GROWTH LESSONS FROM THE REFACTORING COMMUNITY

Danny Dig

June 29, Virtual
IWoR 2020

University of Colorado
Boulder

Go to **www.menti.com** and use the code **80 08 24**

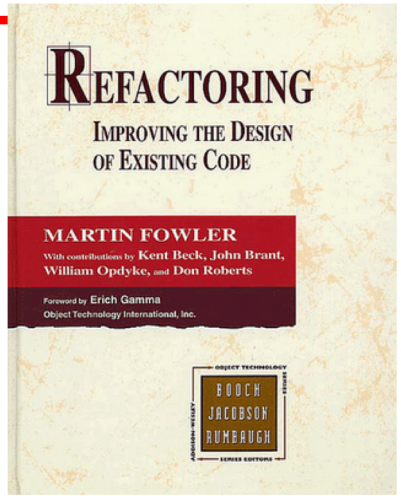
On Aug 5, 2015 …

From personal success to significance

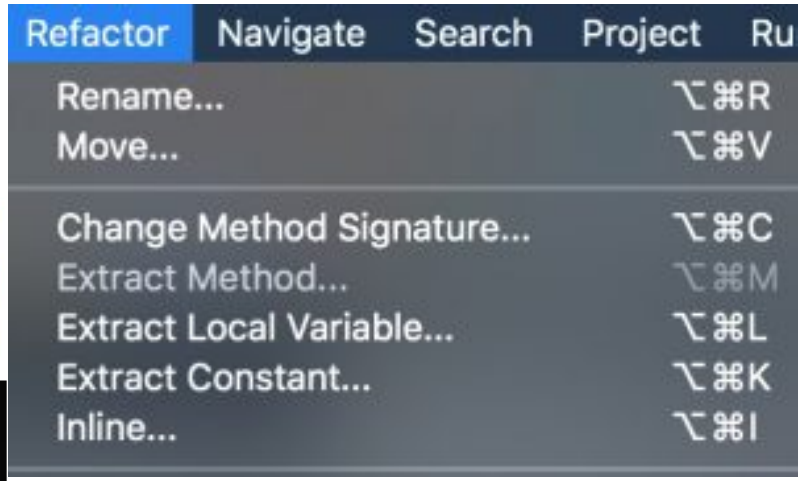
From a ladder climber to a ladder holder

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

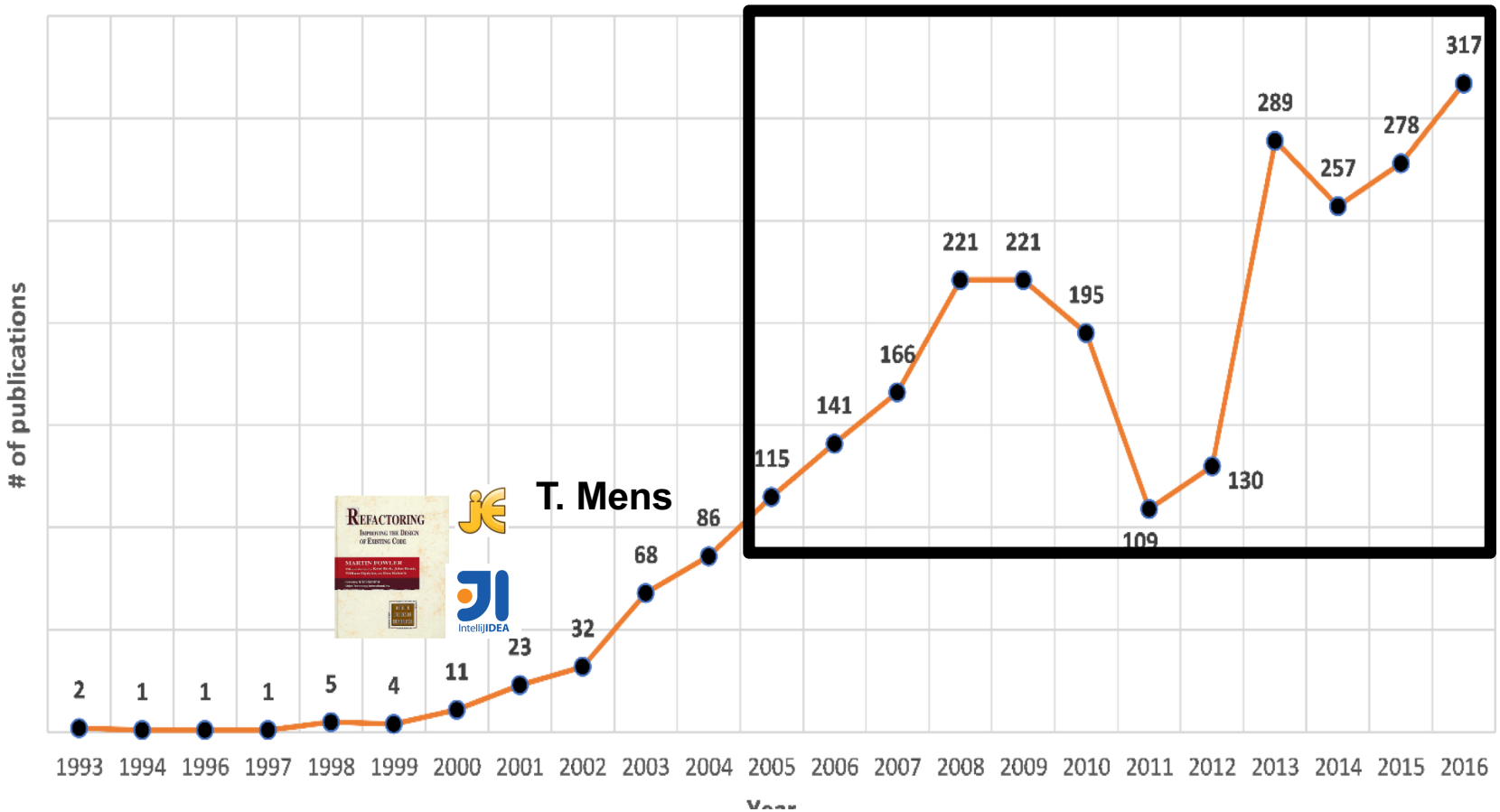
In 2004, I joined the team

To go fast, go alone. To go far, go with others.

A 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

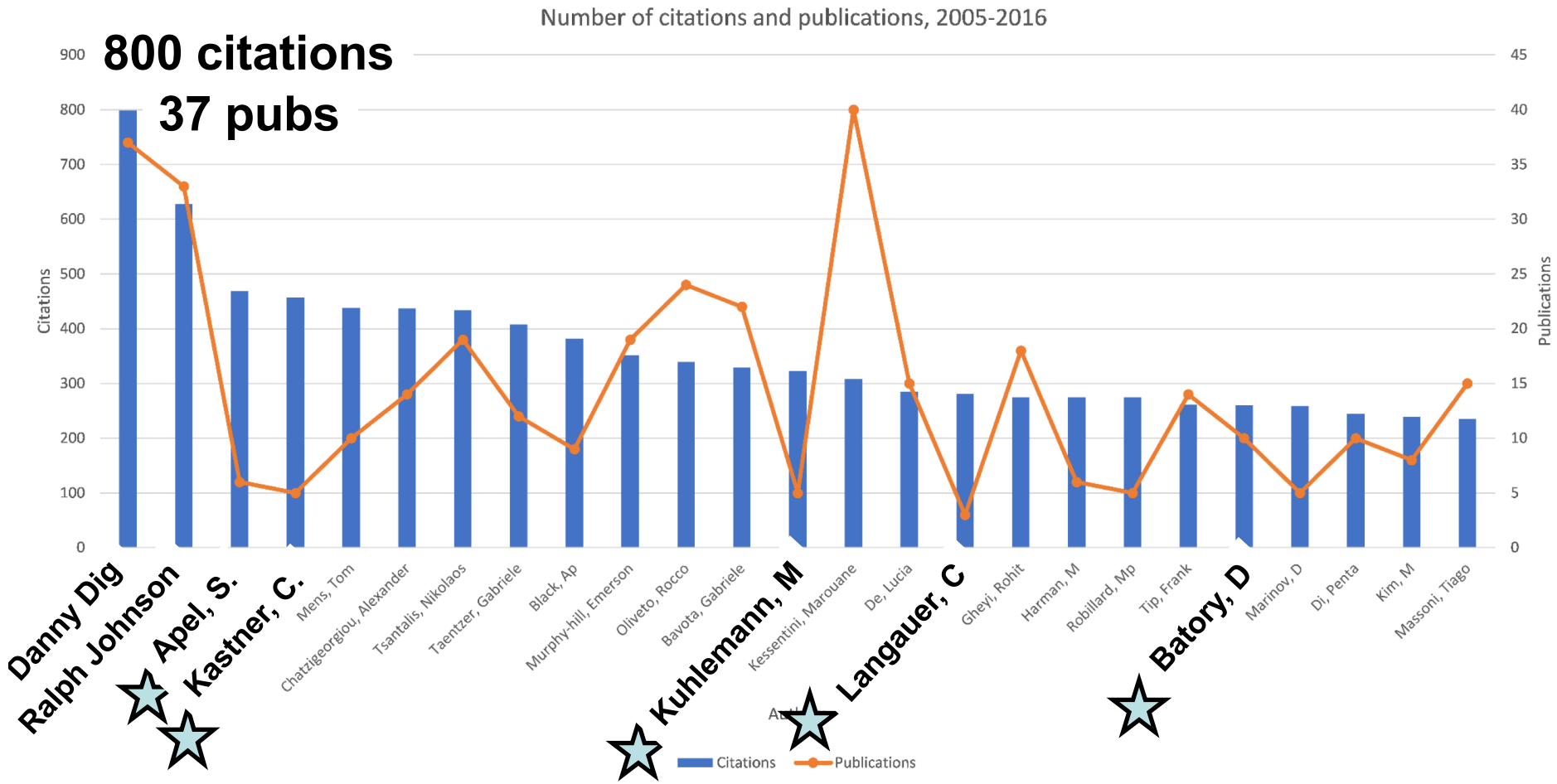
Scopus and Web of Science
- **"Refactoring" in title, abstract, and keywords**
- **yielded 3277 papers**

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**

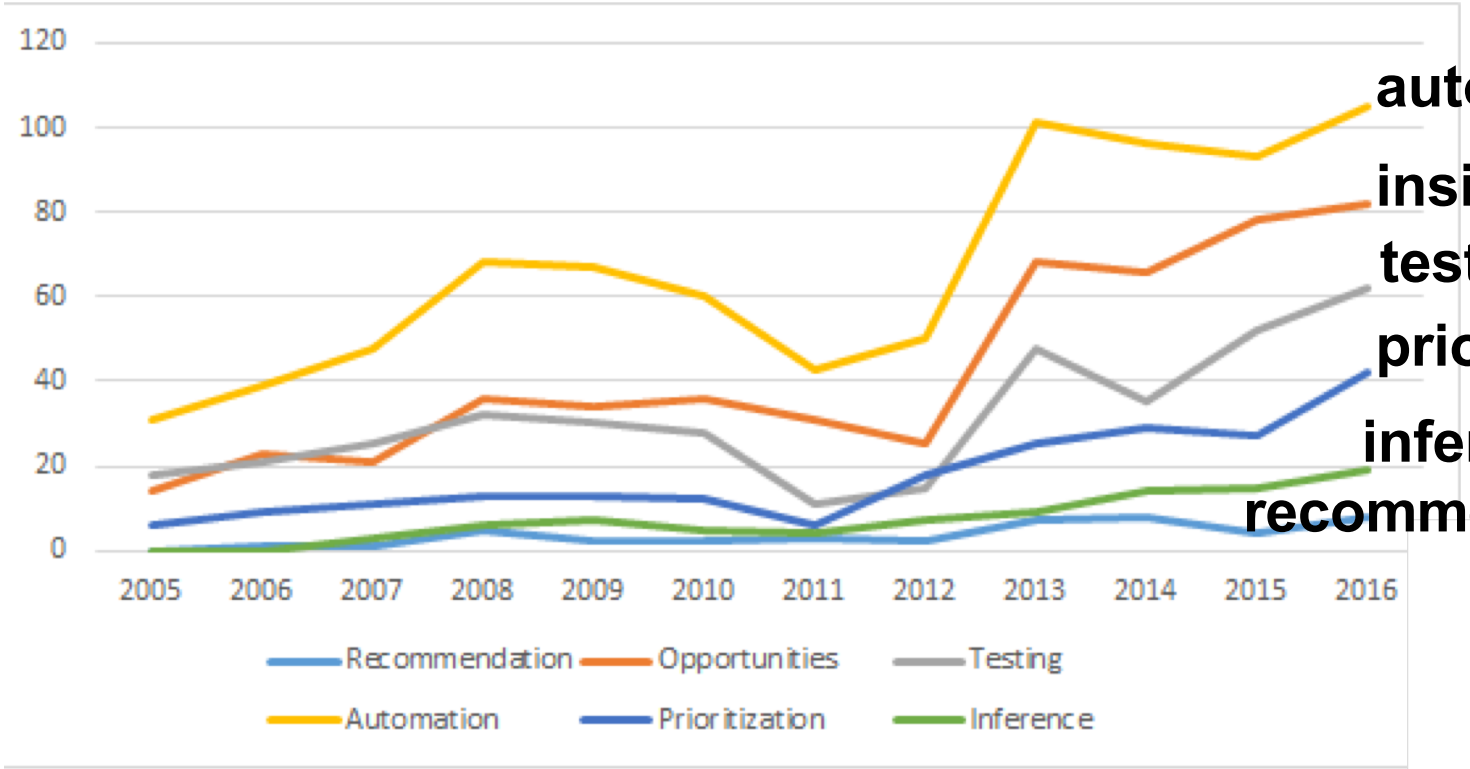
In the end we removed 397 papers

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



For PhD committee, invite 1-2 outsiders of your area

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



automating

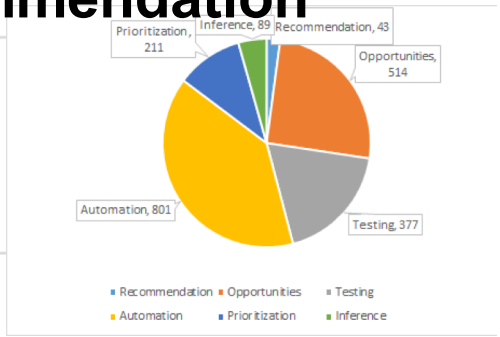
insight

testing

prioritization

inference

recommendation



Expand to meet the needs of new audiences

Our New Focus on Automation

Scalability 1.0: Refactoring to Design Patterns contain hundreds of lower-level refactoring steps [ICSE'16]
- 10x faster than state-of-the-art IDE refactorings

Scalability 2.0: Ultra-large scale refactoring for codebases of Hundreds of Millions LOC (e.g., Google Microsoft scale)
- whole-program analysis is not feasible

The next generation of global, distributed refactoring [ICSE19]
- MapReduce on the cloud: scalable, safe, useful

New Work on Inferring Refactorings

Nikos Tsantallis' RefactoringMiner: commit-based detection [ICSE'18]

- **No similarity thresholds**
- **High accuracy**: 98% precision, 87% recall
- **Ultra-fast**: 58ms on median per commit
- **Better than previous state-of-the-art**: +22% precision, 7x faster

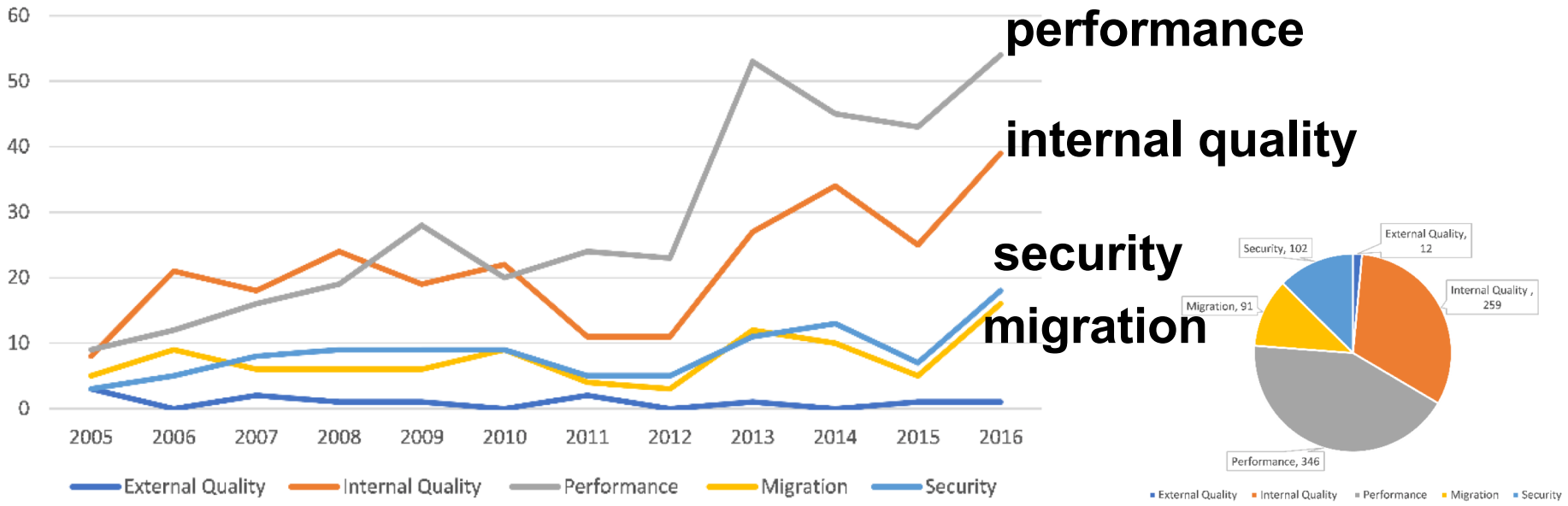
Largest and least biased refactoring oracle up to date

- **6K+** true refactoring instances, **185** open-source projects

Enabling other researchers:

- **21 papers by others, within 1 year of our release**

O3: To Grow, Expand Objectives (the WHY)

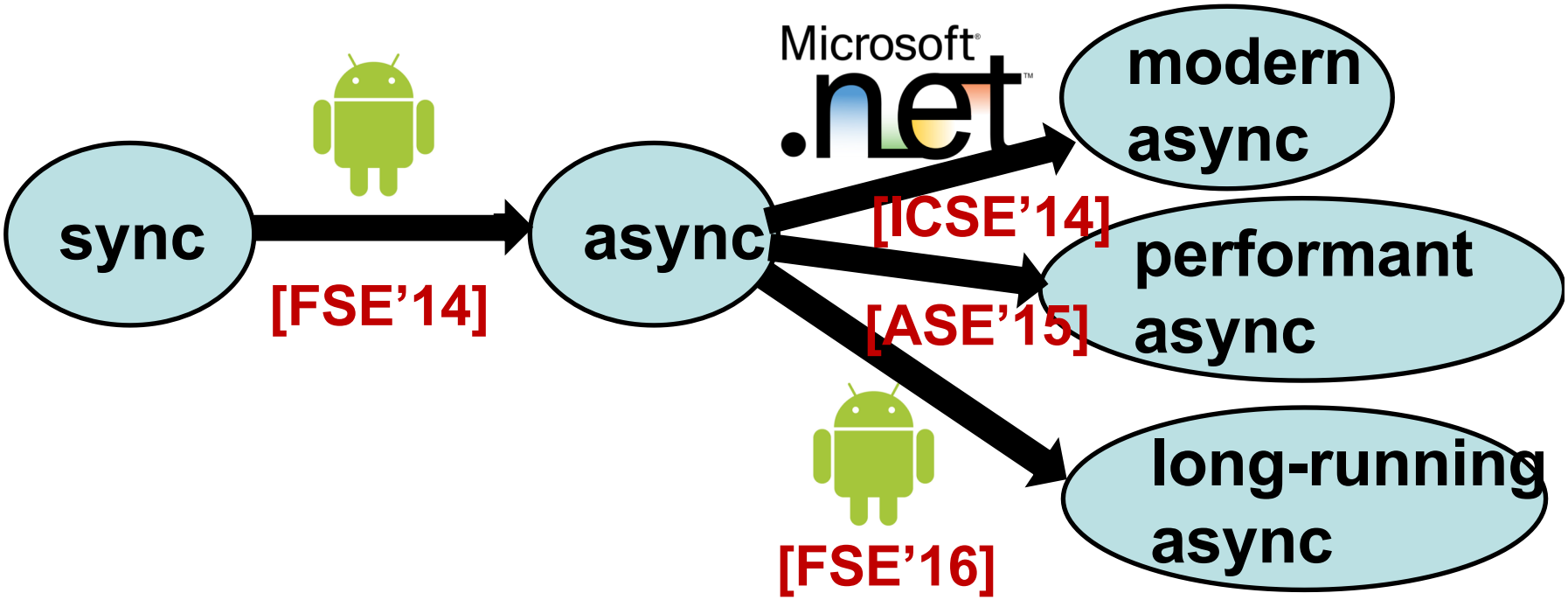


Expand Objectives: new refactoring research is to improve performance, security, migration (beyond internal quality)

Overview of Our Refactorings for Asynchronous Programming for Mobile Apps

Slow operations freeze mobile apps and frustrate users
- 75% of performance bugs in Android [Li et al., ICSE'14]

Culprit: long running operations running in the main UI thread
Solution: refactoring for asynchronous execution



Overview of Our Refactorings for Parallelism

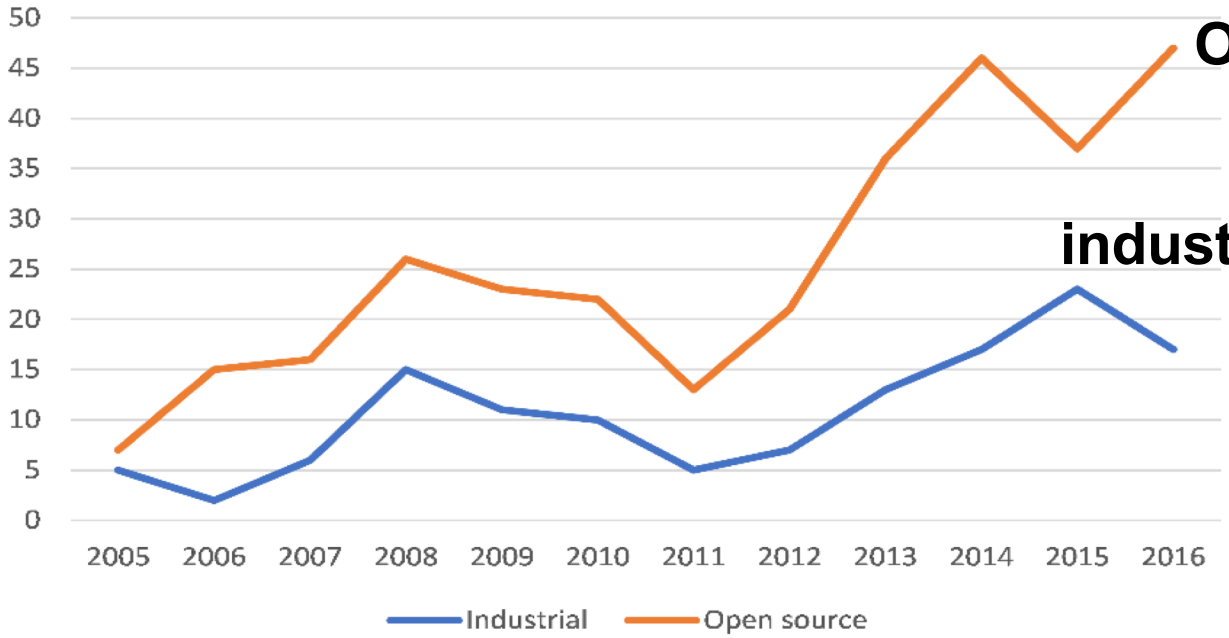
Refactorings for thread-safety
- make class immutable [ICSE'11]
- convert to Atomic* classes [ICSE'09]
- use concurrent collections [ICSE'09]
- infer region annotations [ASE'09]
- atomic check-then-act operations [ICST'13]

Interprocedural analyses:
- control, data-flow
- points-to
- constraint-based

Refactorings for throughput
- parallel recursive divide-and-conquer [ICSE'09]
- loop parallelism via ParallelArray [OOPSLA'10]
- loop parallelism via lambda functional operators [FSE'13]

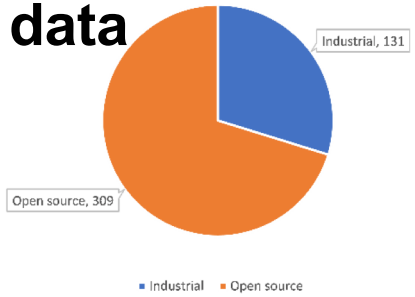
Refactorings for scalability
- Atomic*, concurrent collections [ICSE'09]

O4: To Increase Practical Impact, Go the Extra Mile



Open-source data

industrial data



Industrial collaboration levels:

- **surveys with practitioners**
- **tool validated on industrial codebase**
- **tool licensed to industry, adopted in products**

"There are no traffic jams on the extra mile"

What is Your Dream?
Mine is Practical Impact on SW Development

Automating

-ship with official

NetBeans IDE

Visual Studio

- hundreds of accepted patches

- first open-source refactoring

Google

Refactoring

Inferring

- used at Google

IBM

- dozen labs

founded Workshop on Refactoring Tools, HotSwUp, Dagstuhl S.

Understanding

- shaped APIs in Java and .NET official concurency libraries

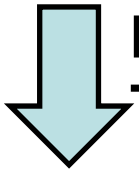
Testing

ORACLE

-learnparallelism.net
150,000+ visitors

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" – Danny Dig '19

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

Big Growth Enabled by Community Engineering

Industry champions: Martin Fowler, Kent Beck, Ward Cunningham, Joshua Kerievsky, Michael Feathers, Uncle Bob

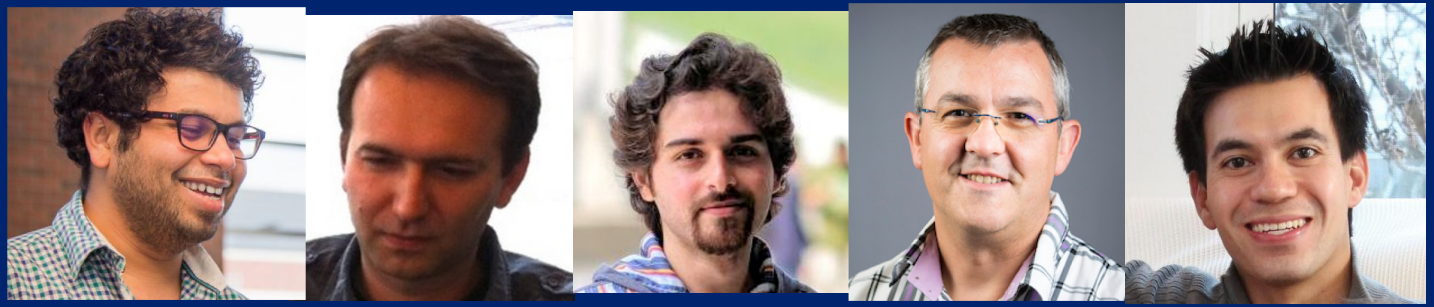
Complementary skills: tool builders, paper writers, curators

Mindset for industrial collaboration and adoption

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

Community: 10 Refactoring Workshops, 1 Dagstuhl
- **first workshop in 2007, 50+ participants, 32 posters**
- **invited all major IDE providers**
- **growing new leaders**

Type Migration in Ultra-large-scale Codebases



Ameya Ketkar — Ali Mesbah — Davood Mazinanian — Danny Dig — Eddie Aftandilian

Google

Why perform type migration?

- **Better performance**

`Function<Double,Double> -> DoubleUnaryOperator` [Ketkar OOPSLA '17]

`Integer -> int`

- **Improve Energy Efficiency**

`HashMap -> ArrayMap –> SparseArray` [Saborido et al., EMSE '18]

- **Better functionality**

`T -> Optional<T>`

- **Library Migration**

`log4J -> sl4j` [Teyton et al., JSEP '14]

- **Fixing API breaking changes**

Updating or moving fields and method [Dietrich et al., ICSME '14]

Existing Refactoring Tools
for Type Migration

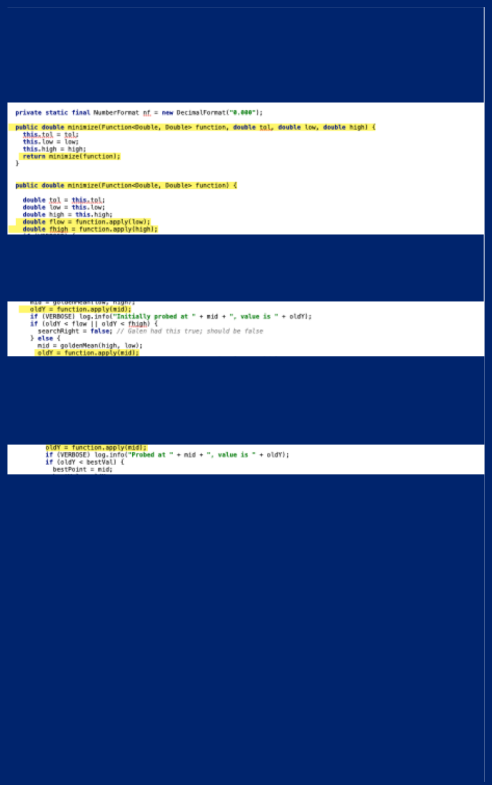
Tool	Config-ility	Safety	Scalability
IntelliJ IDEA	🟡	🔴	🔴
Refactor by example (Refaster)	🟢	🔴	🟢
Type Constraints	🟡	🟢	🔴
T2R (Our approach)	🟡	🟢	🟢

Configurability: Custom type migrations

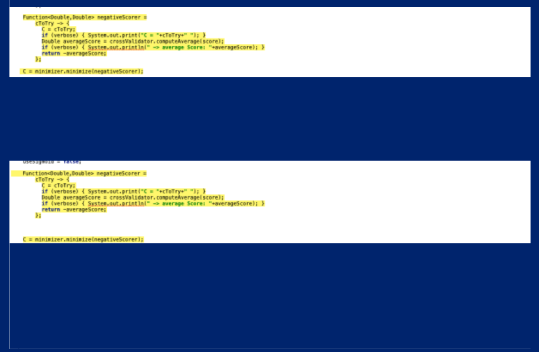
Safety: Type correctness, completeness of mappings

Scalability: 300M LOC

Key Novel Idea 1: Decrease Search Space



GoldenSectionLineSearcher



SVMLightFactory

Reason about elements of the *"Source"* type of the migration

`Function<Double,Double>`

↓

`DoubleUnaryOperator`

Reason about:

- **`Function<Double, Double>`**
- **`Double`**

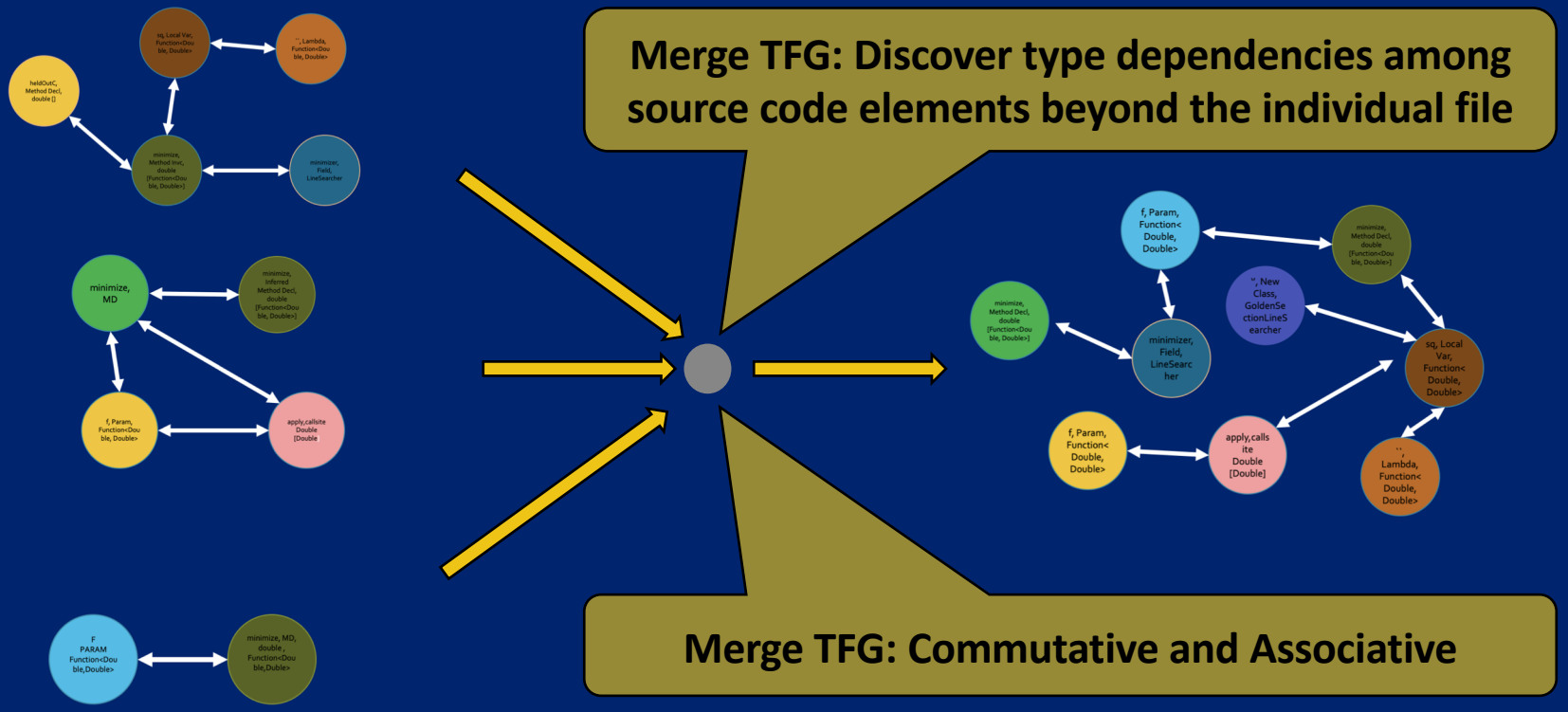
Key Novel Idea 2: Map-Reduce amenable

Type Fact Graphs (TFG)

Nodes: Program elements of type T

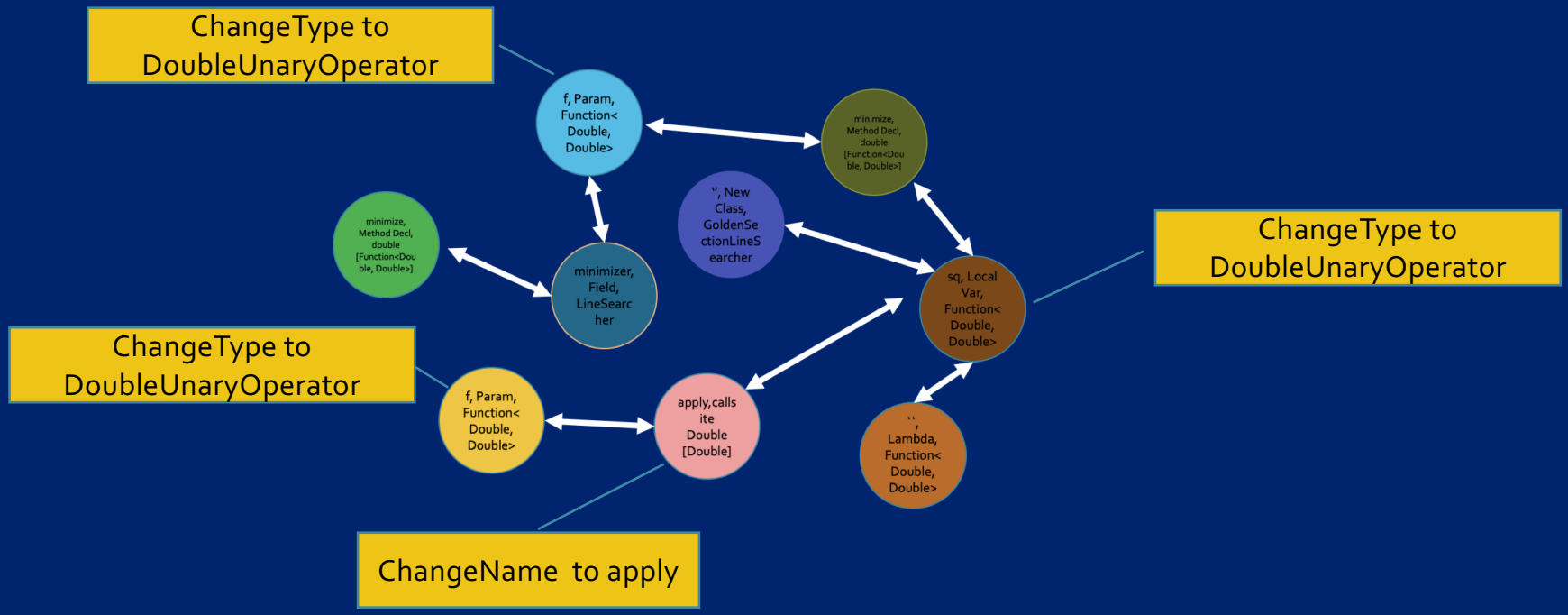
Labelled Edges: Encode type dependencies
e.g., Receiver, Assignment, Invocation

Key Novel Idea 2: Map-Reduce amenable



Apply Transformation Rules

`Function<Double, Double>` ➡ ChangeType to `DoubleUnaryOperator`
`apply` ➡ ChangeName to `applyAsDouble`



Empirical Evaluation

Used T2R to migrate 13 generic functional interface types to their specialized, performant types

Open Source Corpus: 7 best in class, performance-critical applications, 2.6 MLOC

Google Corpus 300M LOC

- **RQ1: Applicability:** How applicable are T2R type migrations?
  A: 139 type migrations in projects previously optimized for specialized functional types

- **RQ2: Scalability:** How scalable is T2R?
  A: 33 minutes for Google's Java codebase

RQ3: Usefulness Do developers find our type migrations useful in practice?

139 patches sent to the original developers as pull request.
126 patches were accepted and 10 still under review

- 2 patches rejected because developer did not like the name “Predicate”
- 1 patch rejected since it affected the public APIs

“This patch was deep inside the database and actually is very important for performance”

“Thanks for the nice contribution. The team would enjoy continuing the discussion.”

“Nice to be able to detect this automatically. Presumably, we have missed a number of them. … You will get a “fame spot” in Speedment”

Summary

A **MapReduce** amenable technique for type migration: emulates inter-procedural analysis at large scale

Type Fact Graphs – models type structure of a program

T2R – our implementation for **specializing Java's functional interfaces**

Evaluation of T2R on **7 open source** projects **and Google's Java code**

- Usability: 98% of the total patches generated were accepted
- Scalability: 300M LOC in 33mins

T2R and corpus available for download

My Most Important Investment

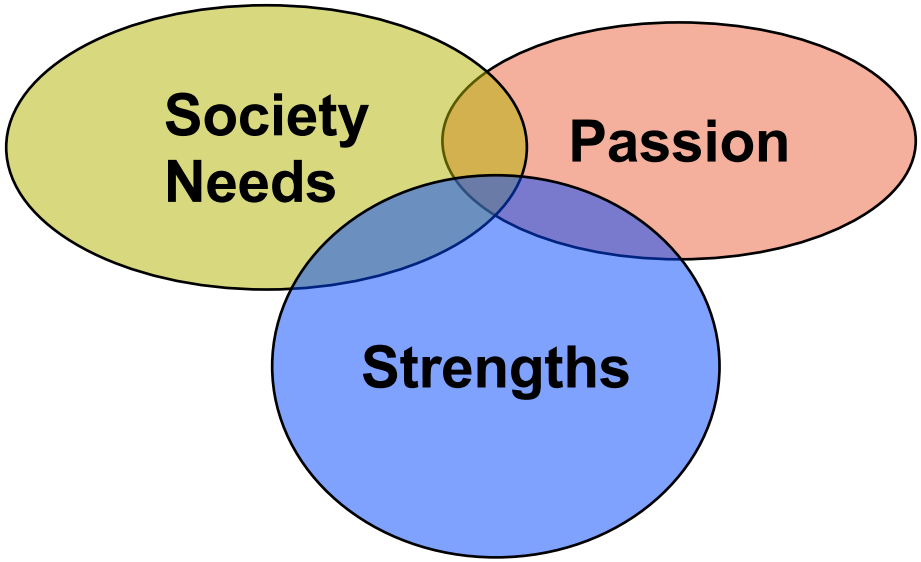
Michael Hilton (PhD'17, now at CMU)
Semih Okur (PhD'16, now at Microsoft)
Yu Lin (PhD'15, now at Google)
Stas Negara (PhD '13, now at Google)
Ameya Ketkar (PhD)
Malinda Dilhara (PhD)
Tom Dickens (PhD)
Sruti Srinivasa (PhD)
Shane McKane (MS'17, now at Intel)
Mihai Codoban (MS '15, now at Microsoft)
Kendall 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)

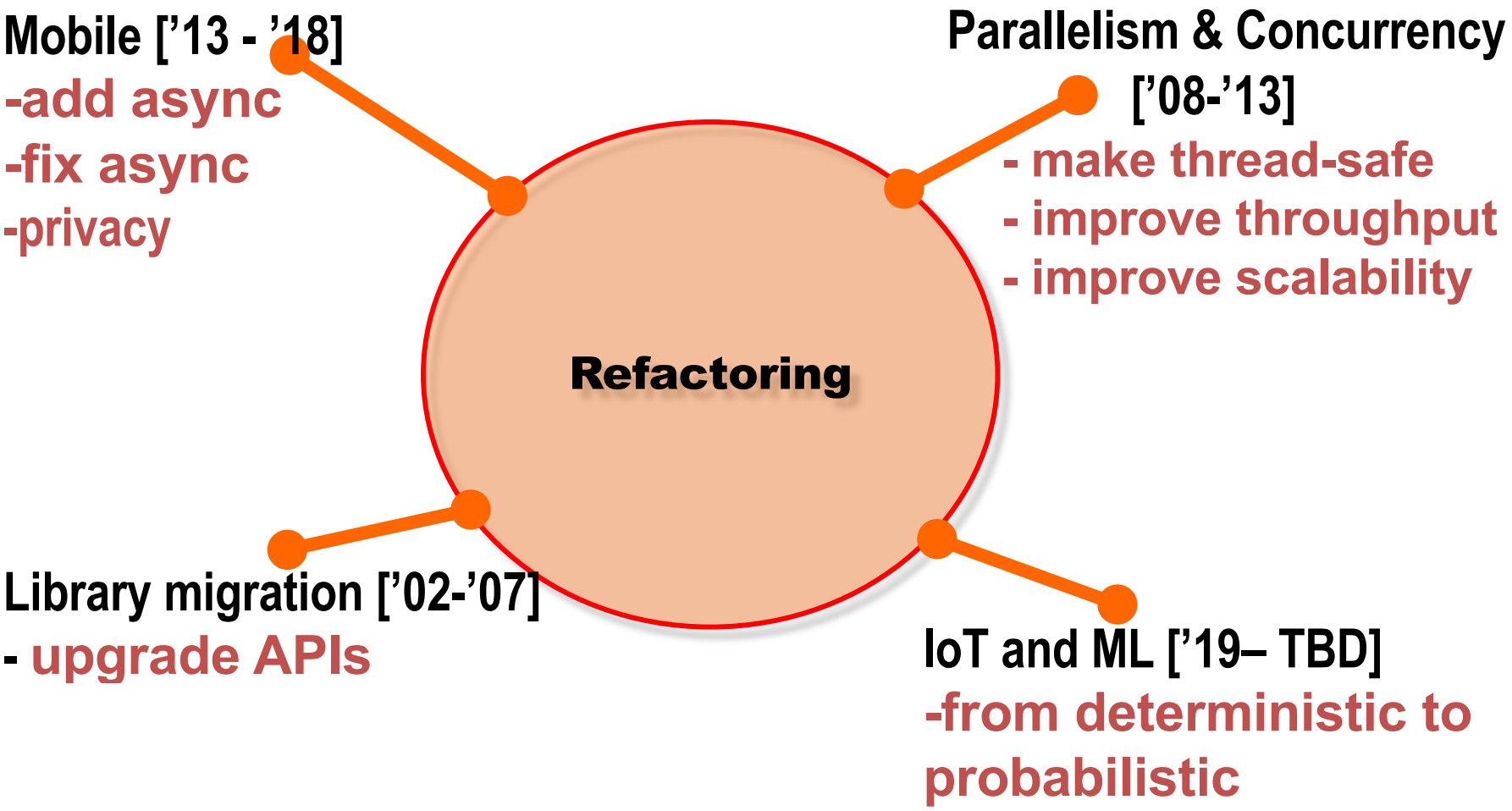
Focus on the Students

"Focus on the students, since graduating great students means you'll produce great research, while focusing on the research may or may not produce great students" **-- the late David Notkin**

Helping students find their voice



Work in Your Strength Zone but Reinvent Yourself



Principles for changing between different programming models


Center on Pervasive Personalized Intelligence
Oregon State University
Boulder
NSF
DAIMLER
Daimler Trucks North America
intel
Itron
Trimble
DELL
NEC
galois
http://ppicenter.org

Listening to Industry during Discovery Visits

Pervasive Personalized Intelligence (PPI)

From Reactive to Predictive Analytics:

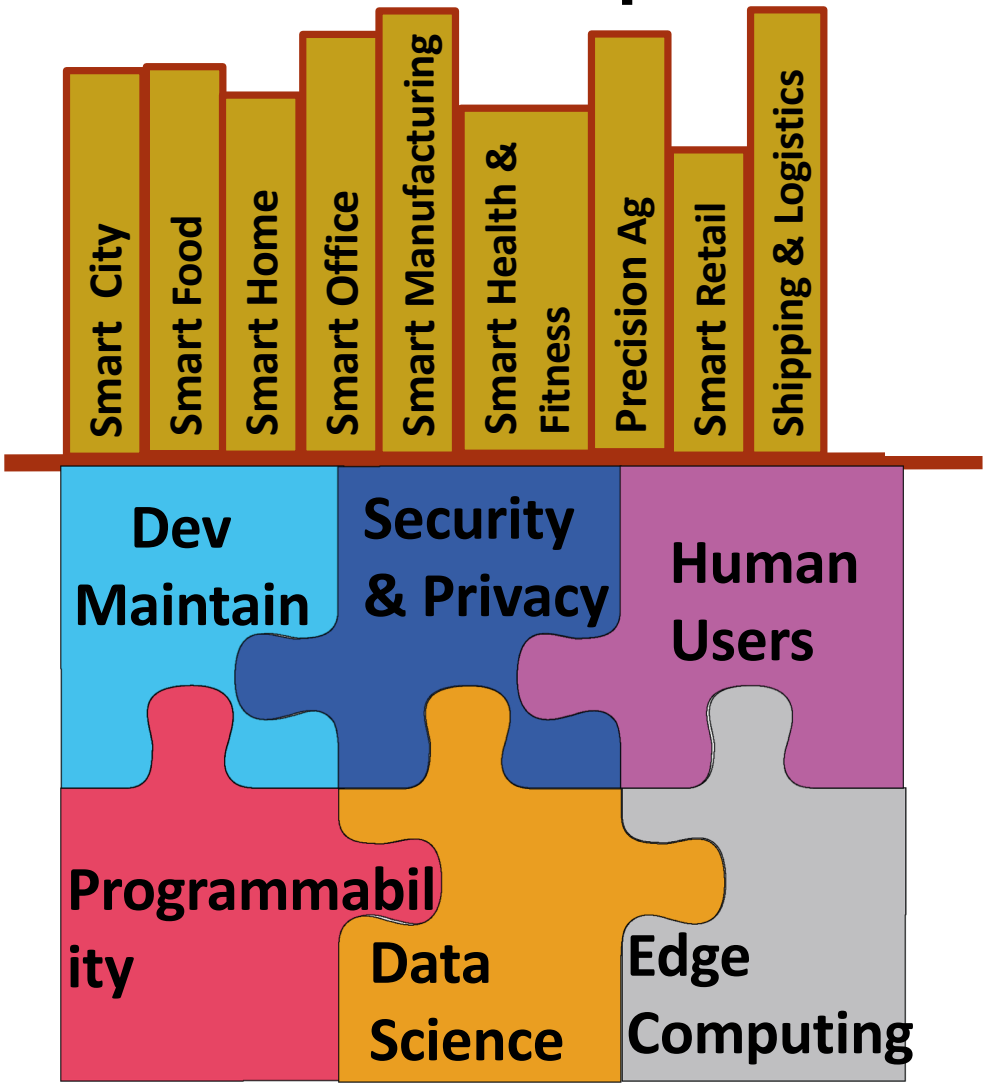
- **City: resource utilization, new infrastructure**
- **Ag: predict diseases, harvest**
- **Industry 4.0: auto-diagnosis on device**

Pervasive to the Edge

Personalized

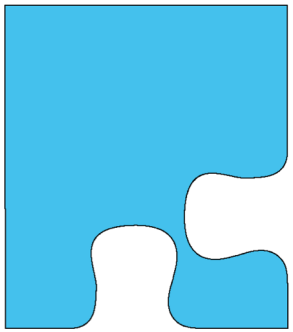
CENTER ON Pervasive Personalized Intelligence

Research Thrusts and Capabilities of PPI



CENTER ON Pervasive Personalized Intelligence

Our New Refactoring for ML Code



90% of software costs due to software evolution

ML algorithms are also code: they need to evolve

Understand ML evolution through formative studies
- quantitative (static analysis) and qualitative

Automate transformations:
- retrofit ML: from deterministic to probabilistic
- model transformations
    - e.g., Eager to Static in TensorFlow
    - evolve model when data changes, without retraining
- improve efficiency of code written by data scientist

Want to Go Fast? Go Alone
Want to Go Far? Go with Others

Learn more at https://ppicenter.org

Join our NSF Center, launching Sept 2020

Introduce us to your industry partners so they join our Founding Members:

Itron NEC Trimble DELL galois intel DAIMLER Daimler Trucks North America

Schedule a Discovery Session with PPI Directors:

Danny Dig (OSU site) danny.dig@colorado.edu

monthly webinar: ppicenter.eventbrite.com

CENTER ON Pervasive Personalized Intelligence

How do you battle distractions (Mind Wandering, Negative Thinking, Uncertain anxiety) during crisis? Join our group

tinyurl.com/dannydig2020