

Improving the Quality of Software Using Testing and Fault Prediction

Professor Iftekhhar Ahmed
Department of Informatics
<https://www.ics.uci.edu/~iftekha/>

About me

- Research focus: Software testing and analysis.
- 4 years of industry experience.
 - Developed the first ever mobile commerce system in Bangladesh.
- IBM Ph.D. Fellowship (2016, 2017).
- Contributor to Linux Kernel.



The Ariane Rocket Disaster (1996)



https://youtu.be/PK_yguLapgA?t=50s

Root cause

- Caused due to numeric overflow error
 - Attempt to fit 64-bit format data in 16-bit space
- Cost
 - \$100M's for loss of mission
 - Multi-year setback to the Ariane program
- Read more at <http://www.around.com/ariane.html>

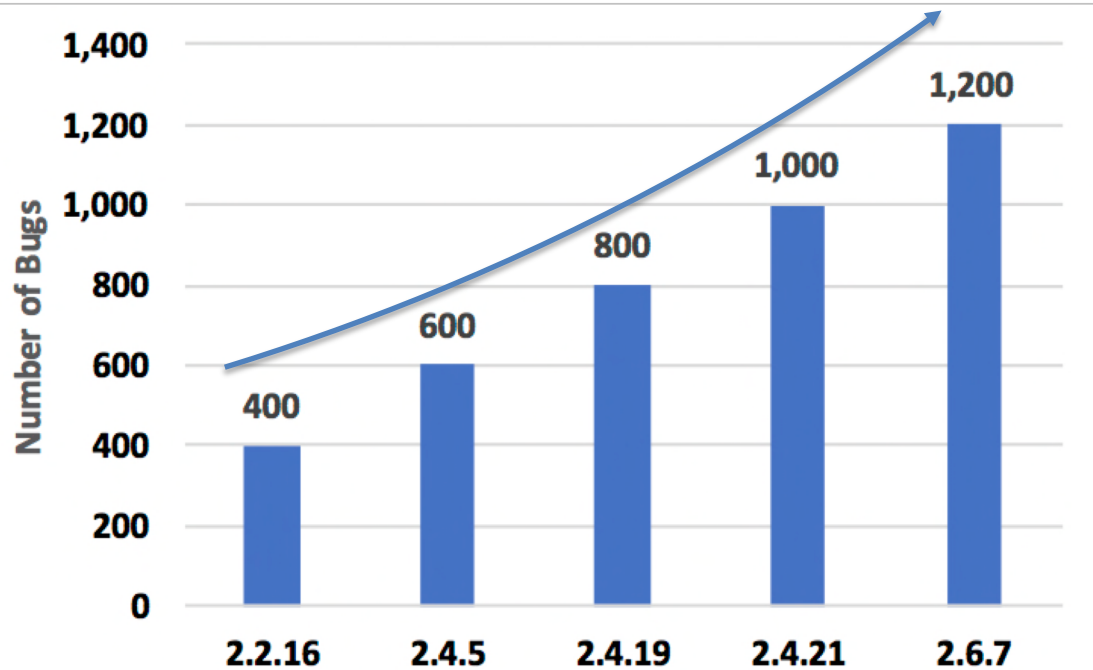


Software is a critical part of our life

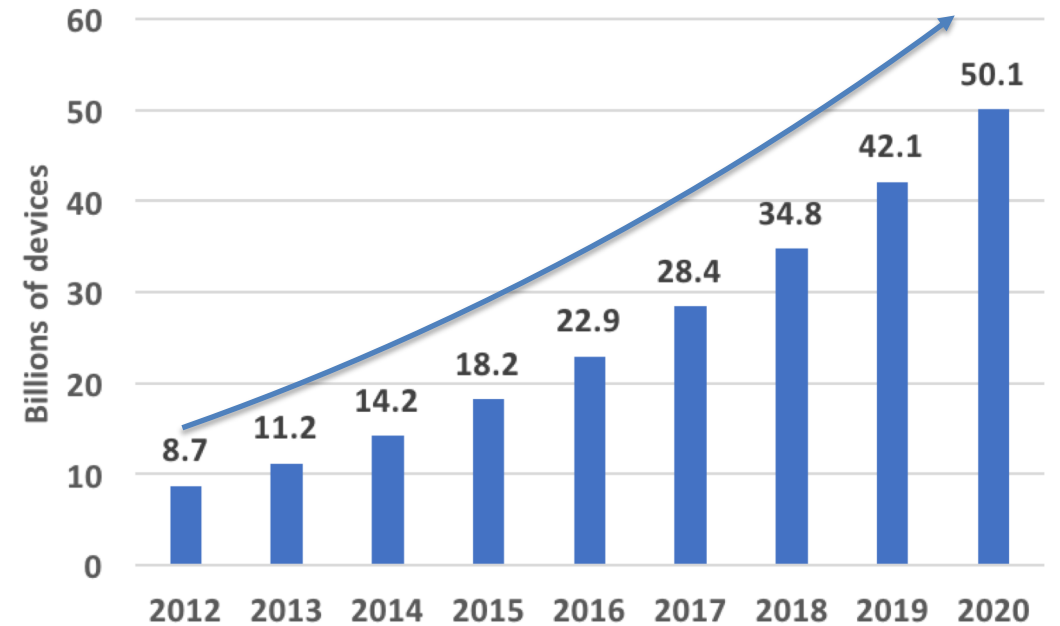


Source: <https://pbs.twimg.com/media/DWwOtruVMAAh1sD.jpg>

Why should we care about software quality?



Code growth and defect in Linux Kernel
(Harris et al. 2016)



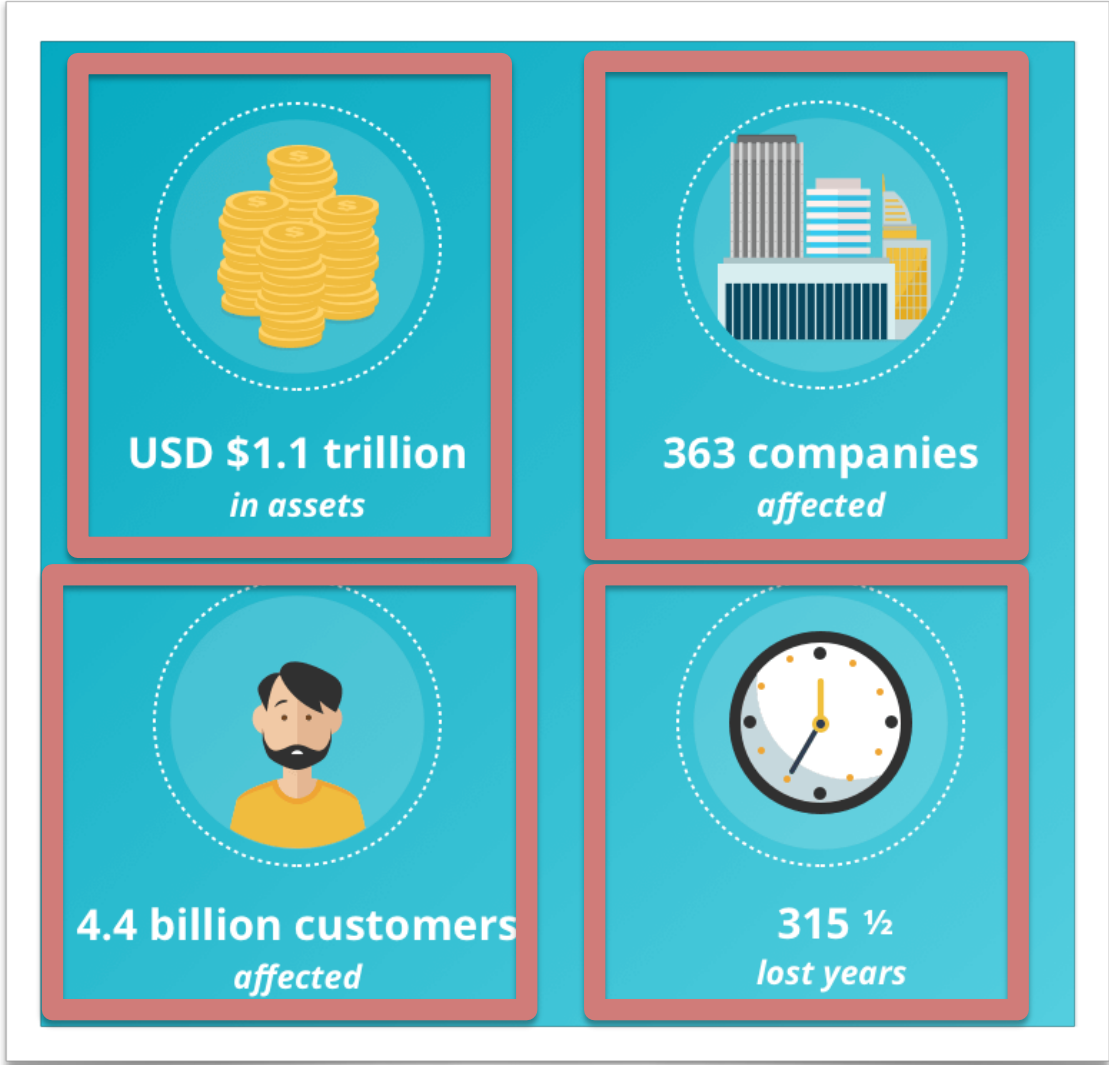
Number of connected devices in IOT

Source: Cisco

Cost of software failure is increasing



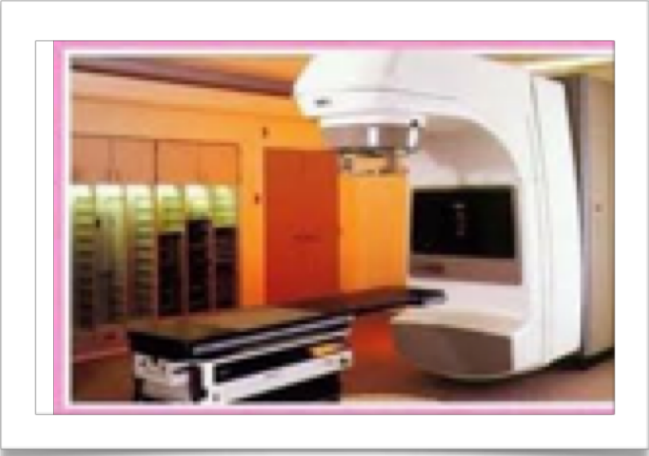
Indian Airlines Flight 605 -- February 14, 1990



The cost of software failure in 2016

Source:Software Fail Watch

What do we do to make software better ?



We also need to think about the developer

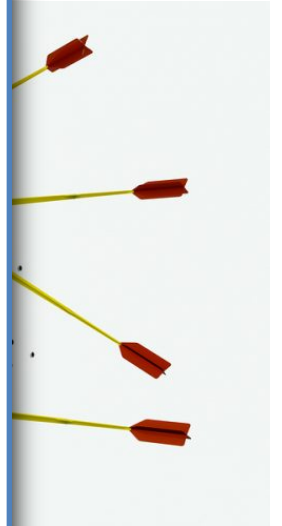
- Lack
- Tool
- Tool
 - Ti
- And

We need tools/techniques that are not only

Scalable, Effective

but also

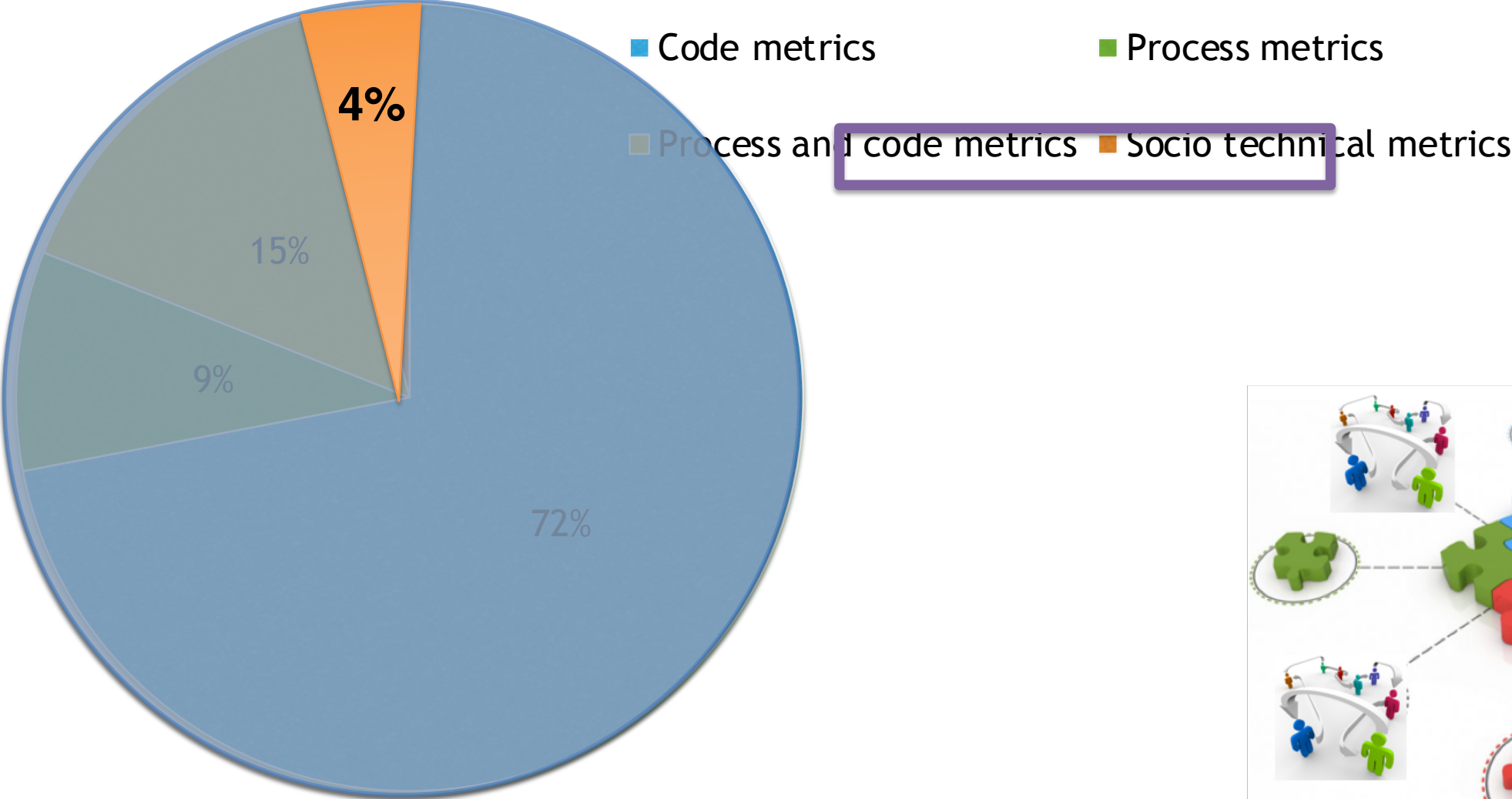
Easy to use



Identifying factors impacting code quality



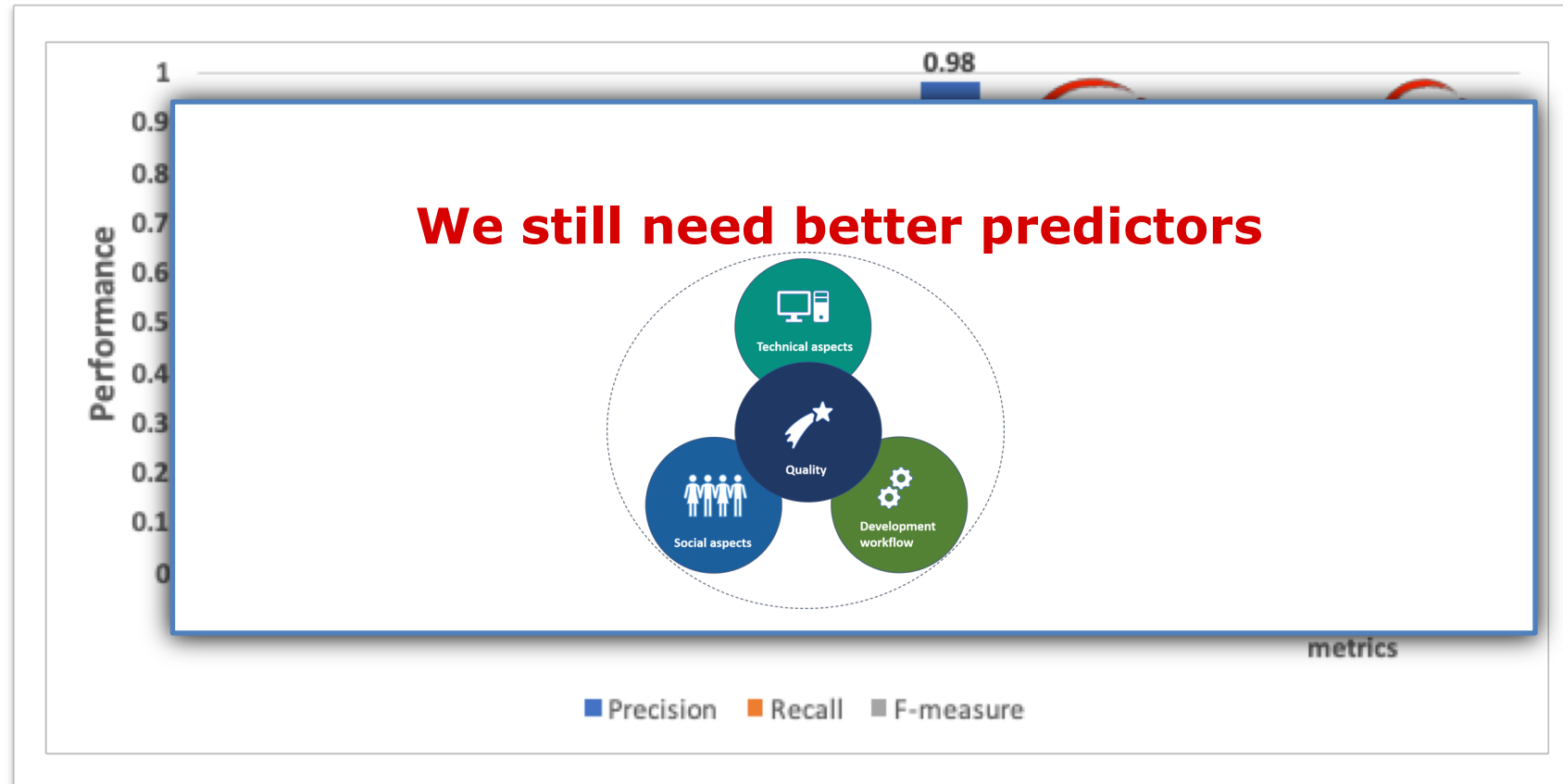
Fault prediction metrics



(Hall et al. 2012)

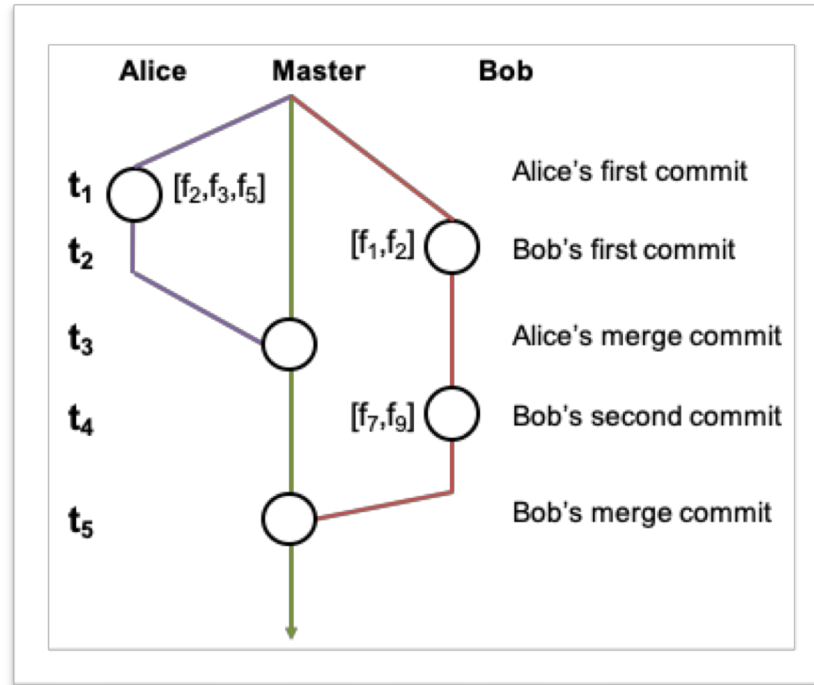


Fault prediction performance








(Hall et al. 2012)


Merge conflict



$$DC = E_A \cap E_B \quad \text{Direct Conflicts: } \{f_2\}$$

 **Some checks were not successful** [Hide all checks](#)
1 errored and 1 successful checks

-   continuous-integration/travis-ci/pr — The Travis CI build could not complete ... [Details](#)
-   continuous-integration/appveyor/pr — AppVeyor build succeeded [Details](#)

 **This branch has conflicts that must be resolved**
Only those with [write access](#) to this repository can merge pull requests.

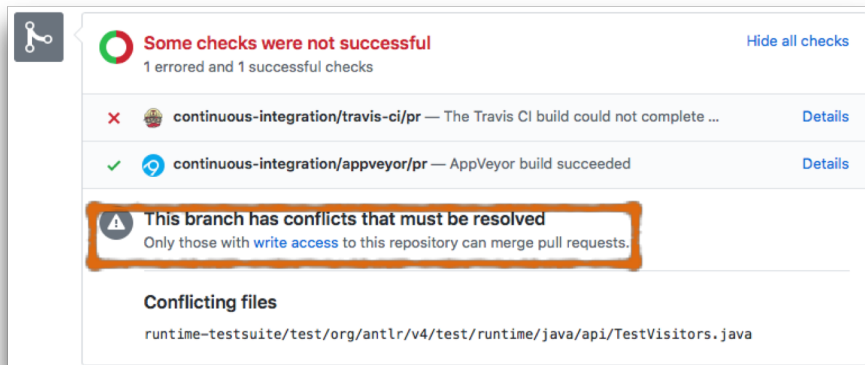
Conflicting files

runtime-testsuite/test/org/antlr/v4/test/runtime/java/api/TestVisitors.java

```
Auto-merging molgenis-web/src/test/java/org/molgenis/web/ErrorMessage.java
Auto-merging molgenis-web/src/main/java/org/molgenis/web/ErrorMessage.java
Auto-merging molgenis-security/src/main/java/org/molgenis/security/MessageDigest.java
CONFLICT (content): Merge conflict in molgenis-security/src/main/java/org/molgenis/security/MessageDigest.java
Auto-merging molgenis-dataexplorer/src/main/java/org/molgenis/dataexplorer/MessageDigest.java
CONFLICT (content): Merge conflict in molgenis-dataexplorer/src/main/java/org/molgenis/dataexplorer/MessageDigest.java
Auto-merging molgenis-data-rest/src/main/java/org/molgenis/data/rest/MessageDigest.java
CONFLICT (content): Merge conflict in molgenis-data-rest/src/main/java/org/molgenis/data/rest/MessageDigest.java
Auto-merging molgenis-core-ui/src/main/java/org/molgenis/ui/admin/MessageDigest.java
Automatic merge failed; fix conflicts and then commit the result.
```

Merge conflict - a socio-technical factor

- Related to **collaborative development** work distribution.
- A developer has to interrupt their work
 - An **immediate concern**.
- They are a common occurrence.
 - In our corpus, **over 19% of merges result in a conflict (6,979 merge conflicts out of 36,111 merges)**



```
Auto-merging molgenis-web/src/test/java/org/molgenis/web/ErrorMessage.java
Auto-merging molgenis-web/src/main/java/org/molgenis/web/ErrorMessage.java
Auto-merging molgenis-security/src/main/java/org/molgenis/security/ErrorMessage.java
CONFLICT (content): Merge conflict in molgenis-security/src/main/java/org/molgenis/security/ErrorMessage.java
Auto-merging molgenis-dataexplorer/src/main/java/org/molgenis/dataexplorer/ErrorMessage.java
CONFLICT (content): Merge conflict in molgenis-dataexplorer/src/main/java/org/molgenis/dataexplorer/ErrorMessage.java
Auto-merging molgenis-data-rest/src/main/java/org/molgenis/data/rest/ErrorMessage.java
CONFLICT (content): Merge conflict in molgenis-data-rest/src/main/java/org/molgenis/data/rest/ErrorMessage.java
Auto-merging molgenis-core-ui/src/main/java/org/molgenis/ui/admin/ErrorMessage.java
Automatic merge failed; fix conflicts and then commit the result.
```

Prior work on merge conflict

- Merge conflict detection (Brun et al. 2013)
- Merge conflict resolution (Apel et al.2013)
- Awareness for reducing merge conflicts (Sarma et al. 2007)
- Merge conflict categorization (Brun et al. 2013)

What is the effect of merge conflict on code quality measured by bug proneness and code smells?

Code smell, a technical factor

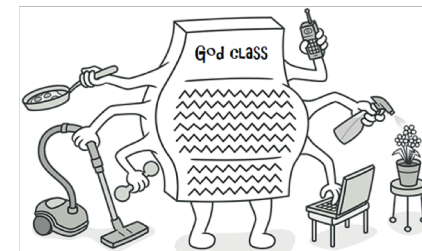
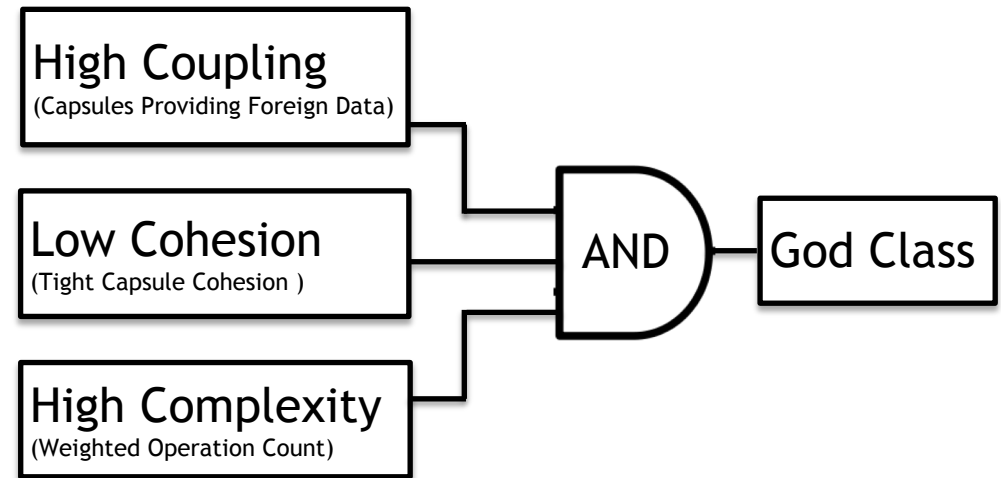
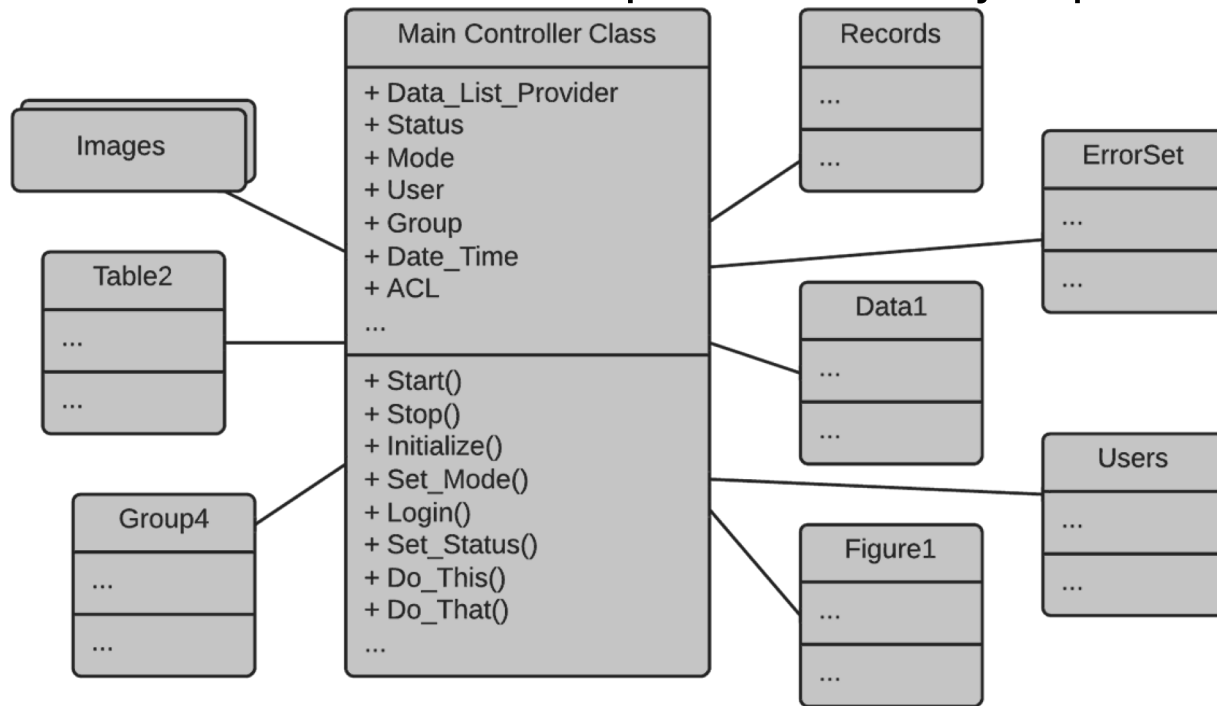
- Developed to identify *future* maintainability problems
- Neither syntax errors nor compiler warnings
- *Symptoms* of poor design or implementation choices



God class

“God class tends to concentrate functionality from several unrelated classes”

Arise when developers do not fully exploit the advantages of object-oriented design

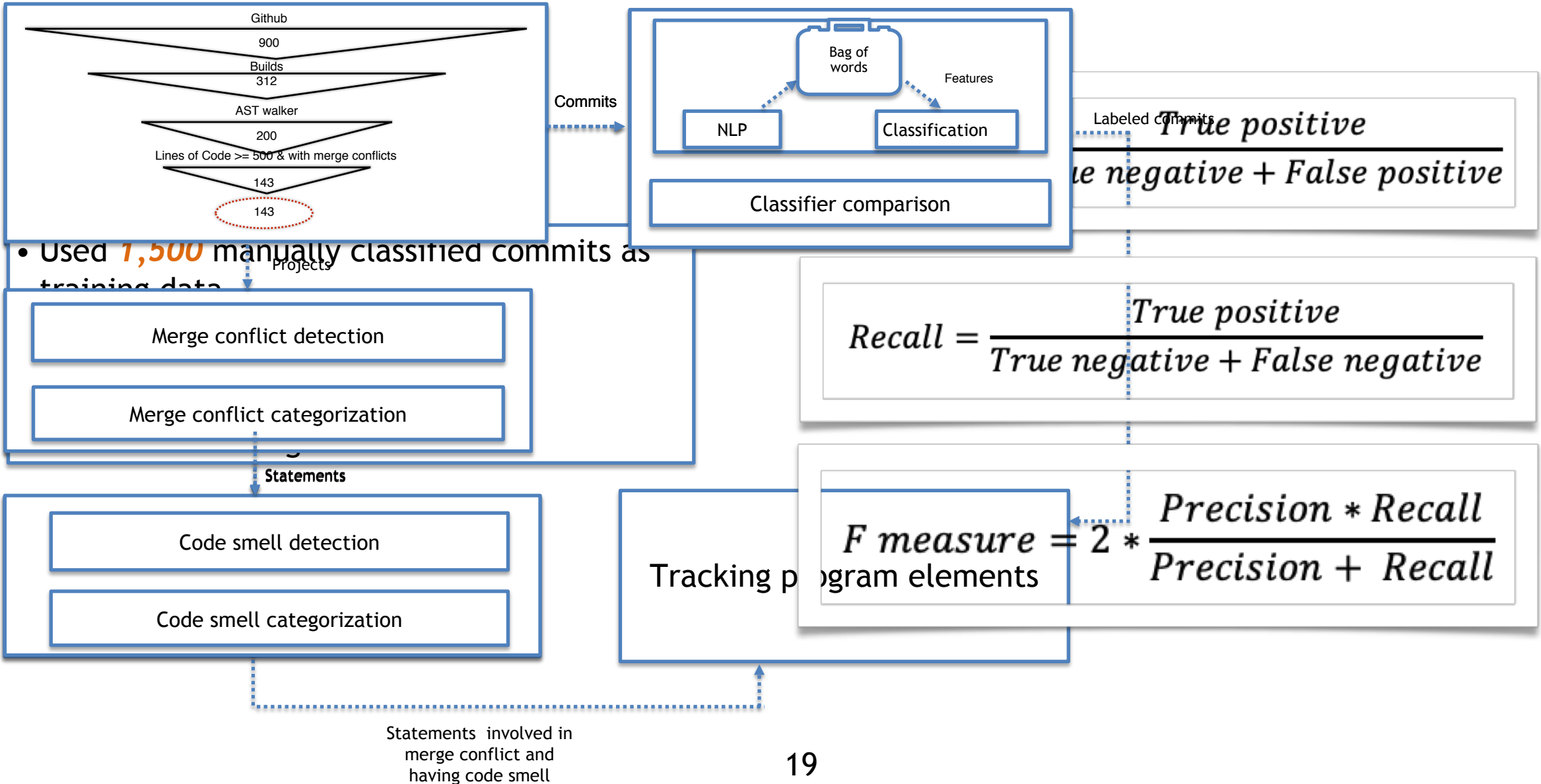


Prior work on code smell

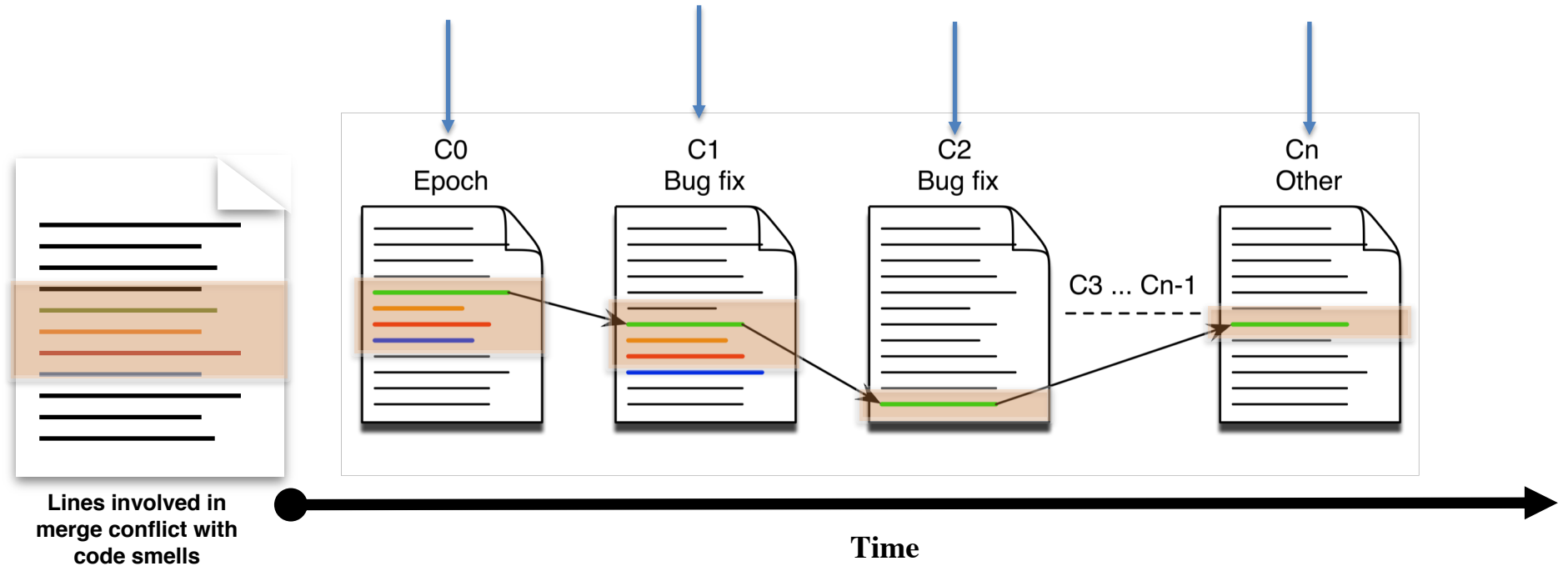
- Detection techniques (Palomba et al. 2013)
- Association with bugs (Oliva et al. 2013)
- Categorizations (Marticorena et al. 2006)

Interaction of code smell and merge conflict on code quality?

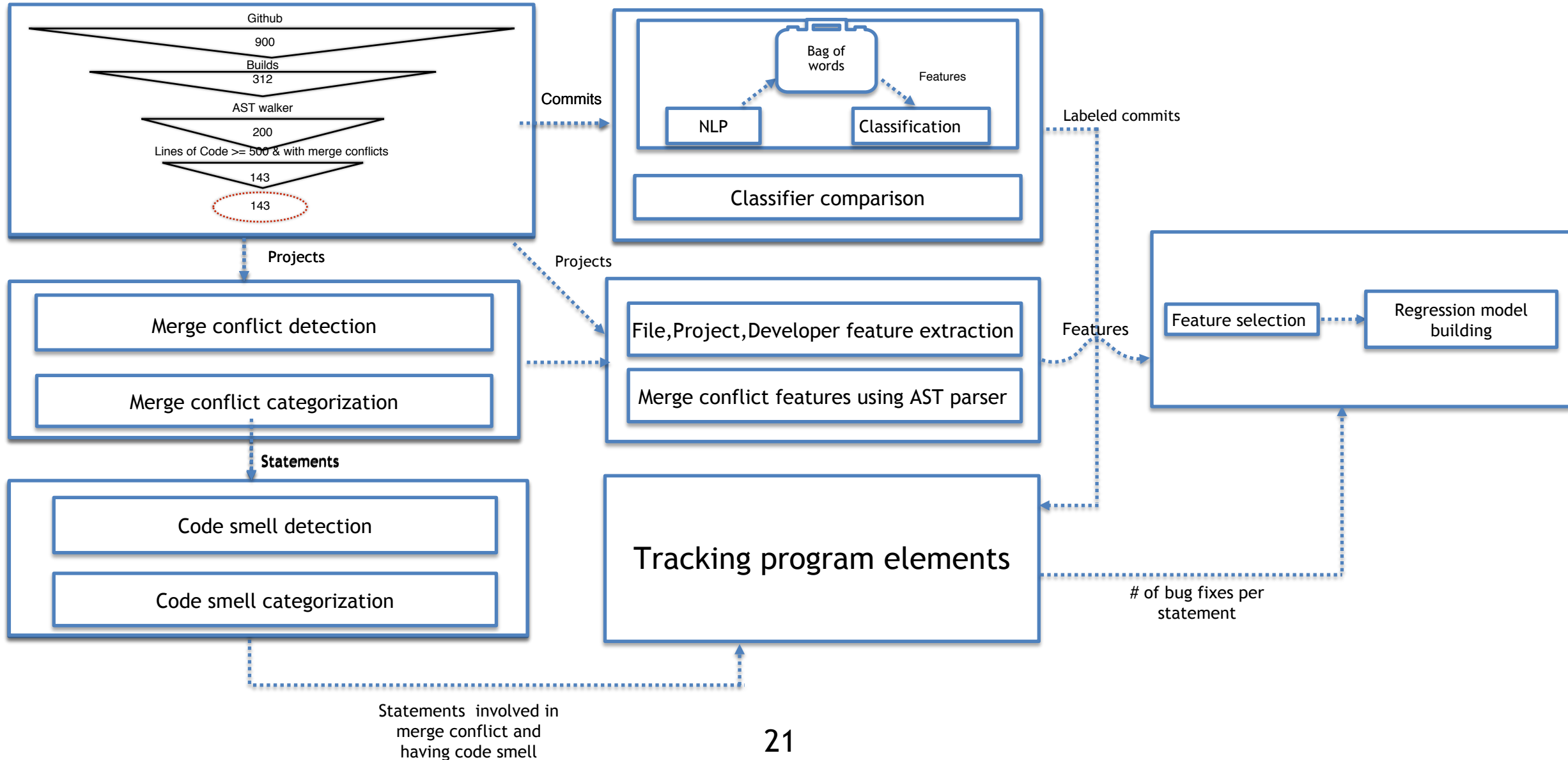
Steps of empirical analysis



Tracking conflicted smelly lines



Steps of empirical analysis





Relationship between code smells and merge conflict

Program elements involved in a merge conflict have an average of **6.54** smells, while those that don't have an average of **1.92**.

Elements involved in a conflict contain 3x more code smells than element not involved in a conflict.

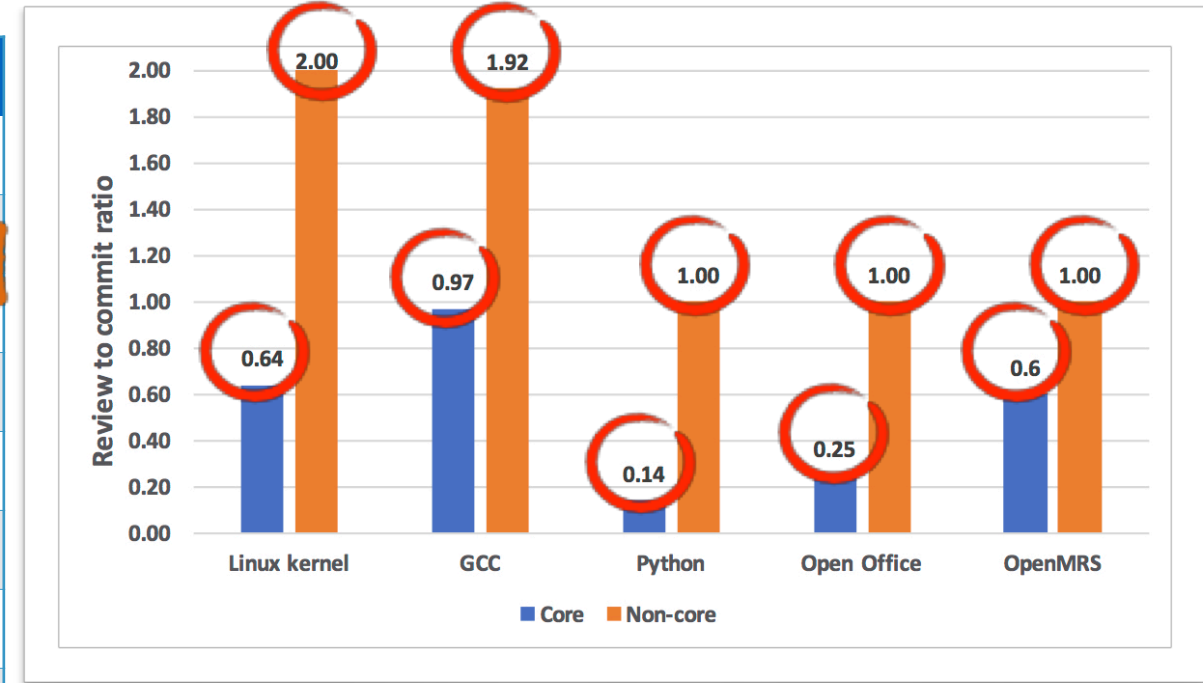
Which code smells are more associated with merge conflict?

Smell	Pearson correlation coefficient with # of conflicts
God Class	0.18
Internal Duplication	0.17
Distorted Hierarchy	0.13

These 3 smells are indicative of bad code structure, at a class level.

What about bugs?

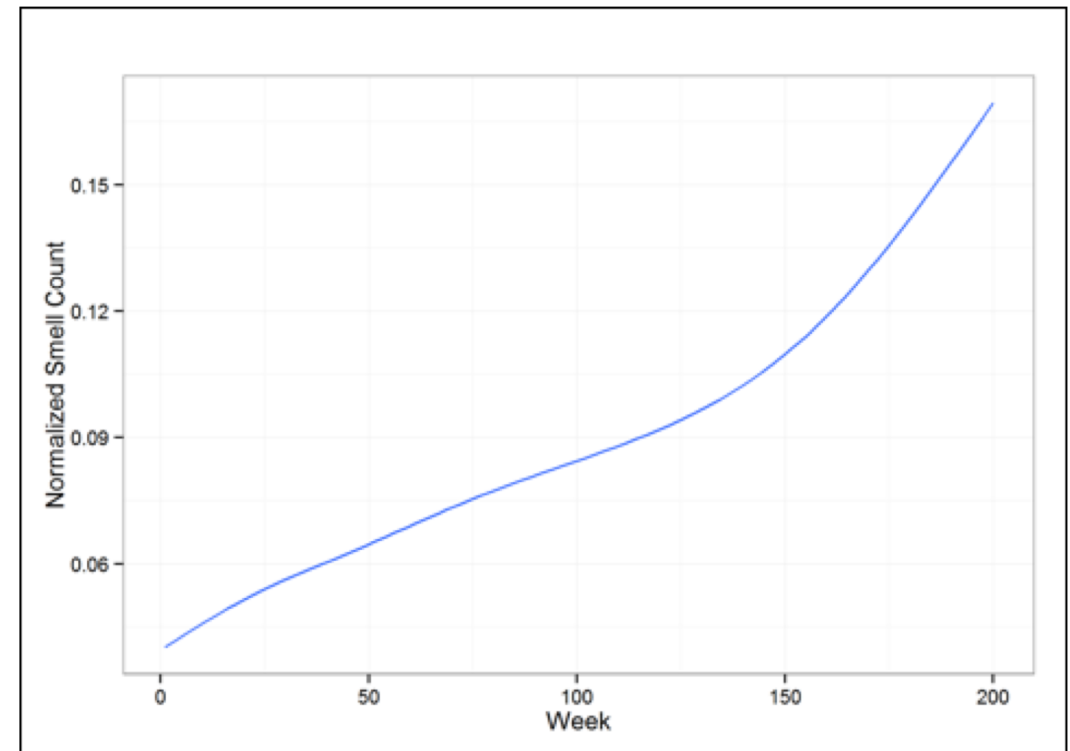
Factor	Coefficients
In Deps	3.19
Out Deps	-0.05
Noncore author	-3.79
No. Authors	0.12
No. Classes	-0.37
No. Methods	0.24
AST diff	0.00
LOC diff	0.01
No. of Smells	0.42



Ahmed et al. 2018 (work in progress)

What does this mean?

- Elements involved in a conflict contain **3x** more code smells than element not involved in a conflict.
 - All smells do not contribute **equally**.
- **Longer a project runs** the more smelly it becomes.
 - More likely to run into merge conflicts.
- A **new socio-technical factor** for bug prediction
 - Statements involved in **a merge conflict with code smells**



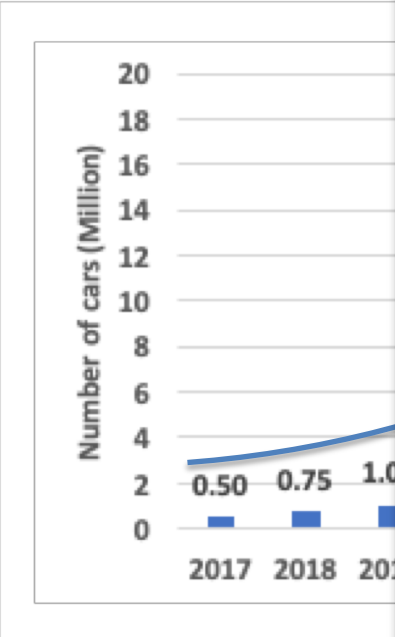
Week-wise average project smelliness
Ahmed et al. 2015

What about systems that behave stochastically?



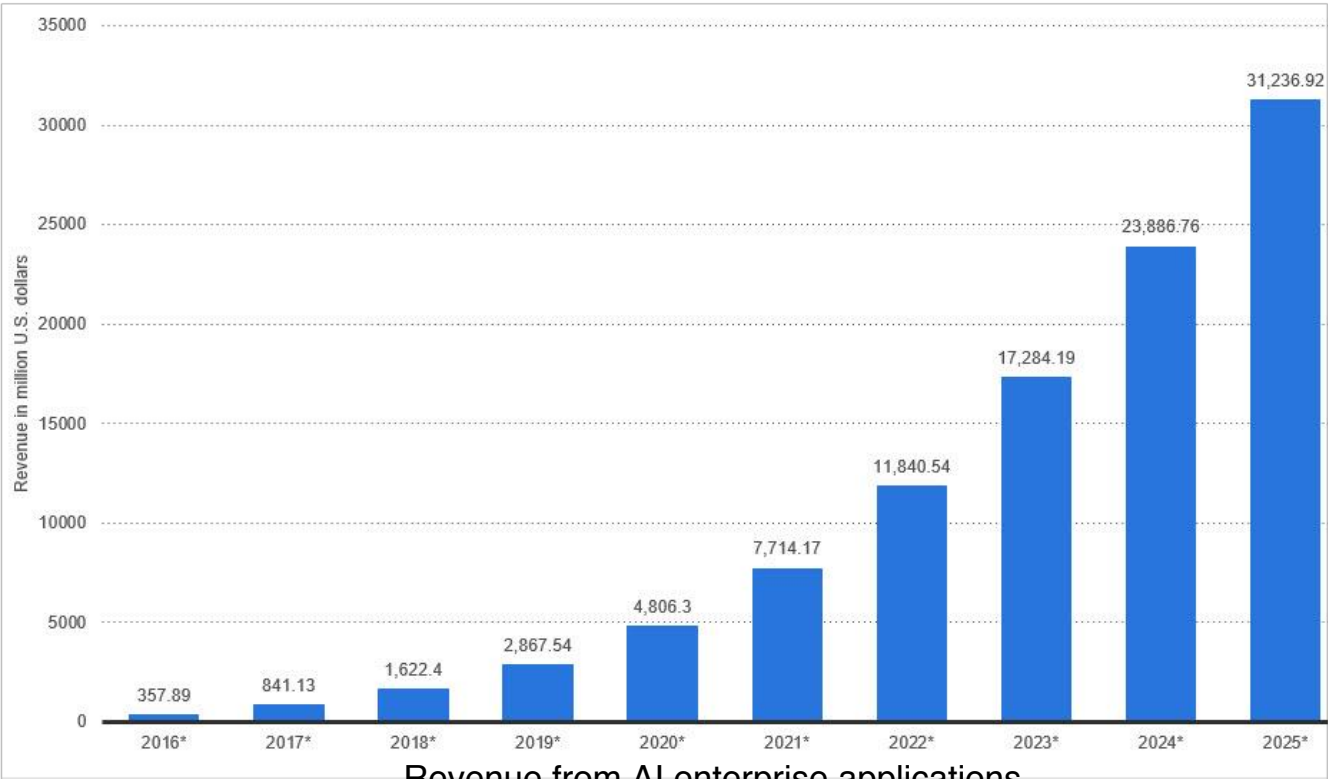
Stochastic systems

- Stochastic in nature
- Bugs in are c



Number of autonou

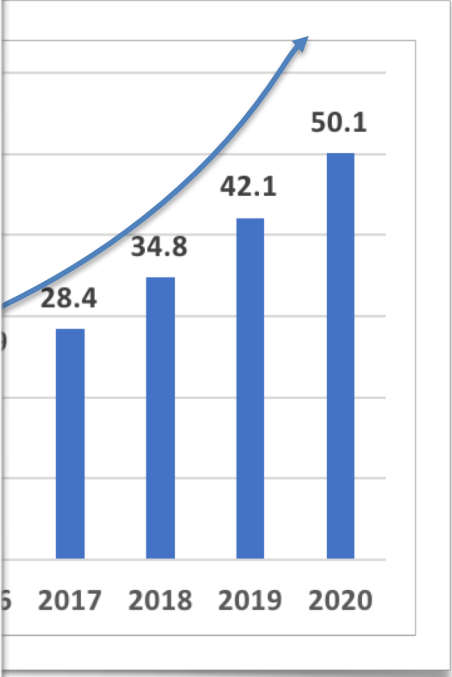
Source: Morgan



Revenue from AI enterprise applications

Source: Statista

Source: Cisco



devices in IOT

Testing challenges for autonomous vehicles



Tesla autopilot failed to recognize a white truck against bright sky leading to fatal crash

Enter mutation analysis

- Addressing the Oracle Problem
- Mutants look like real bugs

$$\Delta = b^2 - 4ac$$

$$d = b^3 - 4 * a * c$$



Mutants **killed** by test cases

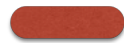
$$d = b^2 + 4 * a * c$$



Test cases



(a = 0, b = 0, c = 0) => (d = 0)



(a = 1, b = 1, c = 1) => (d = -3)

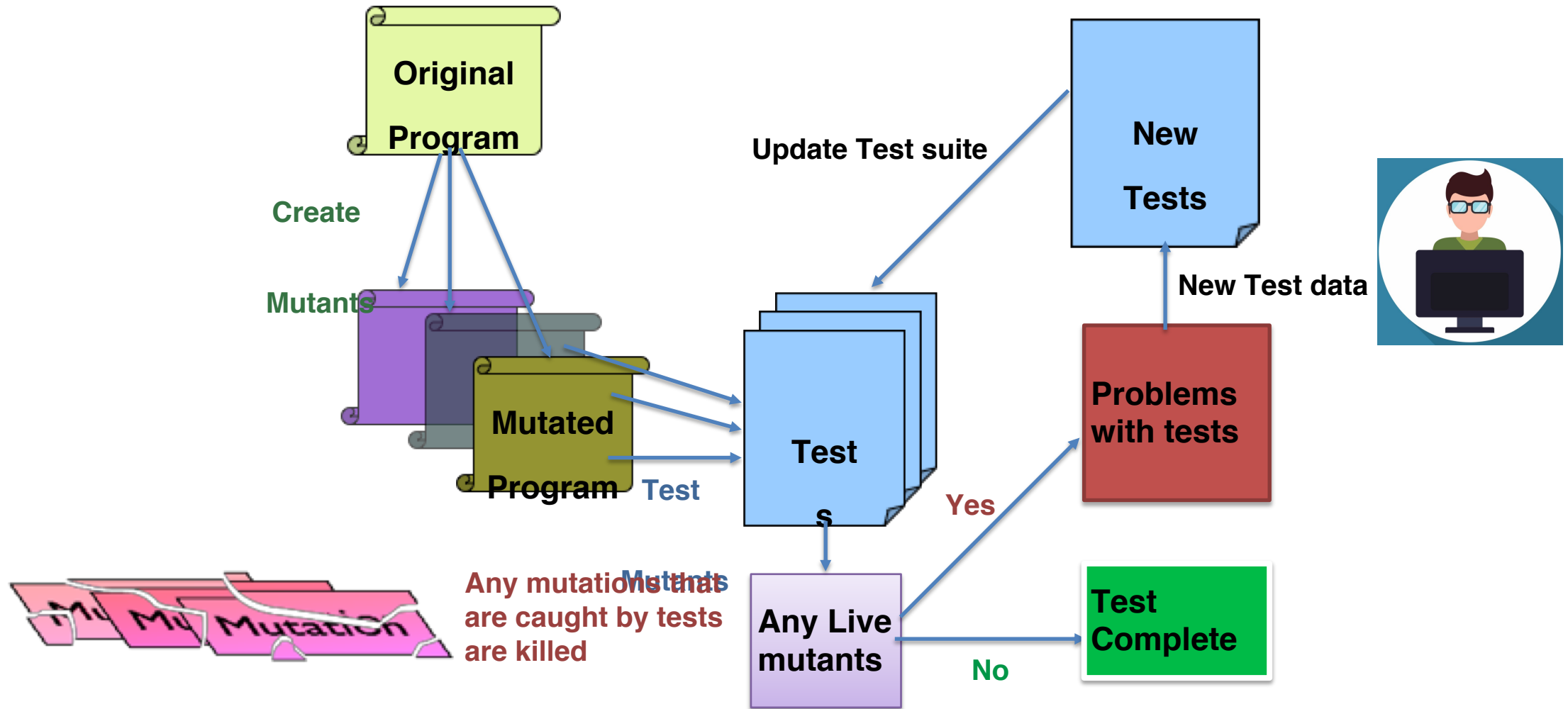


(a = 0, b = 2, c = 0) => (d = 4)

$$d = b^2 - 4 + a * c$$

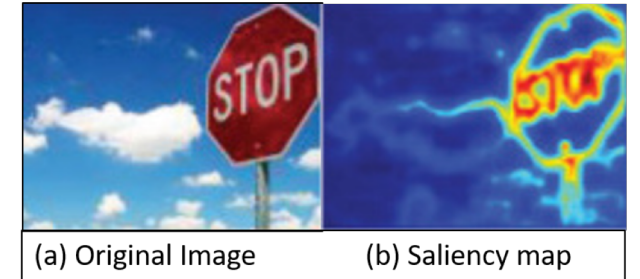


The mutation analysis process



Simulating robust physical perturbations

- Mutating inputs to each subsystem (**Fuzzing**)
- Mutating combinations of subsystems together (**Higher Order Mutants**)
- **Adversarial** testing meets **mutation** testing
- Identifying **important regions** of the image using **saliency map**
- Ensuring **mutated inputs** are **realistic**



Distance/Angle	Subtle Poster	Subtle Poster Right Turn	Camouflage Graffiti	Camouflage Art (LISA-CNN)	Camouflage Art (GTSRB*-CNN)
5° 0°					
5° 15°					
10° 0°					
10° 30°					
40° 0°					
Targeted-Attack Success	100%	73.33%	66.67%	100%	80%

Evtimov et al. 2017

Conclusion

Software is a critical part of our life



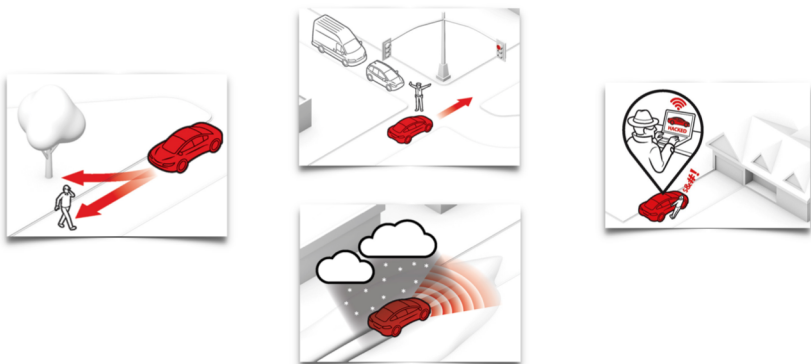
Testing works but ...

- Test coverage is not enough.
 - There is a **limit on return on investment**.
 - High coverage **doesn't guarantee bug free** software.
 - **Continuous improvement related to coverage** (vs. mere measure of "tested or not?") is not evident
- We need to **move beyond coverage** to measure the quality of the tests.

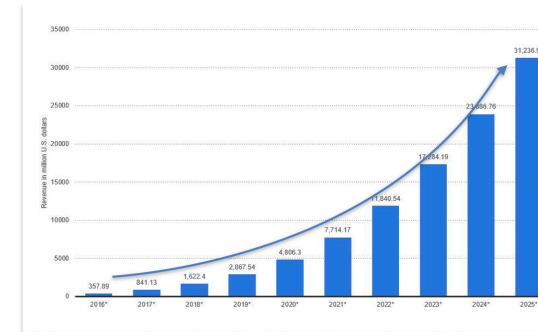
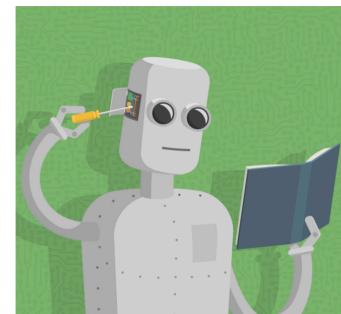
Stochastic systems and Flaky test

- Stochastic in nature
 - Bugs in are often non-deterministic.
- A flaky test is a test which could fail or pass for the same configuration
 - **Harmful to developers because test failures do not always indicate bugs in the code**

Unique challenges



Rise of Machine Learning



Revenue from AI enterprise applications
Source: Statista