An Industrial Case Study on Predicting and Detecting Faulty Programs Using Machine Learning

Wahab Hamou-Lhadj

Université Concordia
Montréal, QC, Canada
wahab.hamou-lhadj@concordia.ca

FETCH'20, Montreal, QC, Canada
February 12, 2020

Software Development Challenges

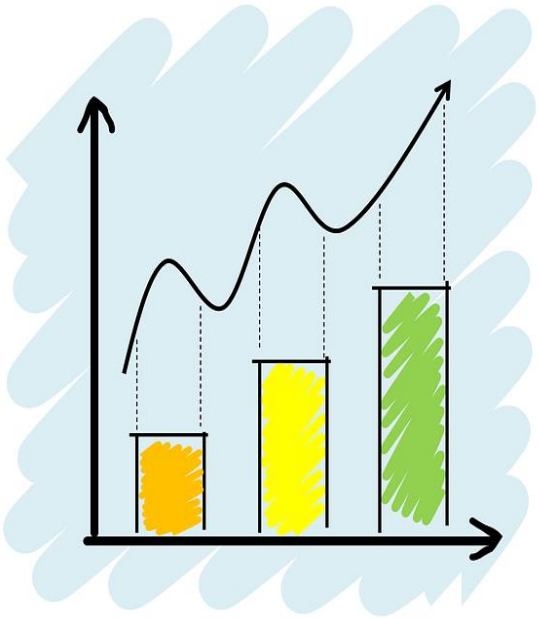
- Increased complexity
- Heavy reliance on people
- Lack of automated tools
- Time to market pressure
- Emerging technologies
- QA trade-offs

Software Maintenance

- **70%** of the overall development cost
- Up to **50%** of maintenance cost is on fixing bugs
- Bugs may have **severe consequences**
- Defects cost the economy **billions of $** annually

Emergence of Software Analytics

- Data-driven SW development and maintenance
- Big Data: source code, bug reports, test cases, logs, user feedback, etc.
- Predictive analytics using ML, DL, CI, and PR
- Information visualization of large-scale data

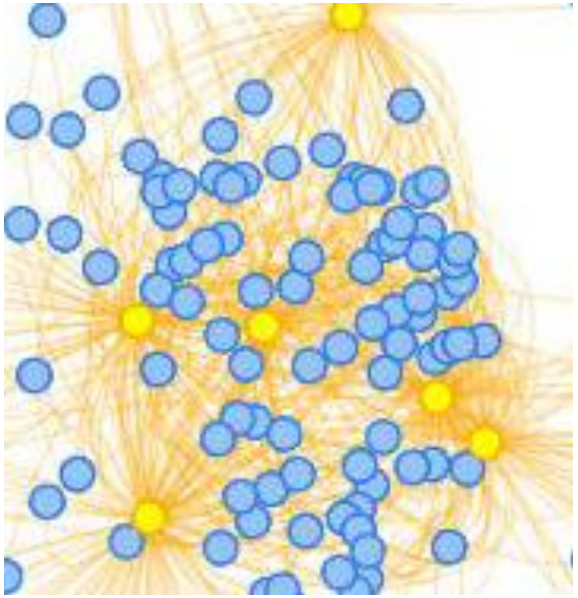


Defect Detection/Prediction Research

- Defect Prediction
  - Statistical analysis
  - Call-graph analysis
  - Analysis of code changes
  - Leverage of historical data
- Automated Patch Generation
  - Development of fixing patterns
  - Reuse of human written patches
  - Directed patches towards specific bug types

Problems with existing techniques

- Offline processing (after the code is built)
- Presence of the entire source code
- Extensive setup and high learning curve
- Lack of clear actions to developers
- High rate of false positives



Our solution: CommitAssistant

- A prototype tool resulting from an NSERC research project between my research lab at Concordia University and Ubisoft Laforge

- Main Features:
  - Detection of bugs at commit-time, i.e., as programmers write code
  - Supports multiple programming languages
  - No external tools or setup required
  - Leverage of historical bugs and fixes
  - High TRL

CommitAssistant Phases

1

Train models of historical defect and healthy commits and associated code

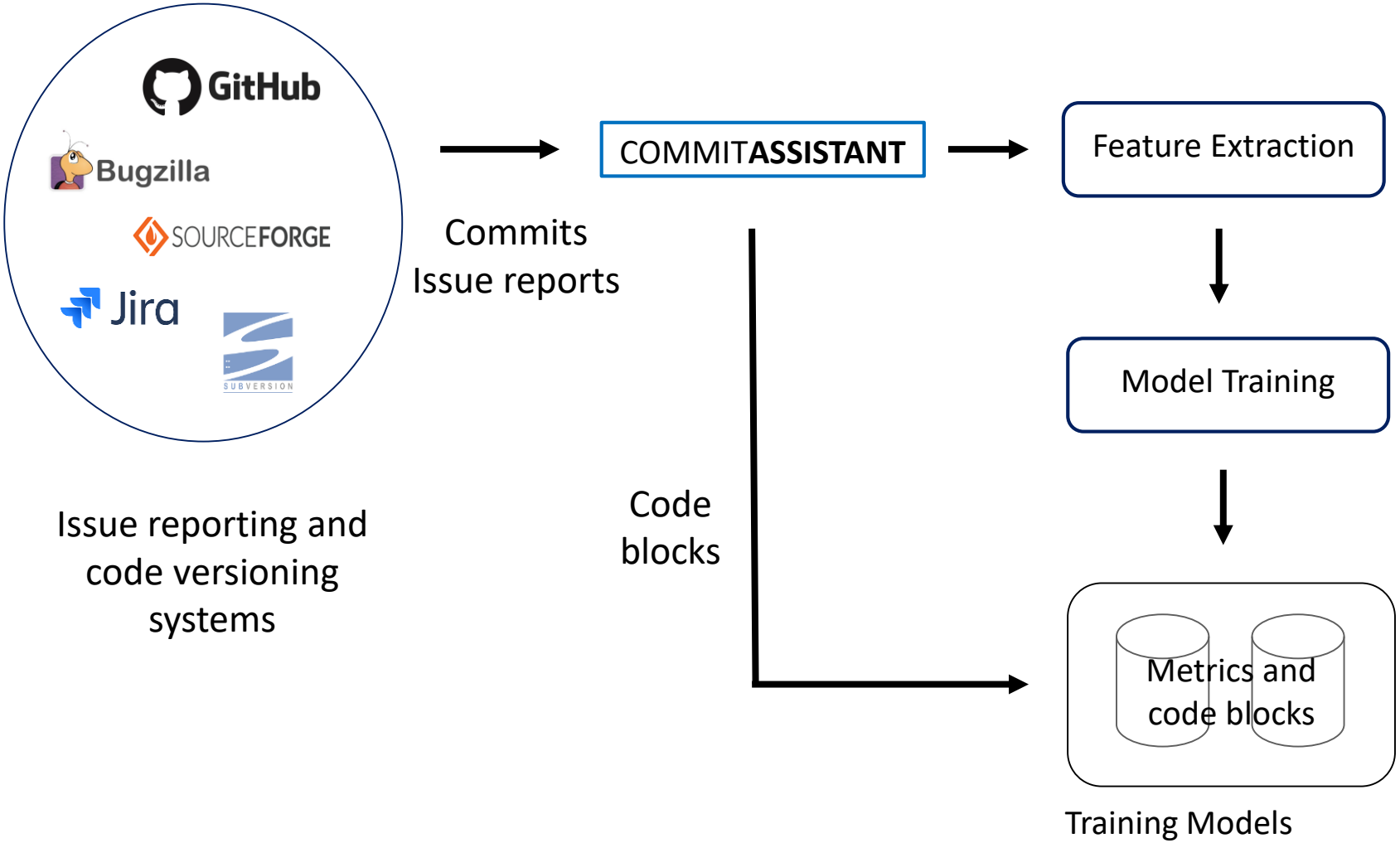
Intercept and analyze developers' commits before they reach the central code repository

2

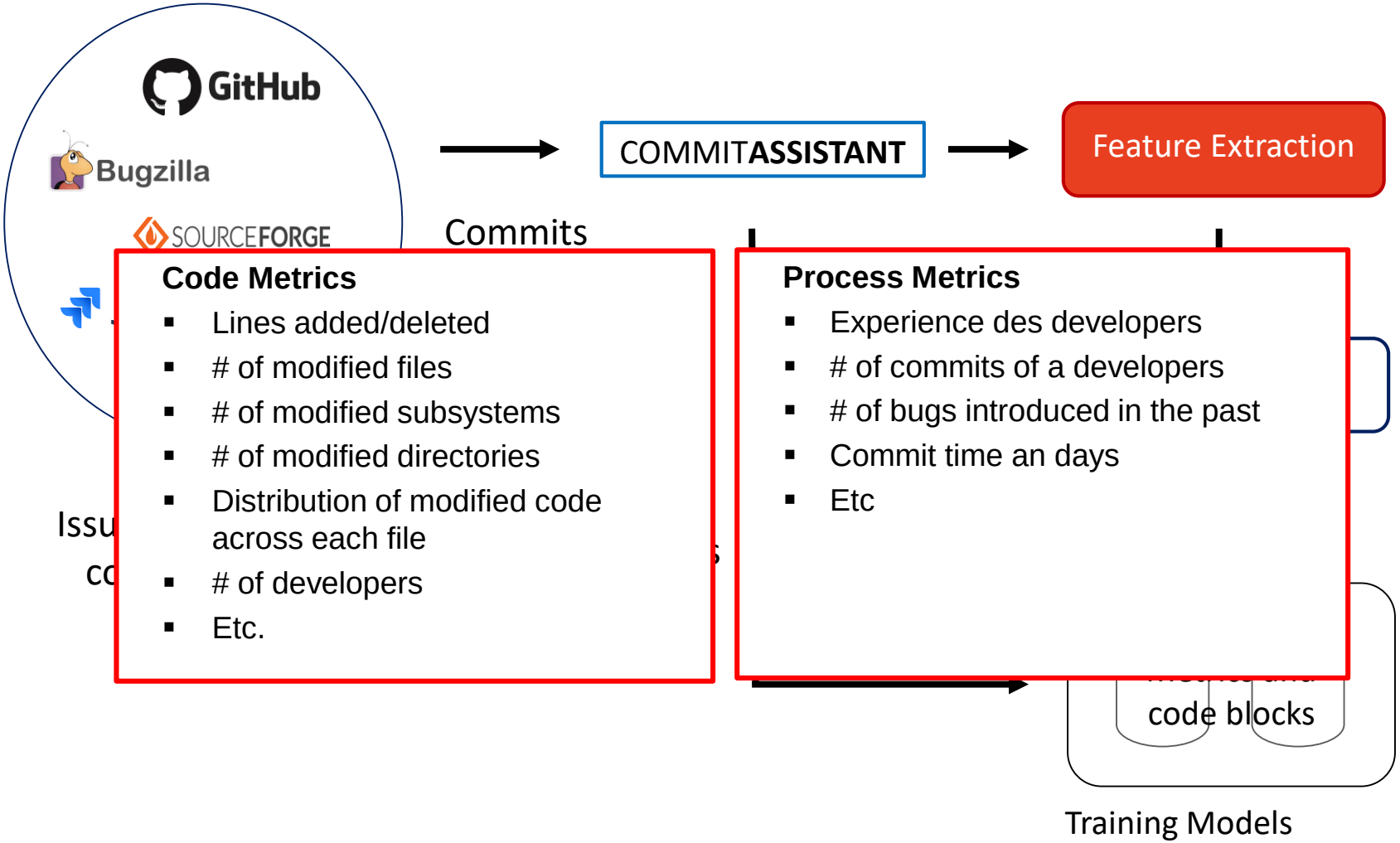
3

Notify developers and **propose fixes** for risky commits

Step 1: Train models



Step 1: Train models



Steps 2, 3: Analyze commits and propose fixes

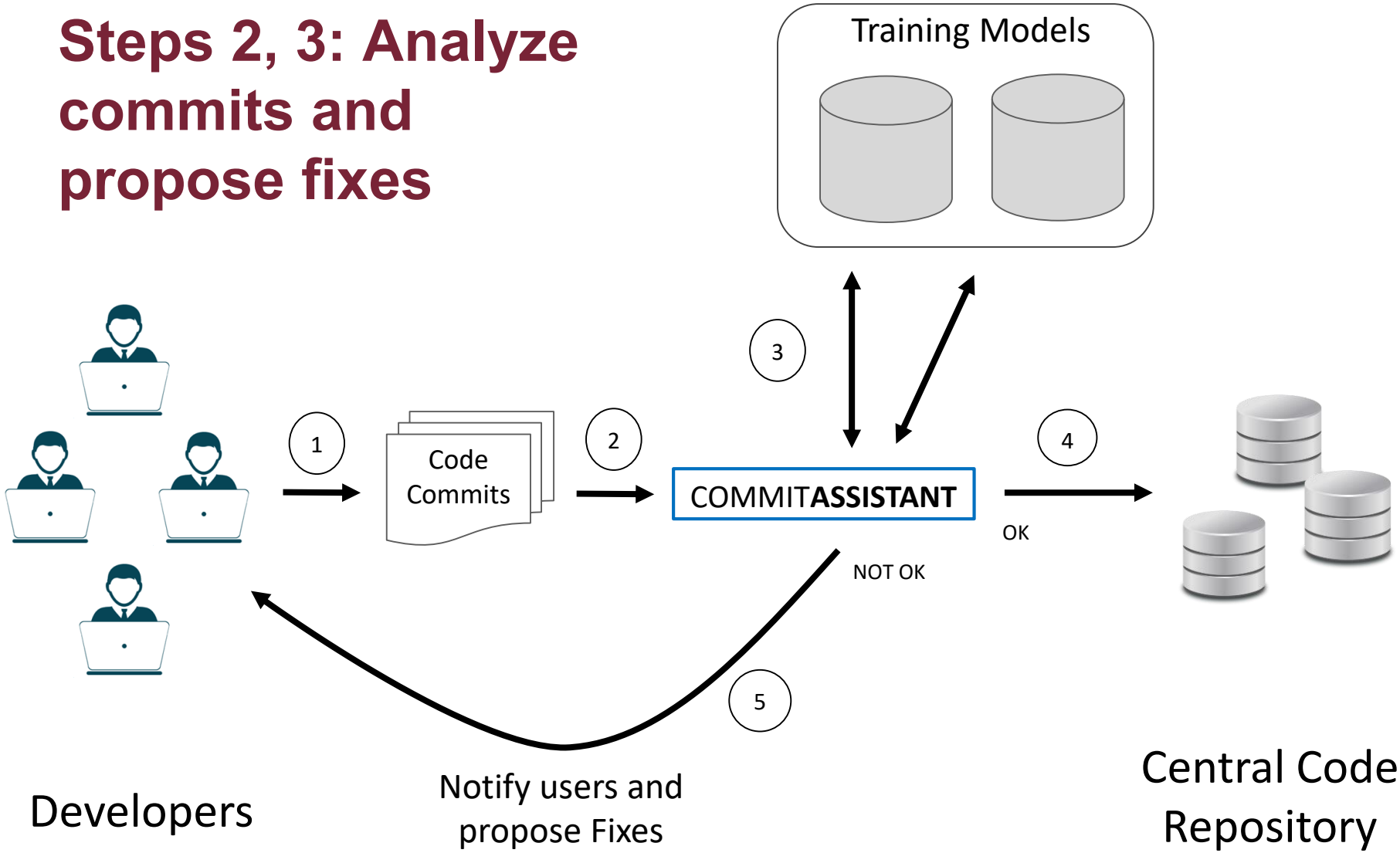


TABLE 3: BIANCA results in terms of organization, project name, a short description, number of class, number of commits, number of defect introducing commits, number of risky commit detected, precision (%), recall (%), F_1-measure (%), the average similarity of first 3 and 5 proposed fixes with the actual fix and the average time difference between detected and original.

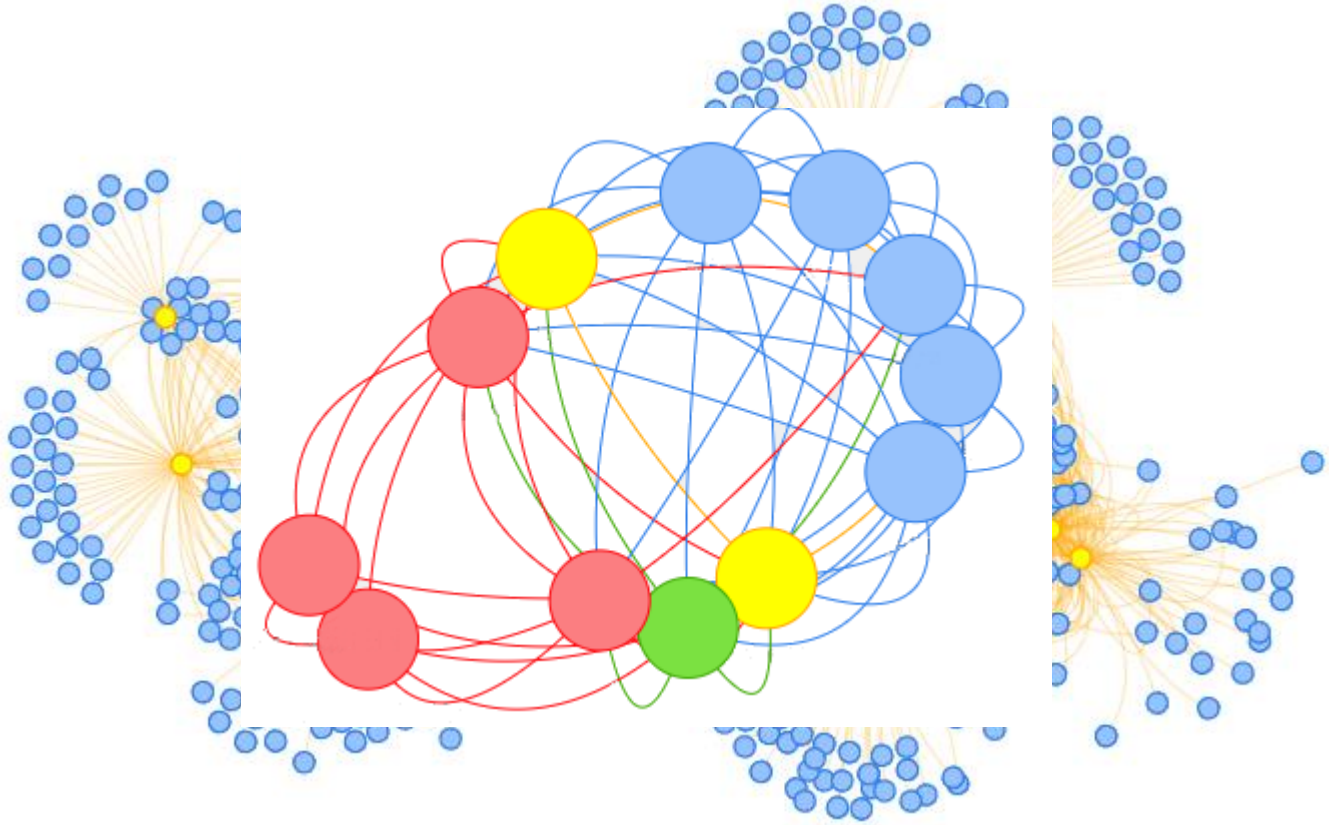
Organization	Project Name	Short Description	NoC	#Commits	Bug Introducing Commit	Detected	Precision	Recall	F_1	Top 5 Fixes Similarity	Top 3 Fixes Similarity
Alibaba	druid	Database connection pool	3,309	4,775	1,260	787	88.44	62.46	73.21	39.97	46.69
	dubbo	RPC framework	1,715	1,836	119	61	96.72	51.26	67.01	60.01	57.14
	fastjson	JSON parser/ generator	2,002	1,749	516	373	95.71	72.29	82.37	18.19	15.23
Apa											48
											68
Clo											10
Dro											52
											56
Ecli											82
											04
Exci											13
Face											03
Goo											59
											70
Goo											59
											53
											97
Gra											93
Jank											48
Jhy											59
Libd											31
Net											57
Ope											66
Ope											00
Orfj											06
Orie											00
Perv											00
Pres											54
Rob											55
Lom											49
Scri											16
											97
											20
Square	okhttp	HTTP+HTTP/2 client	344	2,649	592	474	93.04	80.07	86.07	29.09	24.91
	okio	I/O API for Java	90	433	40	24	100.00	60.00	75.00	31.51	35.50
	otto	Guava-based event bus	84	201	15	15	93.33	100.00	96.55	54.11	49.94
	retrofit	Type-safe HTTP client	202	1,349	151	111	99.10	73.51	84.41	49.88	45.46
StephaneNicolas	robospice	Android library	461	865	113	39	87.18	34.51	49.45	60.90	65.04
ThinkAurelius	titan	Graph Database	2,015	4,434	1,634	527	90.13	32.25	47.51	48.64	50.59
Xetorthio	jedis	Redis client	203	1,370	295	226	92.04	76.61	83.62	25.69	29.45
Yahoo	anthelion	Plugin for Apache Nutch	1,620	7	0	-	-	-	-	-	-
Zxing	zxing	1D/2D barcode image	3,030	3,253	791	123	94.31	15.55	26.70	29.35	37.96
Total			**96,003**	**165,912**	**41,225**	**15316**	**90.75**	**37.15**	**52.72**	**40.78**	**44.17**

Evaluation

- **42 open source projects**
- **Precision = 90%**
- **Recall: 37%**
- **79% of the proposed fixes are accurate**

Project Clustering

We can improve the detection accuracy if we search within inter-related projects



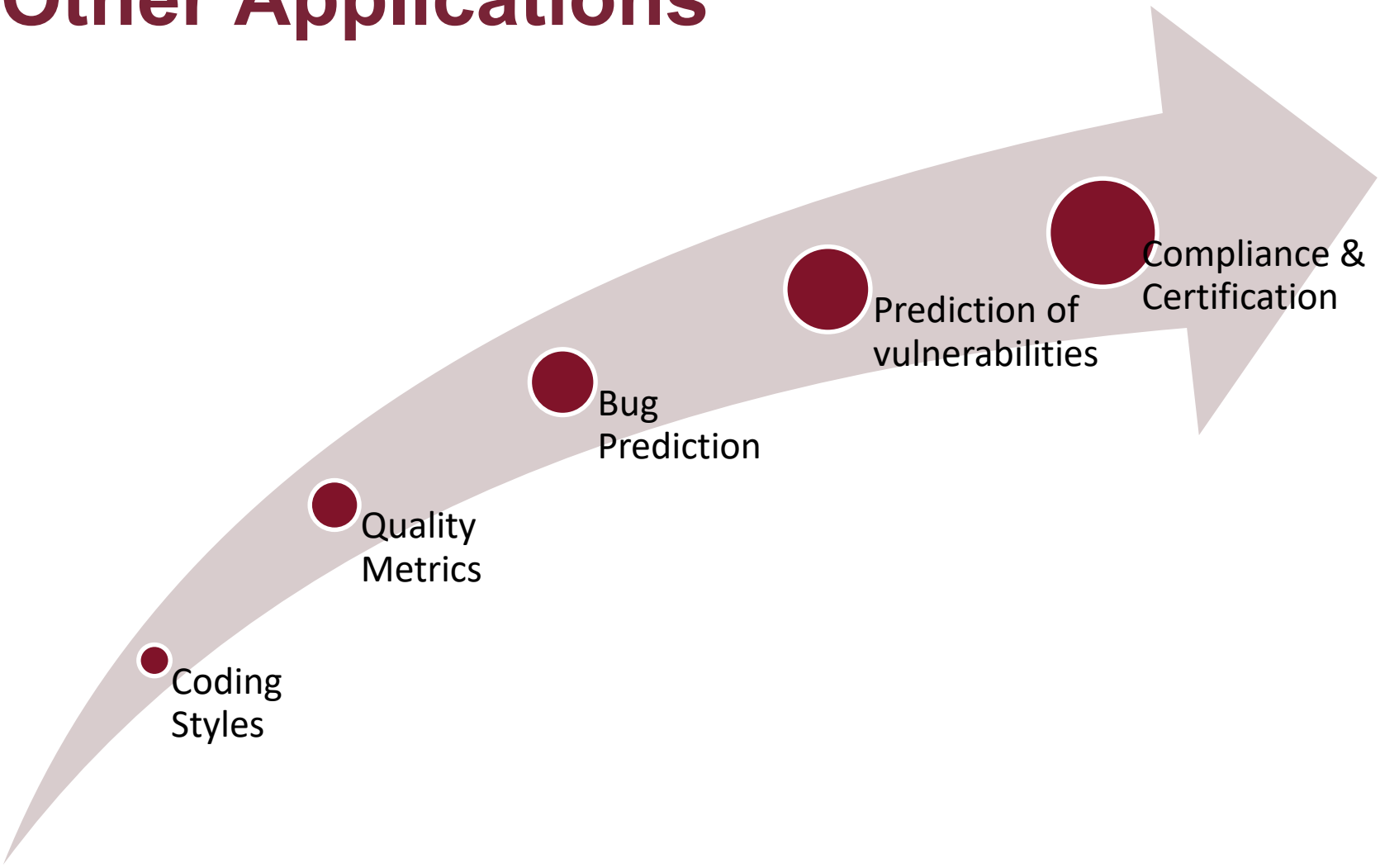
Evaluation of CommitAssistant at Ubisoft

- 12 Ubisoft AAA games
- 10+ millions of LOCs
- Precision = 79%
- Recall = 65%
- 67% of the fixes were deemed acceptable

Impact

- Commit-Assistant is designed **to integrate** well with the workflow of Ubisoft developers
- Clever-Commit (production version of Commit-Assistant) **is widely deployed** at Ubisoft
- Ubisoft announced in a press release that Commit-Assistant can **cut the bug fixing time by 20%**
- **Mozilla** announced that it is working with Ubisoft to use Clever-Commit in the **development of Firefox**

Other Applications



CommitAssistant as JIT Monitoring Tool

Analyzing commits provides real-time view of code quality:

- Num. of introduced bugs
- File metrics
- Subsystem metrics
- Code change density
- Code complexity
- Number of fixes
- Etc.

Open Questions

- How can we apply CommitAssistant to embedded and critical safety systems?
- What is the relationship between commit analysis, testing, tracing and logging, operational intelligence, etc.?
- Can this technology help with certification and compliance of software?
- Is this technology certifiable?

Engineering Complex
Preponderant Software Systems
Toulouse, France
October 16-17, 2019

Conclusion

- Machine learning and AI are needed to reduce overhead of bug fixing

- CommitAssistant:
  - reuses existing knowledge and AI to improve new code
  - improves quality by providing early feedback to developers
  - assists developers on how to fix risky commits
  - works well on Ubisoft systems

CONCORDIA.CA