Analysis and Testing of Deployed Software

Alessandro (Alex) Orso

College of Computing
Georgia Tech
http://www.cc.gatech.edu/~orso

Supported in part by National Science Foundation, the State of Georgia under the Yamacraw Mission, and Boeing Commercial Airplane Group.

A Bug Can Be More Than a Nuisance

- 10 years of development
- $7 billion cost
- $500 million loss
- (no humans onboard)

Ariane 5

"No test was performed to verify that the SRI would behave correctly when being subjected to […] the trajectory of Ariane 5. Had such a test been performed, the failure […] would have been exposed."

(from the report of the Inquiry Board)

Software bugs are costing the U.S. economy an estimated $59.5 billion each year. Improvements in testing and maintenance could reduce this cost by about a third, or $22.5 billion.

(from NIST Estimated Planning Report 02-3)

Goal of My Research

Improve effectiveness and efficiency of testing and maintenance

- Develop new testing techniques (and required analyses)
- Develop systems and infrastructure
- Perform empirical evaluation in realistic settings

Outline

Background

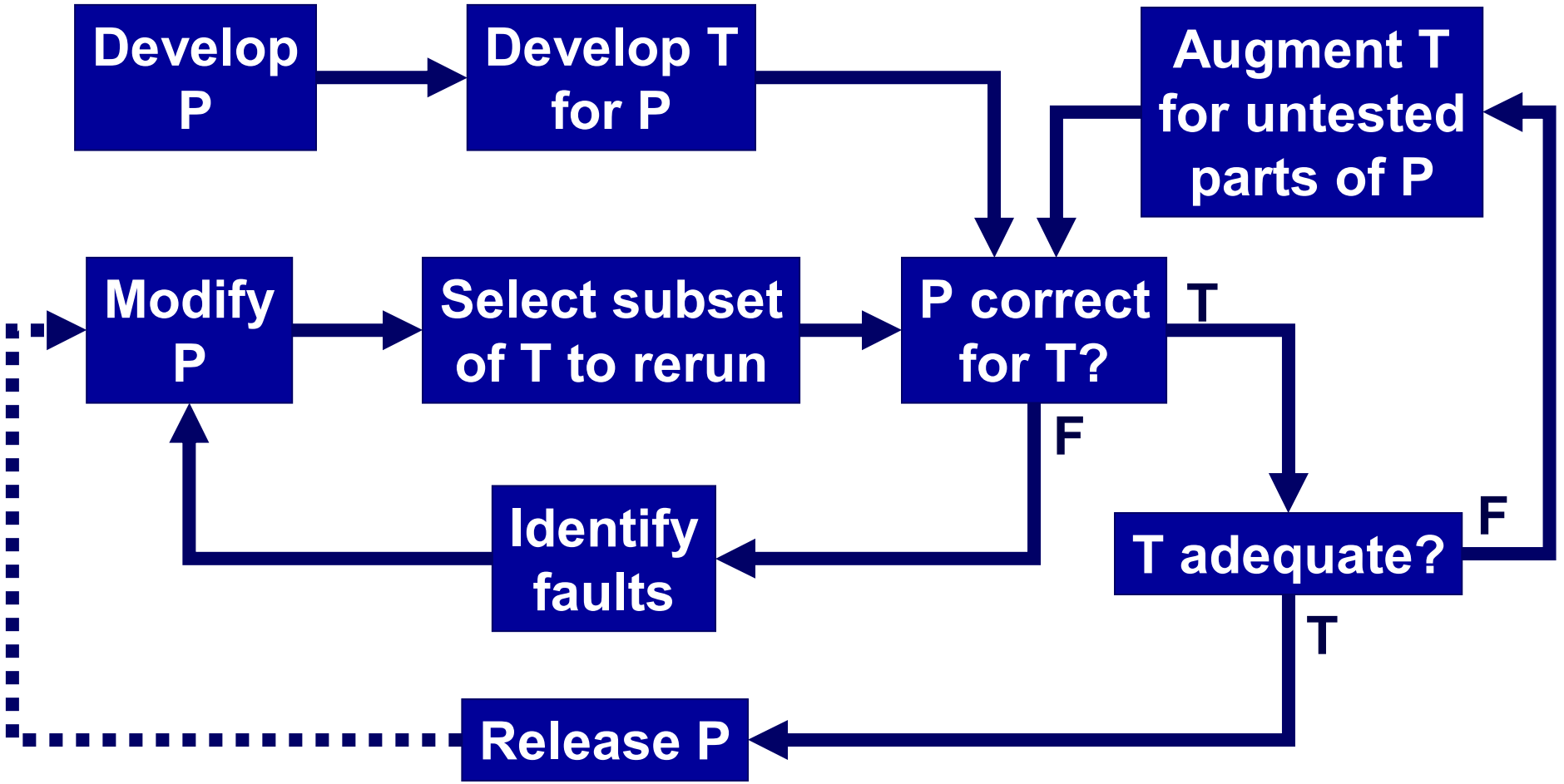
- **Testing and maintenance**
- **My research focus and contributions**

The Gamma Project

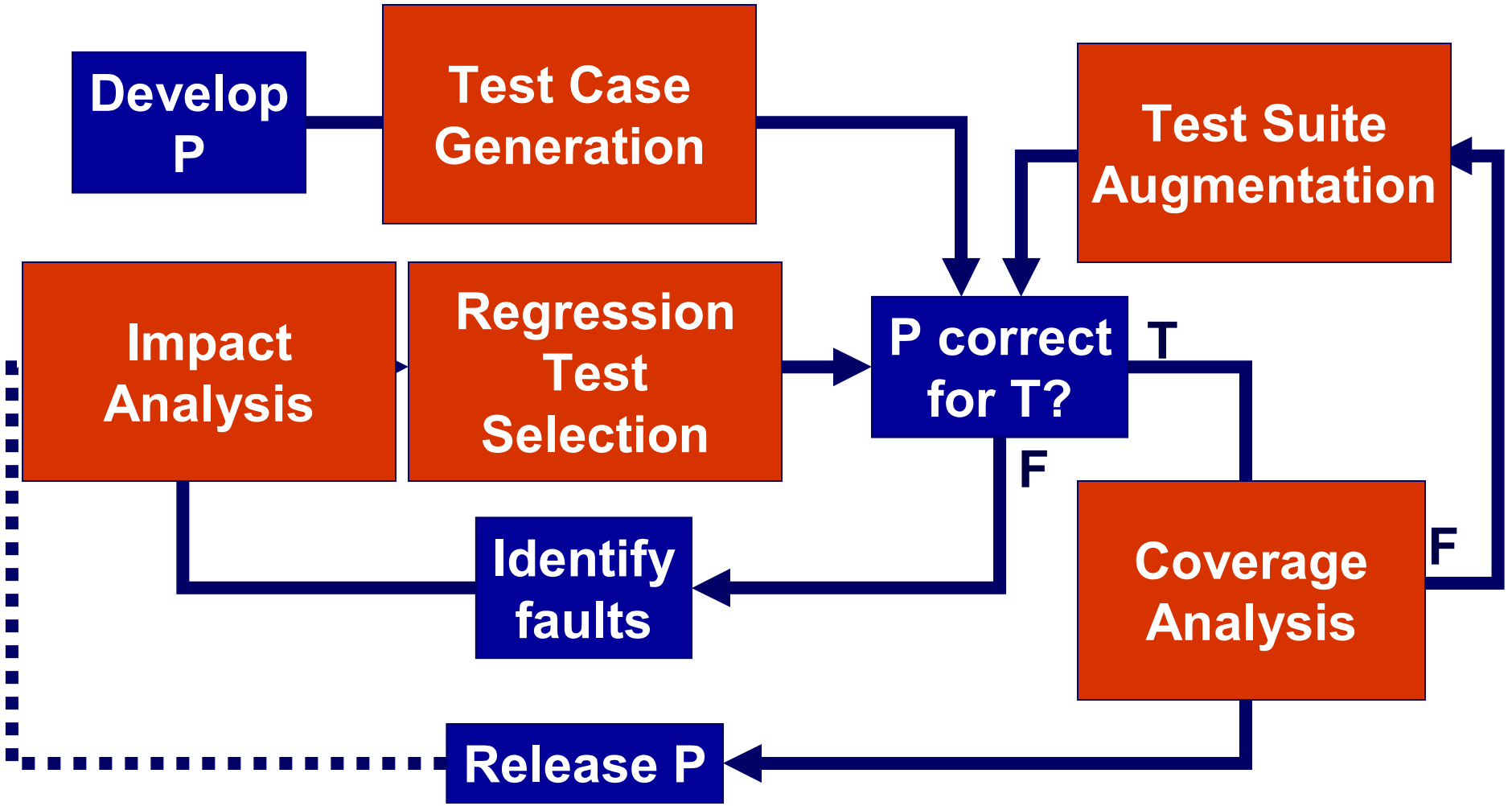
- Overview
- Impact analysis
- Replay of executions

Conclusion and Future Work

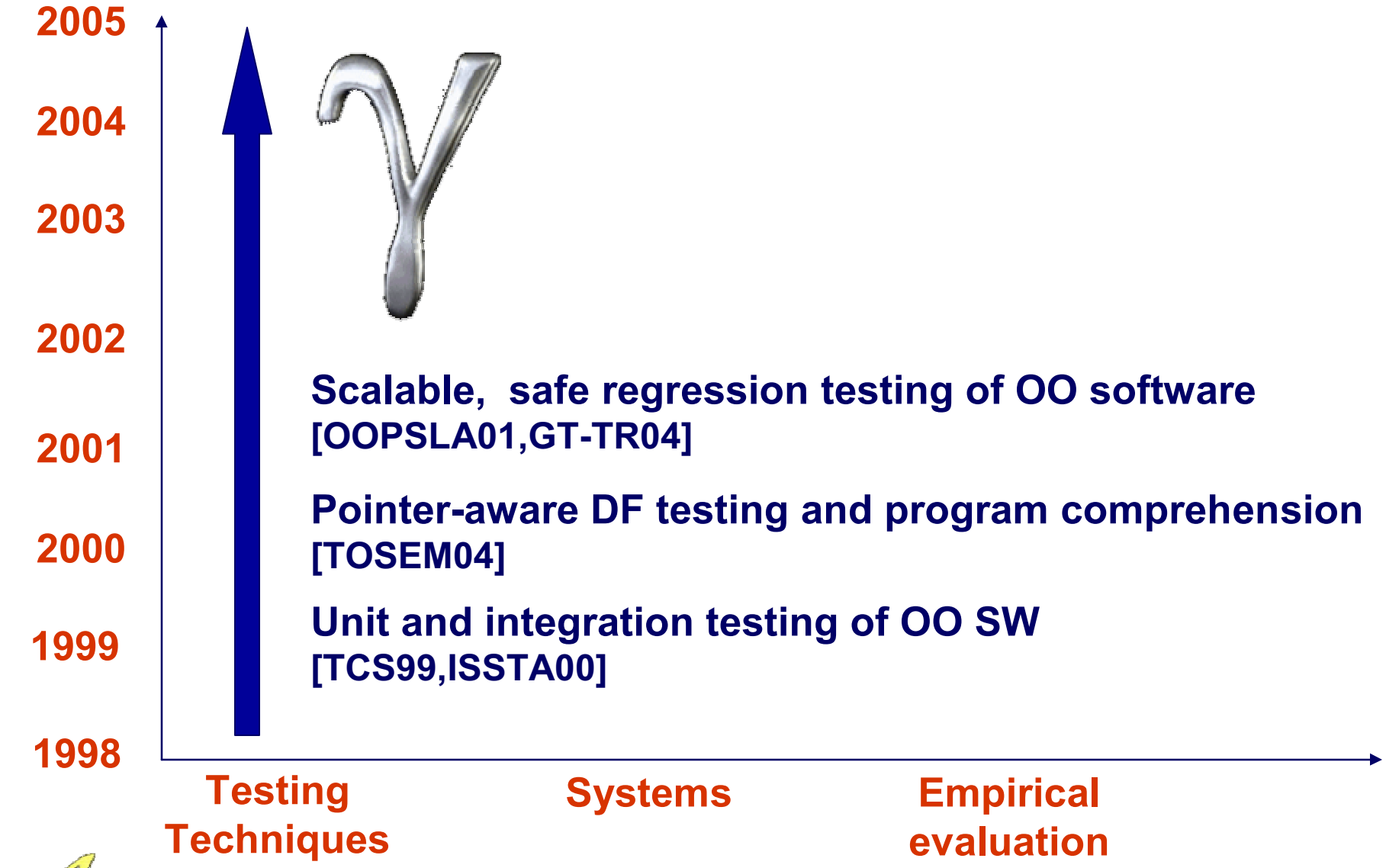
Testing and Maintenance



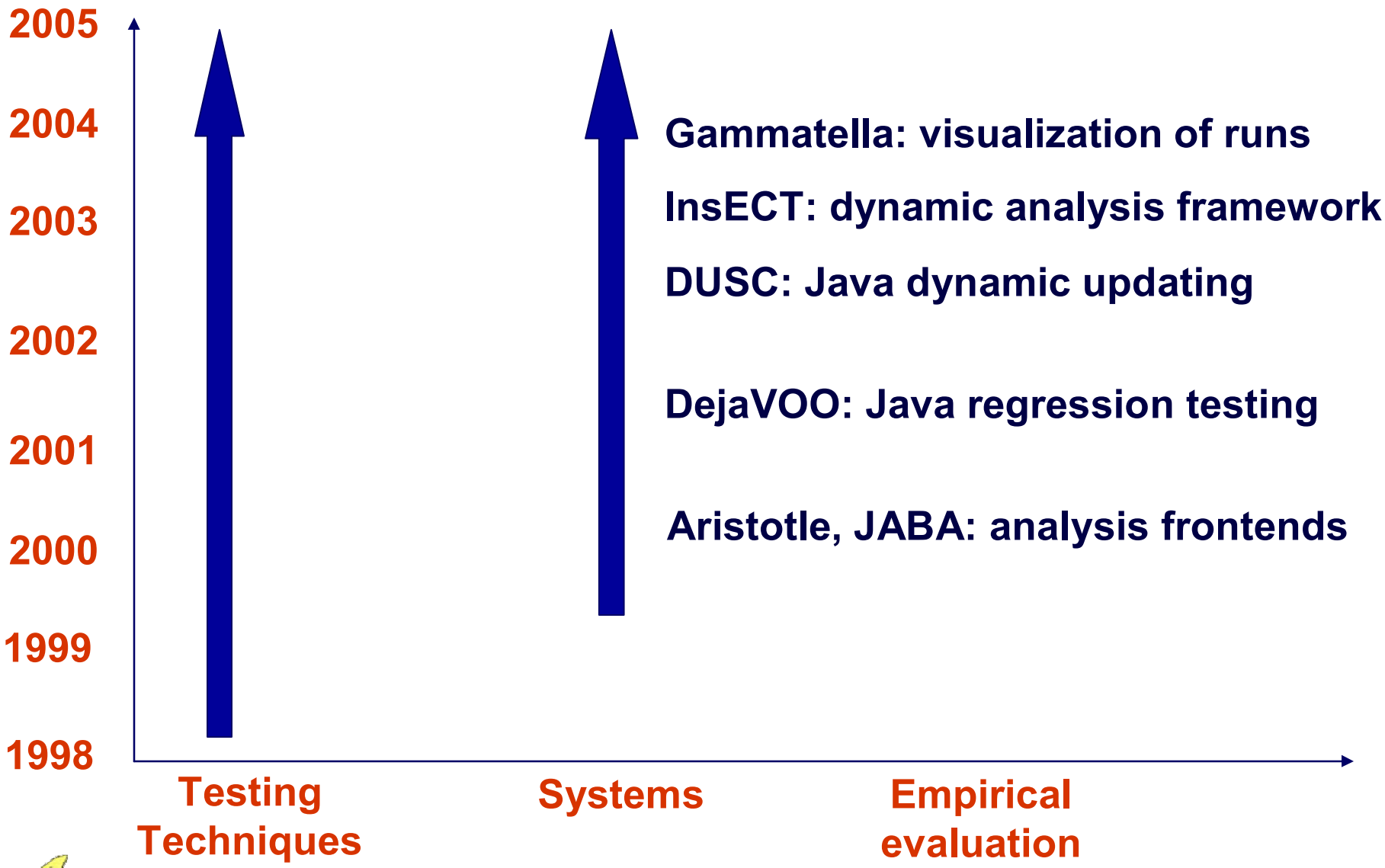
Testing and Maintenance



Contributions



Contributions



Gammatella: visualization of runs

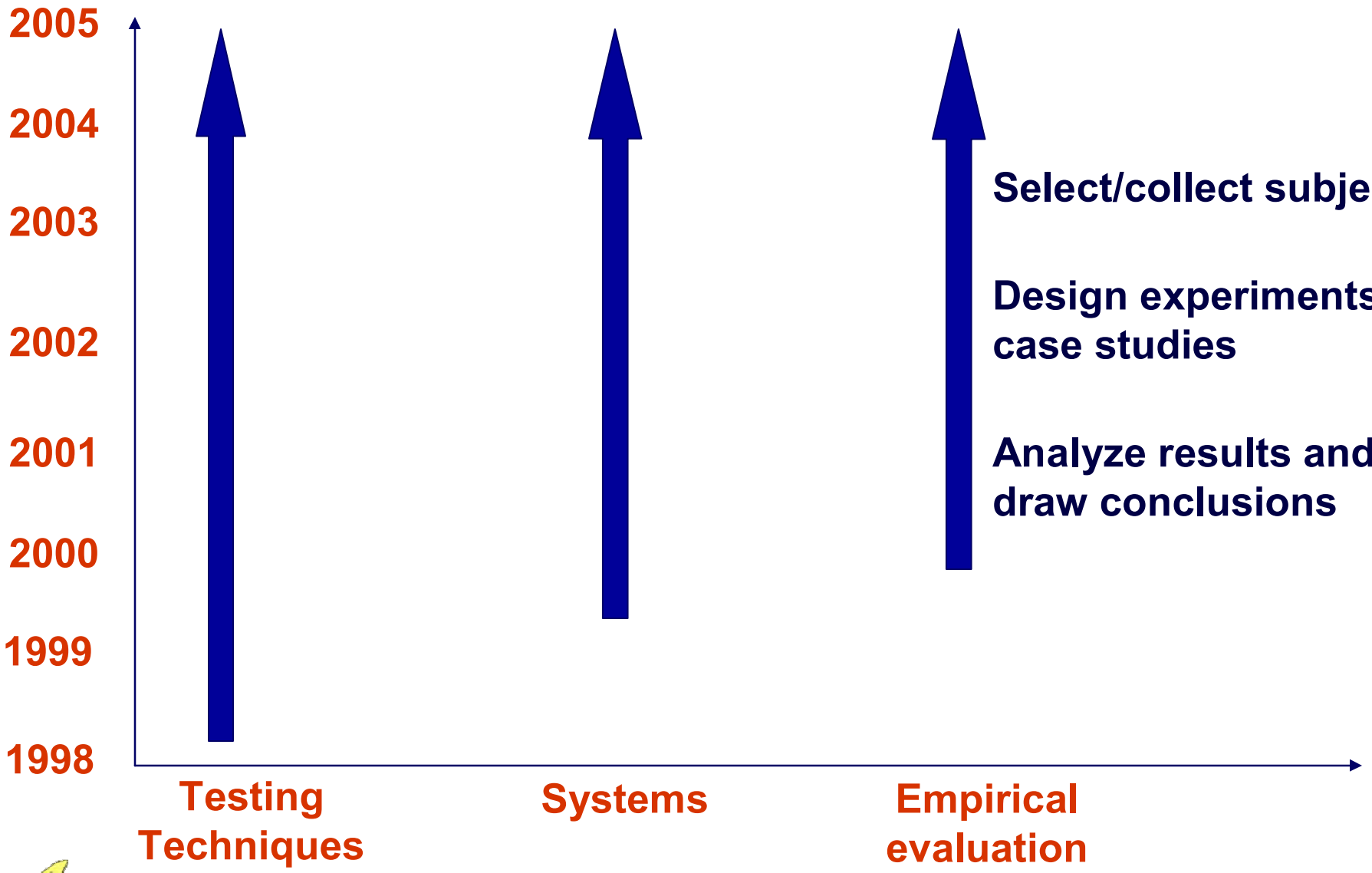
InsECT: dynamic analysis framework

DUSC: Java dynamic updating

DejaVOO: Java regression testing

Aristotle, JABA: analysis frontends

Contributions



Outline

Background

- Testing and maintenance
- My research focus and contributions

The Gamma Project

- **Overview**
- Impact analysis
- Replay of executions

Current and Future Work

Motivation for Gamma

Fundamental shift in software development

- Software virtually everywhere
- Most computers interconnected
- Large amount of user resources

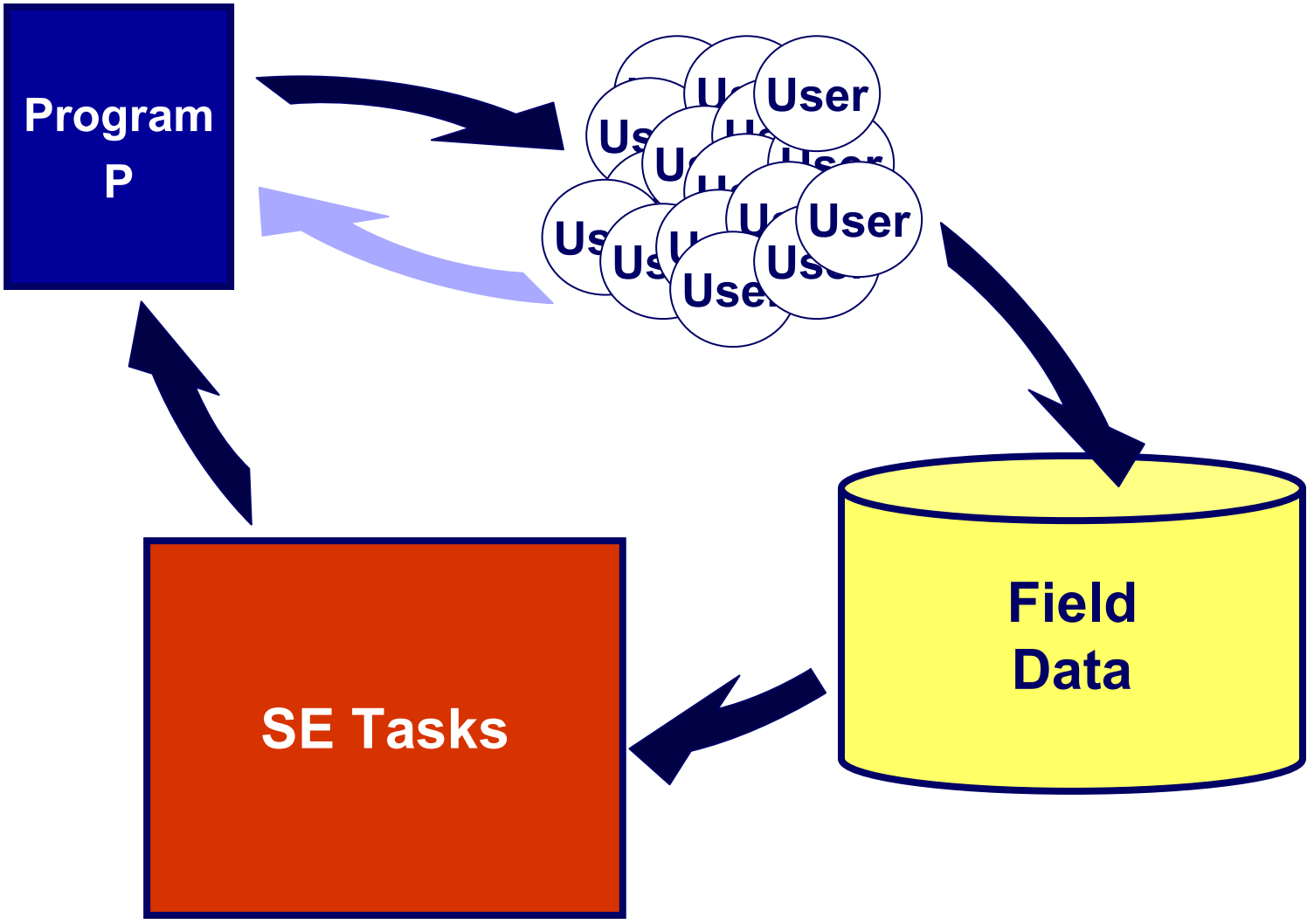
Problems

- Short development cycles
- Increasingly dynamic interactions
- Increased complexity (environments, SW structure, …)

Opportunity to use field data/resources for SE tasks

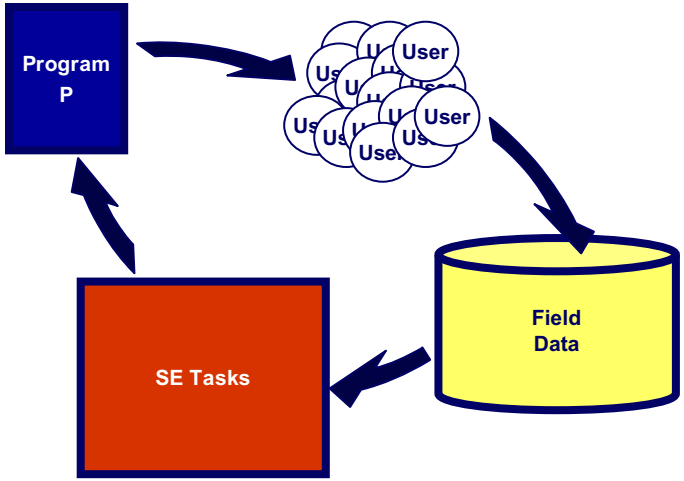
- In-house activities limited by the use of synthetic workloads and configurations
- The Gamma approach aims to overcome these limitations

Gamma: Overall Picture



[ISSTA02]

Gamma: Overall Picture



Challenges

- Applications
  - How to augment existing analyses
  - How to enable new analyses
- Data collection
  - Effectiveness
  - Efficiency
  - Security and privacy

Gamma: Work So Far

Applications

- Coverage analysis for deployed software [PASTE02]
- Impact analysis [FSE03,ICSE04]
- Regression testing [FSE03]
- Visualization of program executions [InfoVis04]
- Partial replay of users' executions [Dagstuhl03]

Data collection

- Software Tomography [ISSTA02,PASTE02]
- Dynamic SW Update [ICSM02]

Outline

Background

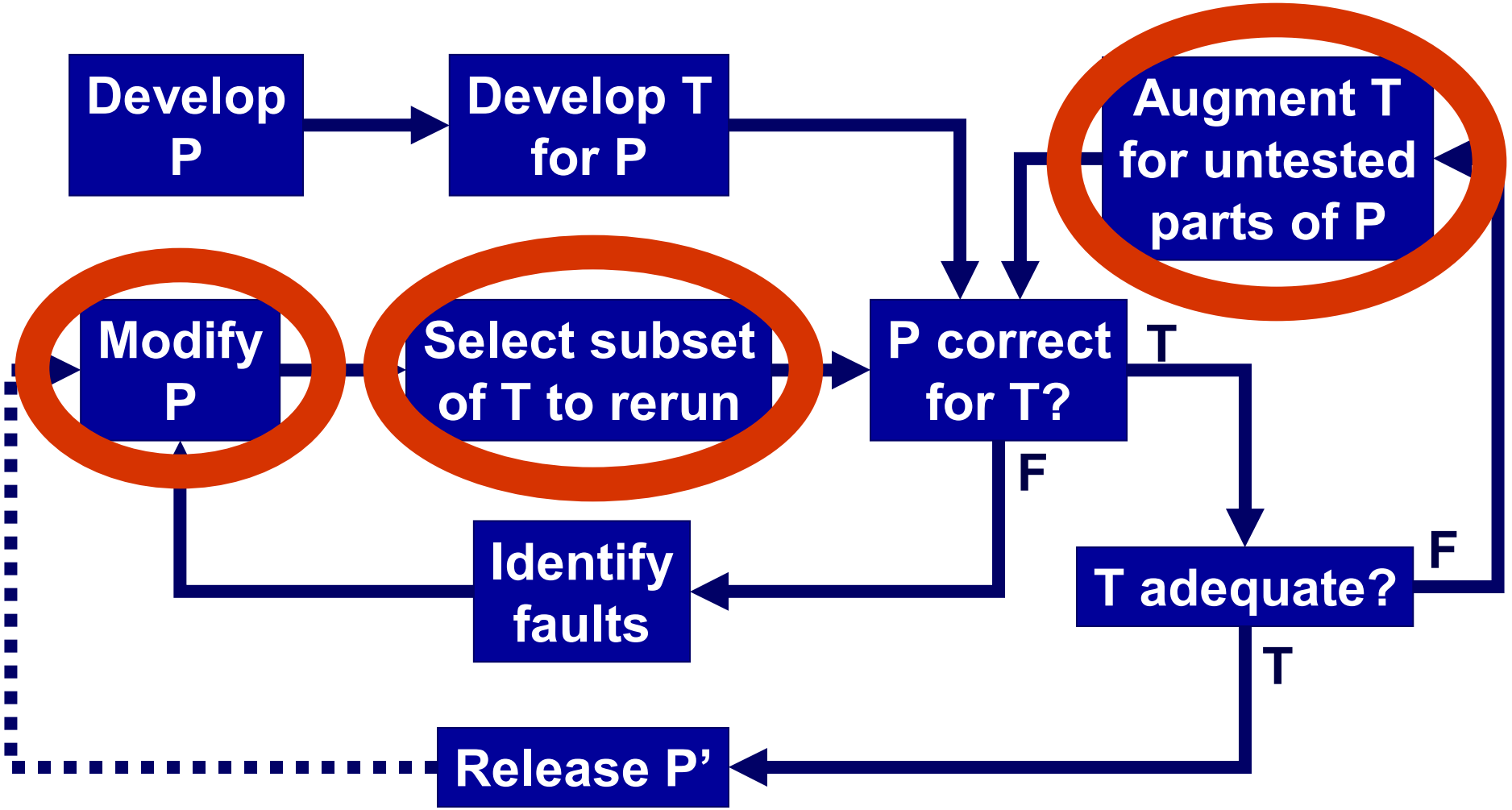
- Testing and maintenance
- My research focus and contributions

The Gamma Project

- Overview
- **Impact analysis**
- Replay of executions

Current and Future Work

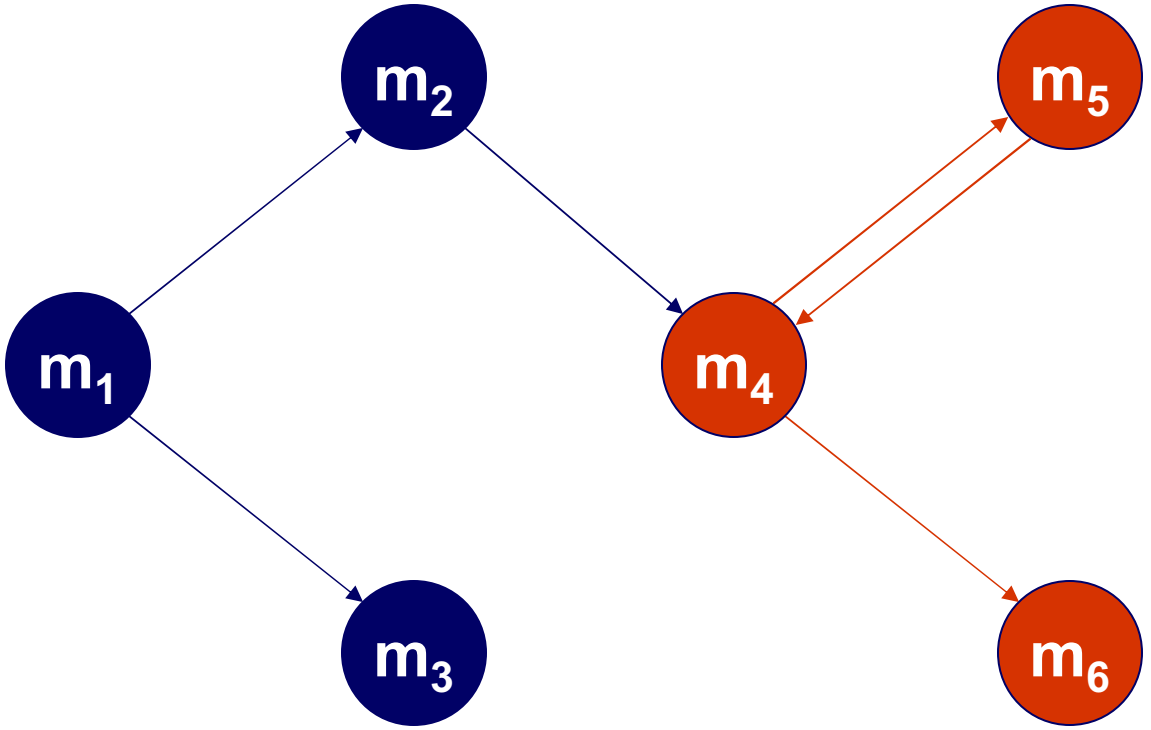
Testing and Maintenance



Call-Graph-Based Impact Analysis

```
m1() {
   if (…) m2();
   m3();
}

m2() {
   for(…) m4();
   …
   exit;
}

m3() {…}

m4() {
    if(…) m5();
    if(…) m6();
}

m5() {
    if(…) m4();
}

m6() {…}
```

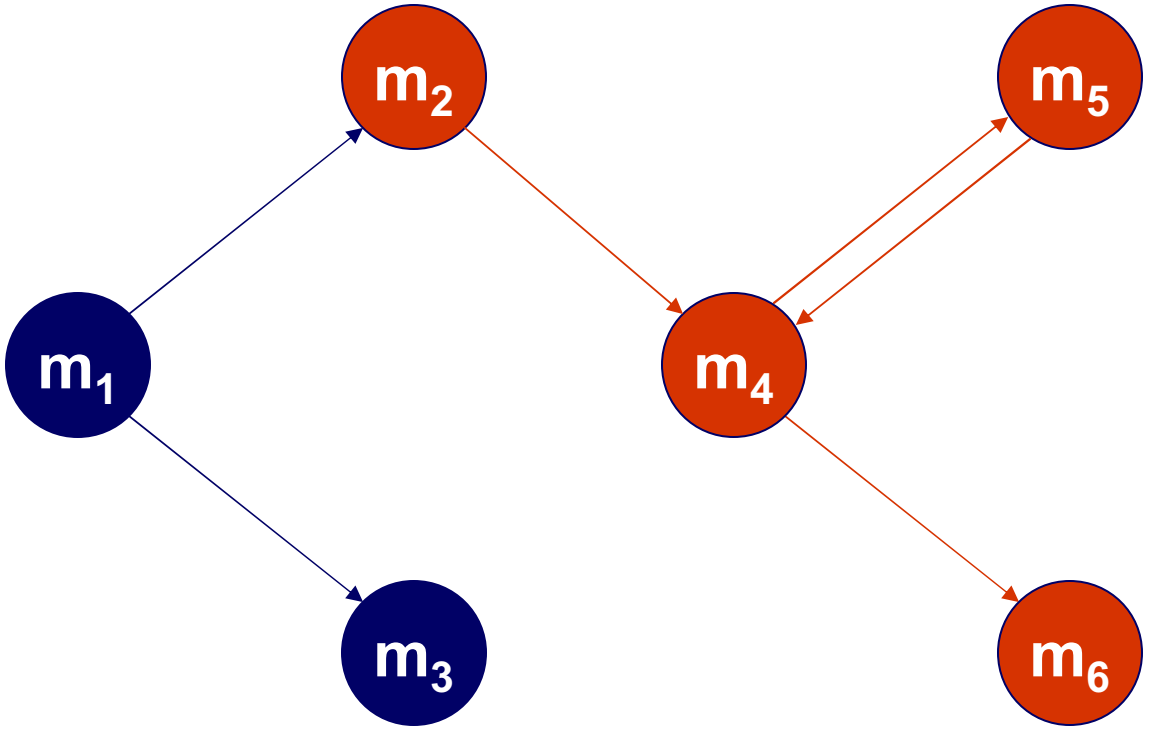
Impact set = { m4,m5,m6 }



Slicing-Based Impact Analysis

```
m1() {
   if (…) m2();
   m3();
}

m2() {
   for(…) m4();
   …
   exit;
}

m3() {…}

m4() {
    if(…) m5();
    if(…) m6();
}

m5() {
    if(…) m4();
}

m6() {…}
```

Impact set = { m2,m4,m5,m6 }



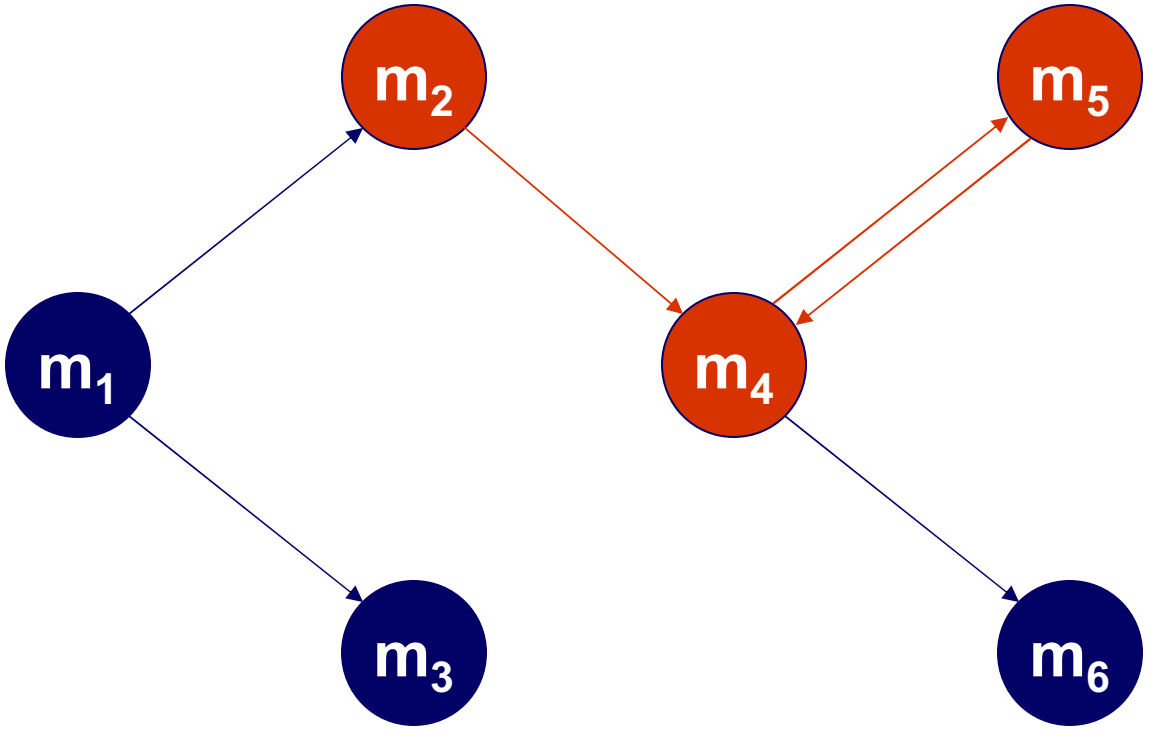
Dynamic Impact Analysis

```
m1() {
   if (…) m2();
   m3();
}

m2() {
   for(…) m4();
   …
   exit;
}

m3() {…}

m4() {
    if(…) m5();
    if(…) m6();
}

m5() {
    if(…) m4();
}

m6() {…}
```

Execution trace1: $m_1\ m_2\ m_4\ m_5\ r\ r\ x$

Execution trace2: $m_1\ m_2\ m_4\ m_6\ r\ r\ x$

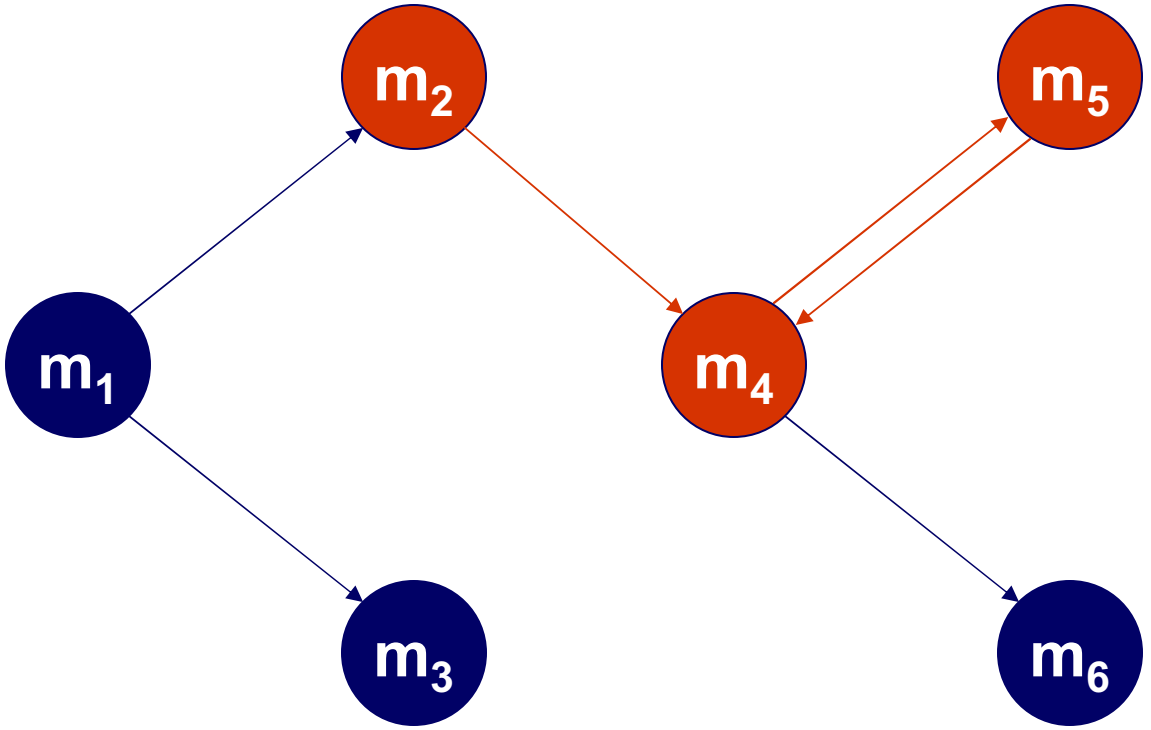
Impact set = { m2,m4,m5,~~m6~~ }



Dynamic Impact Analysis

```
m1() {
   if (…) m2();
   m3();
}
m2() {
   for(…) m4();
   …
   exit;
}
m3() {…}
m4() {
    if(…) m5();
    if(…) m6();
}
m5() {
    if(…) m4();
}
m6() {…}
```

Execution trace: $m_1\ m_2\ m_4\ m_5\ m_4\ r\ r\ r\ m_4\ m_5\ m_4\ r\ r\ r\ m_4\ m_5\ m_4\ r\ r\ r\ m_4\ m_5\ m_4\ r\ r\ m_6\ r\ r\ m_4\ m_5\ m_4\ r\ r\ r\ m_4\ m_5\ m_4\ r\ r\ r\ m_4\ m_5\ m_4\ r\ r\ r\ m_4\ m_5\ r\ r\ m_4\ m_5\ m_4\ r\ r\ r\ m_4\ m_5\ m_4\ r\ r\ r\ m_4\ m_5\ m_4\ r\ r\ r\ m_4\ m_5\ r\ m_6\ r\ r\ m_4\ r\ m_4\ r\ m_4\ m_5\ m_4\ r\ r\ r\ m_4\ m_5\ m_4\ r\ r\ r\ m_4\ r\ \ldots$



Impact Analysis: Goal

Develop an impact-analysis technique that

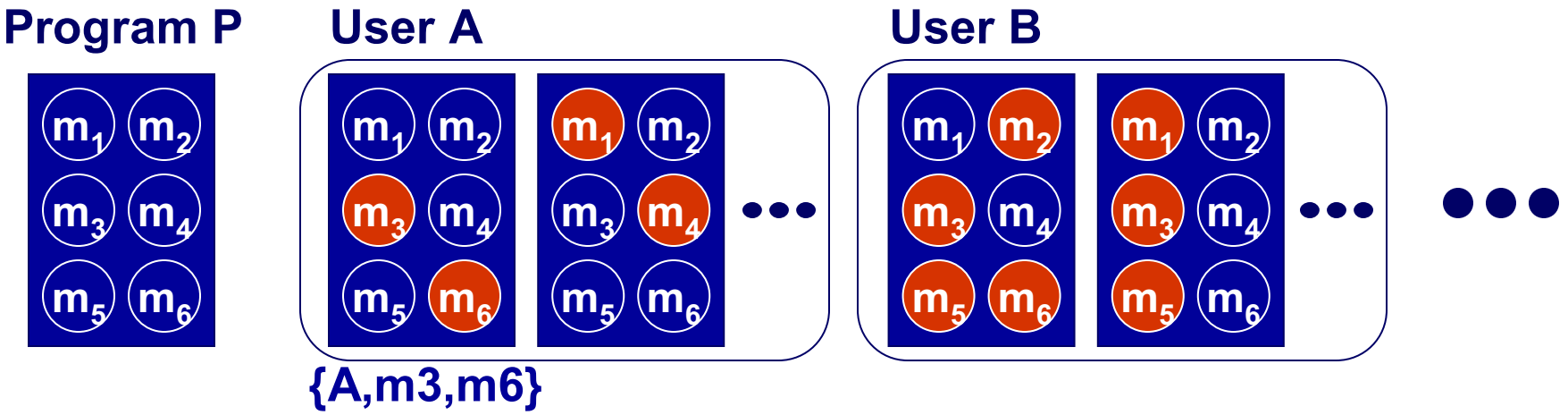
- Reflects actual usage in the field
- Conservative (safe) w.r.t. considered profile
- Lightweight

Lightweight dynamic forward slicing

- Coverage analysis
- Forward reachability

[FSE03]

Coverage Analysis



Field Data

	m_1	m_2	m_3	m_4	m_5	m_6
A1			X			X
B1		X	X		X	X
B2	X		X		X	
A2	X			X		
C1				X		X

Forward Reachability

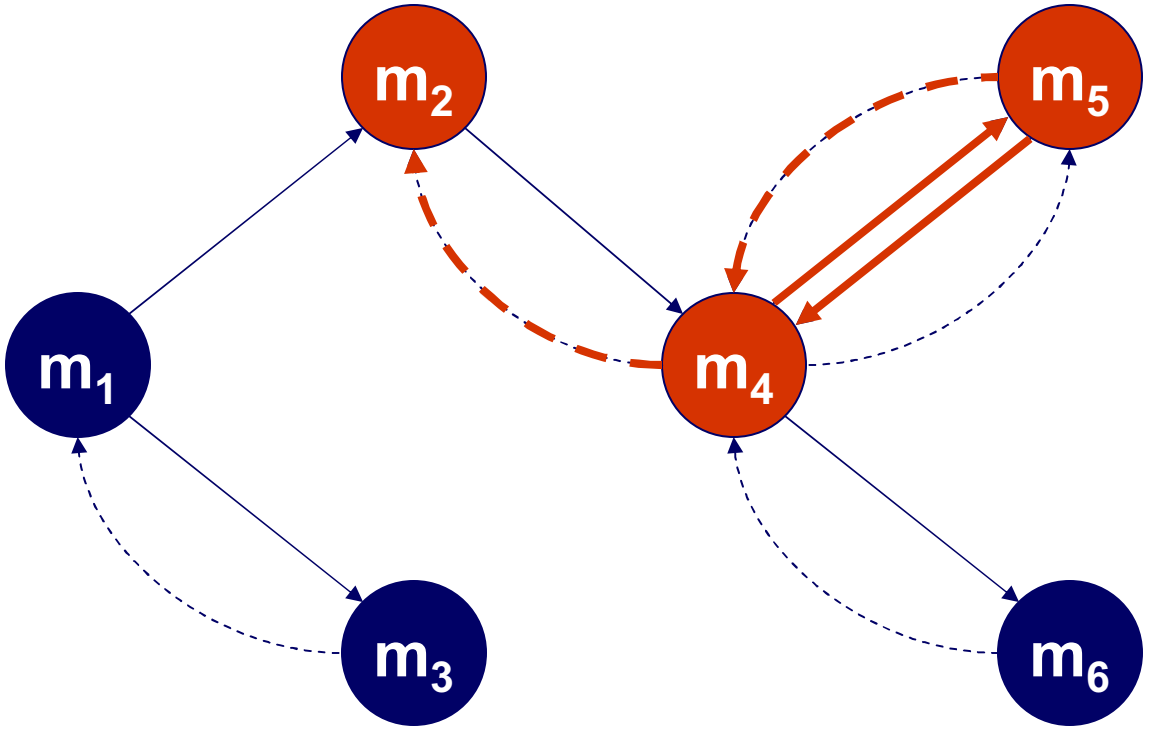
```
m1() {
   if (…) m2();
   m3();
}
m2() {
   for(…) m4();
   …
   exit;
}
m3() {…}
m4() {
    if(…) m5();
    if(…) m6();
}
m5() {
    if(…) m4();
}
m6() {…}
```

A1: $m_1\ m_2\ m_4\ m_5\ r\ r\ x$

B1: $m_1\ m_2\ m_4\ m_6\ r\ r\ x$

	m_1	m_2	m_3	m_4	m_5	m_6
A1	X	X		X	X	
B1	X	X		X		X

Impact set = { m2,m4,m5 }



Forward Reachability

```
m1() {
  if ( ) m2();
}
n
}
n
n
}
n
}
m6() {…}
```

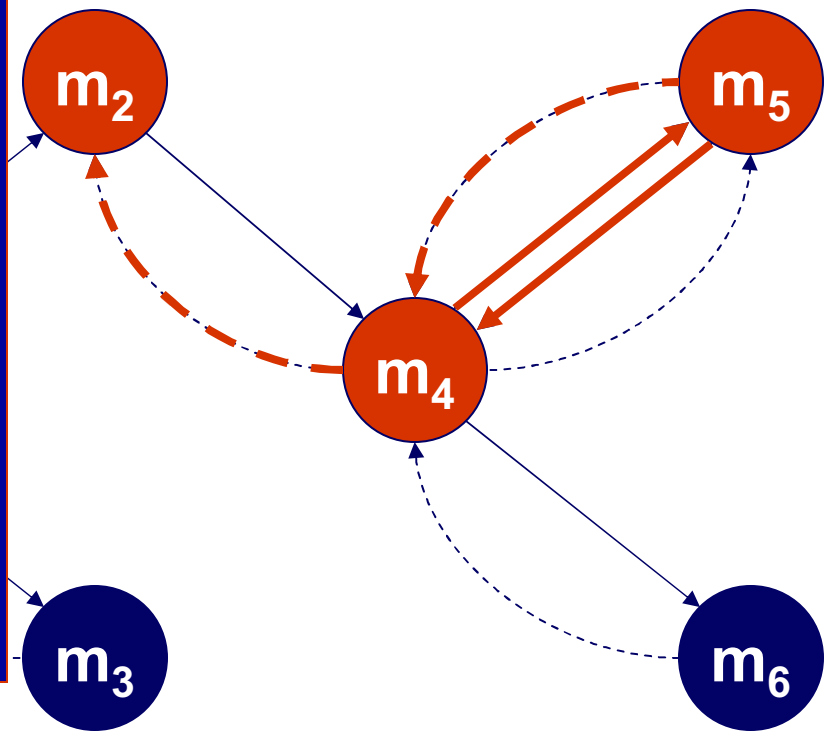
	m_1	m_2	m_3	m_4	m_5	m_6
A1	X	X		X	X	
B1	X	X		X		X

Input:
- program P
- set of changes C
- users' execution data

Output:
- impact set for C

1. Build augmented call graph
2. For each changed method c in C
   a. Identify user executions through c
   b. Mark methods covered by such executions
   c. Perform reachability considering only marked nodes
3. Return the set of reached methods

set = { m2,m4,m5 }



Sources of Imprecision

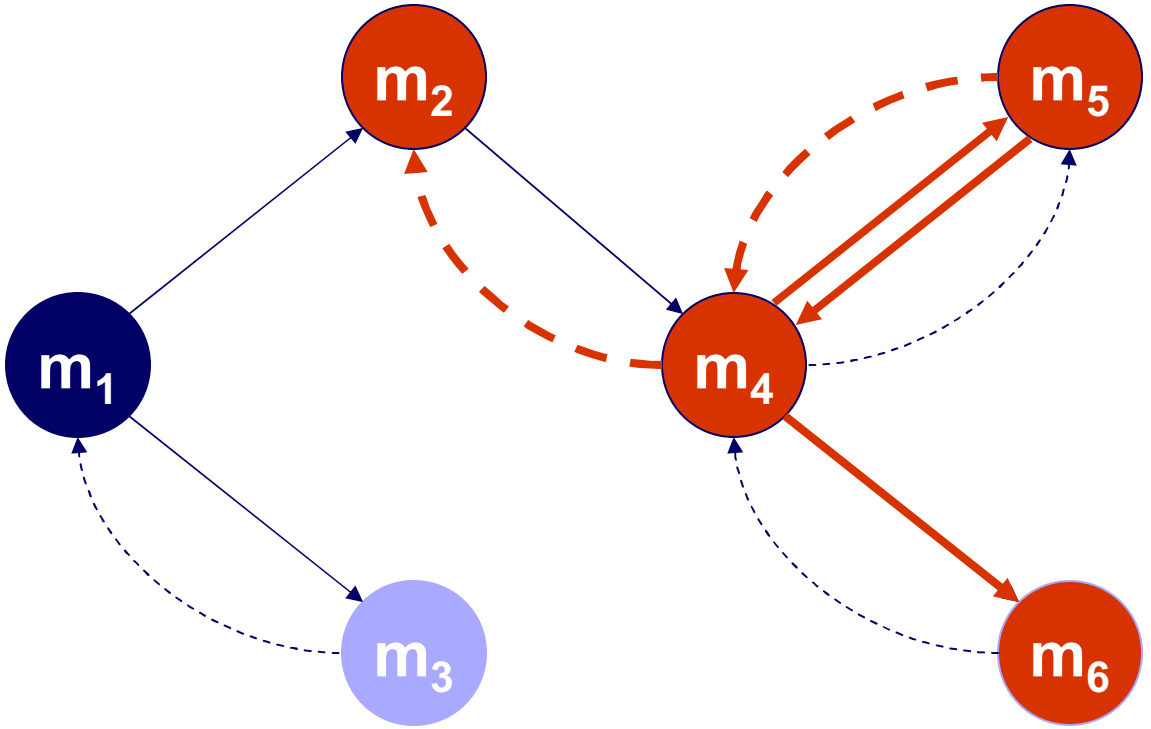
```
m1() {
   if (…) m2();
   m3();
}
m2() {
   for(…) m4();
   …
   exit;
}
m3() {…}
m4() {
    if(…) m5();
    if(…) m6();
}
m5() {
    if(…) m4();
}
m6() {…}
```

A1: $m_1\ m_2\ m_4\ m_6\ r\ m_5\ r\ r\ x$

B1: $m_1\ m_2\ m_4\ m_6\ r\ r\ x$

	m_1	m_2	m_3	m_4	m_5	m_6
A1	X	X		X	X	X
B1	X	X		X		X

Impact set = { m2,m4,m5,m6 }



Study 1

How does the technique compare with alternative approaches?

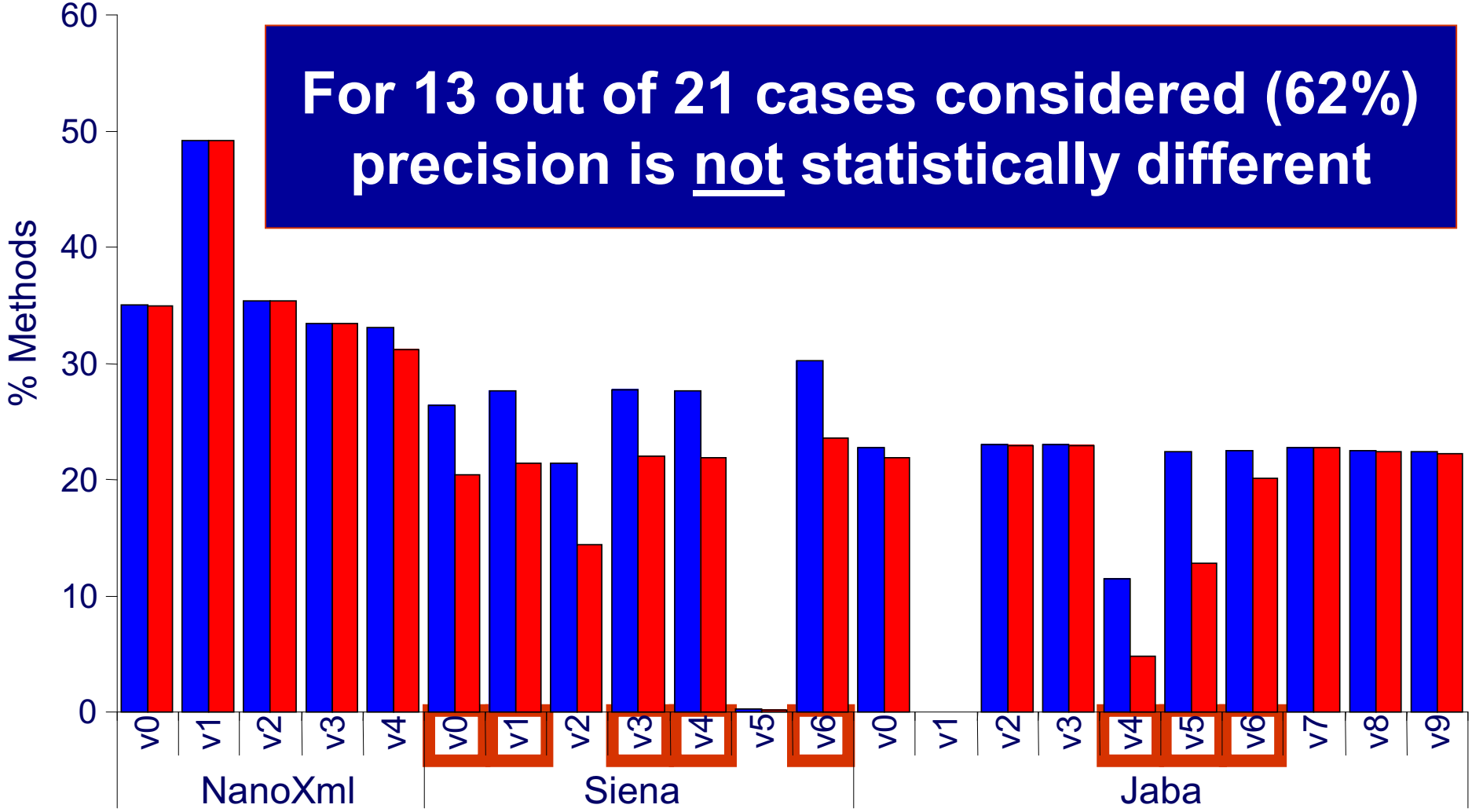
- Dynamic technique based on execution traces
- Static techniques
  - Based on transitive closure on the call graph
  - Based on slicing

Subjects

Program	Versions	Classes	Methods	LOC	Test Cases
NanoXML	6	19	251	3,279	216
Siena	8	24	219	3,674	564
Jaba	11	550	2,695	60,000	125

[ICSE04]

Results: Relative Precision



Difference in precision from 0 to 13.4, with avg. 3.8

Results: Time Overhead

Program		
NanoXML v0		
NanoXML v1	20.19	862.39
NanoXML v2	21.24	1,199.63
Siena v0	391.58	1,807.54
Siena v1	391.12	1,807.19
Siena v2	391.23	1,786.14
Jaba v0		
Jaba v1		
Jaba v2		
Jaba v3	53.71	60,012.22

14min (4,171%)

16hrs (111,634%)

Study 2

Does the use of field data actually yield different results than the use of in-house data?

Subject

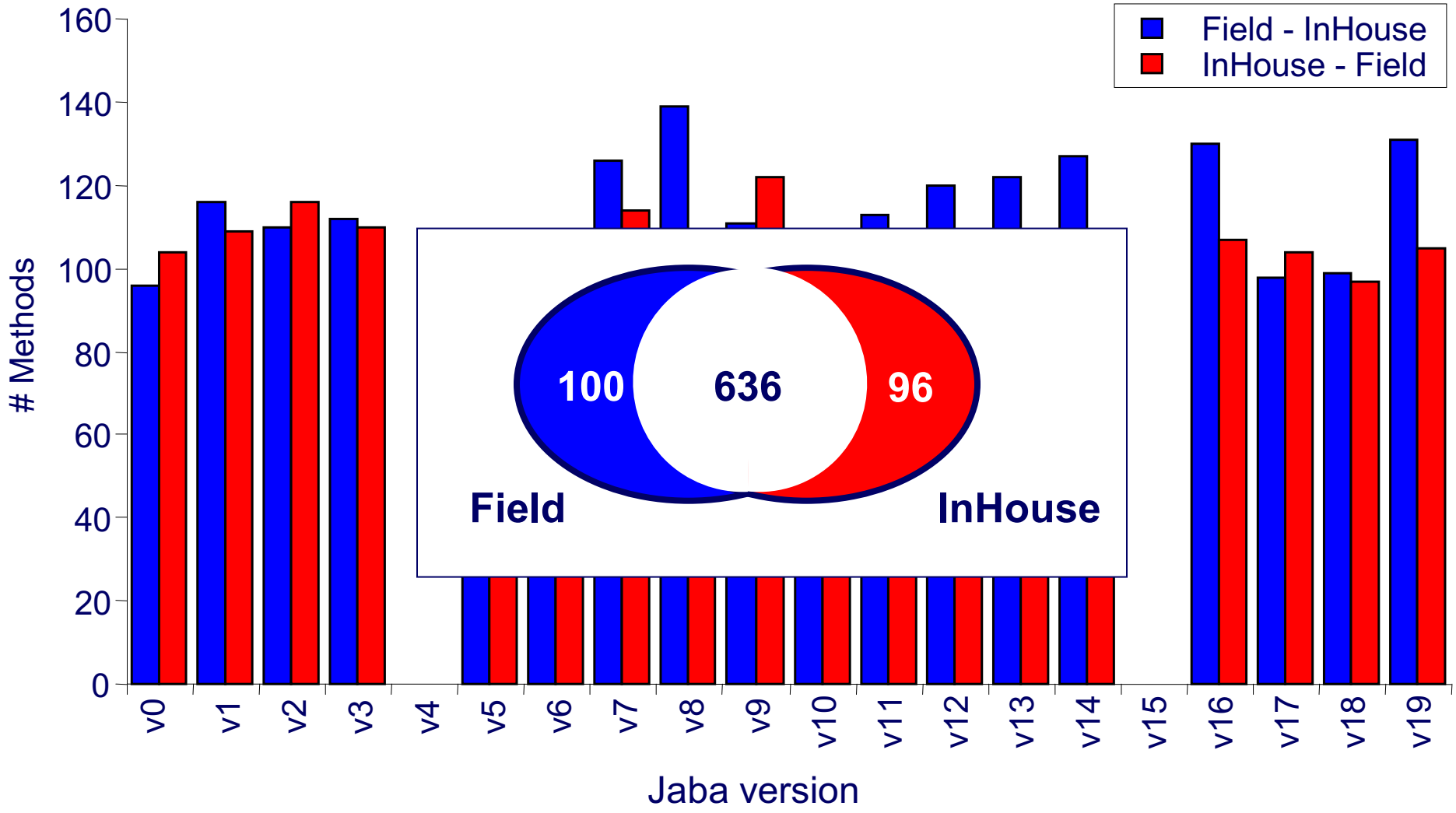
Program	Versions	Classes	Methods	LOC
Jaba	21	550	2,695	60,000

Data

- In-house data: 707 test cases, 63% coverage
- Field data: 1,100 executions (14 users, 12 weeks)

[FSE03]

Results: Differences in Impact Sets



Outline

Background

- Testing and maintenance
- My research focus and contributions

The Gamma Project

- Overview
- Impact analysis
- **Replay of executions**

Current and Future Work

Replaying Executions: Issues

Practicality

- High volume of data
- Hard to capture (custom)

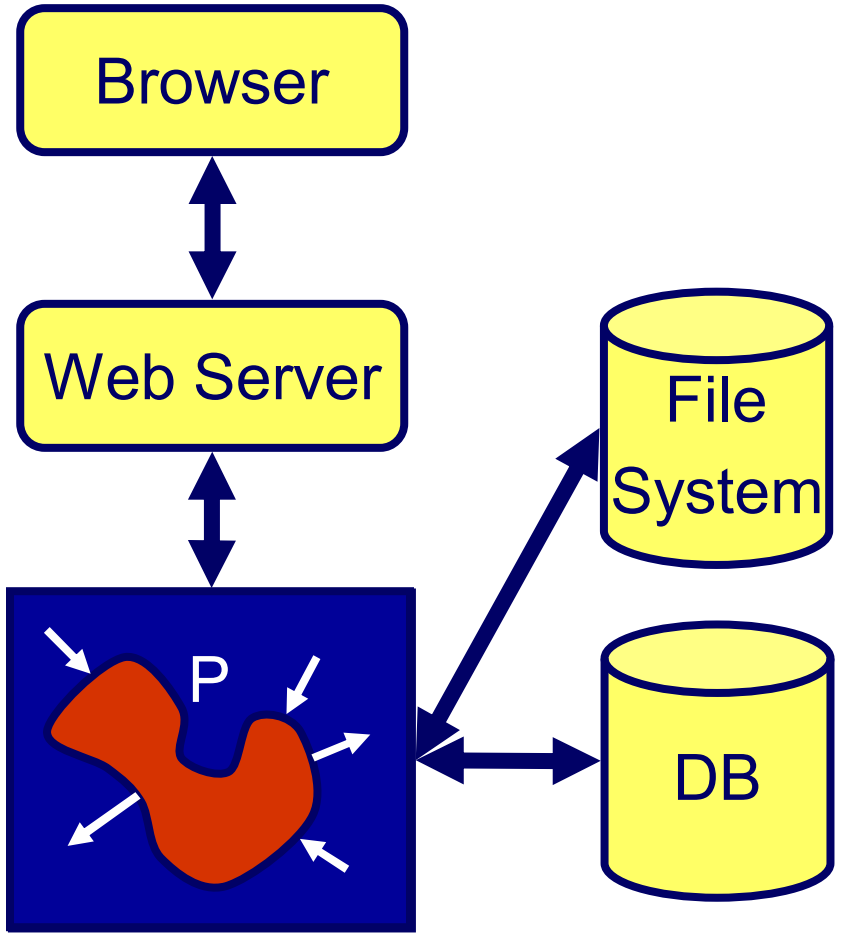
Privacy

- Sensitive information

Safety

- Side-effects

Partial replay

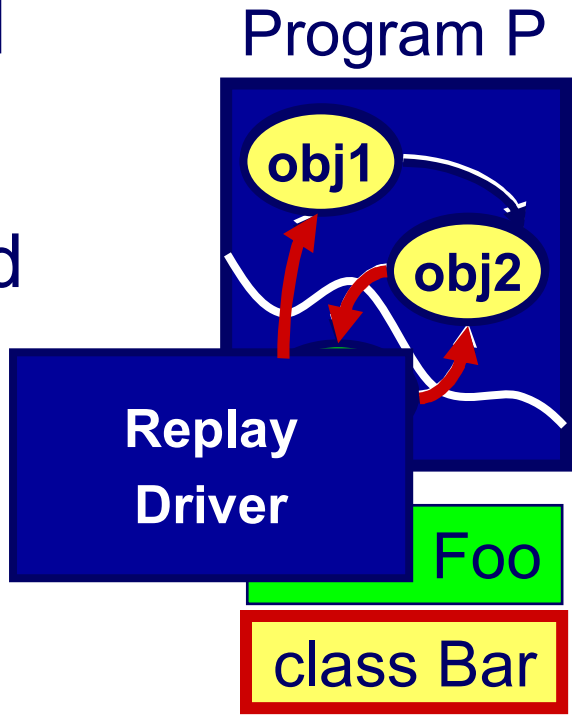


[Dagstuhl03]

Replaying Executions of Subsystems

Capture

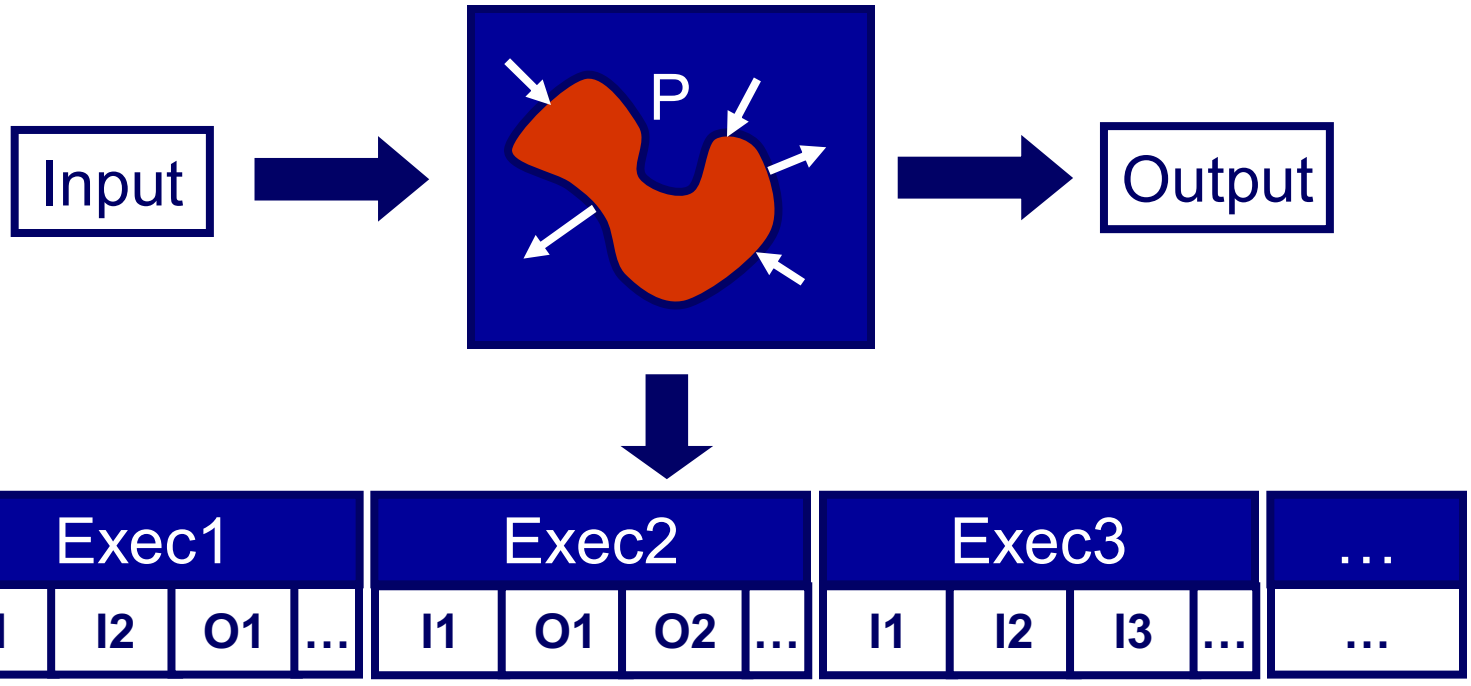
- Select subsystem of interest and its boundaries
- Collect information flowing in and from the subsystem



Replay

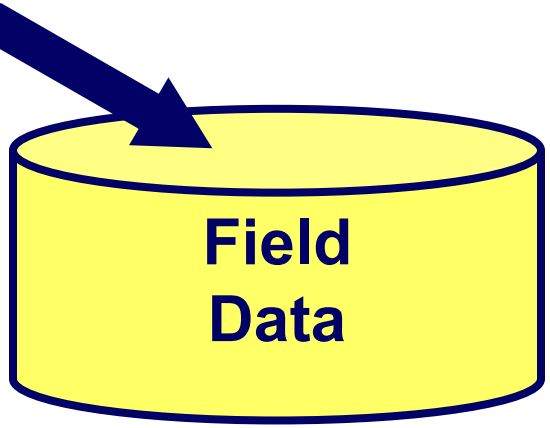
- Automatically generate driver
  - Perform incoming calls
  - Consume outgoing calls

Replaying Executions: Scenarios



Scenarios

- Run your favorite dynamic analysis
- Extract subsystem/unit regression tests



Outline

Background

- Testing and maintenance
- My research focus and contributions

The Gamma Project

- Overview
- Impact analysis
- Replay of executions

Conclusion and Future Work

Related Work

Gamma Project

- Perpetual/Residual testing (Clarke, Osterweil, Richardson, Young)
- Expectation-Driven Event Monitoring (EDEM) (Hilbert, Redmiles, Taylor)
- Skoll (Porter, Schmidt, et al.)
- Bug Isolation (Liblit, Aiken, et al.)

Static and Dynamic Analysis

- Trace-based impact analysis (Law and Rothermel)
- Too numerous to list!

Summary and Conclusions

- Gamma
  - Dynamic impact analysis
  - Replay of users' executions
- Empirical studies
  - User-based dynamic impact analysis is practical
  - Importance of using field data

Ongoing and Future Work

Gamma

- Hybrid technique for impact analysis
- Replay of executions
- Stability of users' behavior
- New analyses (e.g., memory-leak detection)
- Use of statistical analysis
- Larger live experiment

Testing of OO software

- Regression testing of large industrial software
- Augmentation based on change analysis
- Automated support for testing and maintenance

Static and dynamic analysis for security

- Analysis to prevent DOS attacks from mobile code

Collaborators on this work

- Taweesup Apiwattanapong
- Anil Chawla
- Mary Jean Harrold
- Bryan Kennedy
- Jim Law
- Gregg Rothermel

For more information:

- http://gamma.cc.gatech.edu