An Empirical Evaluation and Comparison of Manual and Automated Test Selection

Milos Gligoric, Stas Negara, Owolabi Legunsen, and

Darko Marinov

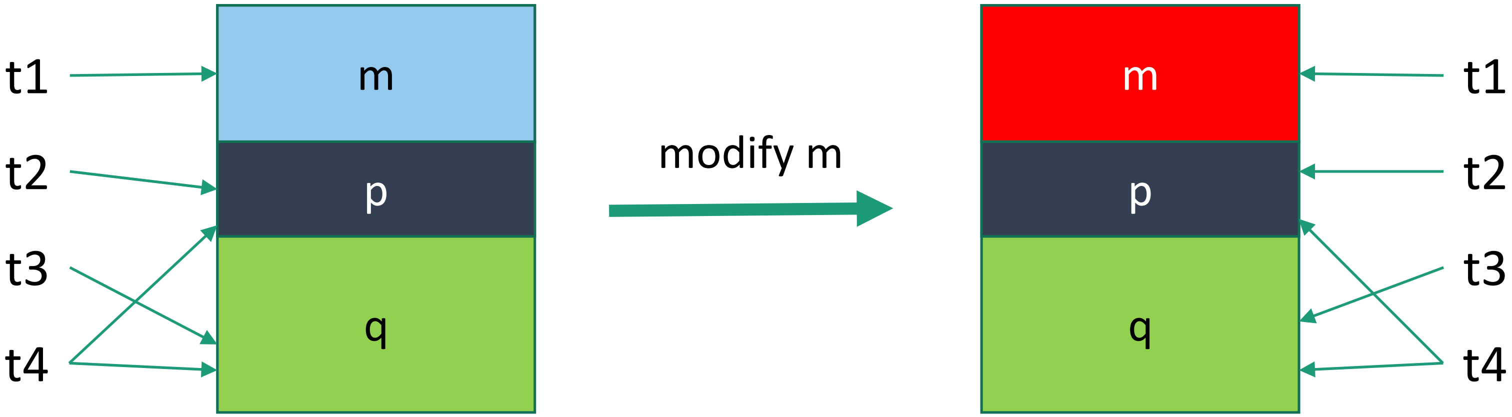
ASE 2014
Västerås, Sweden
September 18, 2014

CCF-1012759, CCF-1439957

ITI RPS #28

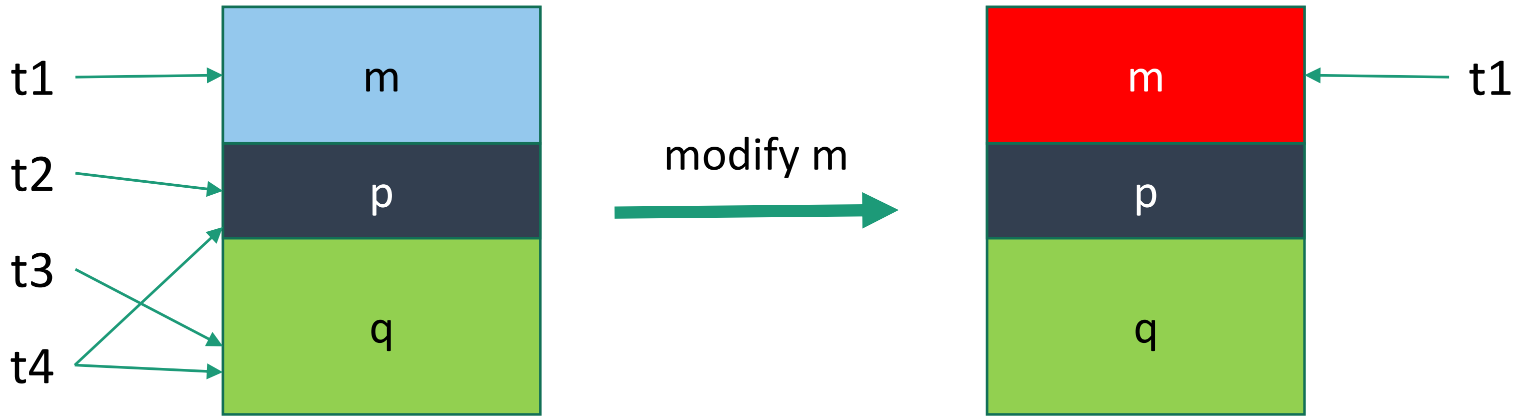
Regression Testing

- Checks that existing tests pass after changes
- RetestAll executes all tests for each new revision
  - ~80% of testing budget, ~50% of software maintenance cost



Regression Test Selection (RTS)

- Selects only tests whose behavior may be affected
- Several optimization techniques have been proposed
- Analyzes changes in codebase
- Mapping from test to various code elements
  - method, statement, edge in CFG

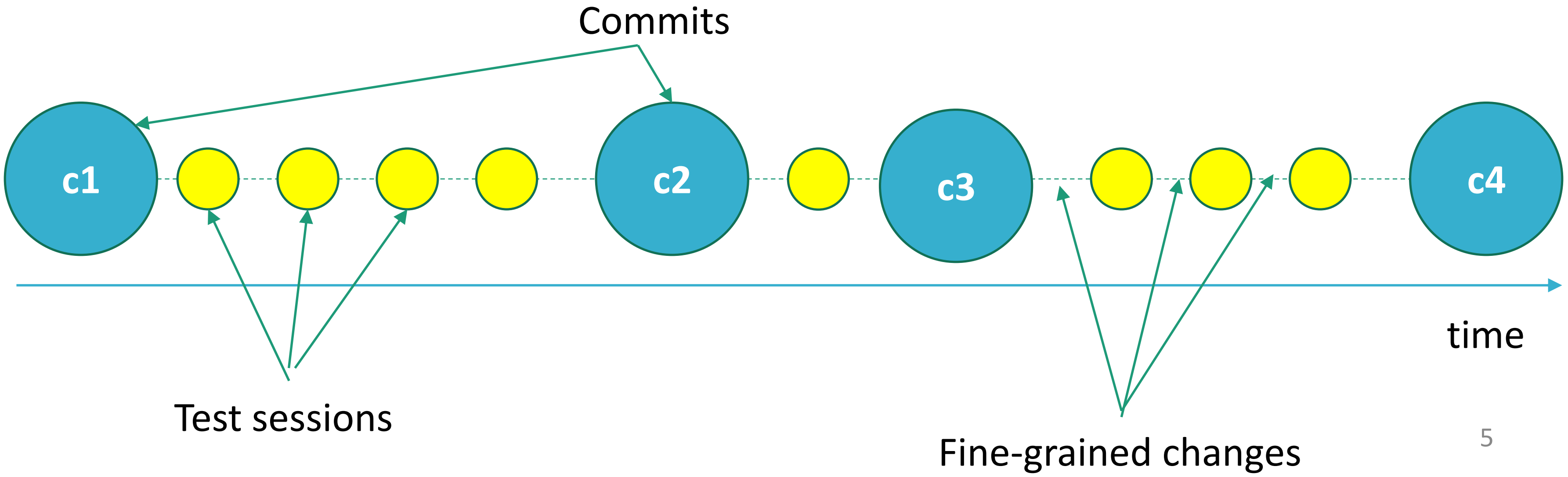


Motivation

- Few systems used in practice: Google TAP
  - Mapping of tests based on dependencies across projects
  - Not applicable to day-to-day work within single project
- No widely adoptable automated RTS tool after ~30 years of research
- Developers’ options:
  - RetestAll (expensive) or manual RTS (imprecise/unsafe)
- No prior study of manual RTS

Hard to Obtain Data

- Data was captured using a record-and-replay tool that was built to study code changes/evolution
- Data by chance had info about test sessions (runs of 1 or more tests)
- Live data allowed us to study manual RTS



Collected Data

- 14 developers working on 17 projects
- 3 months of monitoring
- 918 hours of development, 5757 test sessions, 264,562 executed tests
- 5 professional programmers, 9 UIUC students

Programming Experience (years)	Number of Participants
2-4	1
5-10	8
>10	5

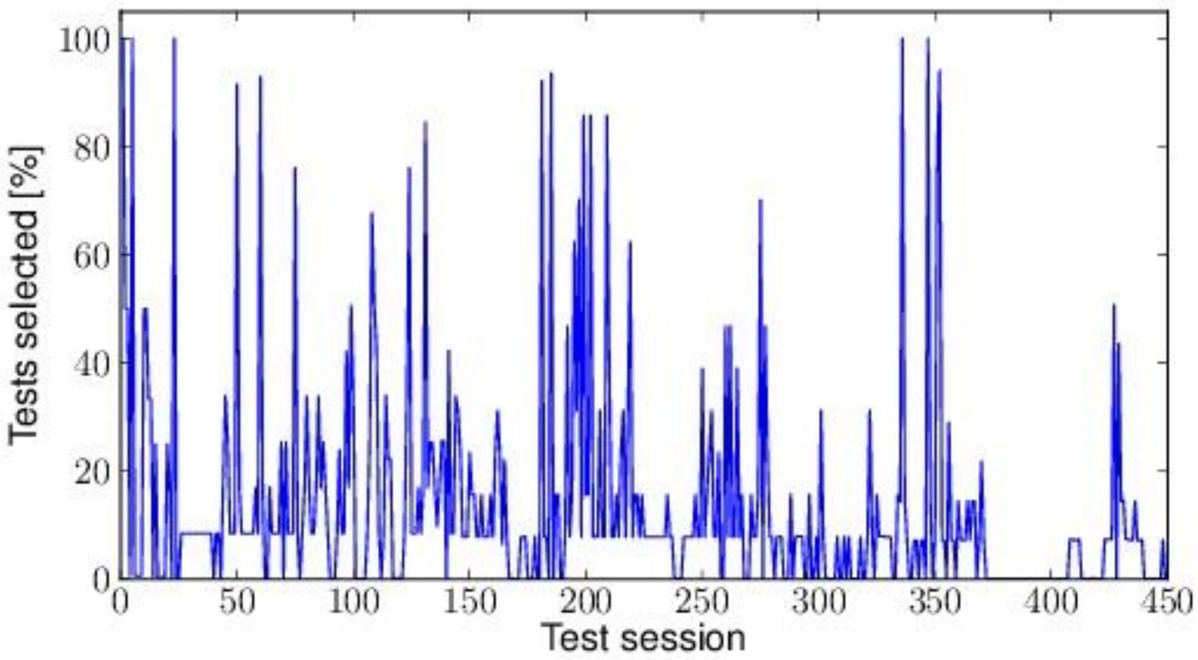
Programming Experience of Study Participants

Research Questions

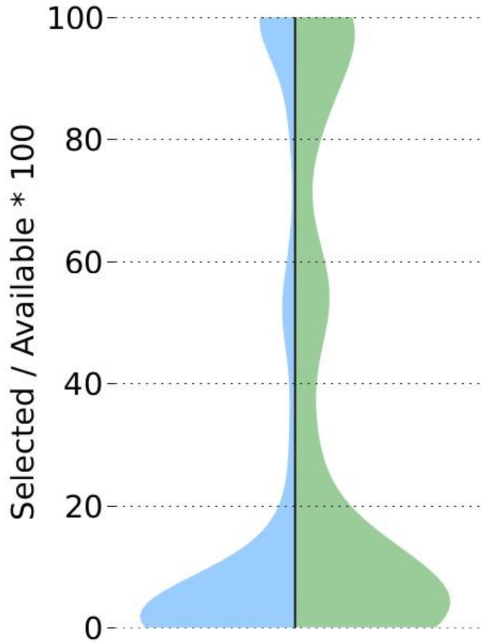
- RQ1: How often do developers perform manual RTS?
- RQ2: What is the relationship between manual RTS and size of test suites or amount of code changes? (Why bother with RTS for small projects?)
- RQ3: What are some common scenarios in which developers perform manual RTS?
- RQ4: How do developers commonly perform manual RTS?
- RQ5: How good is current IDE support in terms of common scenarios for manual RTS?
- RQ6: How does manual RTS compare with automated RTS?

RQ1

How often do developers perform manual RTS?



Manual Selection trends for one study participant



Distribution of Manual RTS ratio for all Participants; they rarely select > 20%

RQ2

What is the relationship between manual RTS and size of test suites or amount of code changes?

- Manual RTS was done regardless of test suite size
  - Max test suite size: 1663
  - Min test size: 6
  - Average time per test: ~0.48 sec
- No correlation between manual RTS and amount of code changes
  - Mean±SD Spearman’s and Pearson’s (w/o single): 0.07±0.10 and 0.08±0.15
  - Mean±SD Spearman’s and Pearson’s (w single): 0.12±0.18 and 0.13±0.09,
- We expected more tests to be run after larger code changes

RQ3

What are some common scenarios in which developers perform manual RTS?

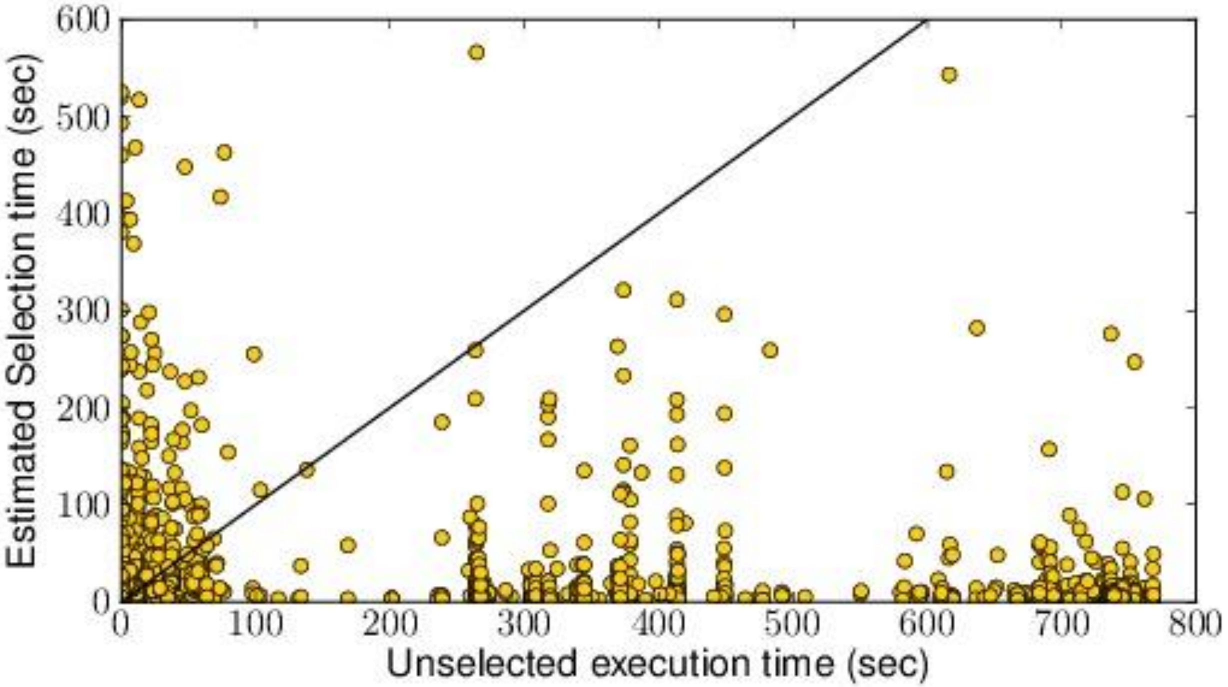
- Debugging
  - Debug test sessions: at least one test failed in preceding test session
  - 2,258 debug test session out of the 5,757

- Performing manual RTS in order to focus, not just for speedup
- This aspect has not been addressed in the literature

RQ4

How do developers commonly perform manual RTS?

- They use ad-hoc ways like comments, launch scripts
- 31% of the time, RetestAll would have been better than manual RTS (above the identity line)



RQ5

How good is current IDE support in terms of common scenarios for manual RTS?

- Limited support for arbitrary selection of multiple tests at once
- VS 2010 requires knowledge of regular expressions & all tests

RTS Capability	Eclipse	Netbeans	IntelliJ	VS 2010
Select single test	+	+	+	+
Run all available tests	+	+	+	+
Arbitrary selection in a node	-	-	±	+
Arbitrary selection across nodes	-	-	±	+
Re-run only previously failing tests	+	+	+	+
Select one from many failing tests	-	-	+	+
Arbitrary selection among failing tests	-	-	+	+

Methodology (RQ6)

- Goal: compare manual and automated RTS
  - We had relatively precise data for manual RTS but challenging to run a tool for automated RTS
- First, we reconstructed the state of project at every test session
- Replayed CodingTracker logs and analyzed the data
  - Discovered that the developer often ran test sessions with no code changes between them
  - For each test session, we ran FaultTracer on the project and compared tool selection with developer selection

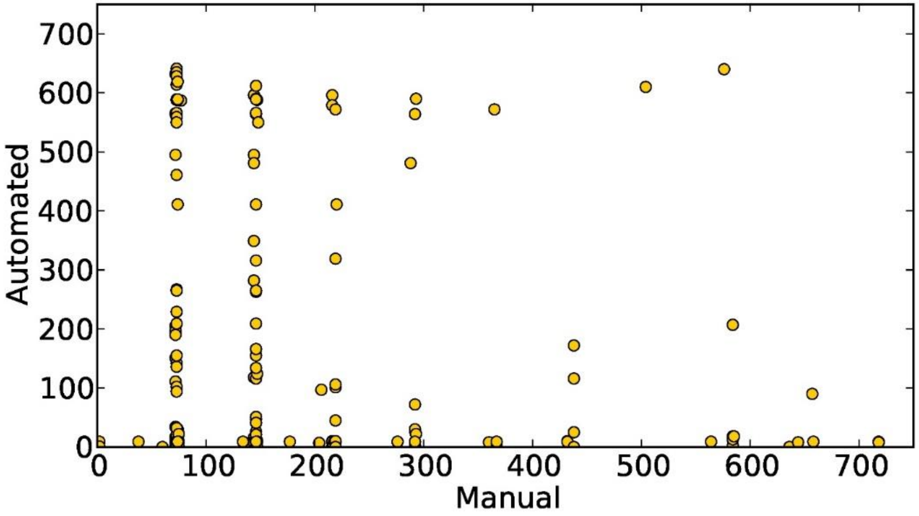
Metrics Used for RQ6 Comparison

- Safety
  - Selects all affected tests
  - RetestAll is always safe
- Precision
  - Selects only affected tests
- Performance
  - Time to select tests and execute them
  - This time should be smaller than time for RetestAll

RQ6 (1)

Comparing manual and automated RTS in terms of precision, safety

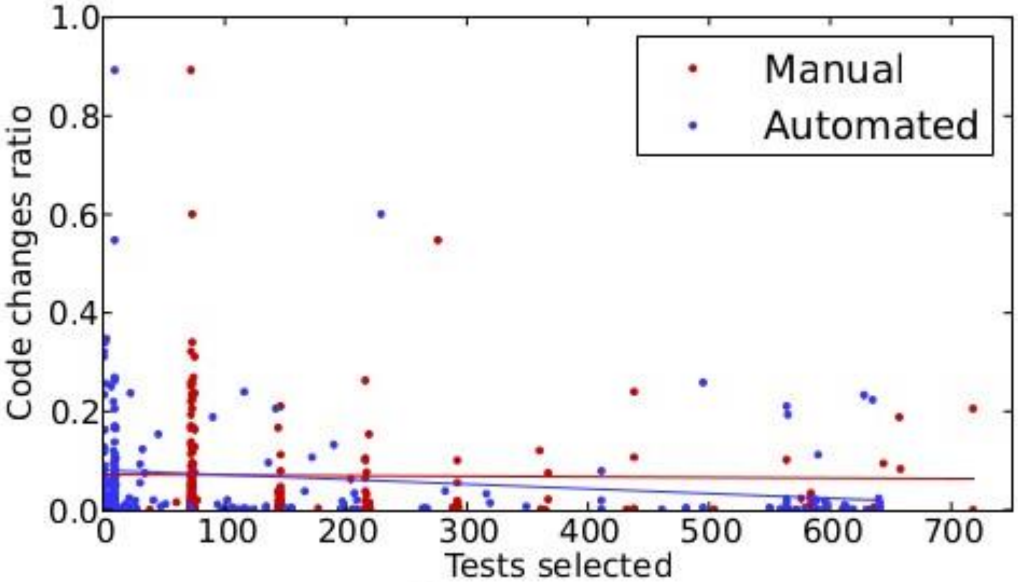
- Assuming automated RTS is safe and precise
- ~70% of the time, Manual RTS > Automated RTS
  - potentially wasting time
- ~30% of the time, Manual RTS < Automated RTS
  - potentially missing faults



RQ6 (2)

Comparing manual and automated RTS in terms of correlation between number of selected tests and code changes

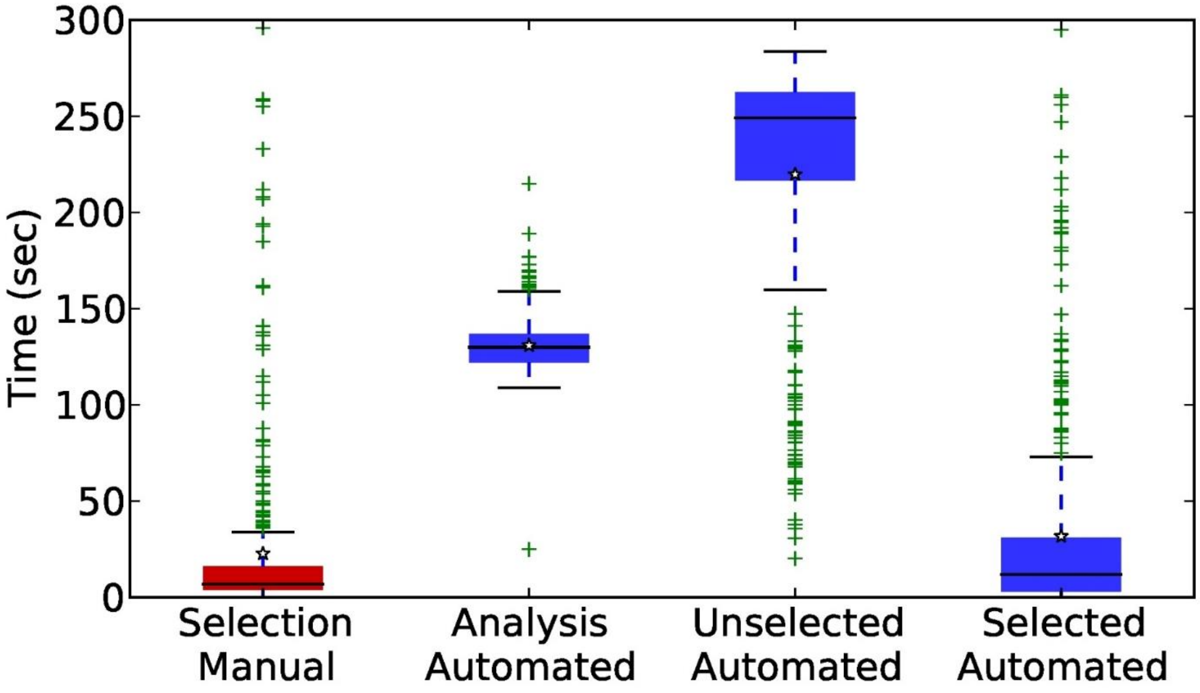
- Very low positive correlation in both
- Slightly more correlation in manual RTS than in automated RTS



RQ6 (3)

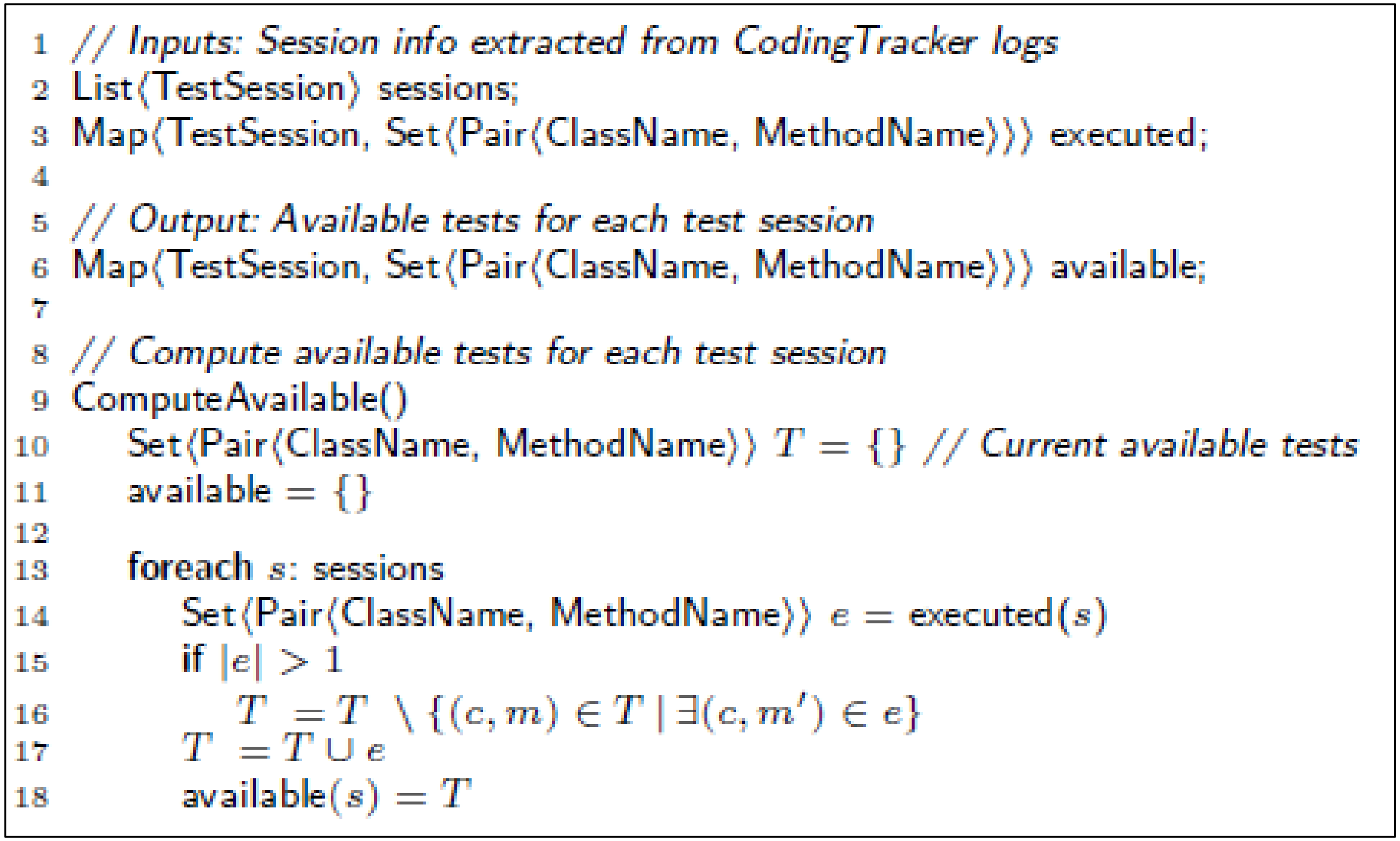
Comparing manual and automated RTS in terms of analysis time

- Automated RTS is slower



Challenges

- CodingTracker doesn’t capture entire state
    - We had to reconstruct state for RQ6
    - We had to approximate available tests



```
// Inputs: Session info extracted from CodingTracker logs
List⟨TestSession⟩ sessions;
Map⟨TestSession, Set⟨Pair⟨ClassName, MethodName⟩⟩⟩ executed;

// Output: Available tests for each test session
Map⟨TestSession, Set⟨Pair⟨ClassName, MethodName⟩⟩⟩ available;

// Compute available tests for each test session
ComputeAvailable()
    Set⟨Pair⟨ClassName, MethodName⟩⟩ $T = \{\}$ // Current available tests
    available = {}

    foreach $s$: sessions
        Set⟨Pair⟨ClassName, MethodName⟩⟩ $e$ = executed($s$)
        if $|e| > 1$
            $T = T \setminus \{(c, m) \in T \mid \exists (c, m') \in e\}$
        $T = T \cup e$
        available($s$) = $T$
```

Our Discoveries (1)

- RQ1: How often do developers perform manual RTS?
- A1: **12 out of 14 developers in our study performed manual RTS**
- RQ2: What is the relationship between manual RTS and size of test suites or amount of code changes?
- A2: **Manual RTS was independent of test suite size, code changes**
- RQ3: What are some common scenarios in which developers perform manual RTS?
- A3: **Manual RTS was most common during debugging**

Our Discoveries (2)

- RQ4: How do developers commonly perform manual RTS?
- A4: **Developers performed manual RTS in ad-hoc ways**
- RQ5: How good is current IDE support in terms of common scenarios for manual RTS?
- A5: **Current IDEs seem inadequate for manual RTS needs**
- RQ6: How does manual RTS compare with automated RTS?
- A6: **Compared with automated RTS, manual RTS is mostly unsafe (potentially missing bugs) and imprecise (potentially wasting time)**

Contributions

- First data showing manual RTS is actually performed
- First study of manual RTS in practice
- First comparison of manual and automated RTS

Conclusions

- Developers could benefit from lightweight RTS techniques and tools
- Need to consider human aspects (e.g. debugging) in RTS research
- Need to balance the existing techniques with the scale at which most developers work
- End goal: adoptable RTS tools

Work in Progress:
Towards Practical Regression Testing

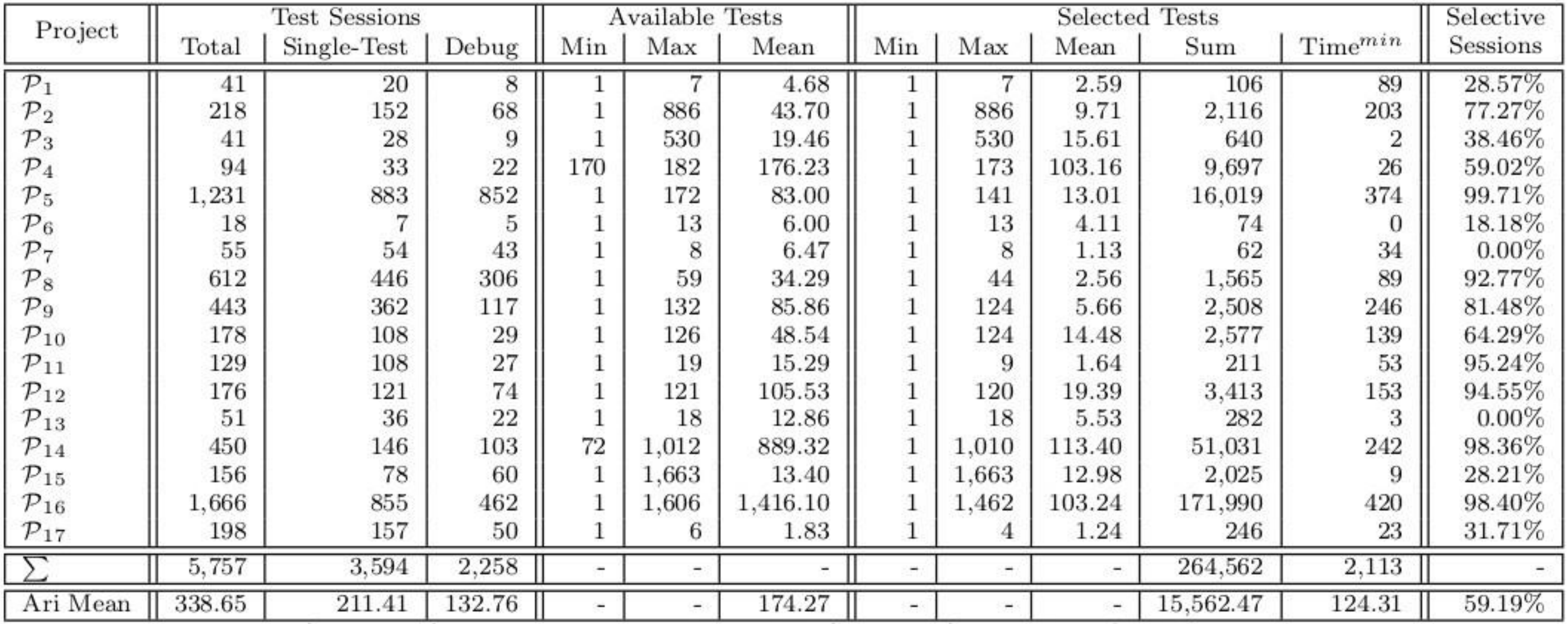
Ekstazi

Led by Milos Gligoric (on job market in 2015)

Questions?

- Do you perform (manual) test selection,
    - If you program…
        - …and test?
- What kind of tool would help you?
- Do you want to collaborate with us?

Extra Slides



Project	Test Sessions			Available Tests			Selected Tests					Selective
	Total	Single-Test	Debug	Min	Max	Mean	Min	Max	Mean	Sum	Time^{min}	Sessions
$\mathcal{P}_1$	41	20	8	1	7	4.68	1	7	2.59	106	89	28.57%
$\mathcal{P}_2$	218	152	68	1	886	43.70	1	886	9.71	2,116	203	77.27%
$\mathcal{P}_3$	41	28	9	1	530	19.46	1	530	15.61	640	2	38.46%
$\mathcal{P}_4$	94	33	22	170	182	176.23	1	173	103.16	9,697	26	59.02%
$\mathcal{P}_5$	1,231	883	852	1	172	83.00	1	141	13.01	16,019	374	99.71%
$\mathcal{P}_6$	18	7	5	1	13	6.00	1	13	4.11	74	0	18.18%
$\mathcal{P}_7$	55	54	43	1	8	6.47	1	8	1.13	62	34	0.00%
$\mathcal{P}_8$	612	446	306	1	59	34.29	1	44	2.56	1,565	89	92.77%
$\mathcal{P}_9$	443	362	117	1	132	85.86	1	124	5.66	2,508	246	81.48%
$\mathcal{P}_{10}$	178	108	29	1	126	48.54	1	124	14.48	2,577	139	64.29%
$\mathcal{P}_{11}$	129	108	27	1	19	15.29	1	9	1.64	211	53	95.24%
$\mathcal{P}_{12}$	176	121	74	1	121	105.53	1	120	19.39	3,413	153	94.55%
$\mathcal{P}_{13}$	51	36	22	1	18	12.86	1	18	5.53	282	3	0.00%
$\mathcal{P}_{14}$	450	146	103	72	1,012	889.32	1	1,010	113.40	51,031	242	98.36%
$\mathcal{P}_{15}$	156	78	60	1	1,663	13.40	1	1,663	12.98	2,025	9	28.21%
$\mathcal{P}_{16}$	1,666	855	462	1	1,606	1,416.10	1	1,462	103.24	171,990	420	98.40%
$\mathcal{P}_{17}$	198	157	50	1	6	1.83	1	4	1.24	246	23	31.71%
$\sum$	5,757	3,594	2,258	-	-	-	-	-	-	264,562	2,113	-
Ari Mean	338.65	211.41	132.76	-	-	174.27	-	-	-	15,562.47	124.31	59.19%

Figure 2: Statistics for projects used in the study; "Selective Sessions" is of multiple-test sessions