIF	Pop-Up	Open Link
High	VPage	IED	NWin
Medium	VIE	NWin	NWin
Low	Auto	NTab	NWin
High	VIE	NTab	NTab
Medium	Auto	IED	NTab
Low	VPage	NWin	NTab
High	Auto	NWin	CTab
Medium	VPage	NTab	CTab
Low	VIE	IED	CTab

Configuration Prioritization

Prioritization

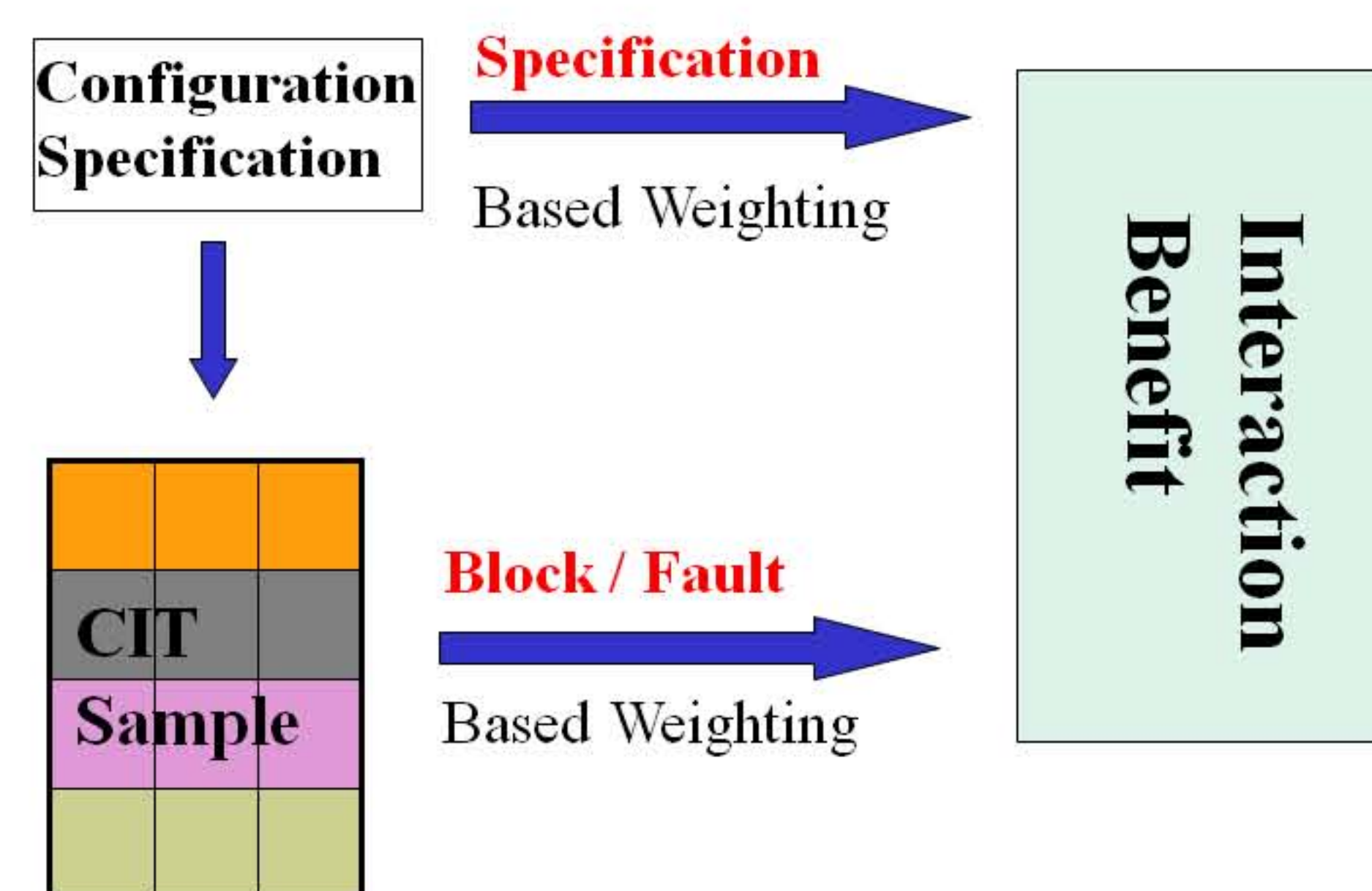
Information

- Specification
- no need for information from prior version
- Block coverage from prior version
- Fault detection from prior version

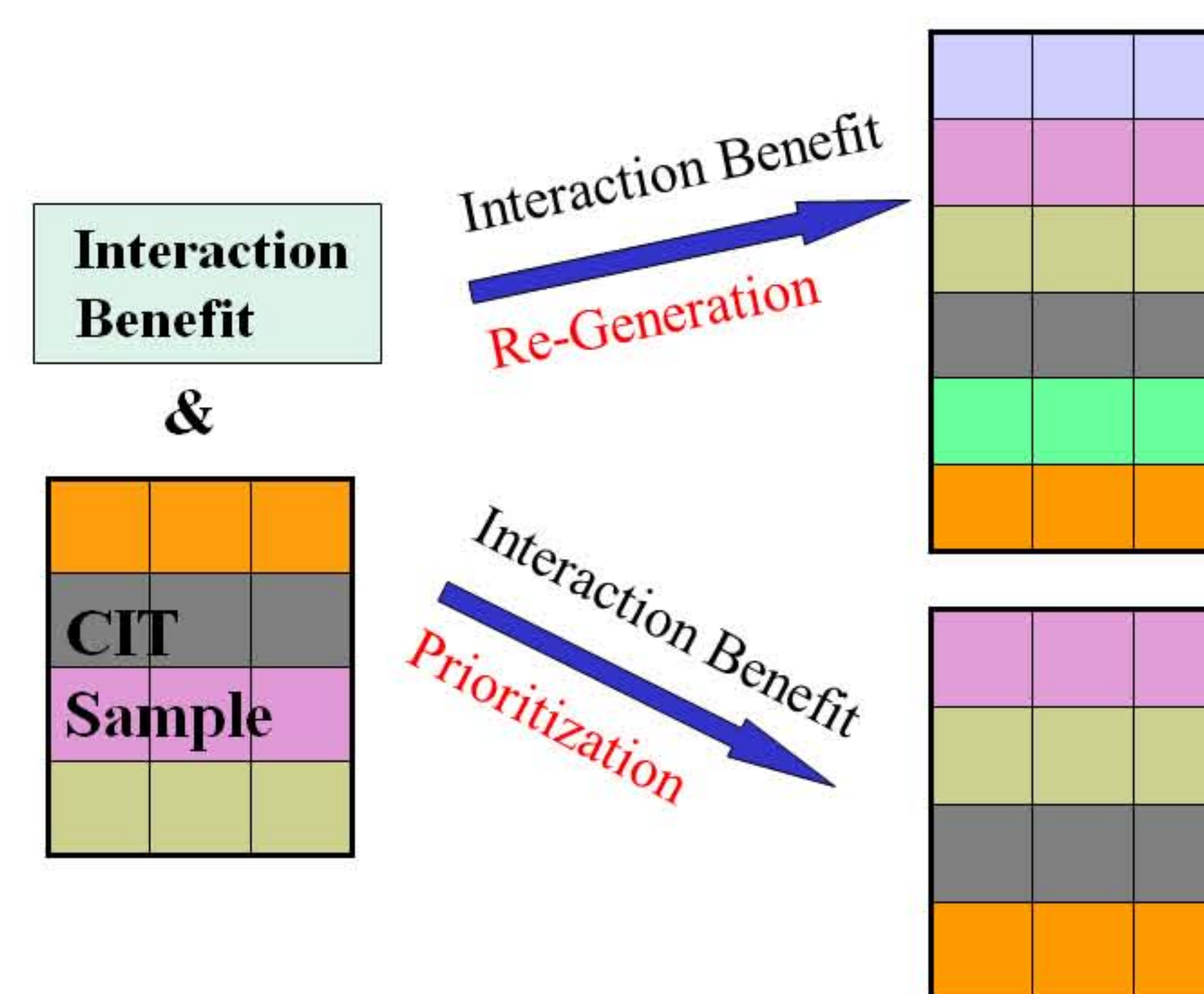
Implementation

- Sampling independent (traditional) prioritization
- Sampling dependent prioritization
 - ❖ CIT Re-generation
 - ❖ CIT Prioritization

Get information for prioritization



Prioritize with the information



Regression Testing and Prioritization

Goal of Regression Testing

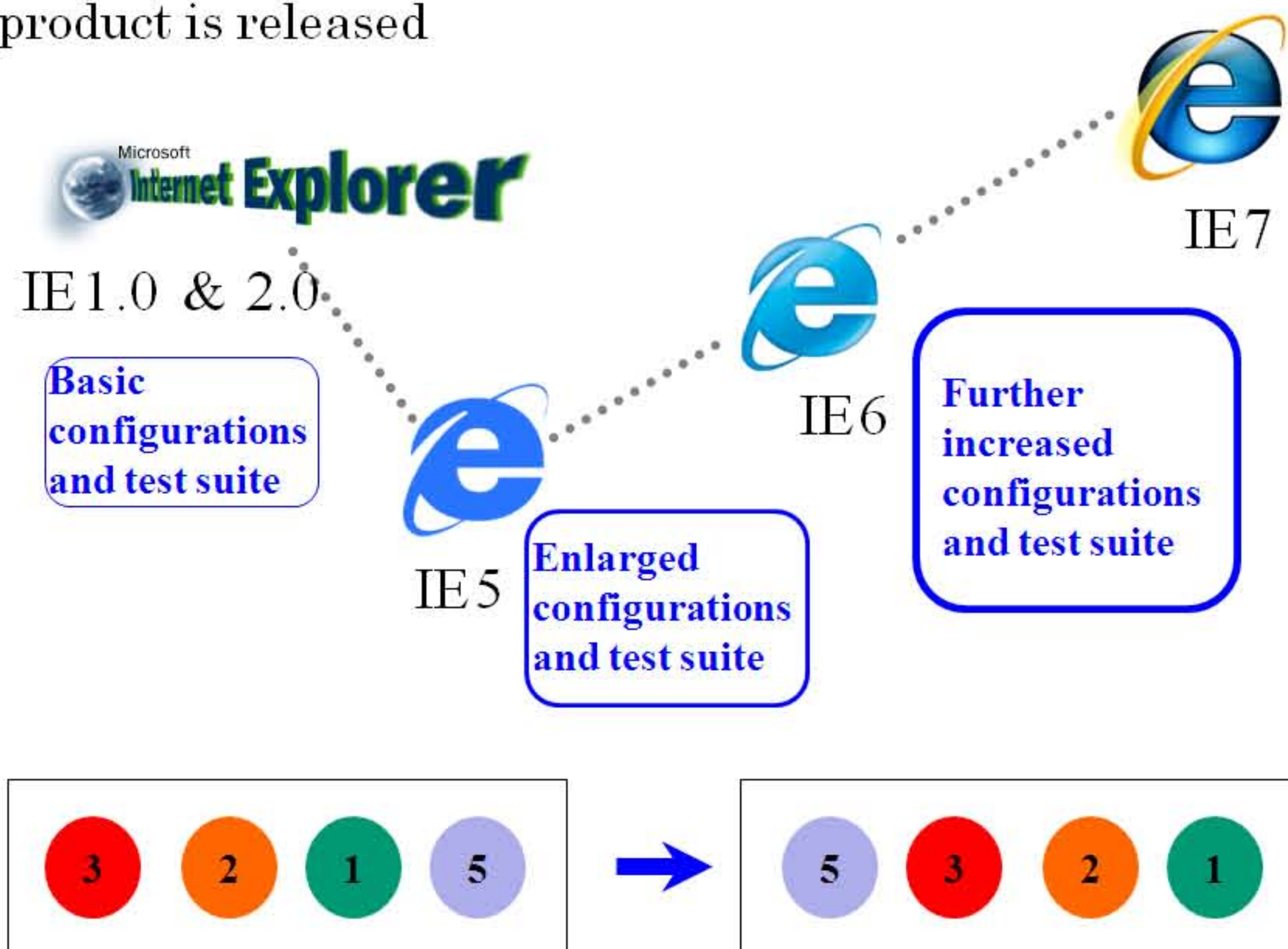
Make sure that modifications do not introduce new faults when each new version of a product is released

Problem

- It is too costly to run even a subset of configurations
- The problem is magnified in regression testing because of increased test cases and configurations due to newly added features / functionality

Solution

- **Prioritization**
- Reorder to run more valuable (i.e. detect more faults) configurations earlier

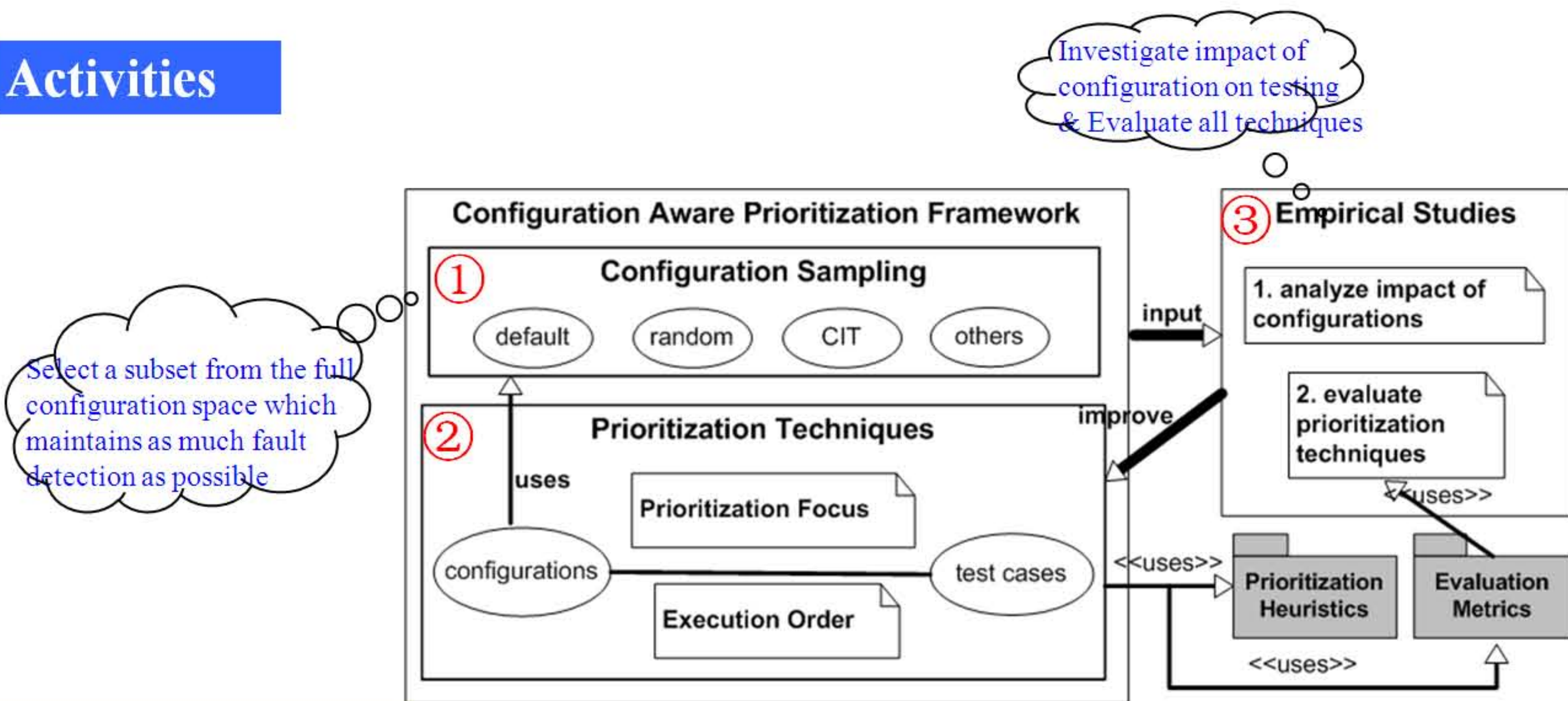


Research Goals & Activities

Goal

To provide cost effective prioritization techniques in regression testing configurable software system.

Activities



Related Work & References

- CIT was created for sampling program input space [D.Cohen *et al.*] and was empirically verified to be effective in many domains of testing [Mandl]
 - Limited to a single version
 - Limited to the test case level
- Prioritization in regression testing: sampling independent [Elbaum *et al.*] & sampling dependent [Bryce & Colbourn]
 - Limited to the test case level
 - Requires test execution information from prior version
 - Sampling dependent technique is not empirically evaluated
- Configuration aware testing: fault localization [Yilmaz *et al.*] and configuration related fault detection [Robinson *et al.*]
 - Limited to a single version

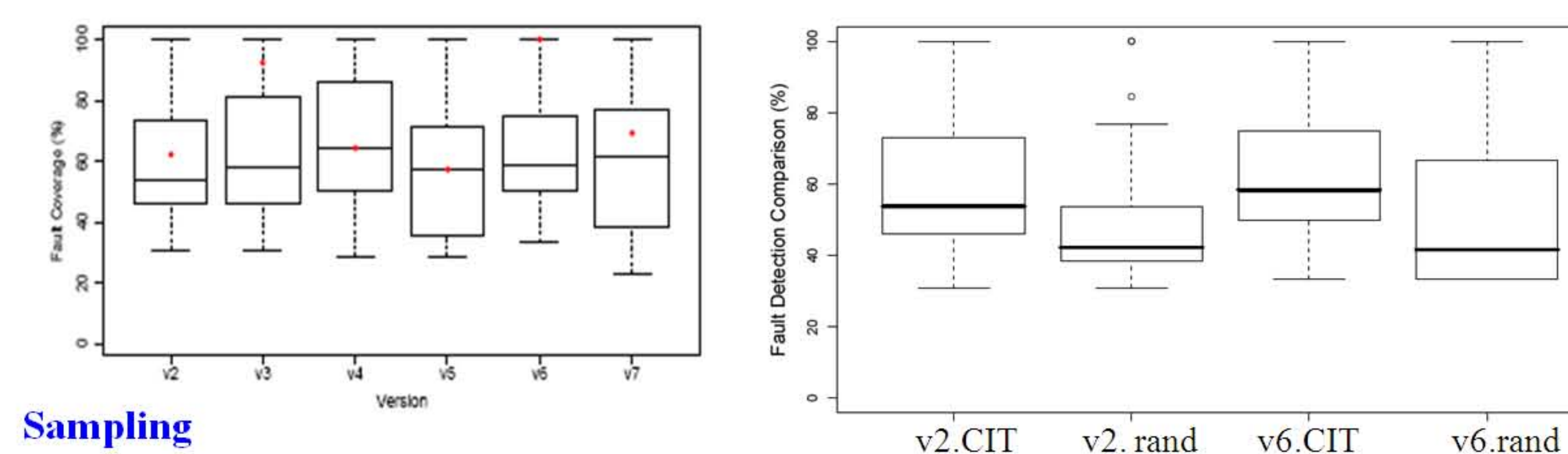
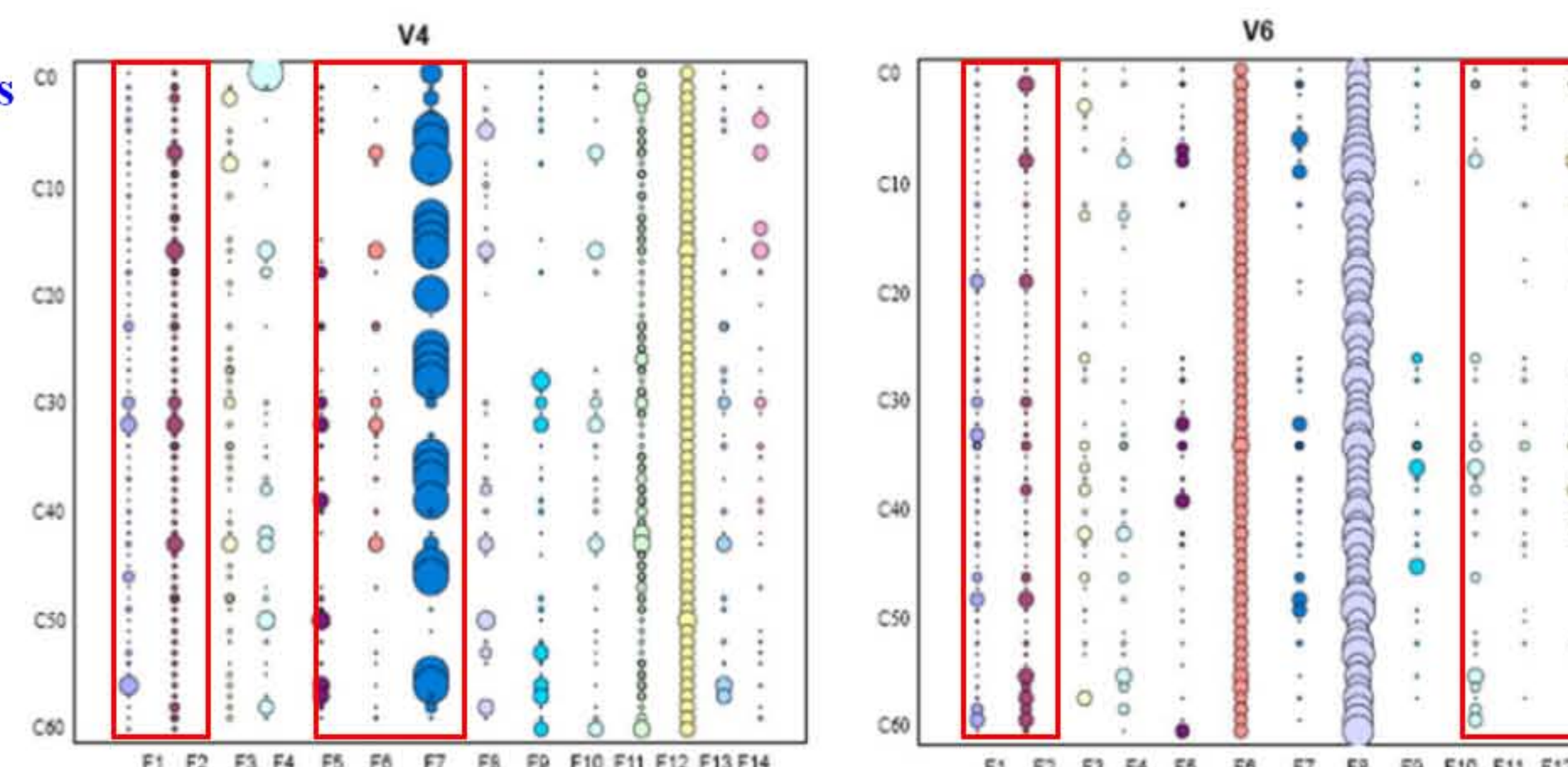
[Bryce & Colbourn] Prioritized interaction testing for pair-wise coverage with seeding and constraints. *Jourl. of Info. and Softw. Tech.*, 48(10):960 - 970, 2006.
 [Elbaum *et al.*] Test case prioritization: a family of empirical studies. *IEEE Trans. Softw. Eng.*, Vol.28, No.2, pp. 159 - 182, Feb. 2002.
 [D.Cohen *et al.*] The AETG system: an approach to testing based on combinatorial design. *IEEE Trans. Softw. Eng.*, 23(7):437-444,1997.
 [Mandl] Orthogonal Latin squares: an application of experiment design to compiler testing. *Commun. ACM* 28, 10 (Oct. 1985), 1054-1058.
 [Yilmaz *et al.*] Covering arrays for efficient fault characterization in complex configuration spaces. *IEEE Trans. Softw. Eng.*, 31(1):20-34, Jan. 2006.
 [Robinson *et al.*] Testing of user-configurable software systems using firewalls. *Int'l. Symp. Softw. Reli. Eng.*, pages 177 - 186, 2008.

Empirical Studies

- Subject: *vim* in SIR (<http://sir.unl.edu/portal/usage.html>)
- Use online documentation along with the *-setall* options within the software to model the system's configuration space
- Seven consecutive versions
- Hand seeded and mutation faults
- Generate 50 of each technique and use the average result

Impact of configurations

- Many faults are dependent on configurations
- Different configurations detect different faults

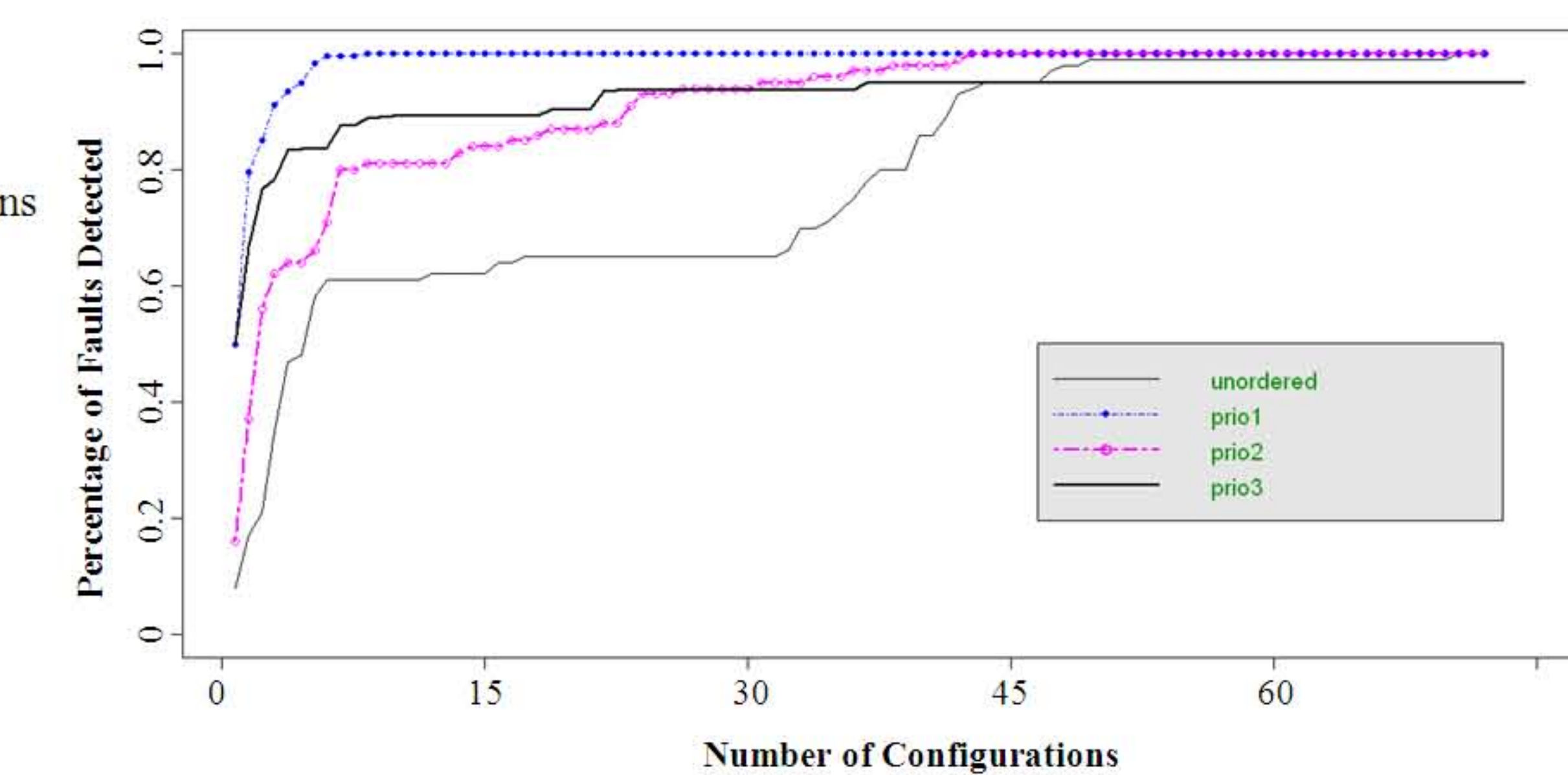


Sampling

- The median fault finding ability of the CIT sample is similar to that of the default configuration
- CIT sampled configurations detect more faults than random configurations

Prioritization

- All prioritized configurations detect faults earlier than unordered configurations
- Specification based prioritization technique is effective: no need for prior testing



Evaluation Metric

- NAPFD
 - Measurement of the area
 - The larger the value, the more effective the prioritization is

version	unordered	pure-FD	p-BC	p-SPEC	regen-FD	r-BC	r-SPEC
NAPFD value (budget: 5/60)							
v3	0.76	0.78	0.78	0.81	0.90	0.90	0.90
v4	0.76	0.90	0.90	0.90	0.90	0.90	0.90
NAPFD value (budget: 10/60)							
... ..							
NAPFD value (budget: 15/60)							
v3	0.92	0.93	0.93	0.94	0.97	0.97	0.97
v4	0.92	0.97	0.97	0.97	0.97	0.97	0.97

Ongoing Work

- Current prioritization techniques are applied at the test case and configuration level independently. Since both have exhibited early fault detection, we expect that a combination of them will yield better performance. We are working on a cost effective prioritization technique model, by incorporating test case prioritization with configuration prioritization, and empirically evaluating them in two configurable software systems.
- Faults appearing in different fragments of code may have different severities, due to several possible reasons: particular fragments of code are more frequently called by users; or particular fragments of code control more important features / functionality of the program, etc. Profiling data may provide information of these factors. We are going to prioritize taking fault severity into consideration.
- Our prioritization techniques to date assume that every test case or configuration has the same cost. This is not always the case in practice. We are going to develop a cost aware prioritization technique.

This work is funded in part by NSF award CCF-0747009 & AFOSR award FA9550-09-1-0129