Machine Learning for Software Engineering

Audition DR2 INRIA
RMoD Team

Romain Robbes, 17/05/2022

18 years in Software Engineering Research

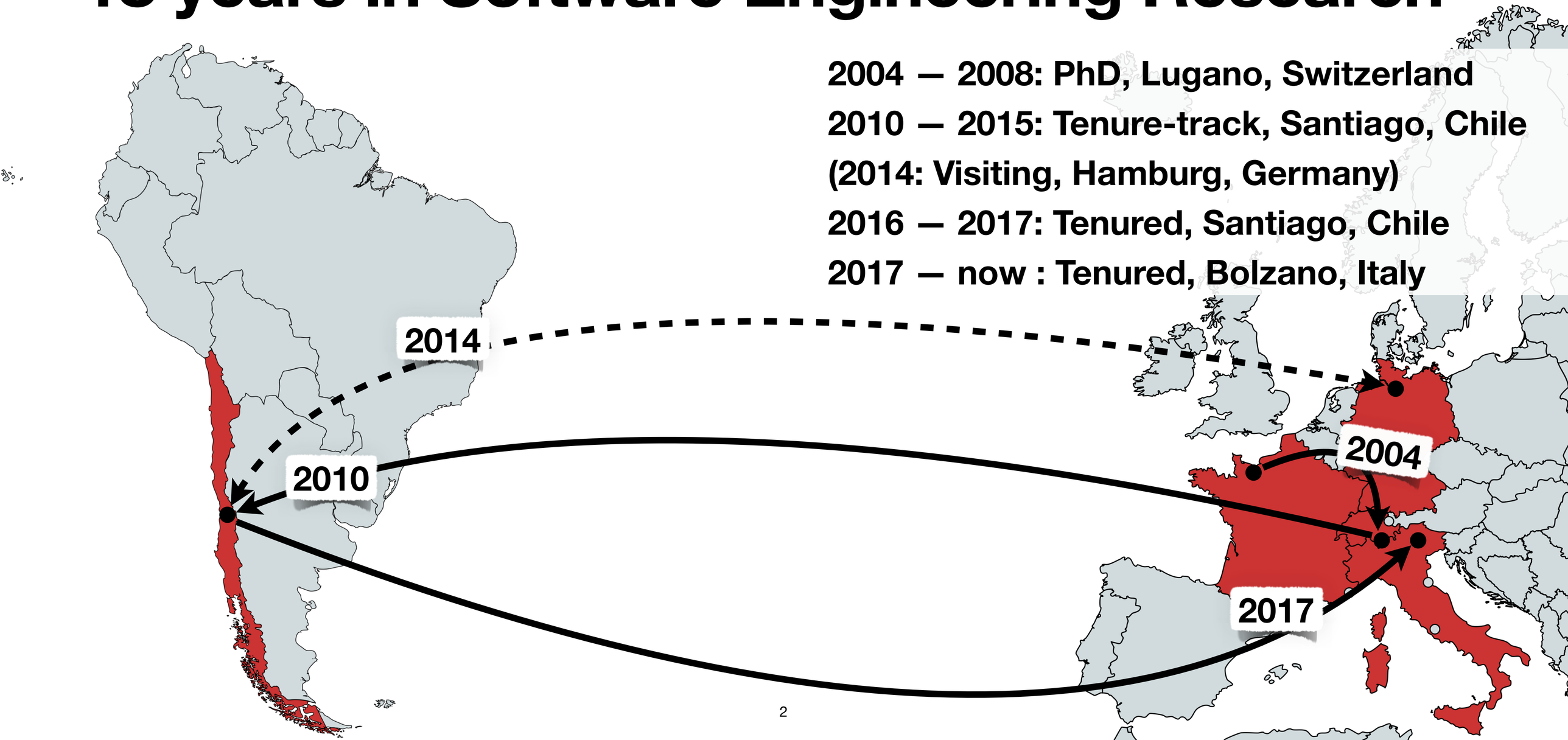
2004 — 2008: PhD, Lugano, Switzerland

2010 — 2015: Tenure-track, Santiago, Chile

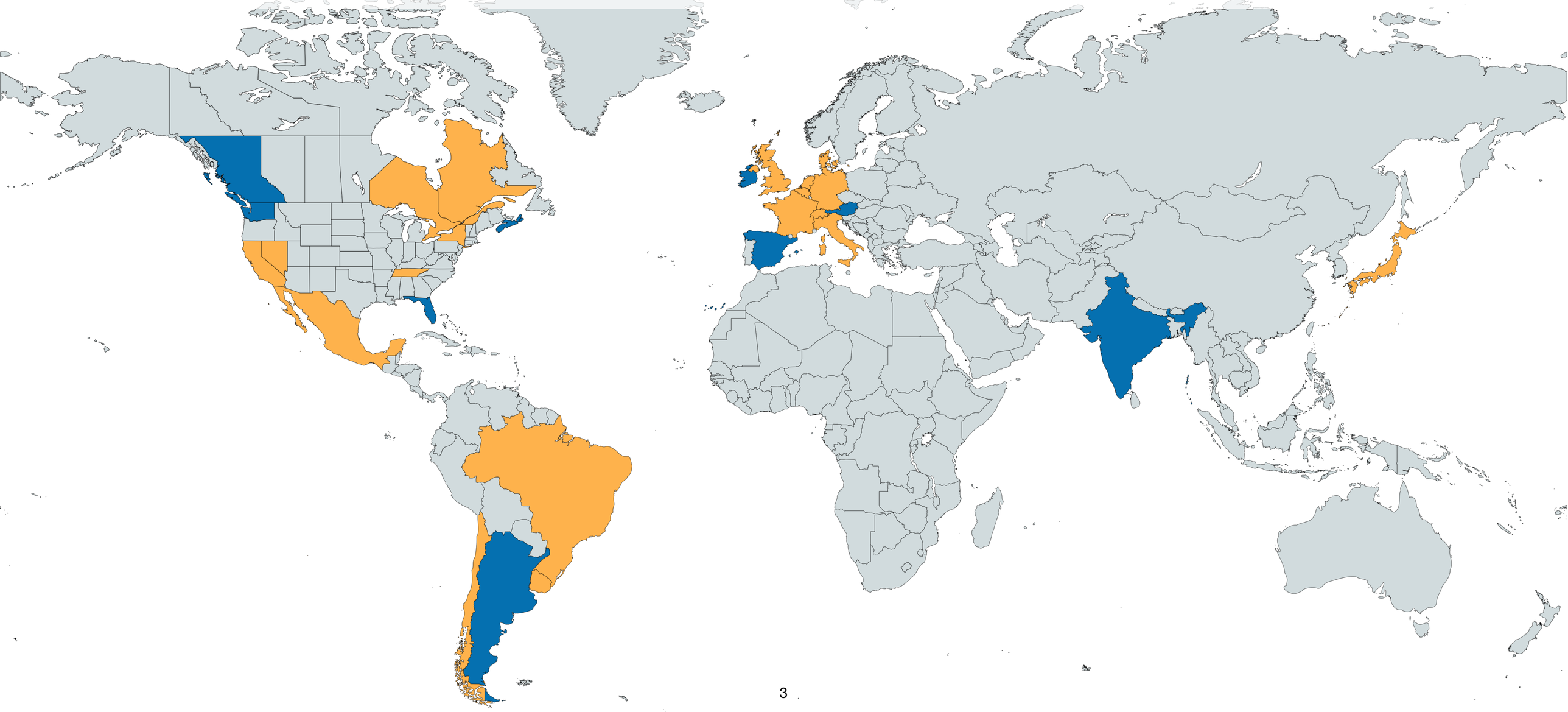
(2014: Visiting, Hamburg, Germany)

2016 — 2017: Tenured, Santiago, Chile

2017 — now : Tenured, Bolzano, Italy



Collaborations: Research and Service



Publications and service in quality venues

12 Full papers in **top conferences**

- ICSE, FSE, ASE
- ECOOP, OOPSLA

3 Track co-chair:
ICSE * 2, OOPSLA

13 Articles in **top journals**

EMSE, TSE, JSS, CSUR

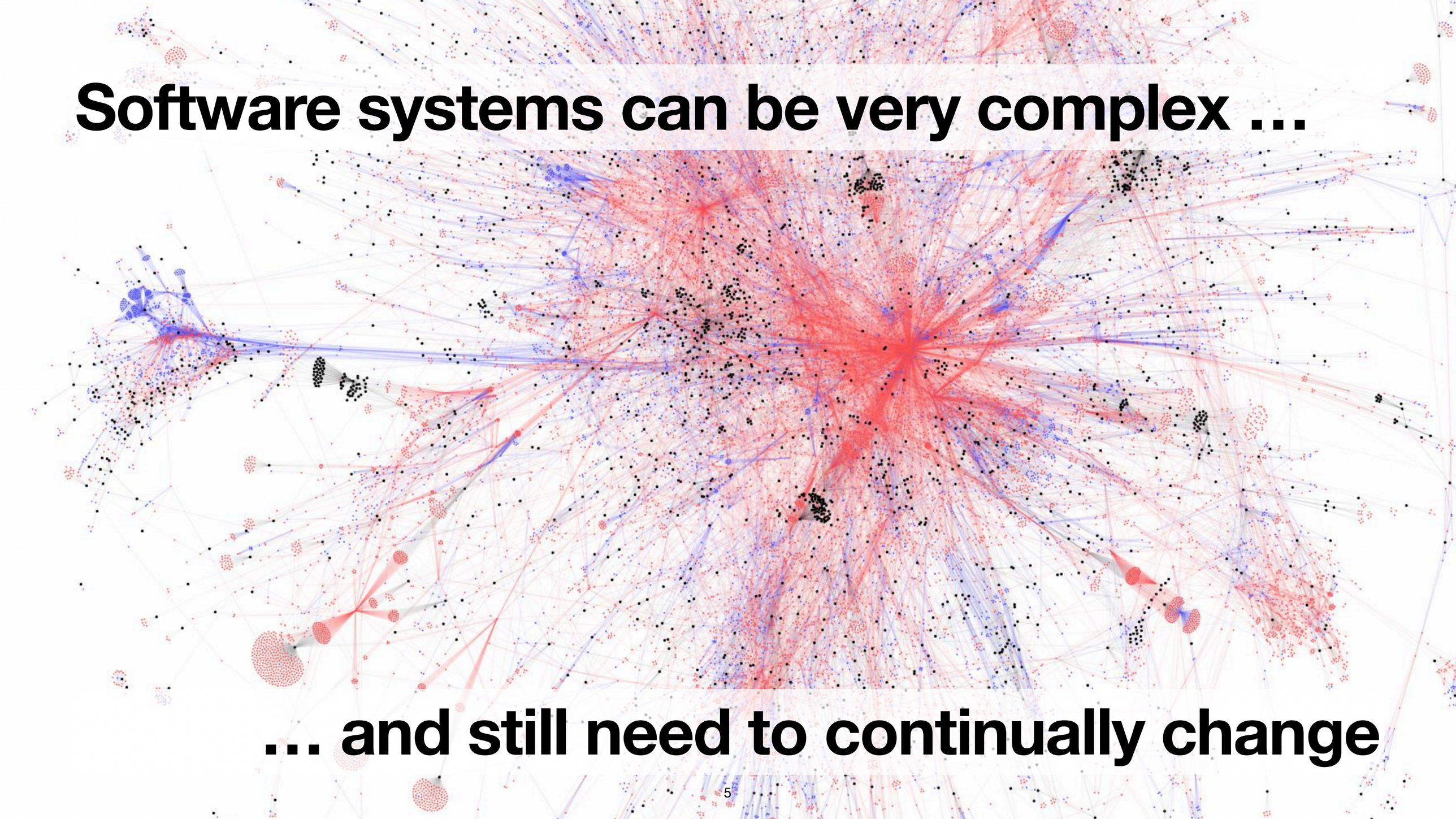
2 Editorial boards:
EMSE, JSS

17 Full papers in **specialized venues**

ICSME, MSR, ICPC,
SANER/WCRE

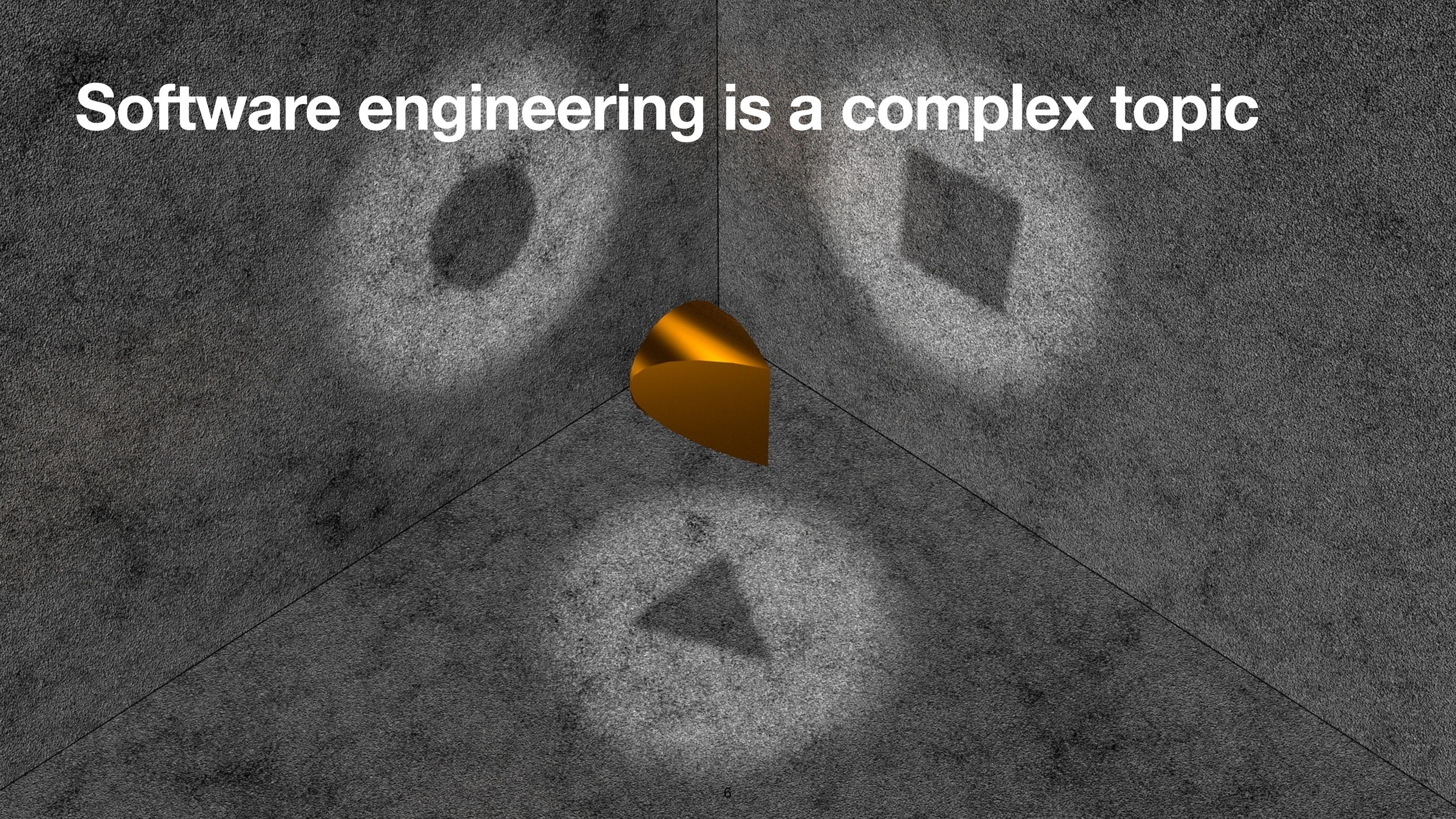
3 PC co-chair:
MSR * 2, WCRE

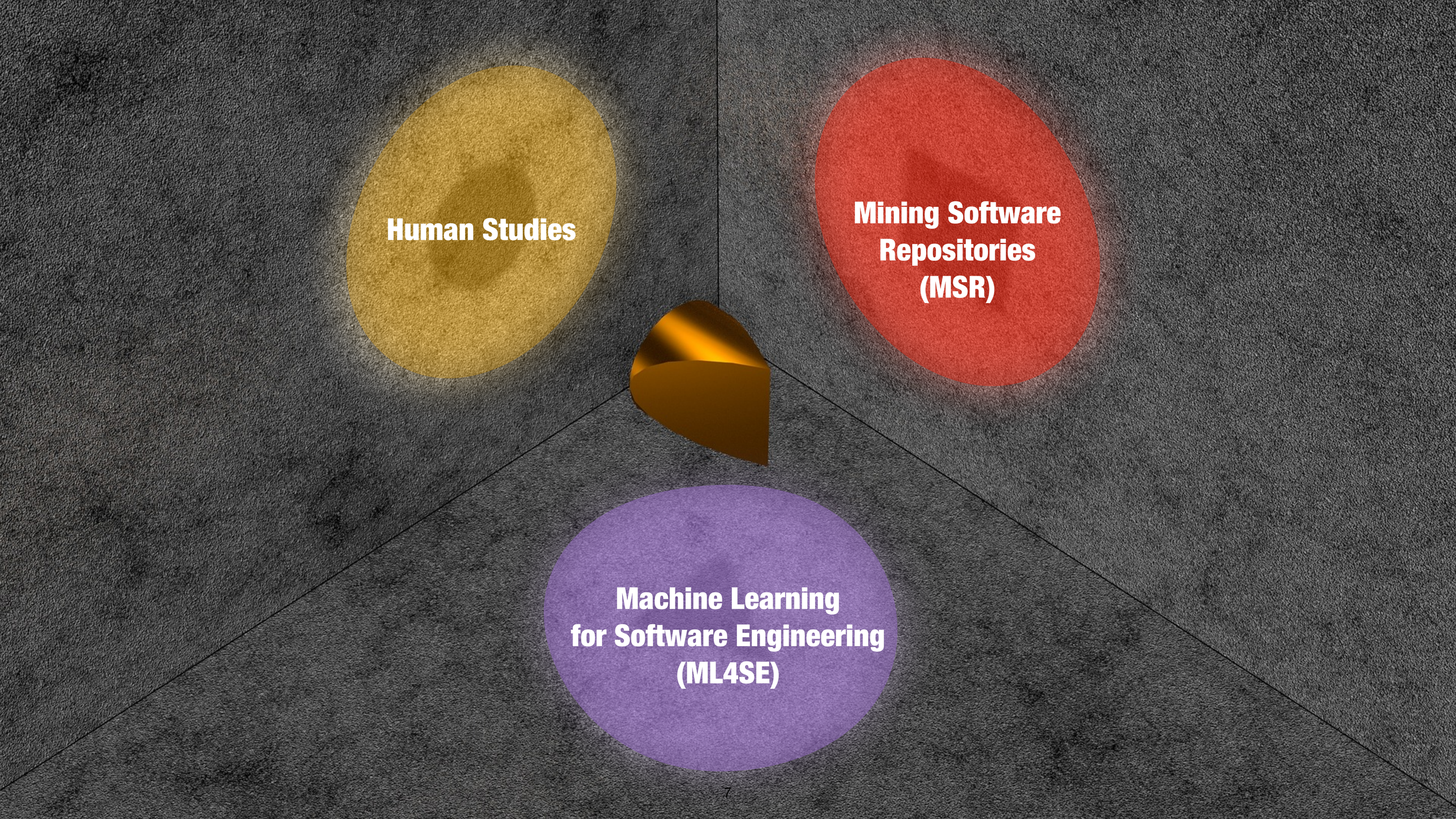
Software systems can be very complex …



… and still need to continually change

Software engineering is a complex topic




Human Studies
Mining Software Repositories (MSR)
Machine Learning for Software Engineering (ML4SE)

Major research contributions

Mining Software Repositories

MSR brings **multiple perspectives** on Software Systems

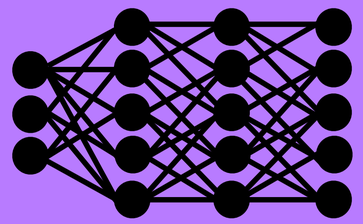
Human Studies

Bring back the human perspective:

- Do we ask the **right questions**?
- Do we have the **right answers**?

ML4SE

Machine Learning for Software Engineering can **integrate** multiple perspectives



Major Research Contributions

Mining Software Repositories

MSR brings **multiple perspectives** on Software Systems

skipped for time, please ask!

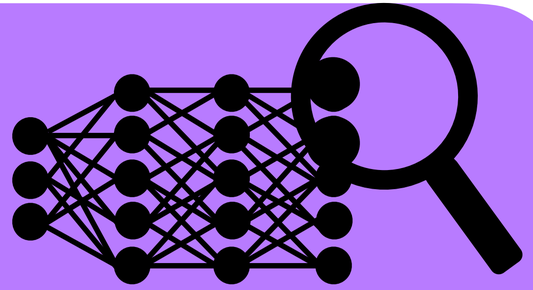
Human Studies

Bring back the human perspective:

- Do we ask the **right questions**?
- Do we have the **right answers**?

ML4SE

Machine Learning for Software Engineering can integrate multiple perspectives



SE artefacts mix structure and language

Hi Alice,

Can you help with this bug?
It's urgent.

```
// updating credentials
User u = DB.getUser(id)
u.setPassword(username)
DB.update(id, u)
```

Cheers,
Bob

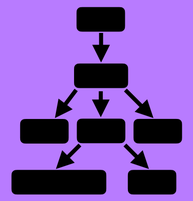
Deep Learning learns vector representations of the meaning of words

Bug =

.8	.2	.7	.1	.9	.5	.7	.3	.1	.3	.4	.9

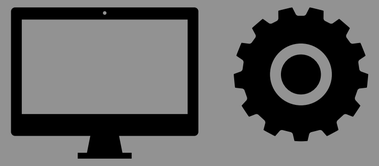
"Everything is a vector" for multiple artefacts

Structured models: trees, graphs



Detecting new kinds of bugs

Program analysis ignores identifiers & comments



```
// … … … … … …
i272.f428(i823);
```

+

Identifiers & comments: critical for humans


```
// updating credentials
u.setPassword(username);
```

=

code-comment incoherence, variable misuse, …

Example: Code Completion

Why this contribution?

Important practical problem

ASE 2008

SE artefacts improve code completion

special issue

ASEJ 2010

Released tool, useful in practice

All three areas, top conferences

ICSE 2020

Making Deep Learning for code completion work

Google Research

THE UNIVERSITY of EDINBURGH

Every programmer uses code completion

alphabeticallySorted
exhaustive
identifier
listWhichIsQuiteLong
manyWhoAre
not
remotelyCloseTo
whatYouWant

→

whatYouWant
shortList

2008 define completion as a task; develop optimized algorithms

2010 implement, evaluate a completion tool, still used in practice

2017 Others struggle with Neural Language Models for completion

2020 success applying Neural Language Models to completion

Every programmer uses code completion

alphabeticallySorted
exhaustive
identifier
listWhichIsQuiteLong
manyWhoAre
not
remotelyCloseTo
whatYouWant

→

whatYouWant
shortList

2008 define completion as a task; develop optimized algorithms

skipped for time

2010 implement, evaluate a completion tool, still used in practice

2017 Others struggle with Neural Language Models for completion

2020 success applying Neural Language Models to completion

Source Code Neural Language Models (NLMs)

Training: predict tokens from context

```
public static void main(String [ ] args) {
    String commandName = args [0] ;
    String uncommonFlag =  ? ? ? ?
```

"Out of the box" use: code completion

Fine-tunable on new tasks

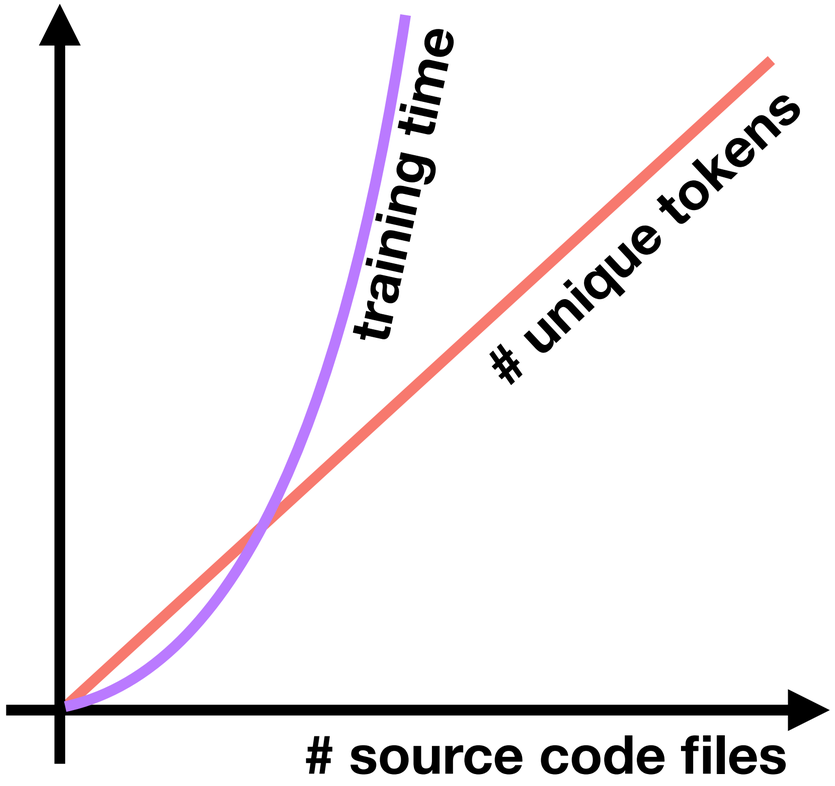
Buggy code?

Yes: 0.1 No: 0.9

Closed vocabulary of pre-defined tokens

Developers create new identifiers at will

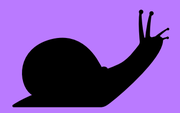
```
foo
Foo
Bar
FooBar
FOO_BAR
a234
a9095
my_taylor_is_rich
MeinTaylorIstReich
mon-tailleur-est-riche
FooBarManagerController
ArbitrarilyLongAndComplexIdentifierIsAllowed
FooBarManagerControllerRuntimeInstantiationException
```



New code contains new tokens, leading to a Vocabulary Explosion

Large vocabulary leads to three issues

Slow training



Does not scale

Rare tokens

?	?	?	?	?	?

Bad vector representations

Unknown tokens

<UNK>

No vector representation

Only ~100 projects

NLM < n-grams

Are Deep Neural Networks the Best Choice for Modeling Source Code?

Vincent J. Hellendoorn
Computer Science Dept., UC Davis

Premkumar Devanbu
Computer Science Dept., UC Davis

Pre-processing works, but not enough

Exhaustive preprocessing on 10,000 Java projects		# Unique Tokens	% Rare Tokens	% Unknown Tokens
No preprocessing	`FooBarManagerController`	11.6M	83%	79%
Coding Conventions	`Foo Bar Manager Controller`	1.3M	81%	20%
Best approach	(Out of a dozen)	0.5M	70%	9%

HUGE vocabulary

Too many rare, unknown tokens

Byte-Pair Encoding learns an **open vocabulary**

BPE builds a vocabulary bottom-up, from frequent character sequences

Previously used in Translation, not yet in NLMs

BPE performs well …

```
FooBarManagerController
Foo Bar ManagerController
```

even in difficult cases …

```
httpclientlib
http client lib
```

and degrades gracefully

```
SehrLangeQuellcodeKennung
Se hr Lan ge Que ll code Ken nun g
```

BPE solves all the vocabulary issues

Exhaustive preprocessing on 10,000 Java projects		# Unique Tokens	% Rare Tokens	% Unknown Tokens
No preprocessing	`FooBarManagerController`	11.6M	83%	79%
Best approach	(Out of a dozen)	0.5M	70%	9%
Byte-Pair Encoding (BPE)		**10K**	**1%**	**0%**
		1,000 X smaller	**Few rare tokens**	**No unknown tokens**

Evaluation and impact

Open-vocabulary NLMs scale:

- **10,000+ software projects**
- **100 X more than previous work**

Thorough evaluation shows large improvements:

- **Java, C, Python**
- **Code completion, bug detection**

Distinguished Paper Award

The 42nd International Conference on Software Engineering
ICSE 2020

acm

SIGSOFT
SPECIAL INTEREST GROUP ON SOFTWARE ENGINEERING

100+ citations in 2 years

data, code, models available

Artifacts Evaluated acm Reusable

Artifacts Available acm

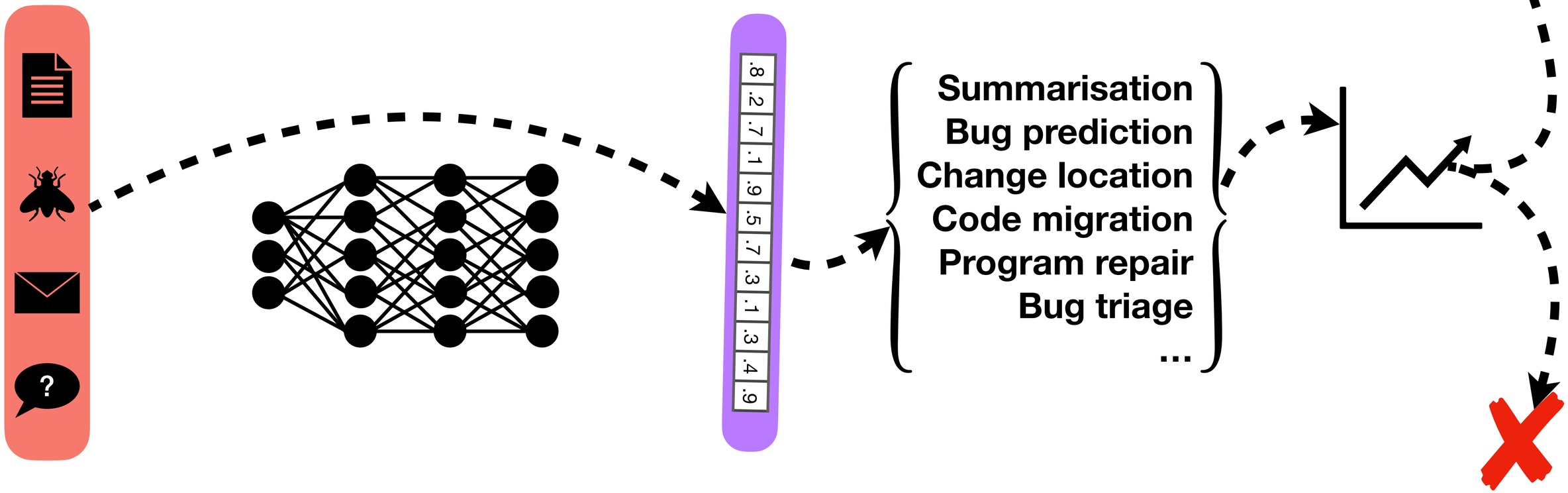
Needed expertise in both ML & SE, willingness to revisit assumptions

Research Program:
ML4SE for Legacy Systems

Challenges in ML4SE for Software Evolution

1 Learning **vector representations** for **complex** SE artefacts

2 **Successfully** apply ML4SE to Software Evolution tasks



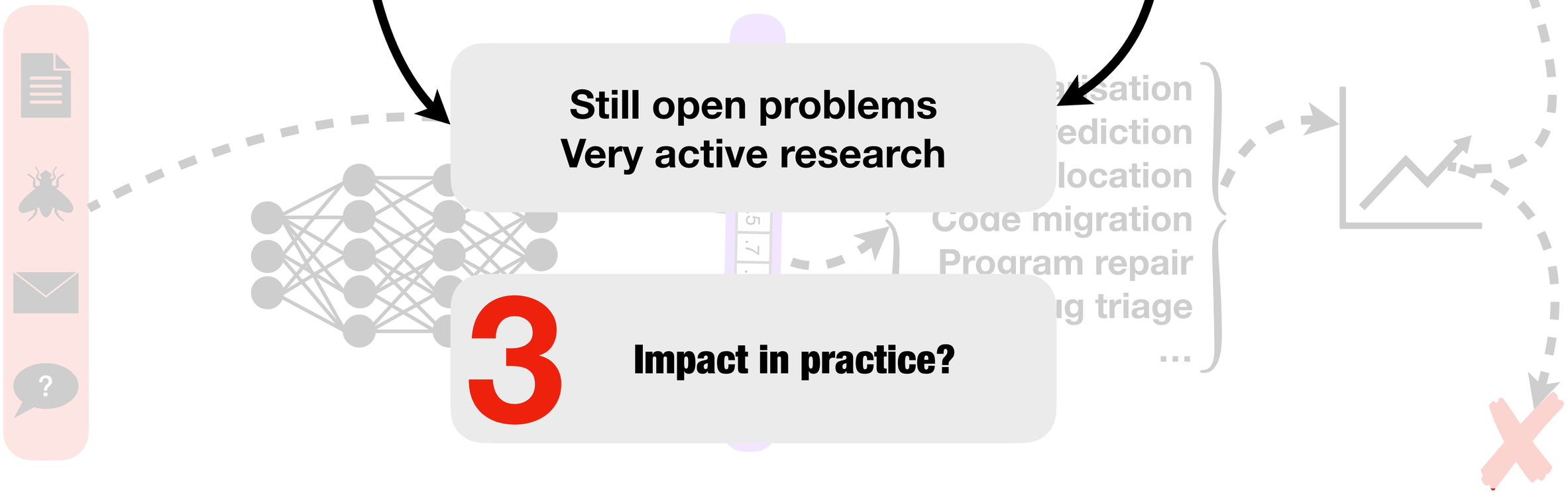
Challenges in ML4SE for Software Evolution

1 Learning **vector representations** for **complex** SE artefacts

2 **Successfully** apply ML4SE to Software Evolution tasks

Still open problems
Very active research

3 **Impact in practice?**



For Source Code NLMs, the answer is “more”

Ours (2020)

1.8 GB

OpenAI’s Codex (2021)

GitHub Copilot

159 GB

55 Million

GitHub repositories
Runs in the cloud
Closed source

DeepMind’s AlphaCode (2022)

715 GB

Are there 55 million repositories of COBOL?

FORTRAN?

MANTIS?

NCL?

4D?

GENERALI

50% of

Generali Belgium

Legacy systems, written in Orphan languages

Decades-old
critical infrastructure for
important institutions

Decades-old
languages
little or no support

Complex code, quality issues
Missing documentation, tests

Online open source presence:
few to none

Knowledge loss over decades
Constant need for evolution

Not enough data
for out of the box ML approaches

Two challenges: scientific and practical

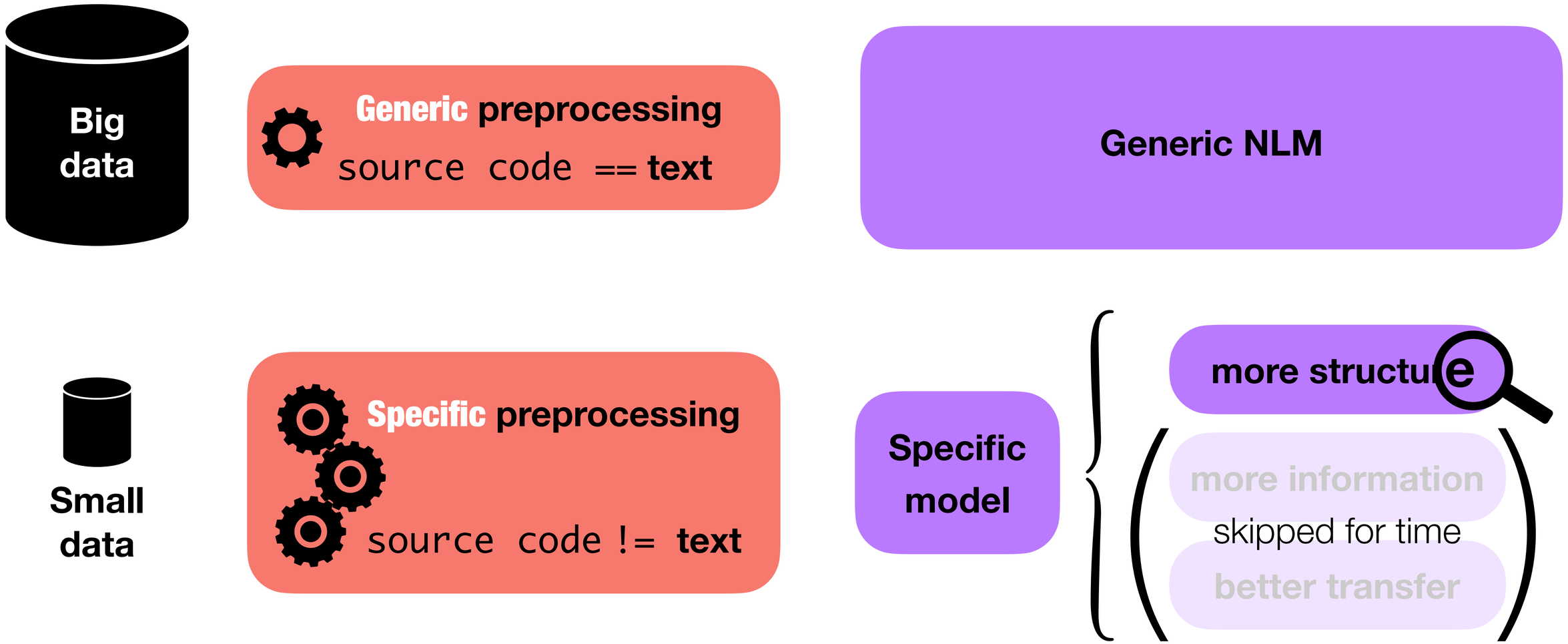
Learn with less data

For orphan languages

In practical settings

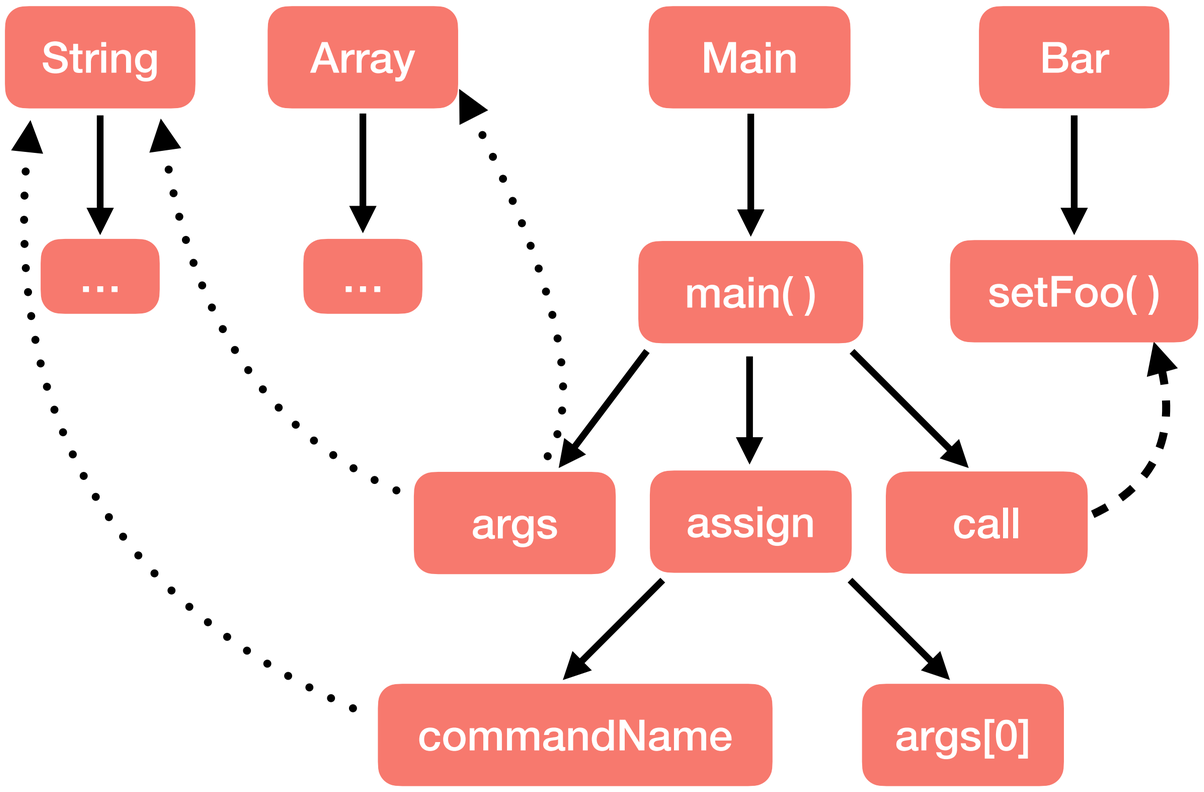
Relevant for legacy systems

Learn with less data, but with more …



Learn from less with **more structure**

```
public static void main(String [ ] args) {
      String commandName = args [0] ;
      String uncommonFlag = args [1] ;
      myBar.setFoo(args[2])
```



Learning implicit information from scratch requires a large model & lots of data

Explicit information: smaller model, less data

Most papers focus on the ML perspective

… data work

Collect and process artefacts

Everybody wants to do the model work, not the …

Develop approach

Run on benchmarks

Practical scenarios require a holistic view

Everybody wants to do the model work, not the …

… scoping	Define the problem
… data work	Collect and process artefacts
…	Develop approach Run on benchmarks
… integration	Tool integration
… evaluation	Practical Evaluations

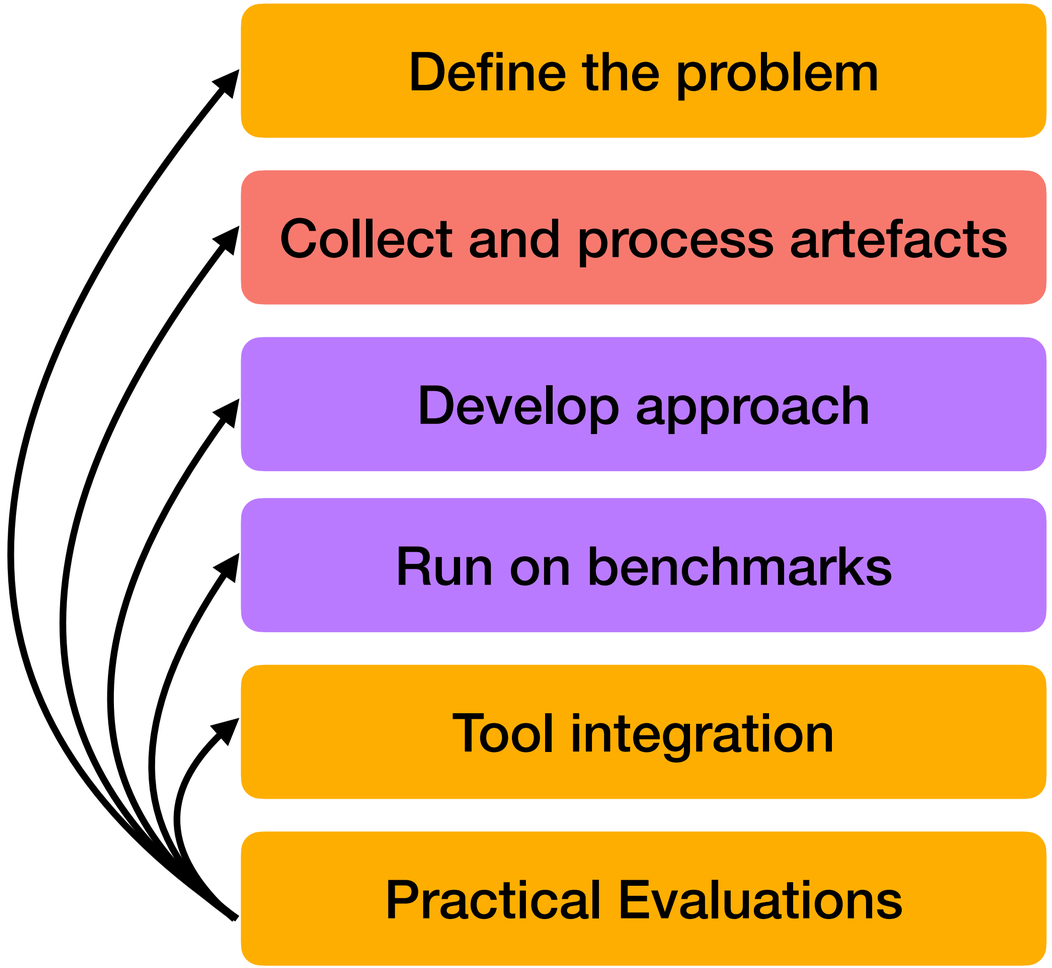
Practical scenarios require a holistic view

We need to do the

- scoping — Define the problem
- data work — Collect and process artefacts
- model work
  - Develop approach
  - Run on benchmarks
- integration — Tool integration
- evaluation — Practical Evaluations

Practical scenarios provide feedback…

… which may lead to **revisiting decisions**



The ideal team for this project is **RMoD**

RMoD has several **industrial partners**

with many **practical problems**

RMoD & I have **extensive previous collaborations**

9 papers:

SATToSE 2019

Integration in RMoD successor EVREF

Axis 1
Software Evolution

- Test generation
- Code migration
- Business rule extraction
- ...

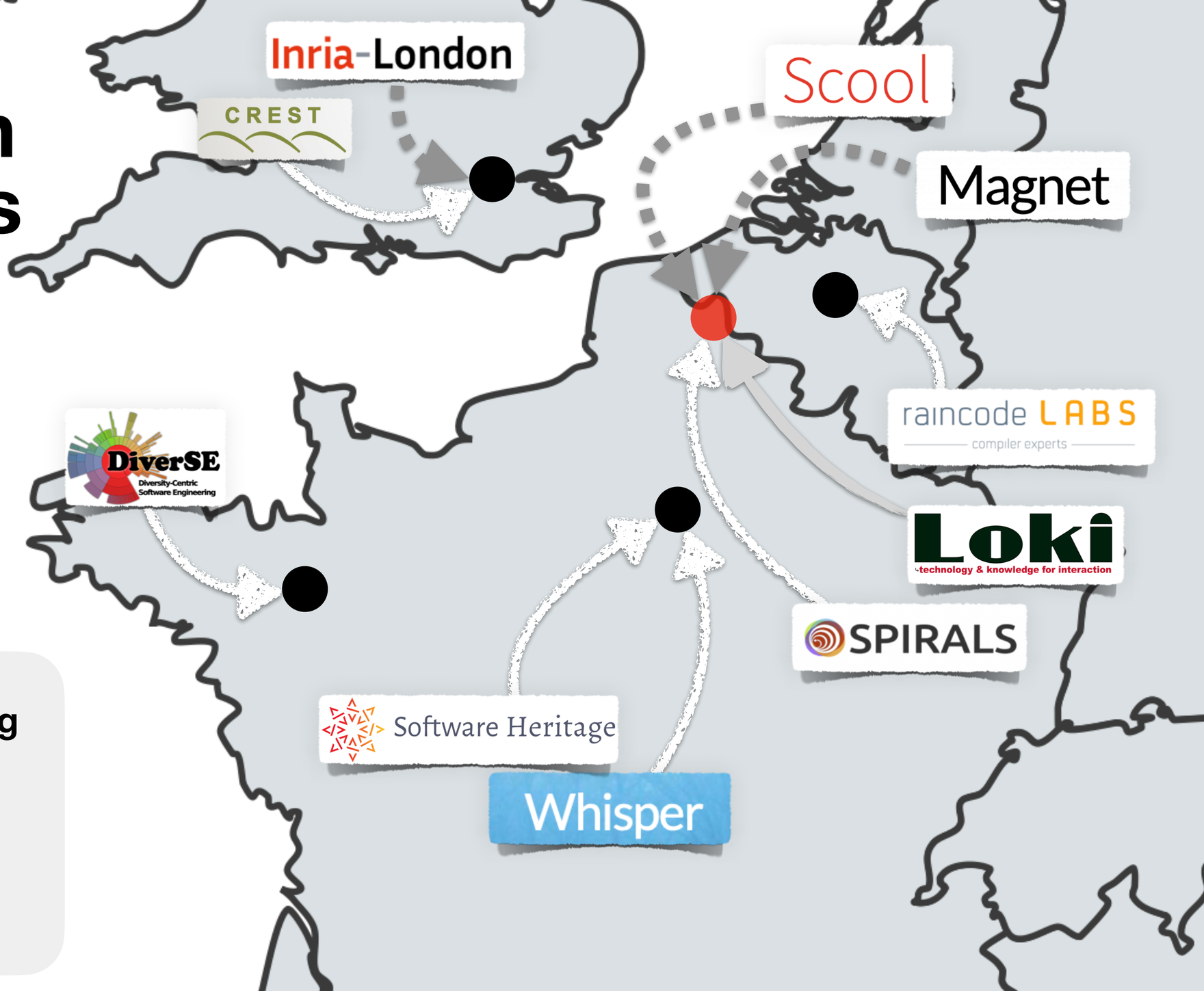
Axis 2
Development Tools

- Bug triage
- Code review support
- Test case prioritisation
- ...

Bringing skills to EVREF

- ML4SE : needed
- MSR : reinforce
- Human Studies : reinforce

Collaboration Opportunities



Software Engineering

Human Computer Interaction

Machine Learning

Major SE Contributions

MSR brings multiple perspectives on Software Systems

Human Studies bring back the human perspective

ML4SE integrates these multiple perspectives

Perspectives on code completion

alphabeticallySorted
exhaustive
identifier
listWhichIsQuiteLong
manyWhoAre
not
remotelyCloseTo
whatYouWant

→

whatYouWant
shortList

- Defined code completion as a task
- Released and evaluated a tool, still used
- Applied Deep Learning to Code Completion

A research program with two challenges

Learn with less data

In practical settings

For orphan languages

Relevant for legacy systems

RMoD is the ideal team

RMod

Its partners have legacy systems & practical problems

Berger Levrault SIEMENS arolla Lifeware

My skills & interests are a great match with RMoD