SENSA: Sensitivity Analysis for Quantitative Change-impact Prediction

Haipeng Cai♦ Siyuan Jiang ♦ Raul Santelices ♦
Ying-jie Zhang* Yiji Zhang ♦

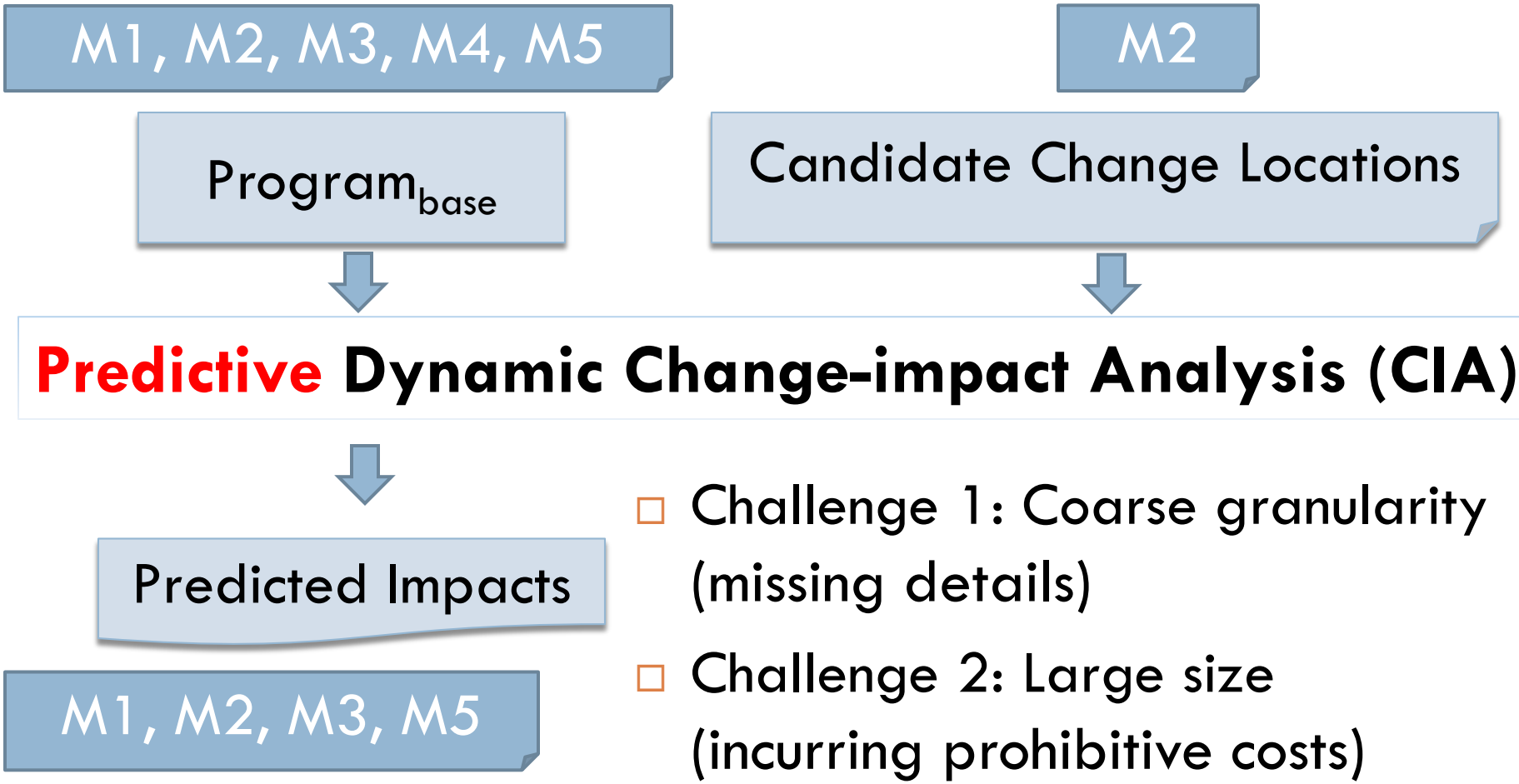
♦ University of Notre Dame, USA
* Tsinghua University, China

UNIVERSITY OF NOTRE DAME
College of Engineering

SCAM 2014

Supported by ONR Award N000141410037

What we do



- Challenge 1: Coarse granularity (missing details)
- Challenge 2: Large size (incurring prohibitive costs)

What we do

M1, M2, M3, M4, M5

M2

$Program_{base}$

Candidate Change Locations

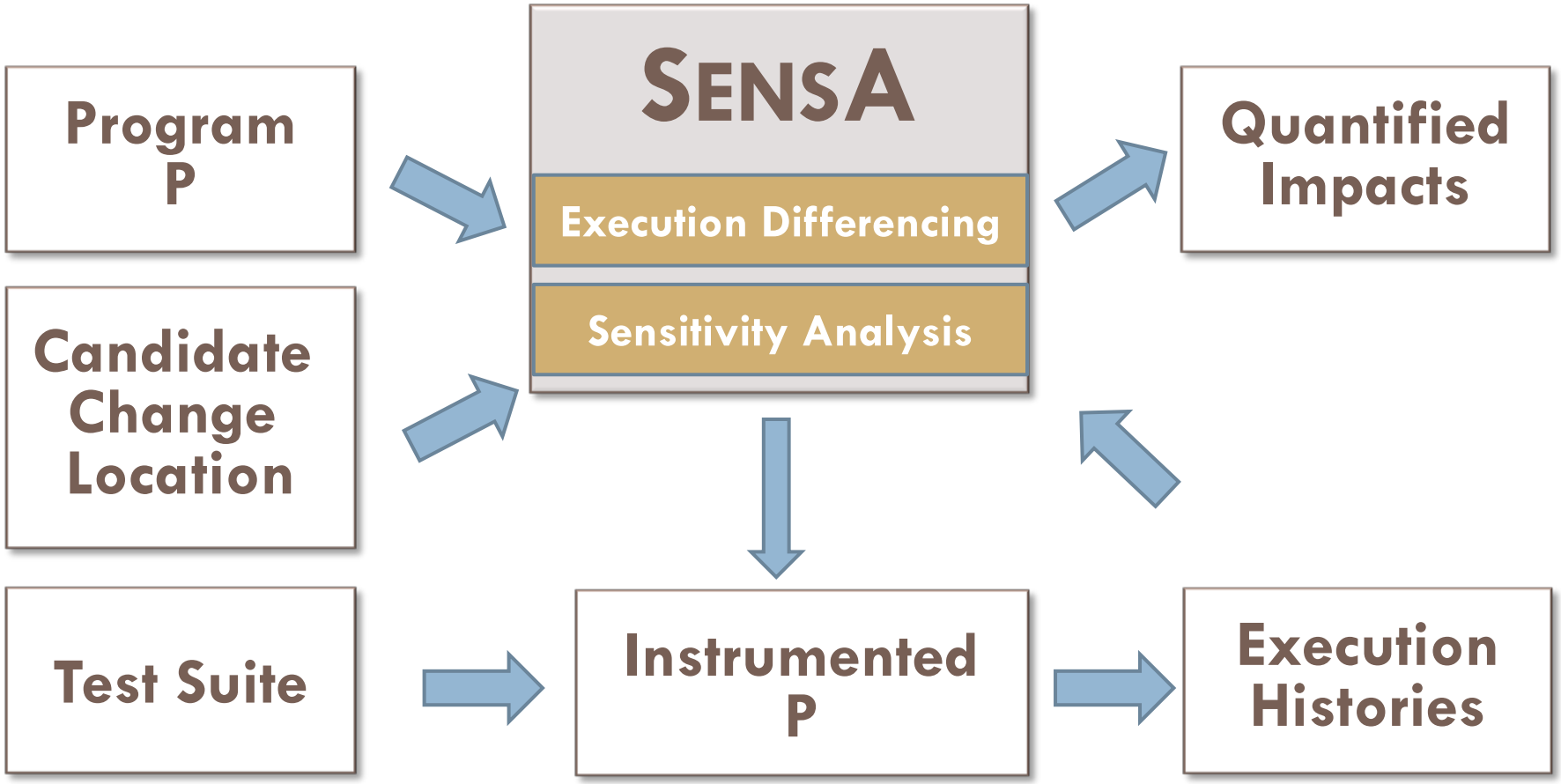
Predictive Dynamic Change-impact Analysis (CIA)

Predicted Impacts

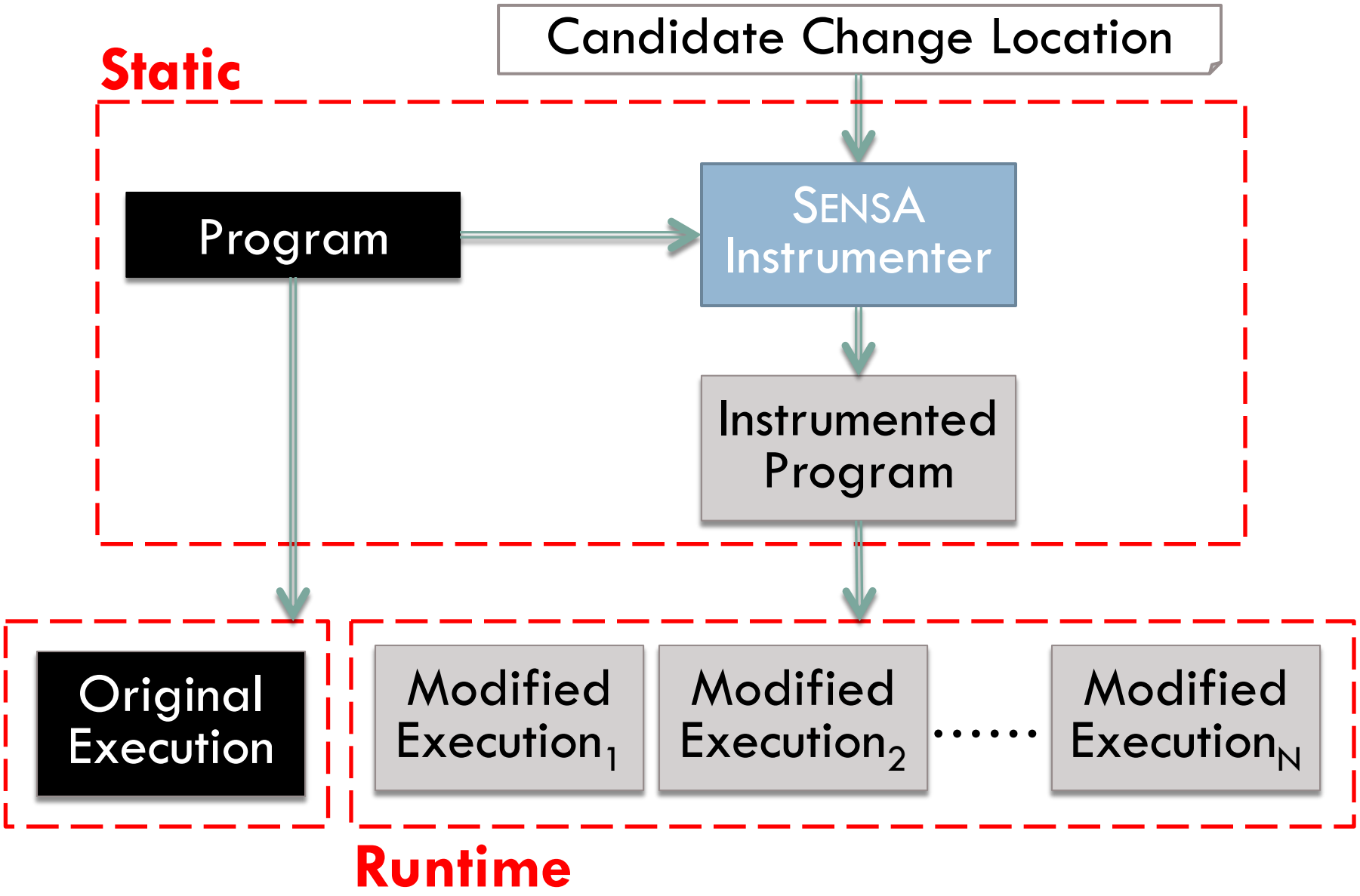
M1, M2, M3, M5

- Challenge 1: Coarse granularity
  - Solution: **statement-level** analysis
- Challenge 2: Large size
  - Solution: **prioritize** change impacts

Technique: overview



Technique: sensitivity analysis



Technique: execution differencing


```
1 public class A {
2     static int M1(int f, int z) {
3         M2(f+z);
4         return new B().M3(f,1); }
5     void M2(int m) {
6         if (m > 0)      // change
7             C.M5(); }}
```

```
8 public class B {
9     public static int t=0;
10    int M3(int a, int b) {
11        int n = b*b - a;
12        return n; }
13    static void M4() {
14        t = 10; }}
```

```
15 public class C {
16     public static boolean M5() {
17         return B.t > 10; }
18
19     public static void M0() {
20         if (A.M1(4,-3) > 0)
21             B.M4(); }}
```

Original Execution

Modified Execution

Execution History

Statement	Value
20	False
6	True
11	-3
12	-3
7	
17	False
4	-3

Results (statements):
6
7
17

Statement	Value
20	False
6	False
11	-3
12	-3
-	
-	
4	-3

Impact set (for statement 6)

How SENSA works


```
1 public class A {
2     static int M1(int f, int z) {
3         M2(f+z);
4         return new B().M3(f,1); }
5     void M2(int m) {
6         if (m > 0)      //change
7             C.M5(); }}
```

```
8 public class B {
9     public static int t=0;
10    int M3(int a, int b) {
11        int n = b*b - a;
12        return n; }
13    static void M4() {
14        t = 10; }}
```

```
15 public class C {
16     public static boolean M5() {
17         return B.t > 10; }
18
19     public static void M0() {
20         if (A.M1(4,-3) > 0)
21             B.M4(); }}
```

Original Execution

Multiple Modified Executions

Statement	Impact Frequency
6	1
7	1
17	1
2	0
:	0
21	0

Only one modified execution for this example

Execution Differencing

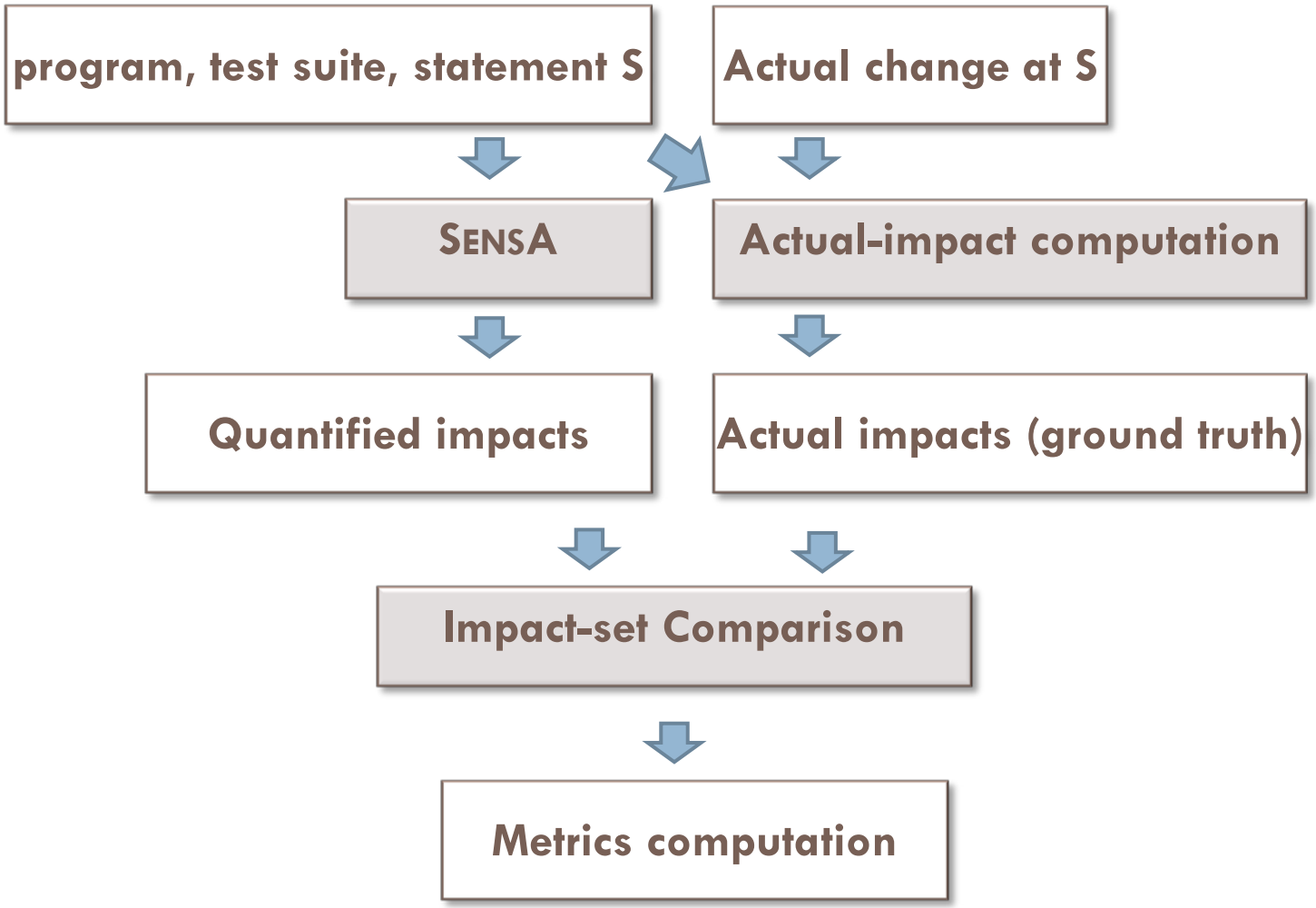
Impact Quantification

Quantified Impact Set

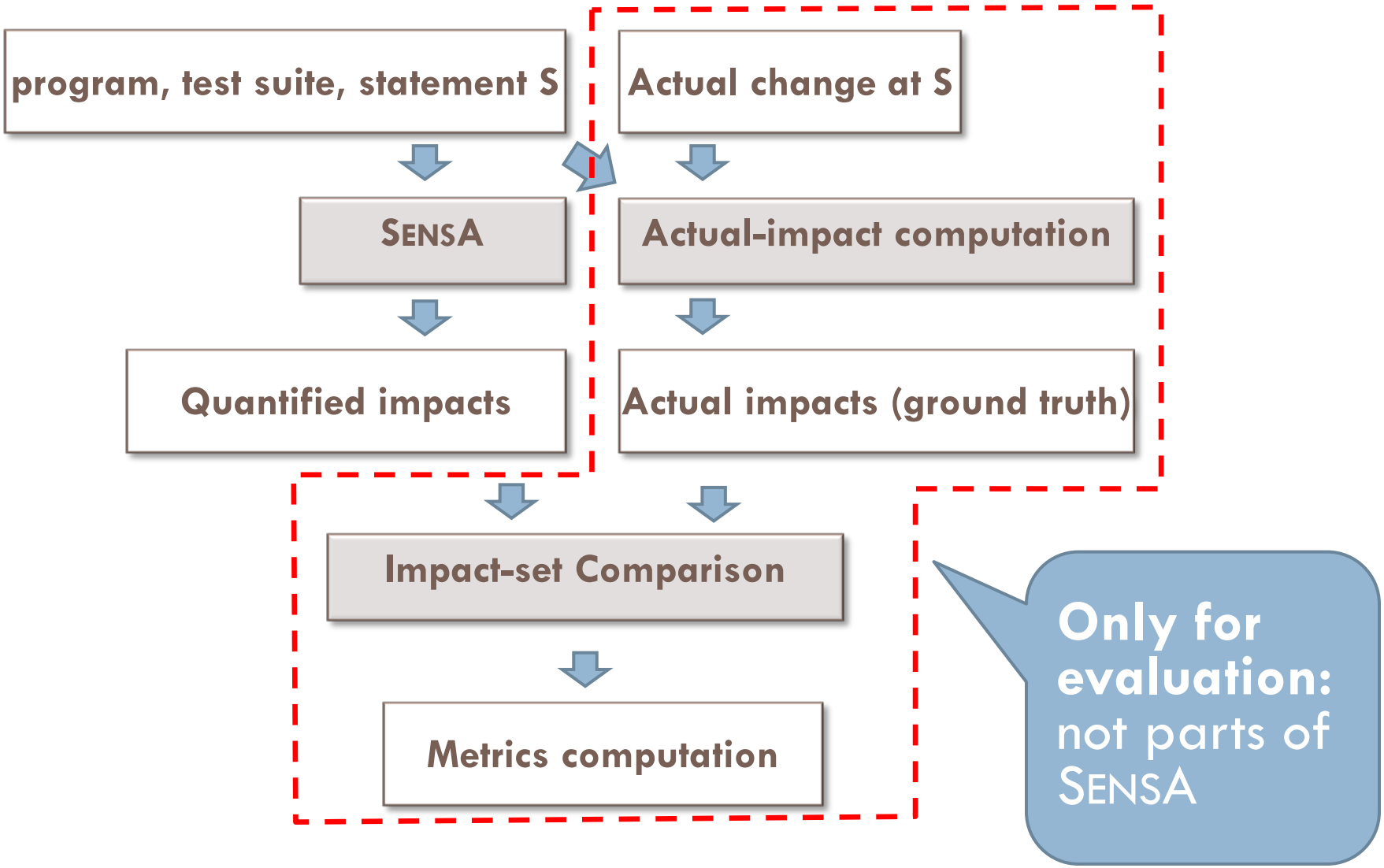
Subject programs and statistics

Subject	Description	Lines of Code	Tests	Changes
Schedule1	Priority Scheduler	290	2,650	7
NanoXML	XML parser	3,521	214	7
XML-Security	Encryption library	22,361	92	7
Ant	Java project build tool	44,862	205	7

Experimental methodology



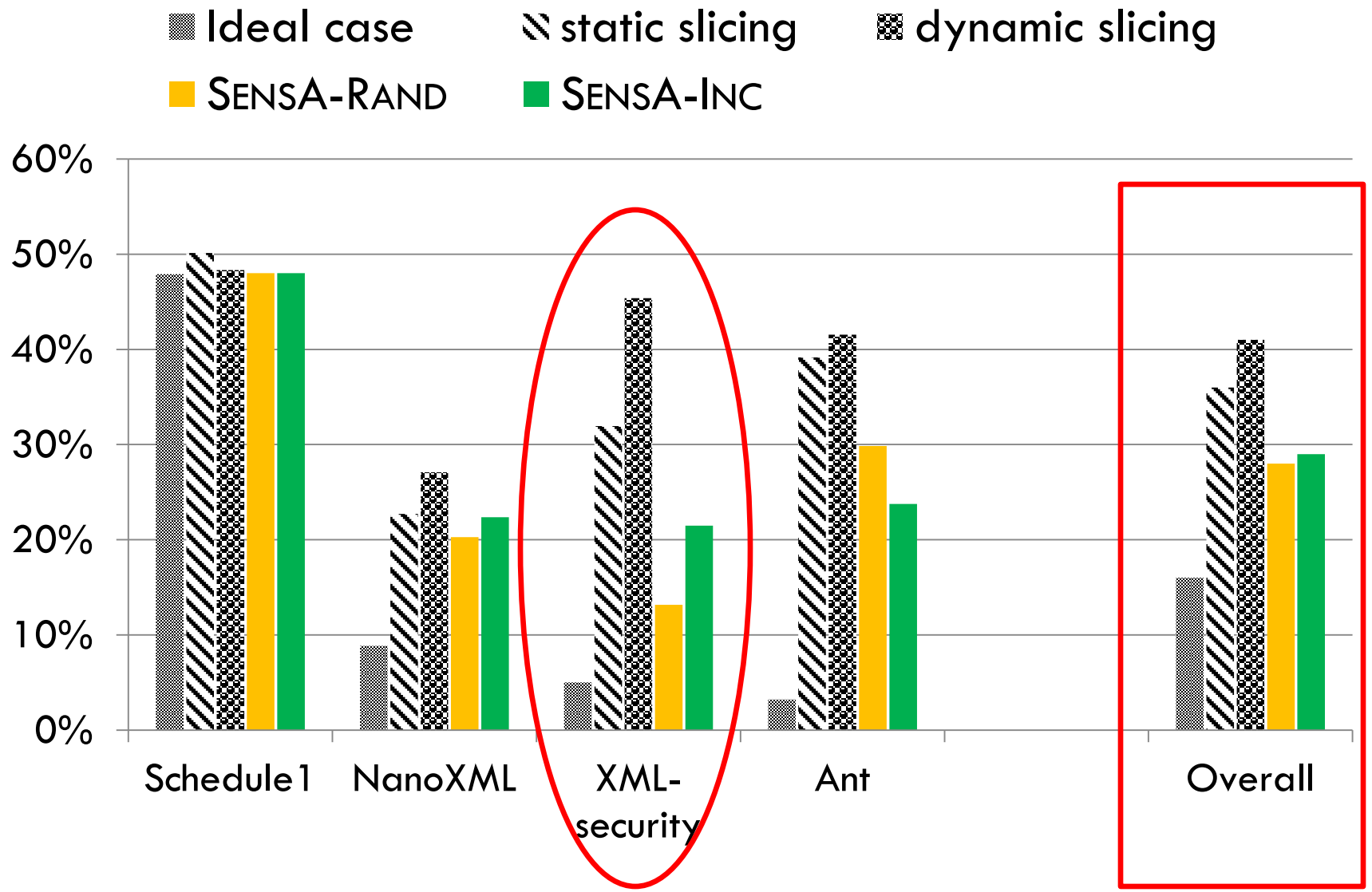
Experimental methodology



Experimental methodology

- Metrics
  - Effectiveness: inspection effort
    - Percentage of worse-case inspection cost
  - Cost: computation time
- Two variants: SENSA-RAND, SENSA-INC
- Compare to: static slicing, dynamic slicing, ideal case
  - Ideal case: best prediction possible
    - use the actual impact set as the prediction result

Results: inspect effort



Results: computation time

Subject	Static analysis	Instrumented run	Post-processing
Schedule1	6 sec	4,757 sec	1,054 sec
NanoXML	17 sec	773 sec	10 sec
XML-Security	179 sec	343 sec	21 sec
Ant	943 sec	439 sec	7 sec

- Static analysis and post-processing cost little time
- Runtime cost dominates the total cost
  - Come from multiple modified executions
  - Can be greatly reduced by executing all modifications in parallel

Results: computation time

Subject	Static analysis	Instrumented run	Post-processing
Schedule1	6 sec	4,757 sec	1,054 sec
NanoXML	17 sec	773 sec	10 sec
XML-Security	179 sec	343 sec	21 sec
Ant	943 sec	439 sec	7 sec

- Static analysis and post-pro ssing cost little time
- Runtime cost dominates the t st
  - Come from multiple mod
  - Can be greatly reduced parallel

Highly Parallelizable

Conclusion

- Contributions
  - A novel approach to quantifying dependencies and, based on that, a quantitative dynamic impact prediction technique
  - An empirical study of the new approach showing the significantly better effectiveness of the new approach than slicing, at reasonable costs
- Future Work
  - To expand the study by including more subjects and more types of changes
  - To apply the dependence-quantification approach to tasks other than impact analysis

Conclusion

Questions

- Contributions
  - A novel approach to quantifying dependencies and, based on that, a quantitative dynamic impact prediction technique
  - An empirical study of the new approach showing the significantly better effectiveness of the new approach than slicing, at reasonable costs
- Future Work
  - To expand the study by including more subjects and more types of changes
  - To apply the dependence-quantification approach to tasks other than impact analysis

Controversial statements

- Test suite augmentation is irrelevant to alleviating the limitation of dynamic analysis that the execute set used does not fully represent the program behavior.

- Quantitative dependence analysis is more effective than traditional non-quantified dependence analysis.