IWCT 2016 (ICST 2016 Workshop), Chicago, Apr 10, 2016

Pairwise Coverage-based Testing with Selected Elements in a Query for Database Applications

Koji Tsumura* **Hironori Washizaki*** Yoshiaki Fukazawa*,
Keishi Oshima† Ryota Mibe†

*Waseda University, Tokyo, Japan

†Hitachi, Ltd., Japan

```
// Get only voided=false items
public List<Item> getUnvoidedItems( ) {
  List<Item> is = getItems( );
  List<Item> ret = new ArrayList<Item>();
  for (Item i : is) {
    // Fault：forget to check voided=false
    ret.add(i);
  }
  return ret;
}
```

```
SELECT
  id,
  name,
  voided,
  price
FROM items
WHERE
  category = 'food';
```

```
// Get only voided=false items
public List<Item> getUnvoidedItems( ) {
  List<Item> is = getItems( );
  List<Item> ret = new ArrayList<Item>();
  for (Item i : is) {
    // Fault：forget to check voided=false
    ret.add(i);
  }
  return ret;
}
```

```
SELECT
  id,
  name,
  voided,
  price
FROM items
WHERE
  category = 'food';
```

id	name	category	voided	price
1	'ABC'	'food'	false	200

100% statement coverage!
But failed to reveal fault.

```
// Get only voided=false items
public List<Item> getUnvoidedItems( ) {
  List<Item> is = getItems( );
  List<Item> ret = new ArrayList<Item>();
  for (Item i : is) {
    // Fault：forget to check voided=false
    ret.add(i);
  }
  return ret;
}
```

```
SELECT
  id,
  name,
  voided,
  price
FROM items
WHERE
  category = 'food';
```

id	name	category	voided	price
1	'ABC'	**'food'**	false	200
2	'DEF'	'clothe'	false	100

100% decision coverage !
But failed to reveal fault.

```
// Get only voided=false items
public List<Item> getUnvoidedItems( ) {
  List<Item> is = getItems( );
  List<Item> ret = new ArrayList<Item>();
  for (Item i : is) {
    // Fault：forget to check voided=false
    ret.add(i);
  }
  return ret;
}
```

```
SELECT
  id,
  name,
  voided,
  price
FROM items
WHERE
  category = 'food';
```

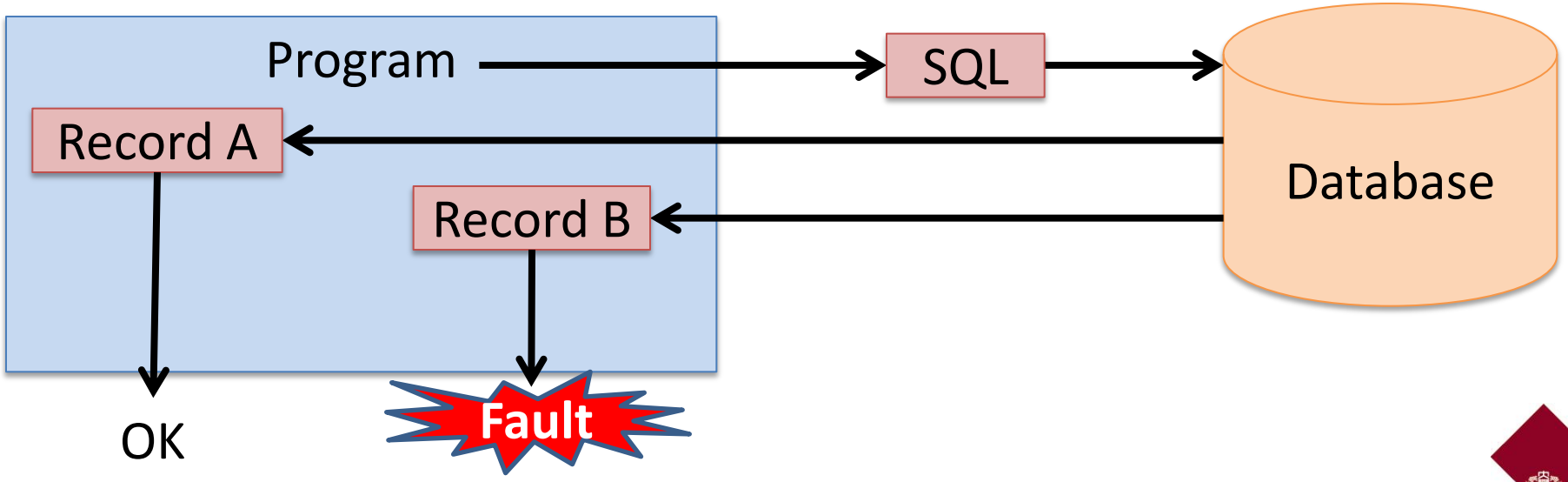
id	name	category	voided	price
1	'ABC'	**'food'**	false	200
2	'DEF'	'clothe'	false	100
3	'GHI'	**'food'**	true	100
4	'JKL'	'clothe'	true	200

100% pairwise coverage for columns { category, voided, price }.

← This reveals the fault!

Testing Database Applications

- Expected behaviors depend on the records obtained by SQL
  - E.g. Some specific items are hidden, …
- We should test them focusing on not only source code, but also SQL
  - Plain Pairwise Coverage (PPC)
  - Selected Pairwise Coverage (SPC)



Plain Pairwise Coverage (PPC)

SELECT Query

```
SELECT
  id, category, voided,
price
FROM  item
WHERE  category =
'food';
```

Test Parameters

Table	Column	Equivalence Classes
item	voided	true, false
	category	'food', 'clothe'

Coverage Criteria

Pair		category	
		'food'	'clothe'
voided	true		
	false		

Plain Pairwise Coverage (PPC)

SELECT Query

```
SELECT
  id, category, voided,
price
FROM  item
WHERE  category =
'food';
```

Test Parameters

Table	Column	Equivalence Classes
item	voided	true, false
	category	'food', 'clothe'

Coverage Criteria

Pair		category	
		'food'	'clothe'
voided	true	×	×
	false	○	×

Query Results

id	category	voided	price
5	'food'	false	100
9	'food'	false	200

PPC = 1 / 4 = 25%

Selected Pairwise Coverage (SPC)

SELECT Query

```
SELECT
  id, category, voided,
price
FROM  item
WHERE  category =
'food';
```

Test Parameters

Table	Column	Equivalence Classes
item	voided	true, false
	category	'food', 'clothe'

Coverage Criteria

Pair		category	
		'food'	'clothe'
voided	true		
	false		

Selected Pairwise Coverage (SPC)

SELECT Query

```
SELECT
  id, category, voided,
price
FROM  item
WHERE  category =
'food';
```

Test Parameters

Table	Column	Equivalence Classes
item	voided	true, false
	category	'food', 'clothe'

Coverage Criteria

Pair		category	
		'food'	'clothe'
voided	true		
	false		

Selected Pairwise Coverage (SPC)

SELECT Query

```
SELECT
  id, category, voided,
price
FROM  item
WHERE  category =
'food';
```

Test Parameters

Table	Column	Equivalence Classes
item	voided	true, false
	category	'food', 'clothe'

- Targeting only selected columns
- Eliminate impossible combination by checking WHERE

Coverage Criteria

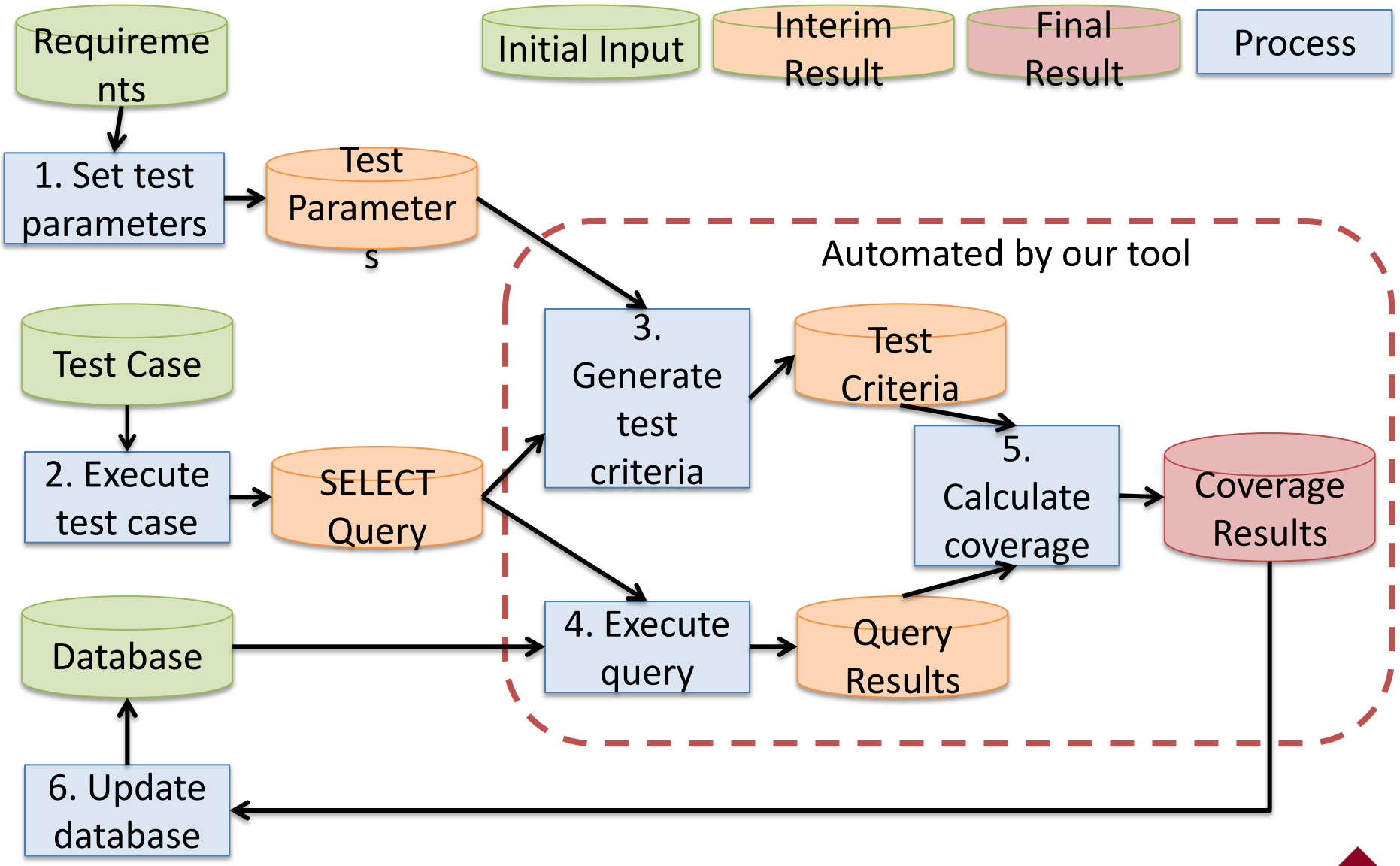
Pair		category	
		'food'	'clothe'
voided	true	×	
	false	○	

Query Results

id	category	voided	price
5	'food'	false	100
9	'food'	false	200

SPC = 1 / 2 = 50%

SPC/PPC Testing Framework (PPC/SPCT)



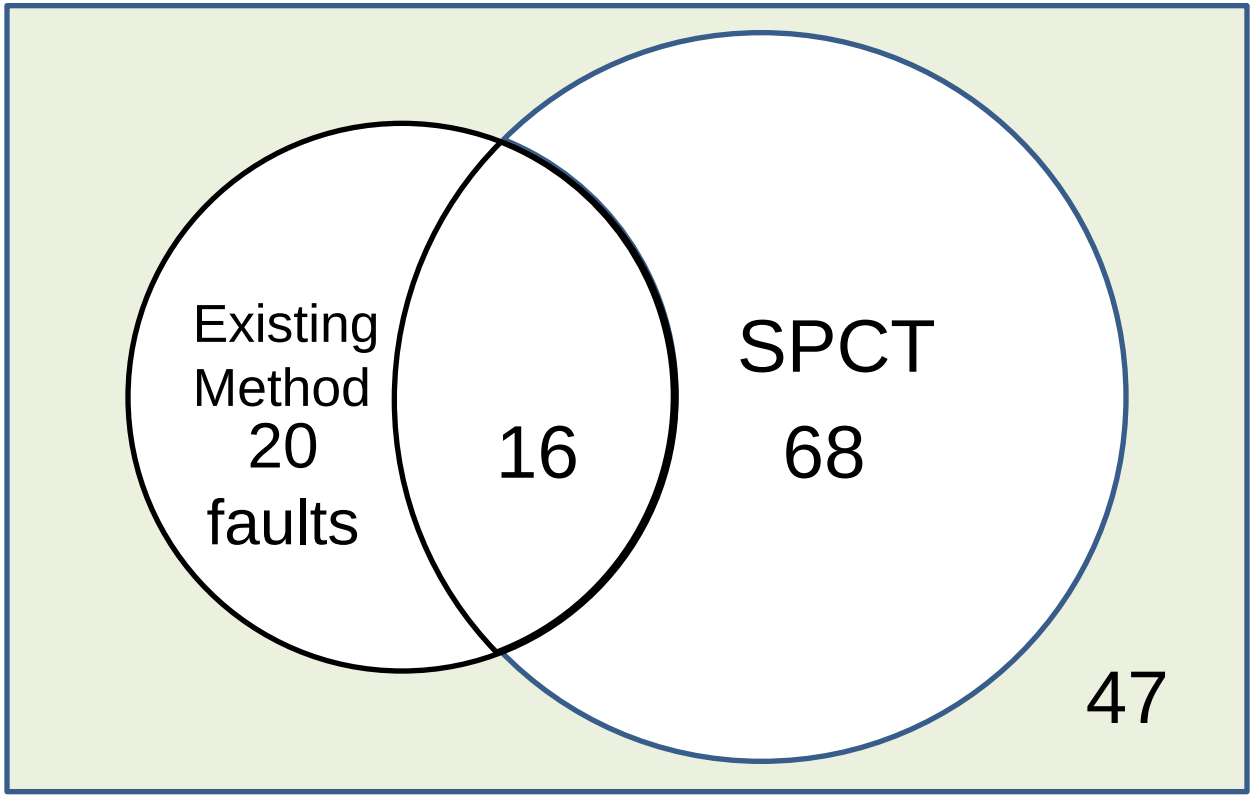
Case Study

- RQ1：Can SPCT detect faults that existing testing method cannot?
  - Fault detection analysis
- RQ2：How many records are needed to reach 100% SPC coverage?
  - Cost calculation
- Target: two OSSs
  - OpenMRS[2], Broadleaf Commerce（BLC）[3]
  - 100 tables, 1000 columns
- Comparative existing method
  - Full predicate coverage: MC/DC coverage for SQL queries [1]

[1] Tuya et al. “Full predicate coverage for testing SQL database queries”（STVR 2010）
[2] OpenMRS（http://openmrs.org/）
[3] Broadleaf Commerce（http://www.broadleafcommerce.org/）

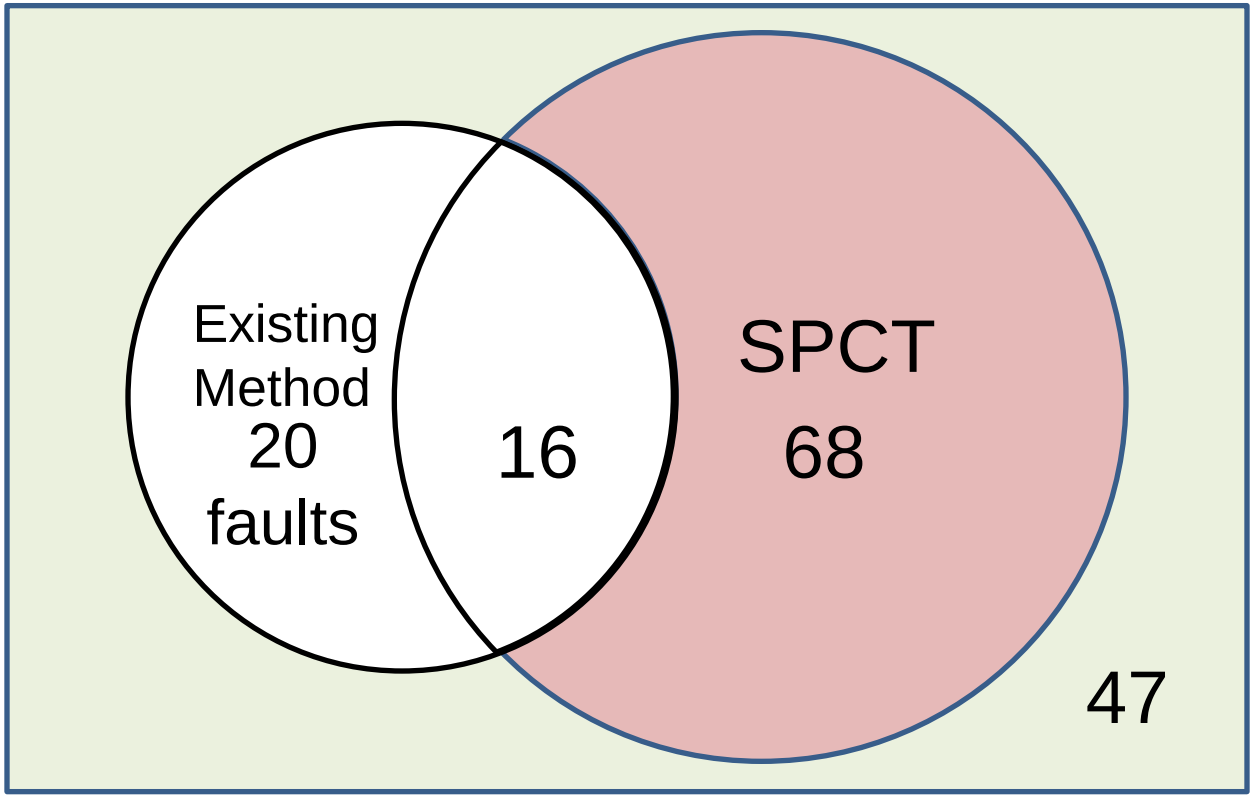
RQ1. Can SPCT detect faults that existing method cannot?

- Fault Detection Analysis



RQ1. Can SPCT detect faults that existing method cannot?

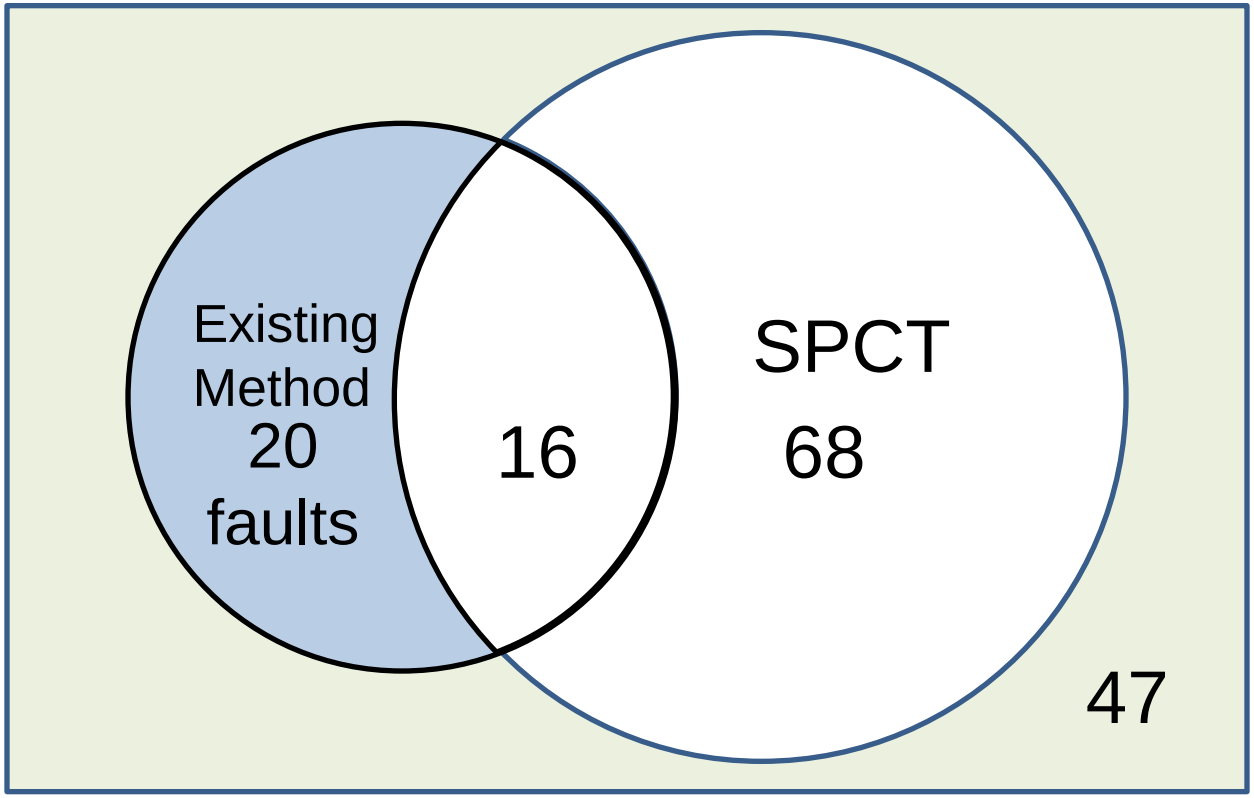
- Fault Detection Analysis



SPCT can detect 68 faults of 115 faults, which are not detected by the existing method.

RQ1. Can SPCT detect faults that existing method cannot?

- Fault Detection Analysis



SPCT can detect 68 faults of 115 faults, which are not detected by the existing method.

However, 20 faults are detected only by the existing method.

RQ2 : How many records to reach 100% SPC coverage?

- Cost Calculation

Target	Number of Records			Cost Ratio (%)	
	Existing	PPCT	SPCT	SPCT/Existing	SPCT/PPCT
OpenMRS	529	1506	1300	245.75	86.32
BLC	405	1341	1331	328.64	99.25
Total	934	2847	2631	281.69	92.41

RQ2 : How many records to reach 100% SPC coverage?

- Cost Calculation

Target	Number of Records			Cost Ratio (%)	
	Existing	PPCT	SPCT	SPCT/Existing	SPCT/PPCT
OpenMRS	529	1506	1300	**245.75**	86.32
BLC	405	1341	1331	**328.64**	99.25
Total	934	2847	2631	**281.69**	92.41

SPCT requires 2-3 times more records than the existing method.

RQ2 : How many records to reach 100% SPC coverage?

- Cost Calculation

Target	Number of Records			Cost Ratio (%)	
	Existing	PPCT	SPCT	SPCT/Existing	SPCT/PPCT
OpenMRS	529	1506	1300	**245.75**	86.32
BLC	405	1341	1331	**328.64**	99.25
Total	934	2847	2631	**281.69**	92.41

SPCT requires 2-3 times more records than the existing method.

However, SPCT requires fewer records (0.87 times) than PPCT.

RQ2 : How many records to reach 100% SPC coverage?

- Cost Calculation

Target	Number of Records			Cost Ratio (%)	
	Existing	PPCT	SPCT	SPCT/Existing	SPCT/PPCT
OpenMRS	529	1506	1300	245.75	86.32
BLC	405	1341	1331	328.64	99.25
Total	934	2847	2631	281.69	92.41

- Tables are divided into more small units
  - BLC joins more tables in a query
- SPCT ignores redundant pairs by focusing only on WHERE clause. However, SPCT does not focus on the conditions in JOIN clause.
  - SPCT can ignore more redundant pairs by focusing on JOIN clause.

Conclusion

- Contributions
  - We proposed new testing coverage PPC and SPC and method SPCT for database applications
    - by focusing on the elements in SELECT queries
  - We implemented a tool to calculate PPC and SPC
  - We applied SPCT and PPCT to two open source software
    - PPCT and SPCT can detect 68.54(%) of faults that existing testing method cannot.
- Future Work
  - Ignore more redundant pairs for efficiency
  - Automation (e.g. Test Parameter Setting)

10th IEEE International Conference on Software Testing, Verification and Validation

Mar 13-18 (due Sep 2016)

aster.or.jp/conference/icst2017/