MAPPING THE LANDSCAPE OF REFACTORING RESEARCH

AKA – REFACTORING THE REFACTORING

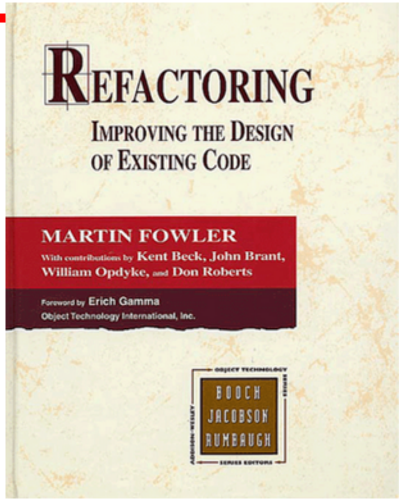
Danny Dig

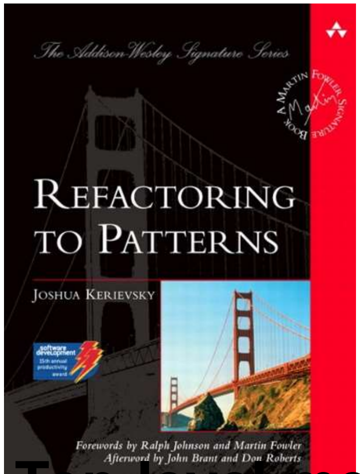
Oregon State University

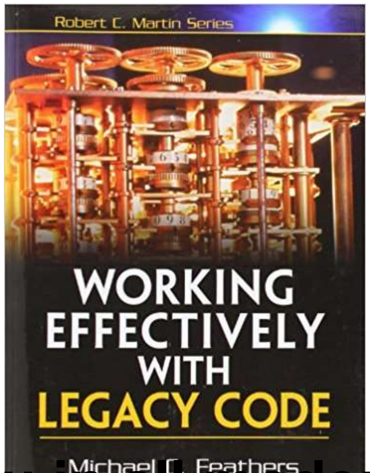
University of Illinois at Urbana-Champaign

What is Refactoring?

“A change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behaviour” – M. Fowler [1999]





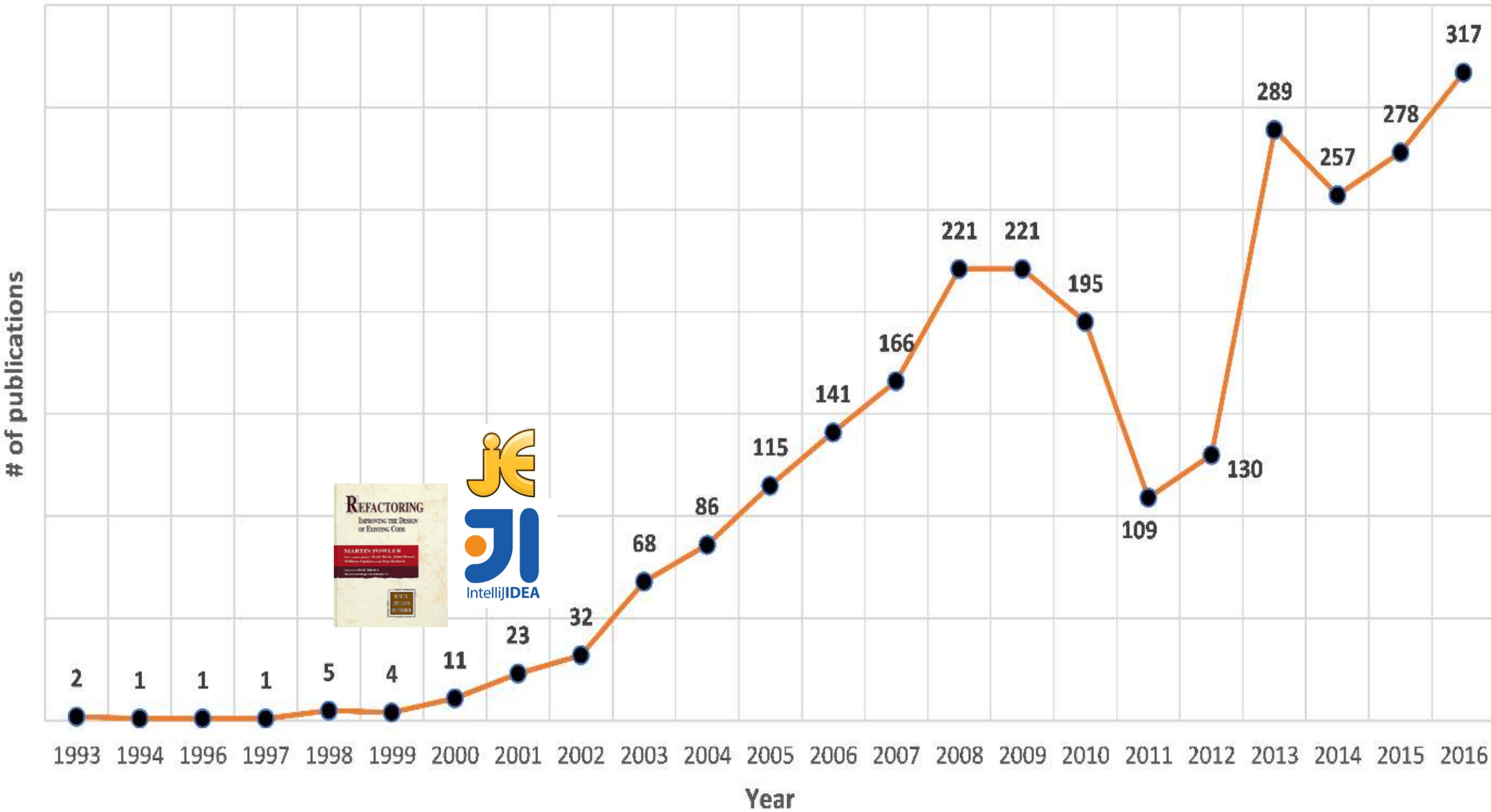


Top-level menu in all modern IDEs

- In 2000, I created the first open-source refactoring tool

Refactoring research growth

2,880 refactoring papers (4,944 authors) since 1990



The Humble Beginnings

First refactoring paper:
- Bill Opdyke and Ralph Johnson [SOPPA'90]: ***Refactoring, an Aid in designing application frameworks and evolving OO systems***

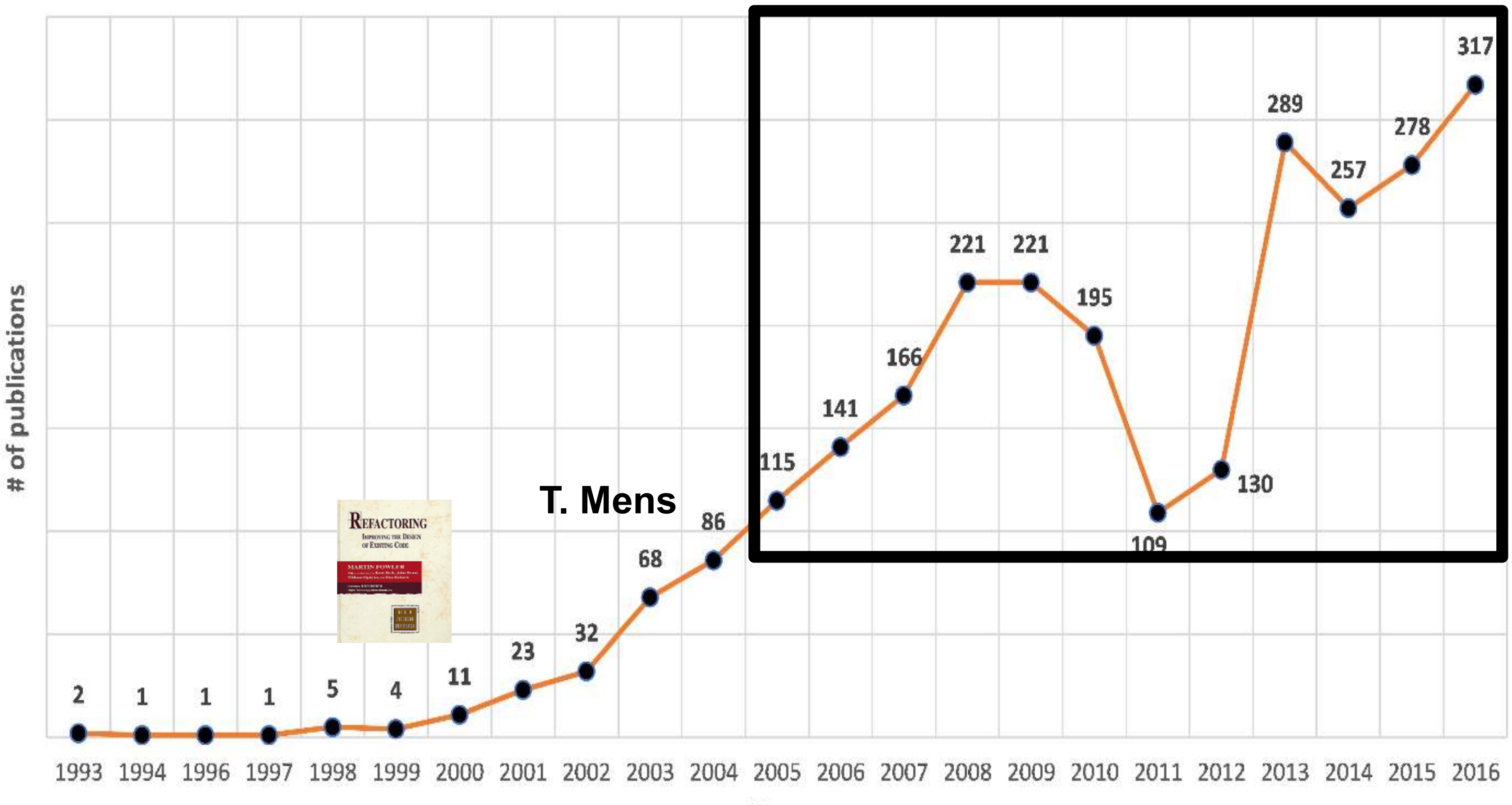
PhD dissertations:
- **Bill Griswold '91 at U of Washington**
- **Bill Opdyke '92 at U of Illinois**
- **Don Roberts '99 at U of Illinois**

Refactoring research hard to publish in early 90s
- conflated with the compiler community

Most recent Decade of Refactoring Research

2,880 refactoring papers since 1990

2,442 papers between 2005-2016



Corpus of Papers

Work done by Marouane Kessentini and his team at Michigan

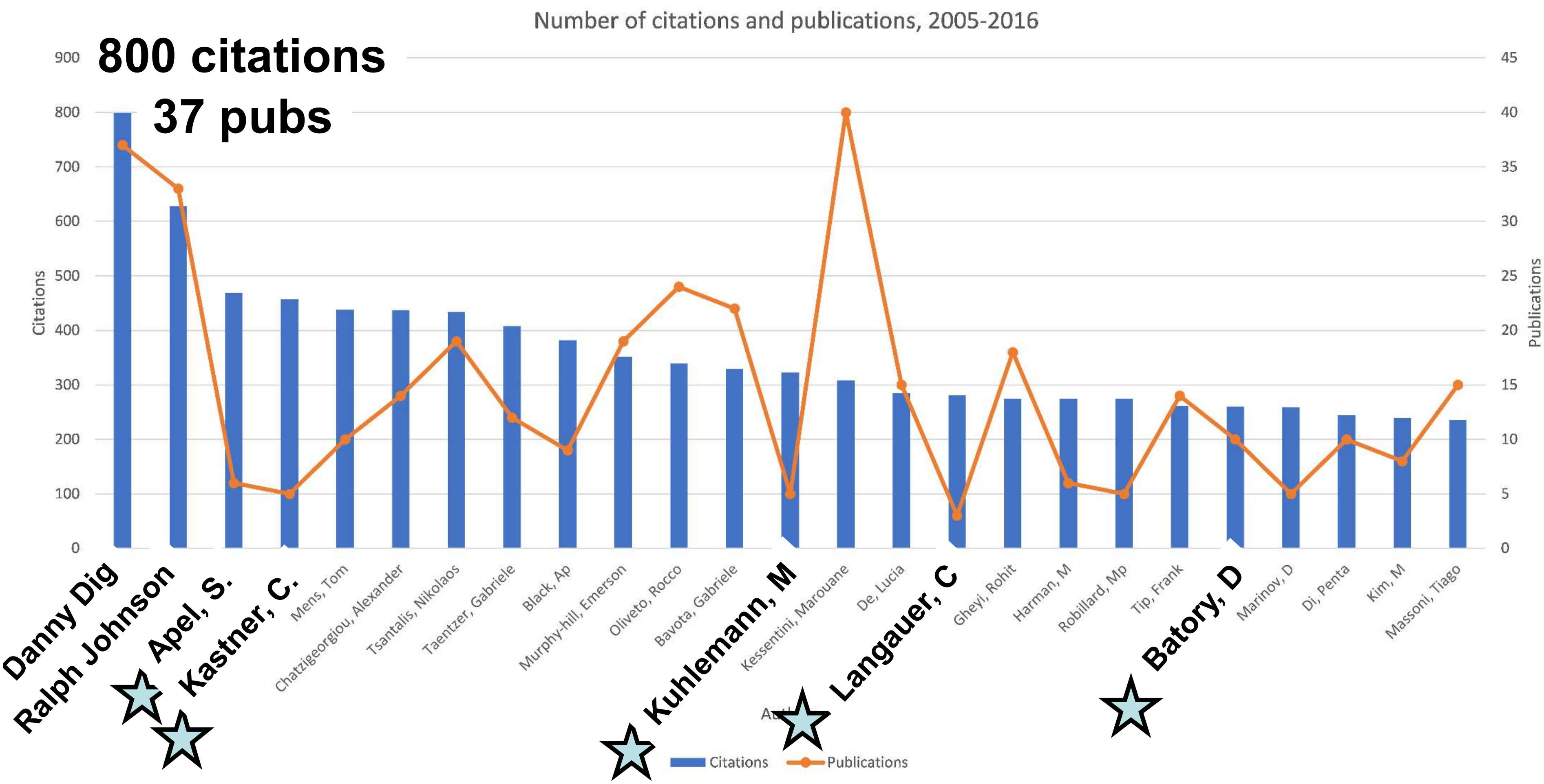
Scopus and Web of Science
- "Refactoring" in title, abstract, and keywords
- yielded 3277 papers

Refactoring definition:
- transformation with behavior preservation

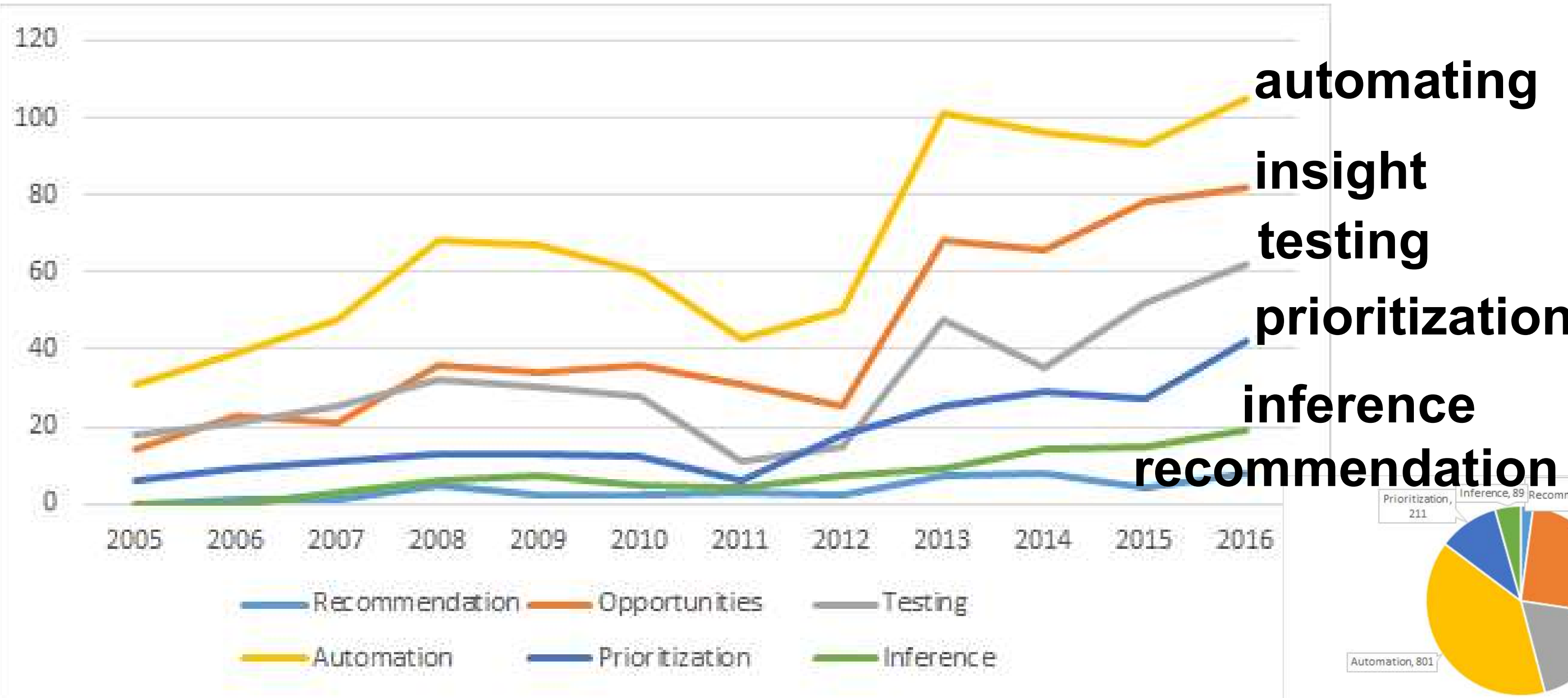
Manual validation of ALL papers:
- each paper analyzed title, abstract (and sometimes content)
- 4 grad students who took a graduate class on Softw QA,
- Kessentini (faculty) looked at the contentious papers

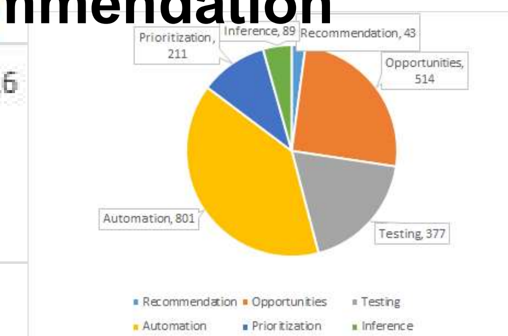
In the end we removed 397 papers

O1: To Grow, Welcome Outsiders, Champions from Other Communities



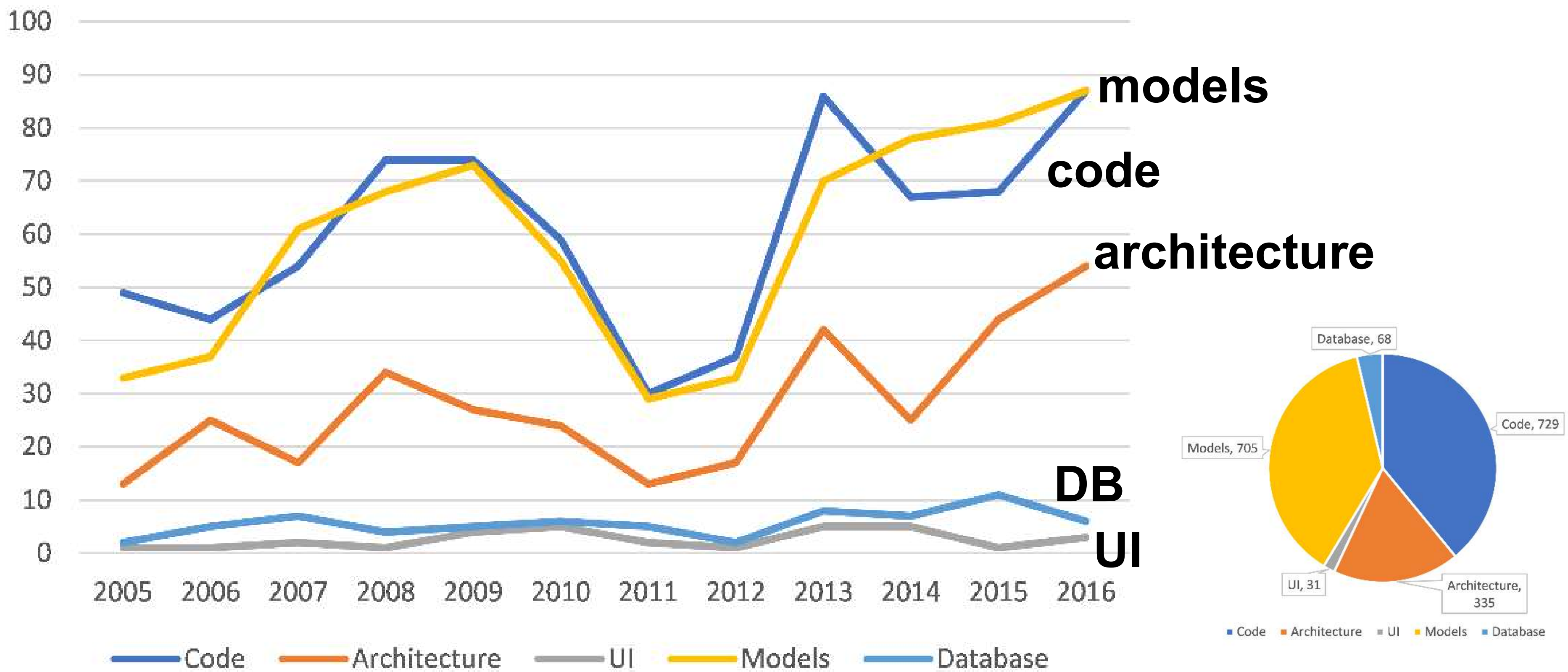
O2: To Grow, Expand Focus of Interest (the WHAT)





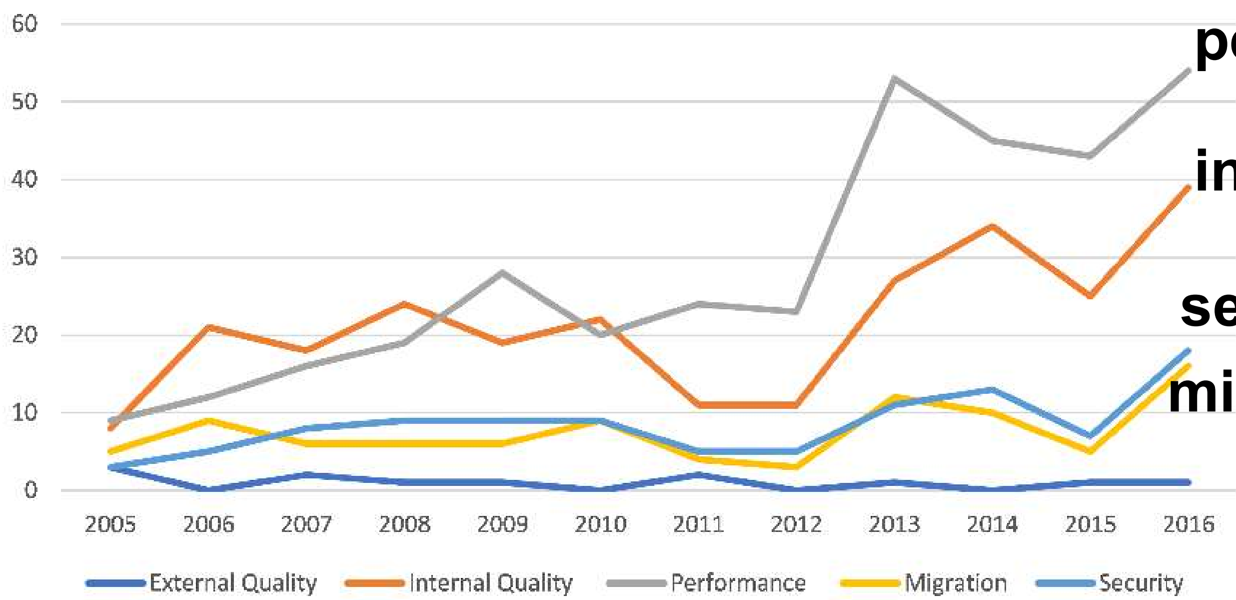
Expand focus to meet new needs that you can serve

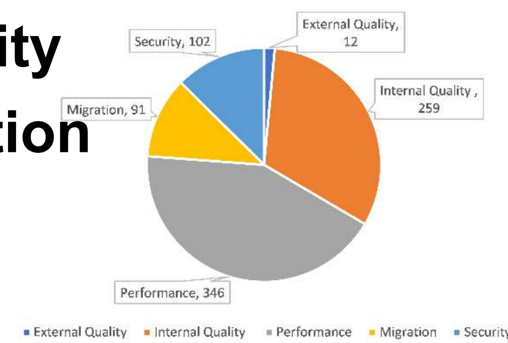
O3: To Grow, Expand the Target Artefacts



Expand target: new refactoring research is about change to the code, models, architecture, DB, UI

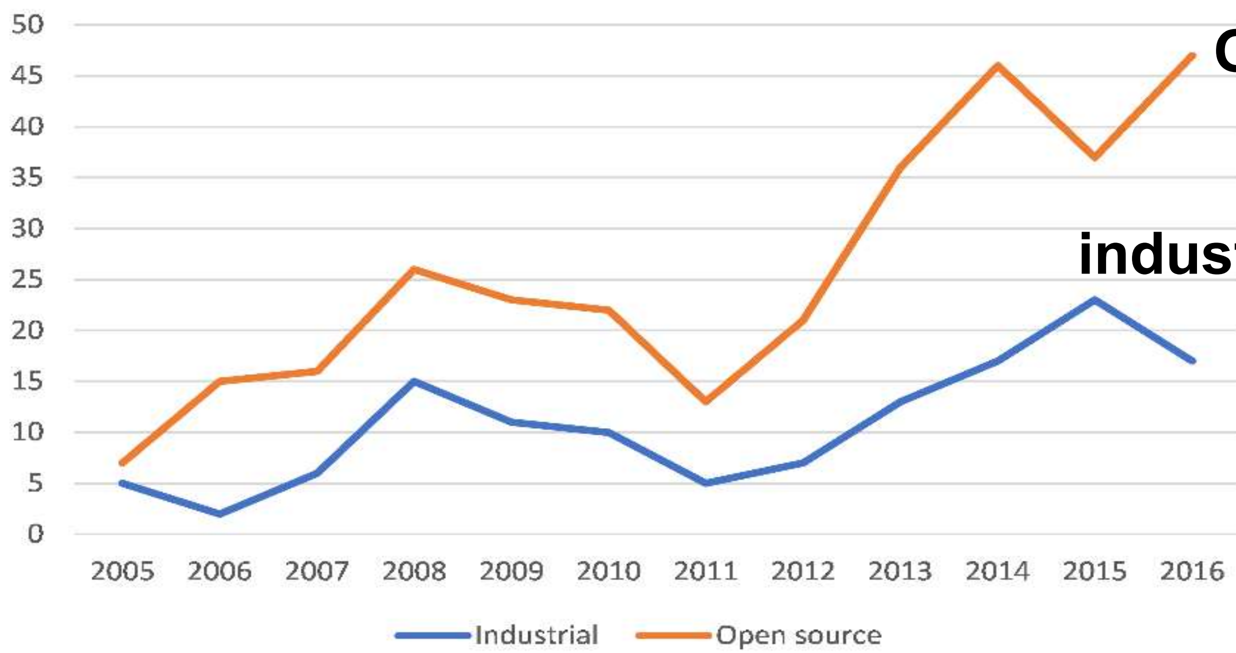
O4: To Grow, Expand Objectives (the WHY)





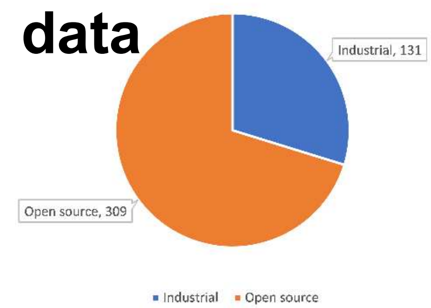
Expand Objectives: new refactoring research is to improve performance, security, migration (beyond internal quality)

O5: To Increase Practical Impact, Work with Industry



Open-source data

industrial data



Industrial collaboration levels:
- surveys with practitioners
- tool validated on industrial codebase
- tool licensed to industry, adopted in products

Big Growth of the Field: Expanding Definition

"A change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behaviour" – M. Fowler '99

Expanded Focus, Objectives, Techniques

"Automation/insight/testing/prioritization of changes to the artifacts of software to improve non-functional requirements and without changing its proper, intended behaviour" – D. Dig '17

Communities that thrive are going to be more accepting of new ideas

Big Growth Enabled by Community Engineering

Industrial champion(s): M. Fowler, Kent Beck, Ward Cunningham

Complementary skills: tool builders, paper writers, curators

Mindset for industrial collaboration and adoption

Shared platform:
- Eclipse (Erich Gamma + Frank Tip), analysis frameworks

Community infrastructure: 7 Refactoring Workshops, Dagstuhl
- **first workshop in 2007, 50+ participants, 32 posters**
- **invited all major IDE providers**
- **growing new leaders**

The Role of Refactorings in API Evolution

Danny Dig and Ralph Johnson
Department of Computer Science
University of Illinois at Urbana-Champaign
201 N. Goodwin
Urbana, IL 61801, USA

Published ICSM 2005, receives Most Influential Paper Award in 2015

Abstract

Frameworks and libraries change their APIs. Migrating an application to the new API is tedious and disrupts the development process. Although some tools and ideas have been proposed to solve the evolution of APIs, most updates are done manually. To better understand the requirements for migration tools we studied the API changes of three frameworks and one library. We discovered that the changes that break existing applications are not random, but they tend to fall into particular categories. Over 80% of these changes are refactorings. This suggests that refactoring-based migration tools should be used to update applications.

component and application developers. What is a suitabl representation for the changes that happened in a compo nent? Can it be gathered automatically? Does this represer tation carry both the syntax and the semantics of changes Can it lead to safe, automatic updating of component-base applications? How much of the effort spent on updatin component-based applications can be saved?

Although there are principles of software evolution th are true for software in any language, programming lar guages have an impact on software evolution. We are pa ticularly interested in the evolution of object-oriented con ponents (we refer to both library and framework as con ponent, unless a distinction is necessary). Classes contain mixture of private and public methods. The public methoc are the ones that are meant to be used by application prc grammers. The set of public methods of a class library mak

Breaking API Changes Cause Problems for Applications

eclipse

51

LOG4J

38

Struts

Web Application Development

136

11

JHotDraw

58

High-level goal: reduce the burden of reuse on maintenance

Either reduce the amount of change,
Or reduce the cost of adapting to change

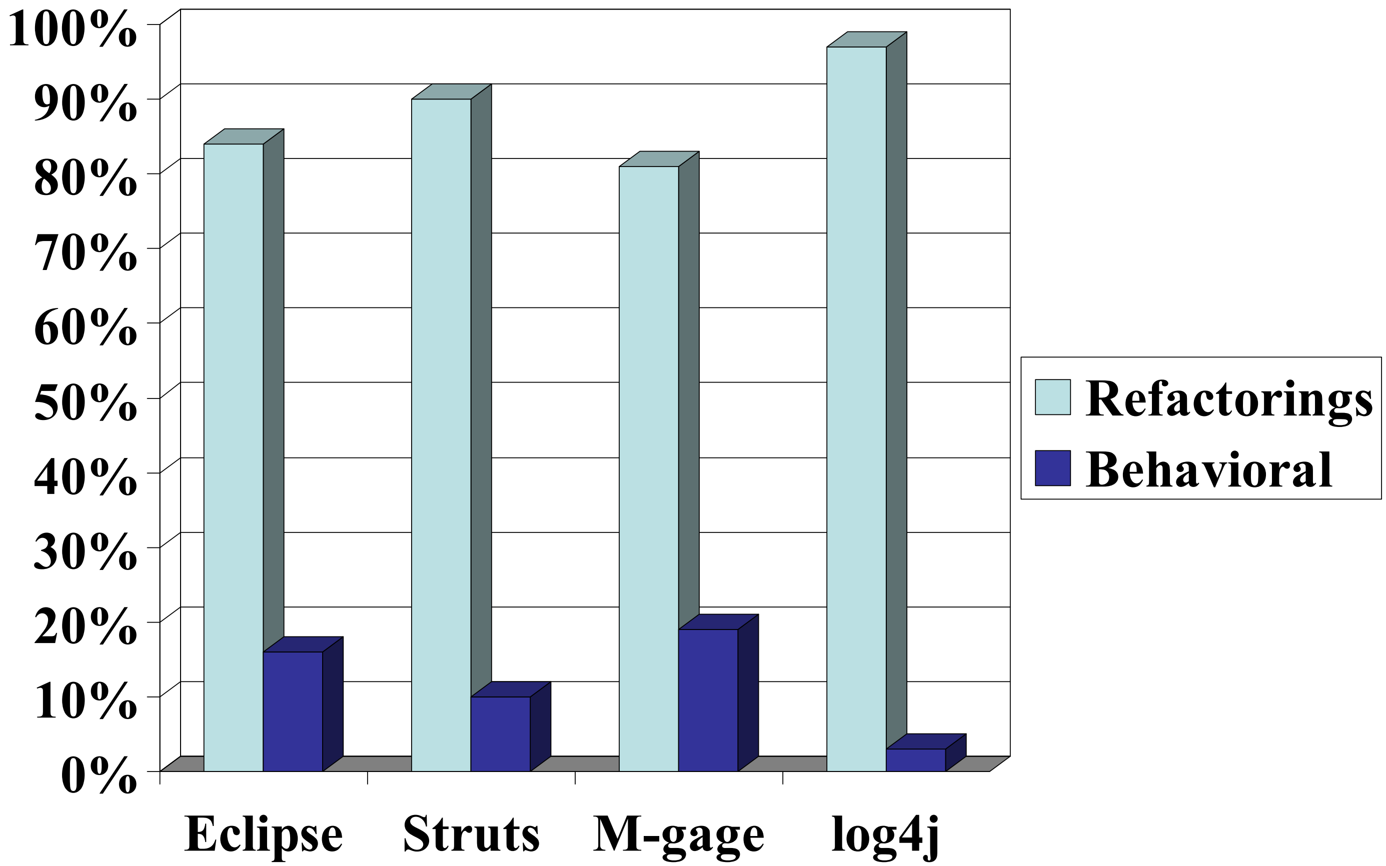
RQ1: Which component changes break compatibility?

RQ2: What is a suitable representation for these changes?

RQ3: Does this representation carry both the syntax and the semantics of changes?

We studied the evolution of real components

Main Result: majority of breaking API changes are Refactorings



Monitoring Refactorings as Objects of Change

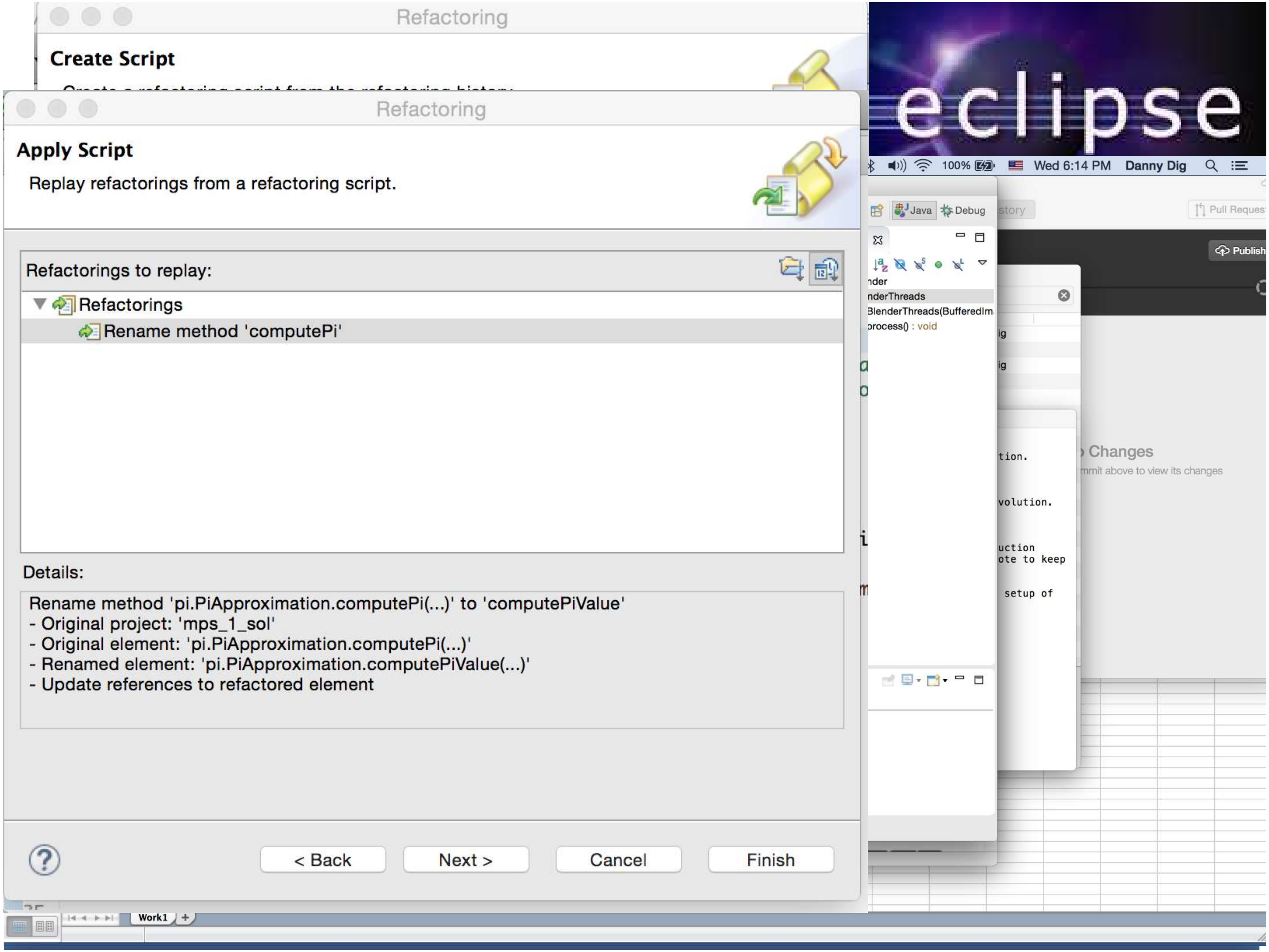
Most of API breaking changes are refactorings

+

Refactorings carry both the syntax and semantics of the change

Monitor component changes that are refactorings

Replay them on the client code


Refactoring
Create Script
Refactoring
Apply Script
Replay refactorings from a refactoring script.
Refactorings to replay:
Refactorings
Rename method 'computePi'
Details:
Rename method 'pi.PiApproximation.computePi(...)' to 'computePiValue'
- Original project: 'mps_1_sol'
- Original element: 'pi.PiApproximation.computePi(...)'
- Renamed element: 'pi.PiApproximation.computePiValue(...)'
- Update references to refactored element
< Back
Next >
Cancel
Finish
eclipse
100%
Wed 6:14 PM
Danny Dig
Java
Debug
Pull Request
Publish
nderThreads
BlenderThreads(BufferedIm
process() : void
Changes
mmit above to view its changes
Work1

Key Questions Regarding Refactorings as Objects of Change

Q1: How many API changes are caused by refactorings?

A: more than 80% **[Dig et al.: ICSM'05, JSME'06]**

Q2: Can refactorings be detected automatically?

A: practical accuracy **[Dig et al. ECOOP'06, Kim et al. ICSM'10, Tsantallis et al. – ICSE'18]**

Q3: Can refactorings be incorporated automatically?

A: refactoring-aware merging **[Dig et al.: ICSE'07, TSE'08]**

Q4: Can applications be shielded from Refactoring API changes?

A: Binary adaptation **[Dig et al. ICSE'08, Savga et al ICSE'08,]**

Work by others

Love/hate relationship with refactoring

API analysis [Robbes et al: ICSE'11,Yu et al:ASE'11,Hybl et al:OOPSLA'13, Robbes et al: FSE'12, Negara et al: ECOOP'13, Vasquez et al: FSE'13, Pinto et al:FSE'12, Kapur et al: OOPSLA'10, Businge at al: SQC15, Batory: CC'07]

Work on automatic upgrades [Nguyen et al: OOPSLA'10, Dagenais et al: ICSE'08, Li et al: ASE'12, Wu et al: ICSE'10]

HotSWUp series of workshops [HotSWUp'08, HotSWUp'09, HotSWUp'11, HotSWUp'12, HotSWUp'13, HotSWUp'14]

Study [Cossete & Walker - FSE'12]**: reactive/postmortem techniques have success rate 20%**

Reflections and Lessons I am Learning

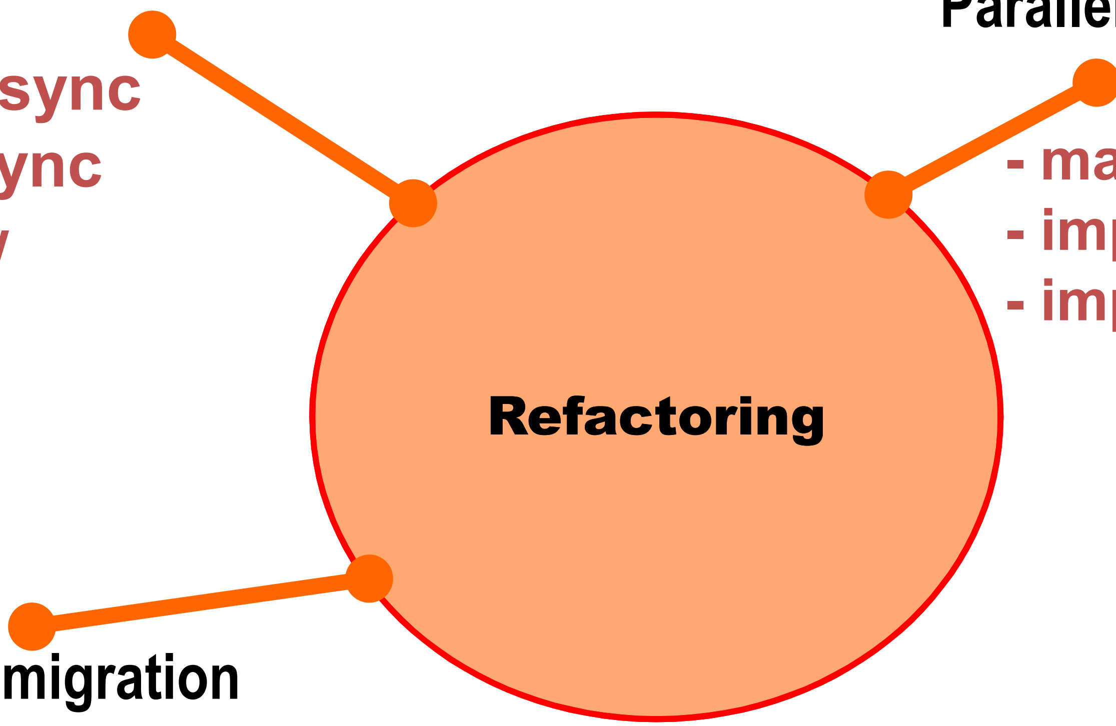
On Aug 5, 2015 …

A life of significance: intentionally serve others

L1: Work in Your Strength Zone but Reinvent Yourself

Mobile
- -add async
- -fix async
- -privacy

Parallelism & Concurrency
- - make thread-safe
- - improve throughput
- - improve scalability



Library migration
- - upgrade APIs

Principles for changing between different programming models

L2: Find Your Dream and then Live It

Refactoring

Automating

-ship with official

- hundreds of accepted patches

- first open-source refactoring

Inferring

- used at
- dozen labs

founded Workshop on Refactoring Tools, HotSwUp, Dagstuhl S.

Understanding

- shaped APIs in Java and .NET official concurency libraries

-learnparallelism.net 150,000+ visitors

Testing

L3: Proactively Look for Opportunities, but Be Flexible

Expected Company	Actual Company	Expected Target	Actual Target
IBM	ORACLE	Lambda Expressions	Lambda Expressions
Google	Google	Async Programming	Type migration at scale

L4: To Grow Others, Grow Yourself

Do you have a plan for your personal growth?
How do you get better at what you do?
How do you improve your relationships?
How do you gain insight?

My Most Important Investment

Michael Hilton (PhD’17, now at CMU)
Semih Okur (PhD’16, now at Microsoft)
Yu Lin (PhD’15, now at Google)
Stas Negara (PhD ‘13, now at Google)
Ameya Ketkar (PhD)
Tom Dickens (PhD)
Sruti Srinivasa (PhD)
Shane McKane (MS’17, now at Intel)
Mihai Codoban (MS ‘15, now at Microsoft)
Kendall Bailey (MS ‘15, now at Intel)
Cosmin Radoi (MS ‘13, now PhD student UIUC)
Sandro Badame (MS ‘12, now at Google)
Fredrik Kjolstad (MS 2011, now PhD student MIT)
Binh Le (MS 2009, SW developer)
Can Comertoglu (MS 2009, now at Microsoft)

Jacob Lewis (Summer’16 – ‘17)
Jonathan Harijanto (Summer’16 –’17)
Lily Mast (Summer’15)
Elias Rademacher (Summer’15 - current)
Nicholas Nelson (Summer 2014-15)
Sean McDonald (Summer’14 –Fall’15)
Hugh McDonald (Summer’14 – Fall’15)
Alexandria Shearer (Summer’12)
Kyle Doren (Summer’12)
Lyle Franklin (UIUC, Summer’12)
Alex Gyori (UIUC, Summer’12)
Yuwei Chen (UIUC, Spring 2012)
Anda Bereckzy (UIUC, Fall’11-Spring’12)
Alex Sikora (UIUC, Fall’11)
Jack Ma (UIUC, Summer’11)
Lorand Szacaks (UIUC, Summer’11)
Caius Brindescu (UIUC, Summer’11)
Mihai Codoban (UIUC, Summer ‘11)
Mihai Tarce (UIUC, Summer’09)
Cosmin Radoi (UIUC, Summer’09)
John Marrero (MIT, Spring’08 – Summer’08)

Call to Action

Big growth enabled by "refactoring" the refactoring

Teamwork makes the dream work

Change is the only guaranteed constant

L1: work in your strength zone, but reinvent yourself
L2: find your dream and then live it
L3: proactively look for opportunities, be flexible
L4: to grow others, first grow yourself