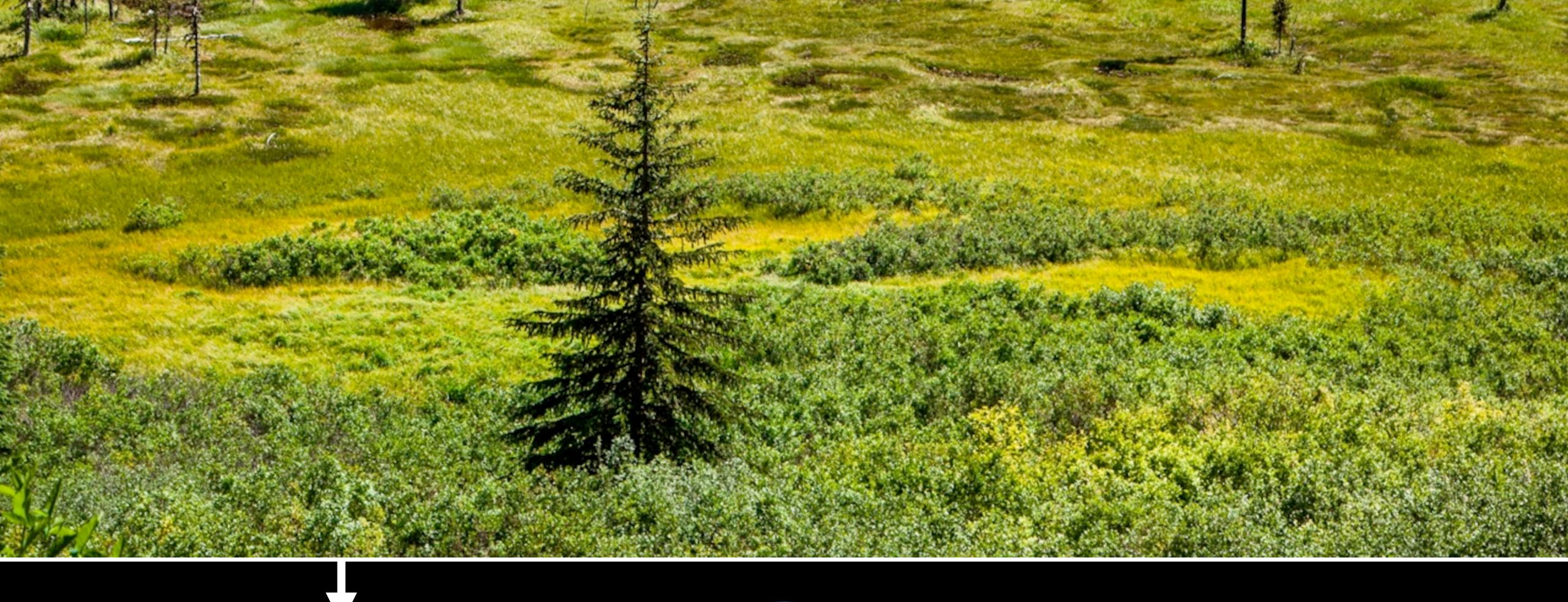
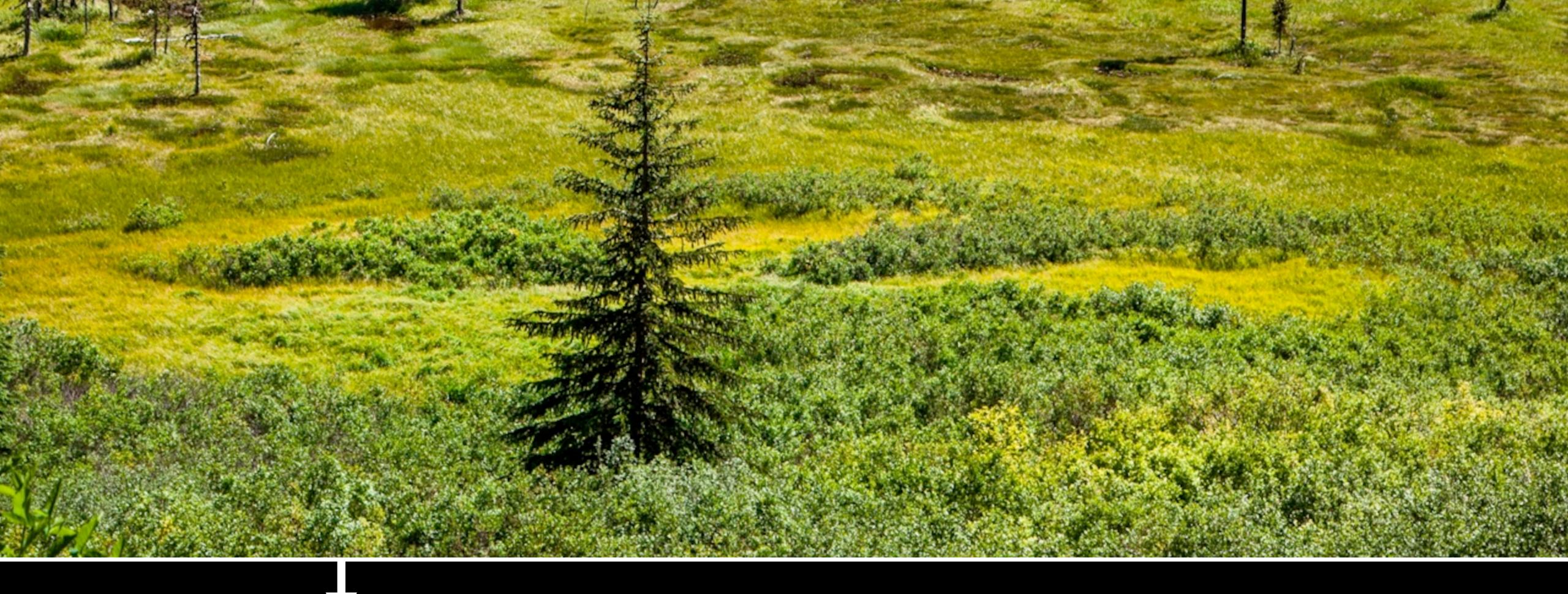


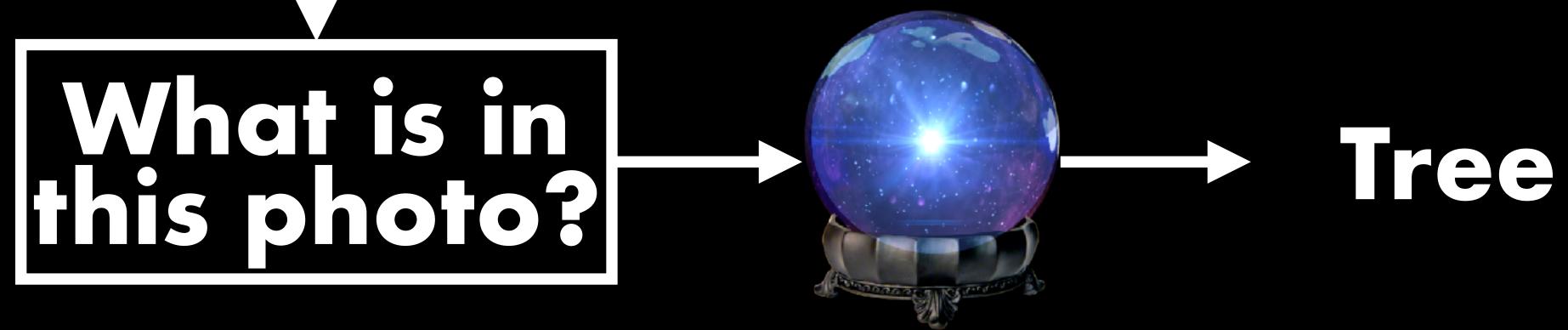
What is in this photo?



What is in this photo?





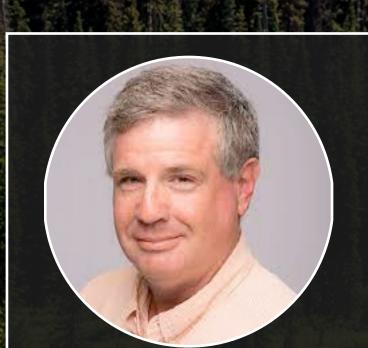




Promoting Situational Awareness in Code Review REFERENS

Shane McIntosh





Michael W. Godfrey





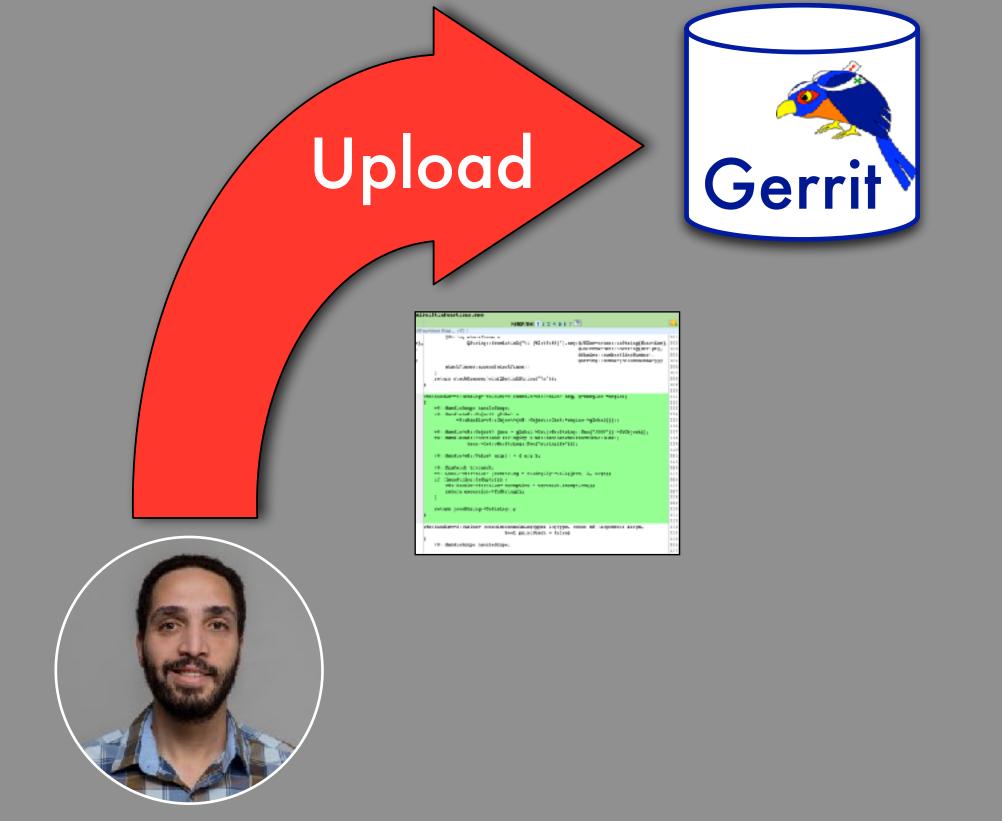




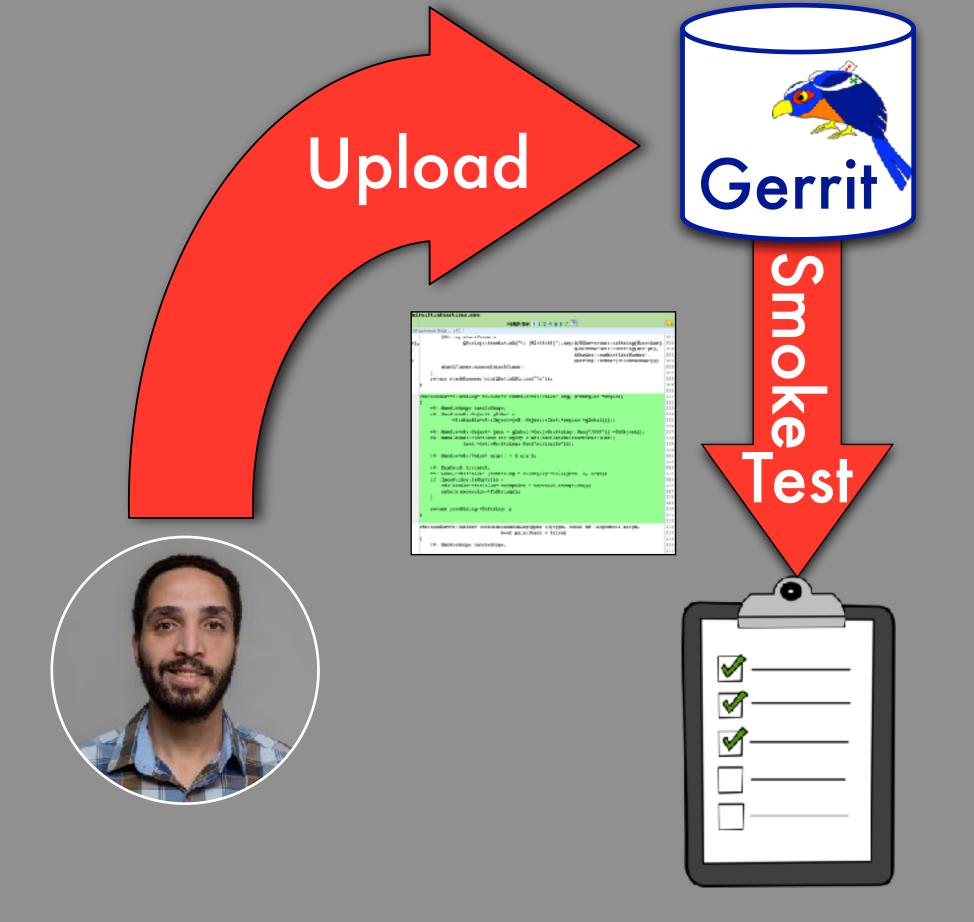




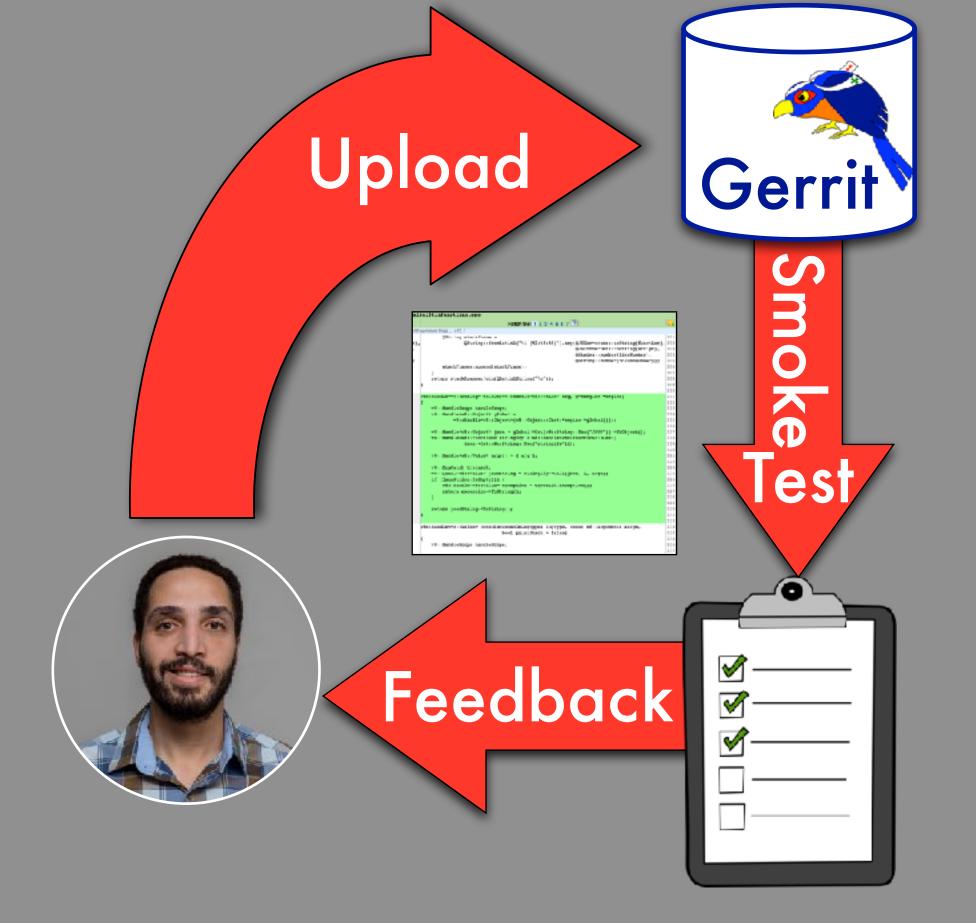




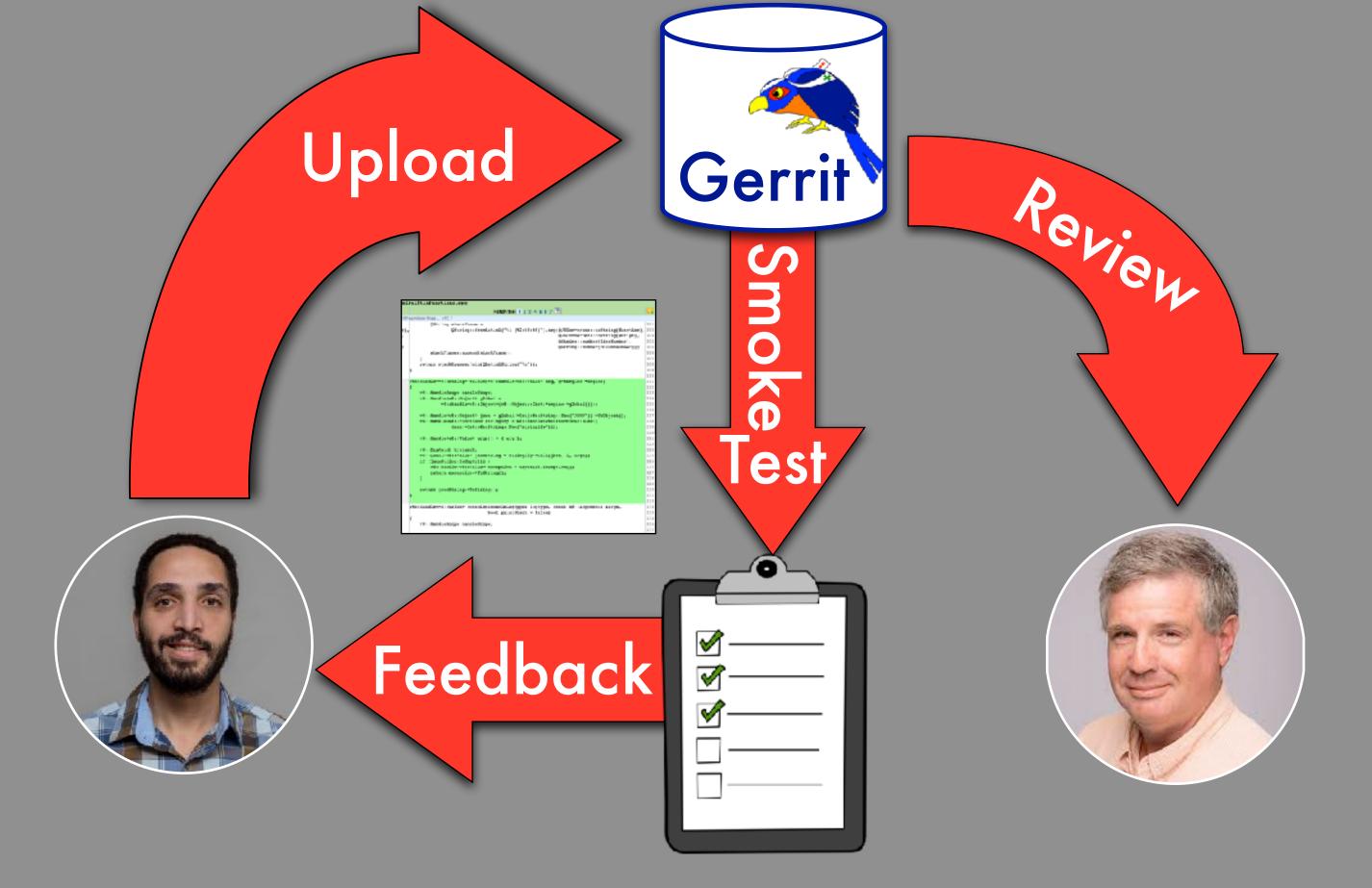




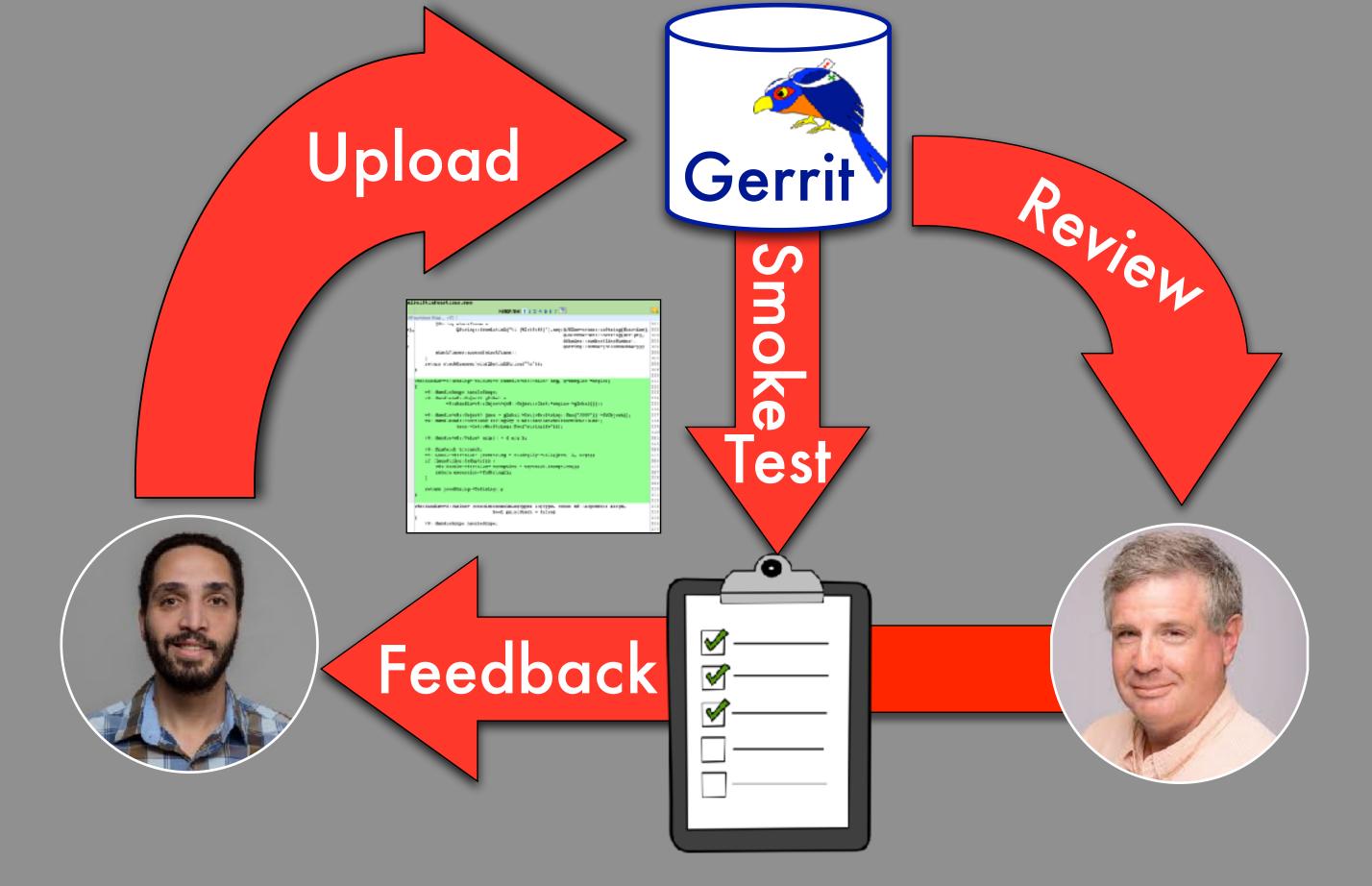




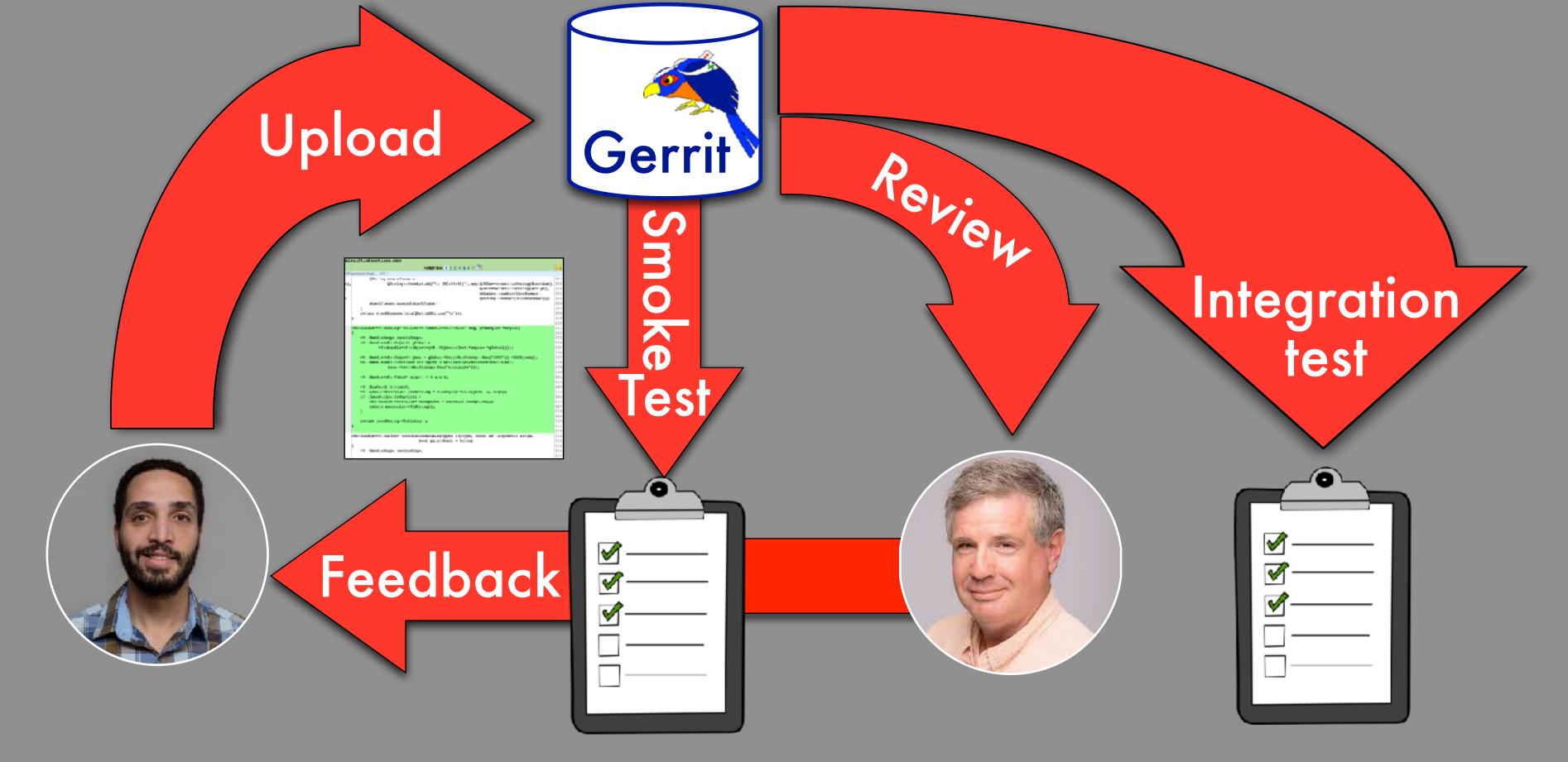




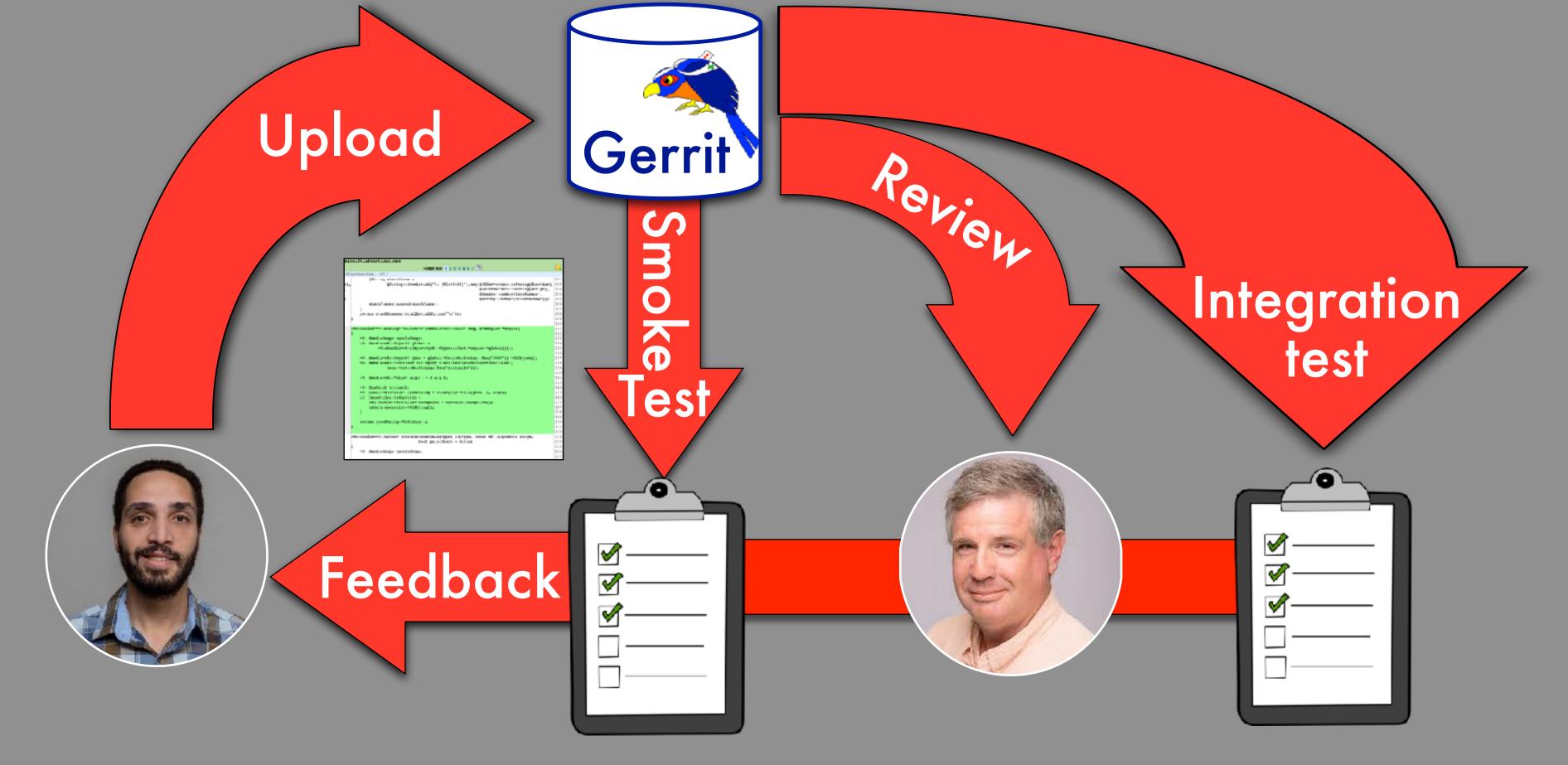




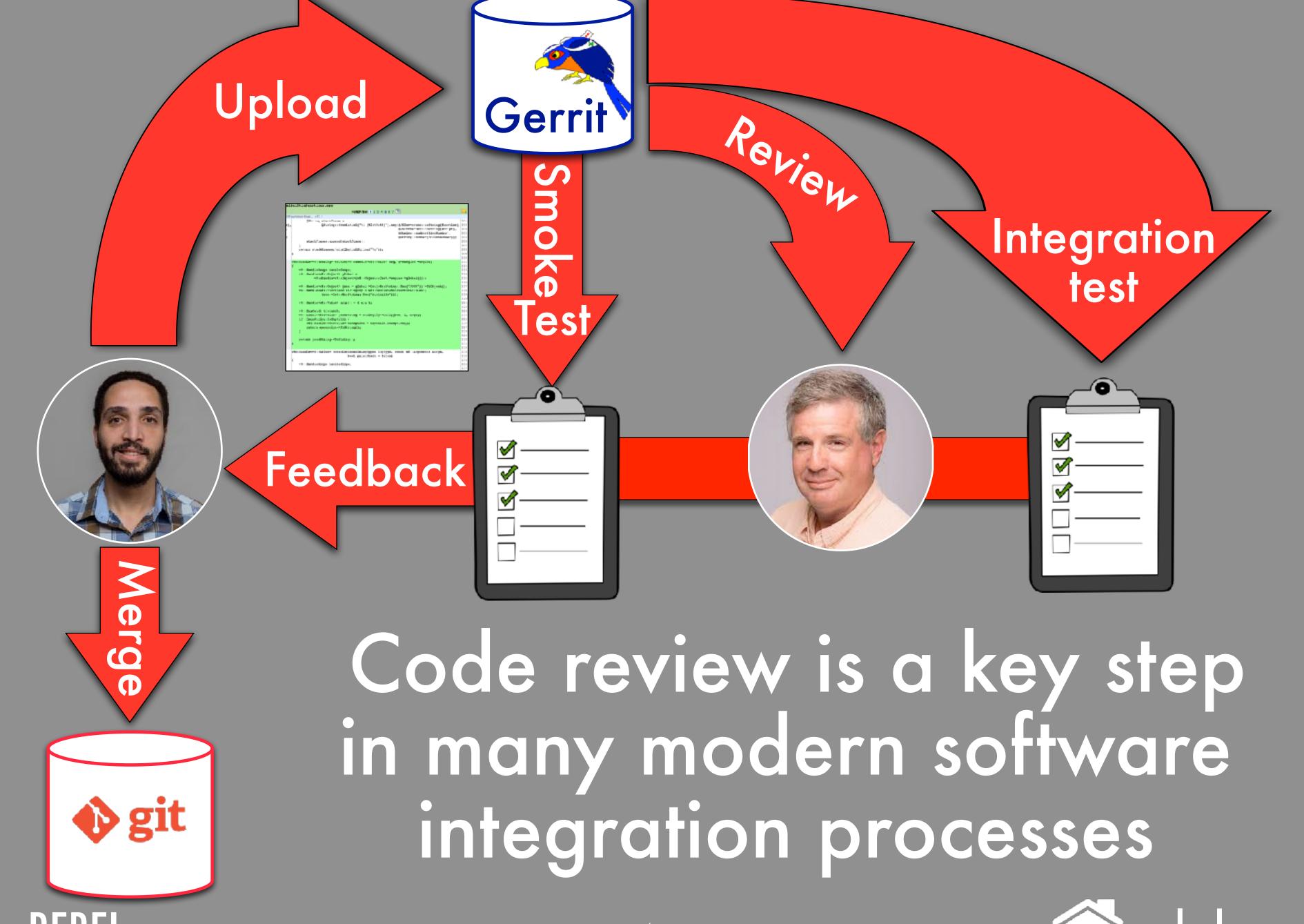
















Expectations, Outcomes, and Challenges of Modern Code Review
[Bacchelli and Bird, ICSE 2013]

Convergent Contemporary Software
Peer Review Practices
[Rigby and Bird, FSE 2013]







Expectations, Outcomes, and Challenges of Modern Code Review
[Bacchelli and Bird, ICSE 2013]

Convergent Contemporary Software
Peer Review Practices
[Rigby and Bird, FSE 2013]

Ke







Early Detection of Defects



Discuss Alternative Approaches

Expectations, Outcomes, and Challenges of Modern Code Review [Bacchelli and Bird, ICSE 2013]

Convergent Contemporary Software Peer Review Practices [Rigby and Bird, FSE 2013]







Early Detection of Defects



Peer Mentorship



Discuss Alternative Approaches

Expectations, Outcomes, and Challenges of Modern Code Review

[Bacchelli and Bird, ICSE 2013]

Convergent Contemporary Software Peer Review Practices [Rigby and Bird, FSE 2013]







Early Detection of Defects



Discuss Alternative Approaches



Peer Mentorship



Communicate Changes

Expectations, Outcomes, and Challenges of Modern Code Review [Bacchelli and Bird, ICSE 2013]

Convergent Contemporary Software Peer Review Practices [Rigby and Bird, FSE 2013]







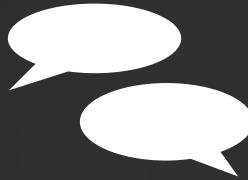
Early Detection of Defects



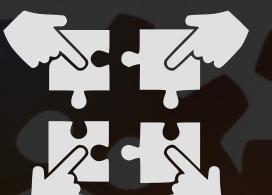
Discuss Alternative Approaches



Peer Mentorship



Communicate Changes



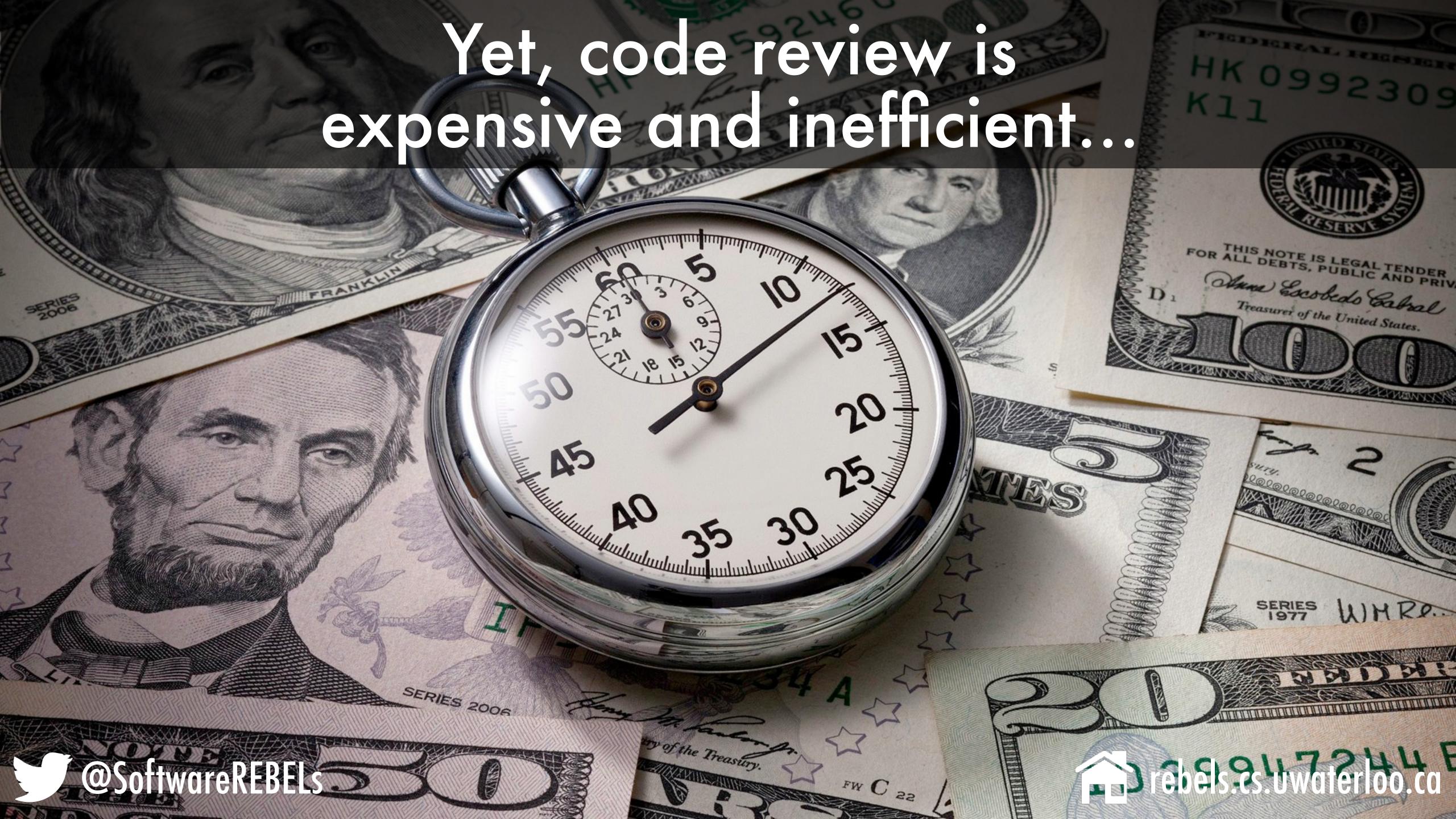
Collaborative Problem Solving

Expectations, Outcomes, and Challenges of Modern Code Review [Bacchelli and Bird, ICSE 2013]

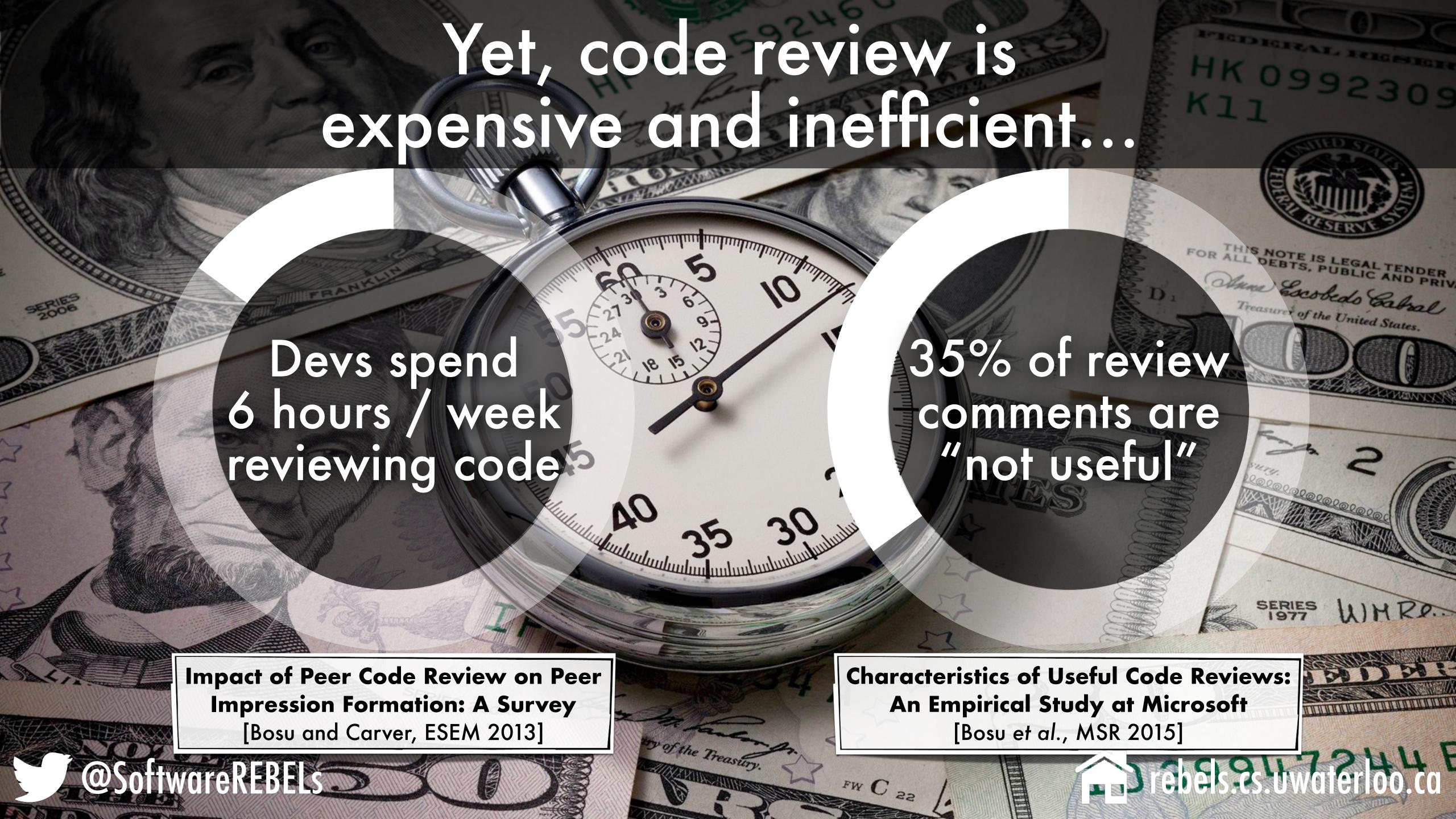
Convergent Contemporary Software **Peer Review Practices** [Rigby and Bird, FSE 2013]











...And code review is still largely based on sleuth work using textual differences

```
m_fetch = emscripten_fetch(&attr, request.url().toString().toUtf8());
q_wasmReplyFetchList.insert(m_fetch->id, (quintptr)this);

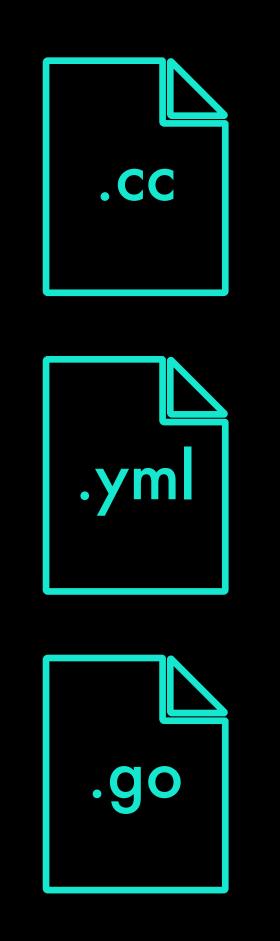
Reviewer

I take it emscripten_fetch(&attr, ...)'s return has the same userData, i.e. this, as attr, and mirrors this in its
__attributes (i.e. m_fetch.__attributes is attr, give or take copying, and m_fetch.userData == attr.userData).
So abort()'s clearing of m_fetch.__attributes.userData makes it possible for us to see that m_fetch has been aborted.
However, I'm unclear on how that clears m_fetch.userData - is this property actually mediated by an operator.() that consults __attributes ? - where the code below that's checking userData is still as expected is checking fetch.userData, not fetch.__attributes.userData.
```





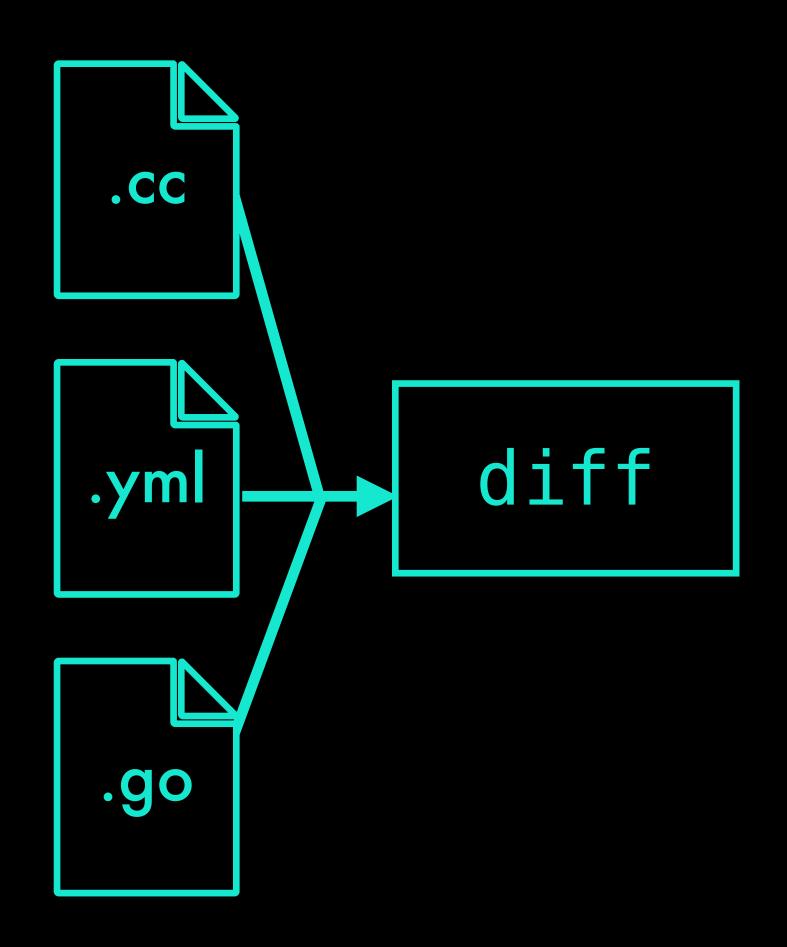
Text-based differences are pragmatic,





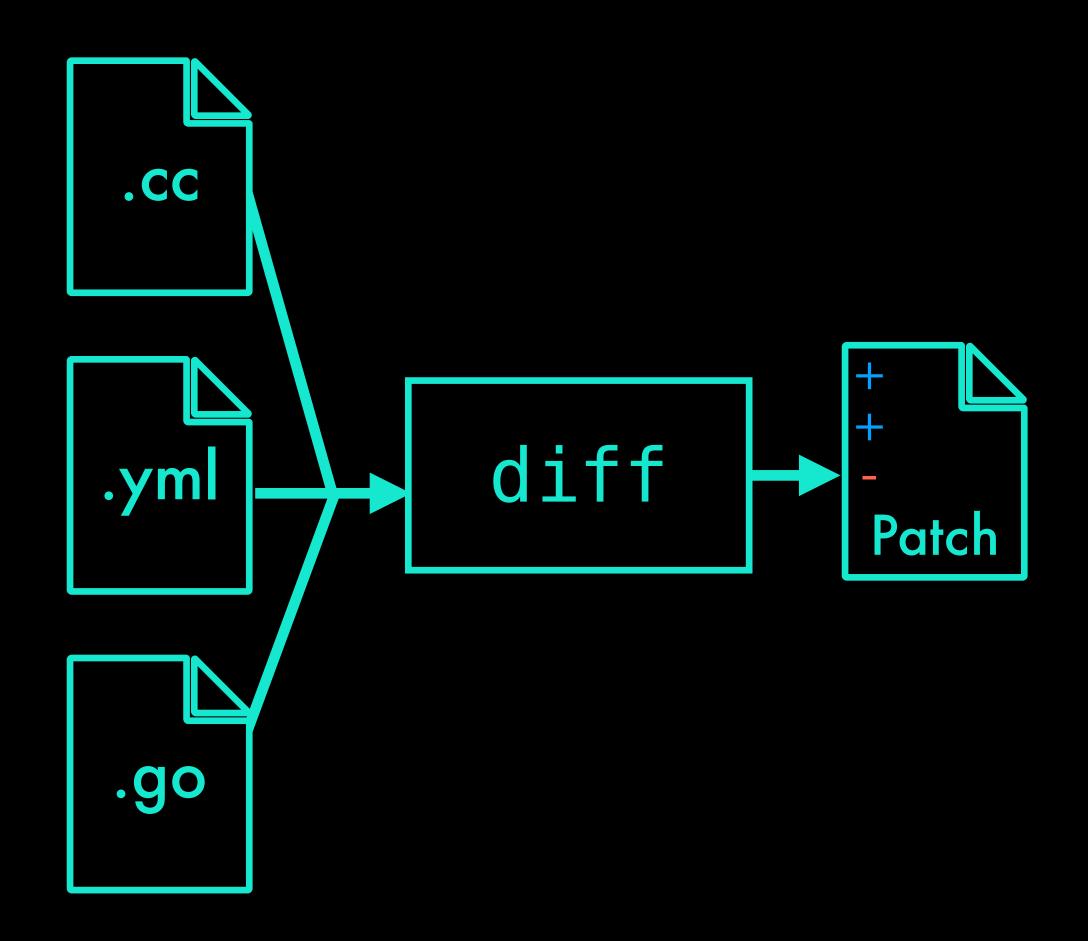


Text-based differences are pragmatic,



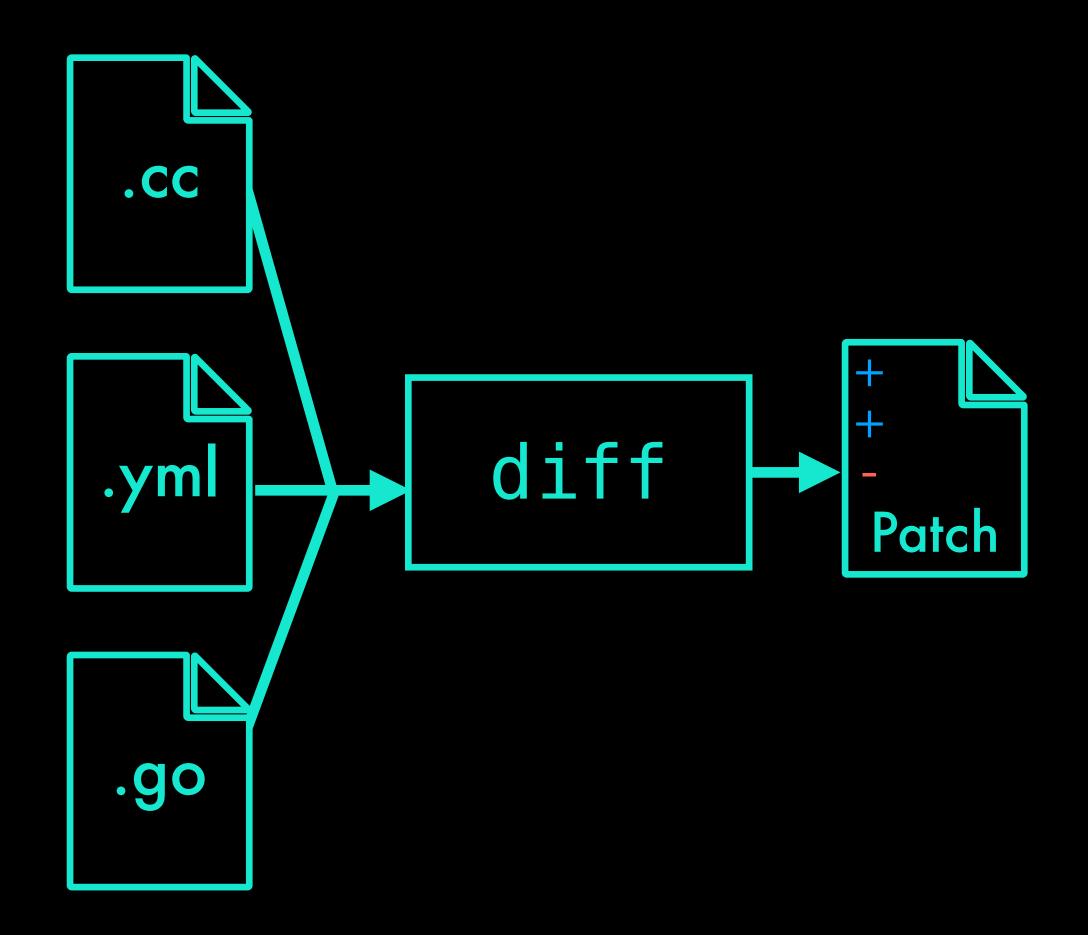


Text-based differences are pragmatic,



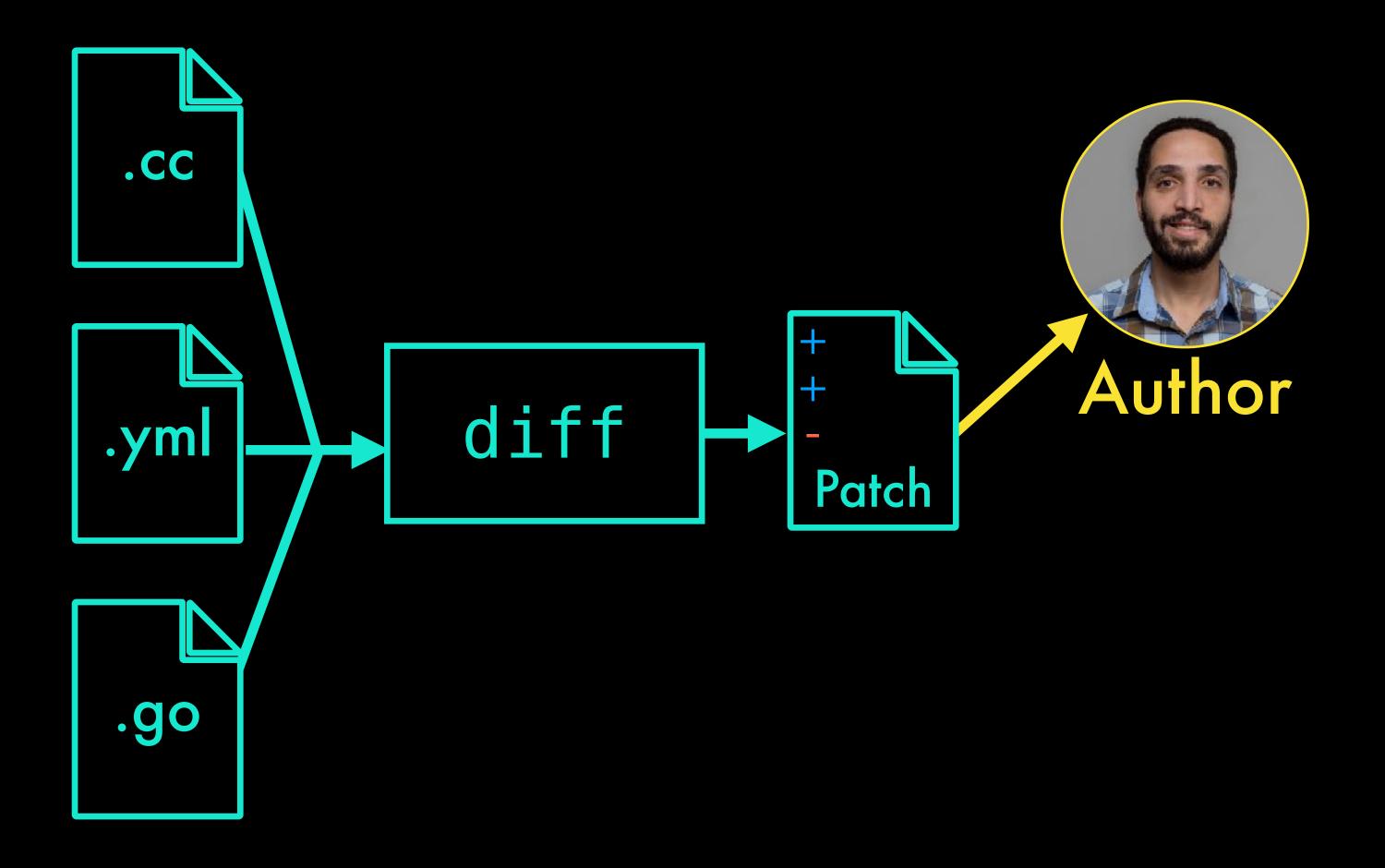




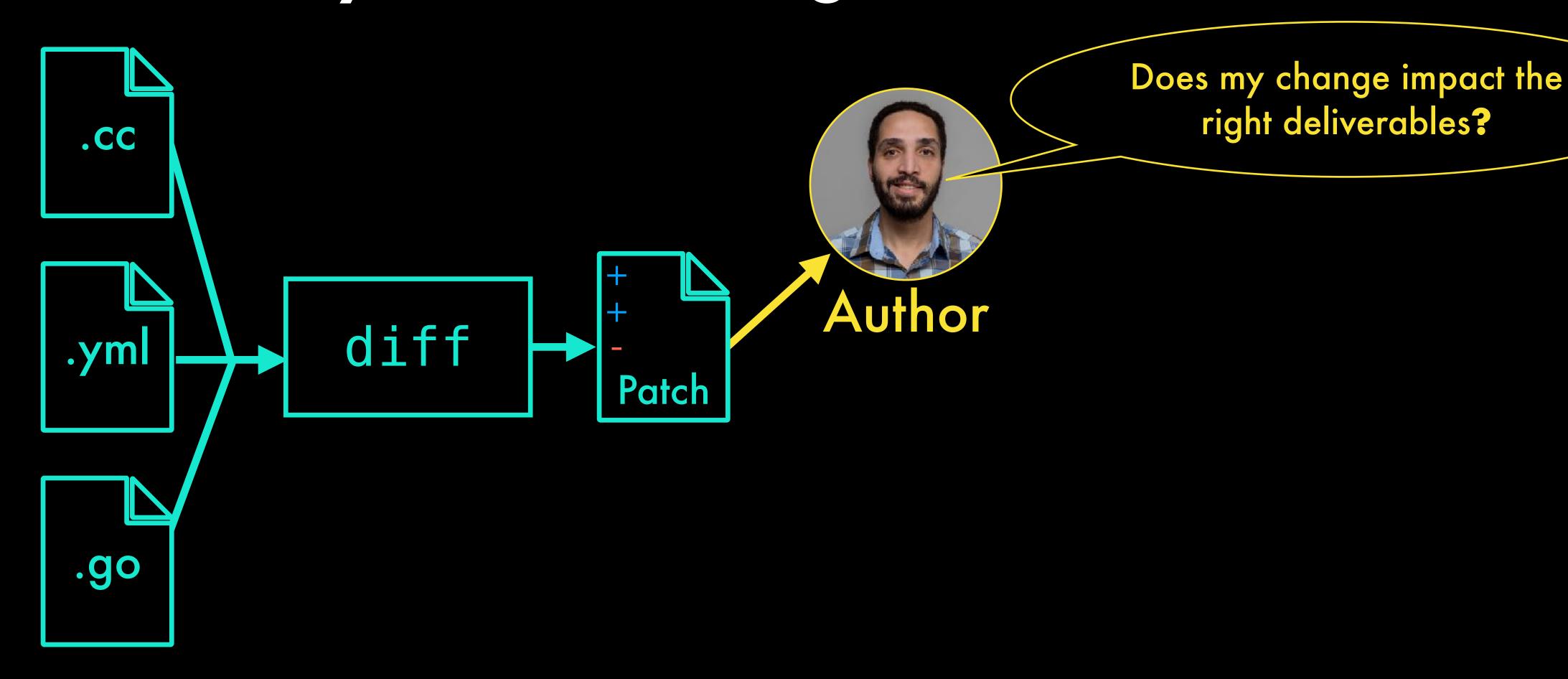






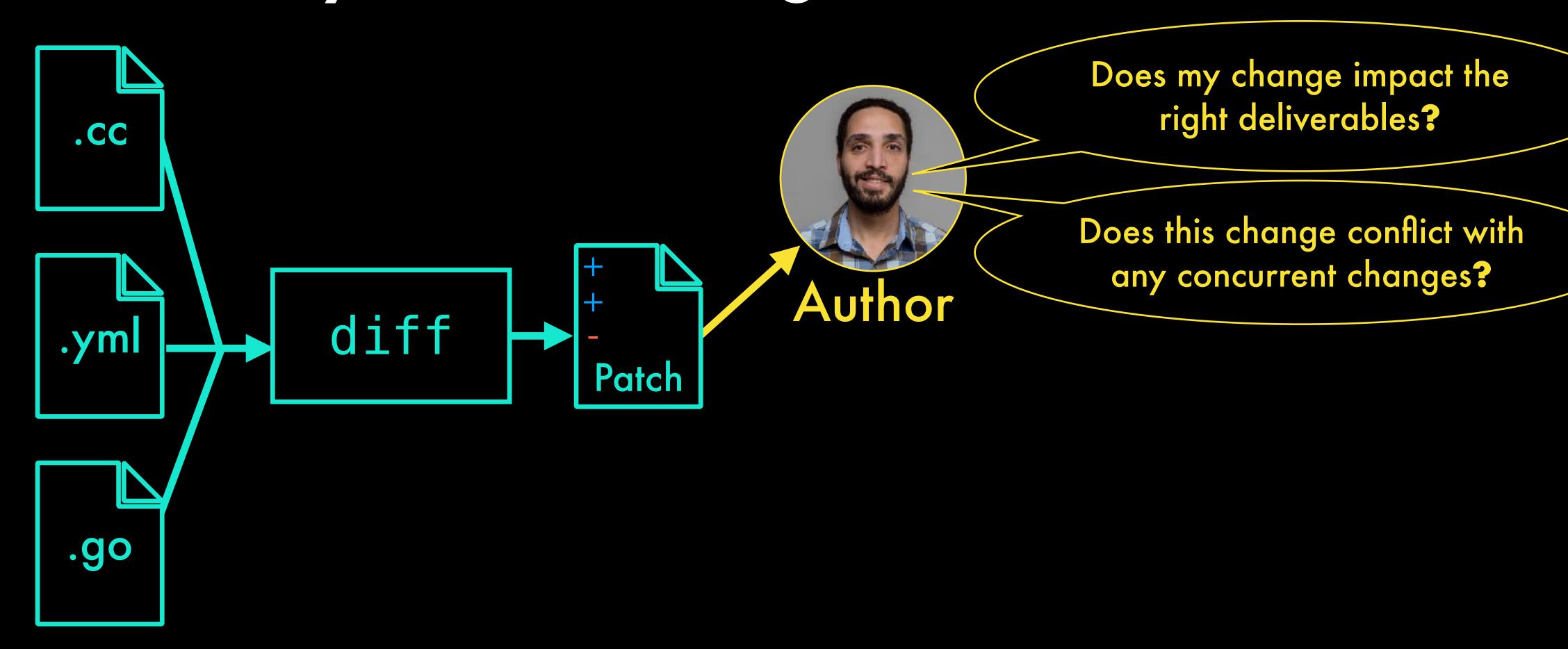






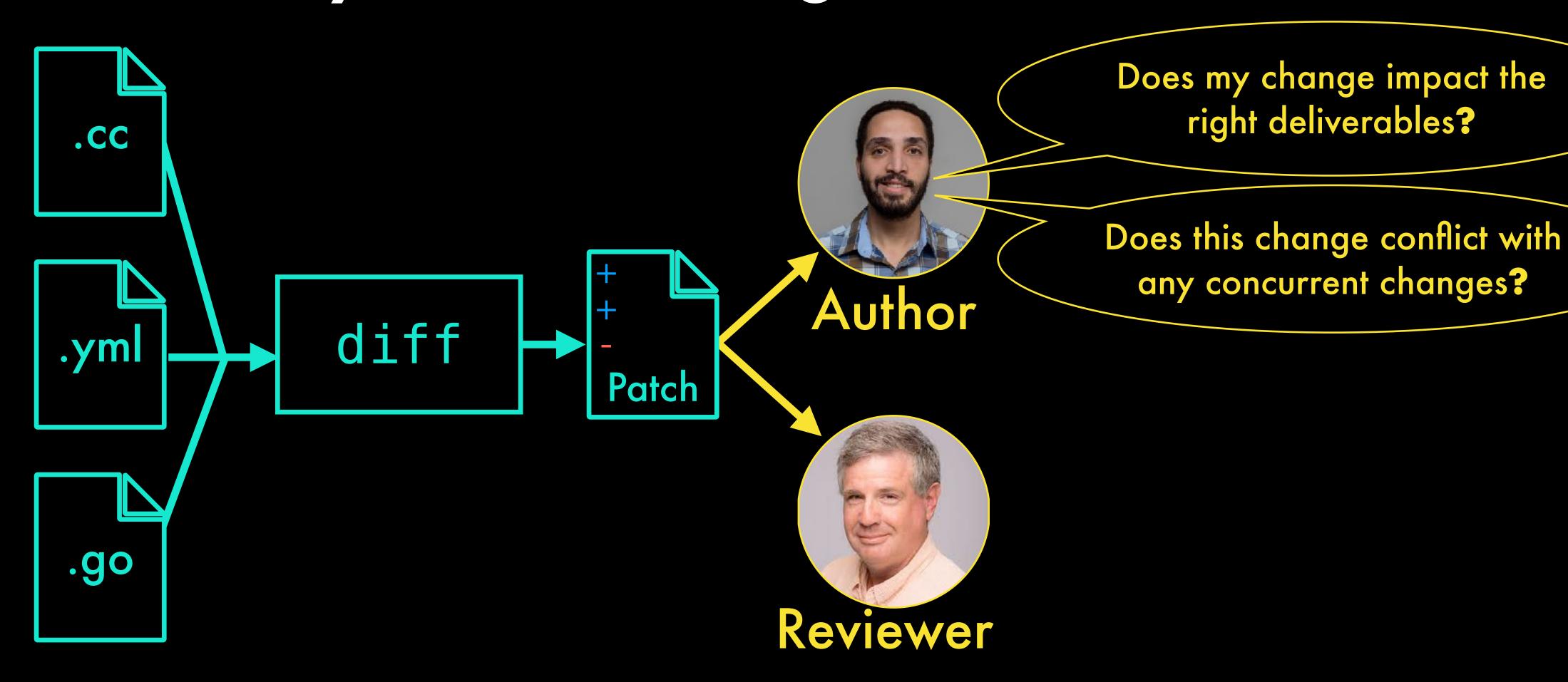




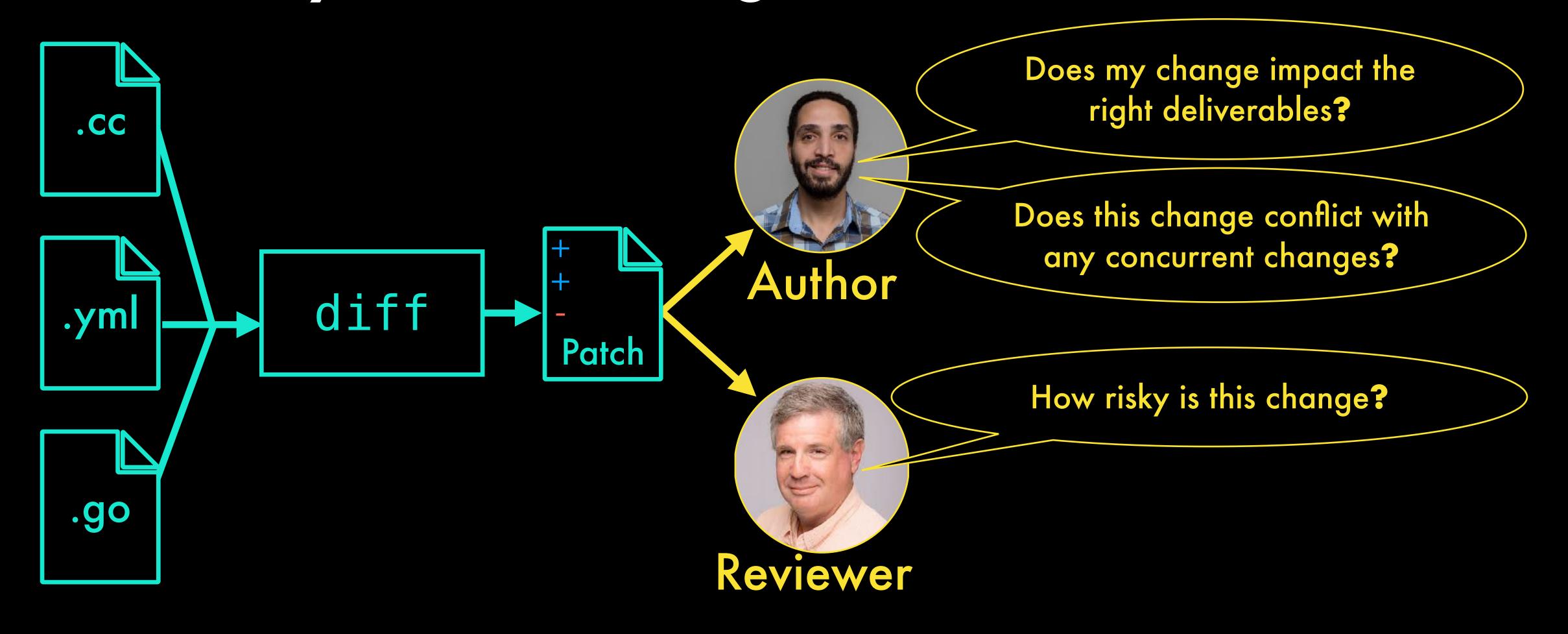




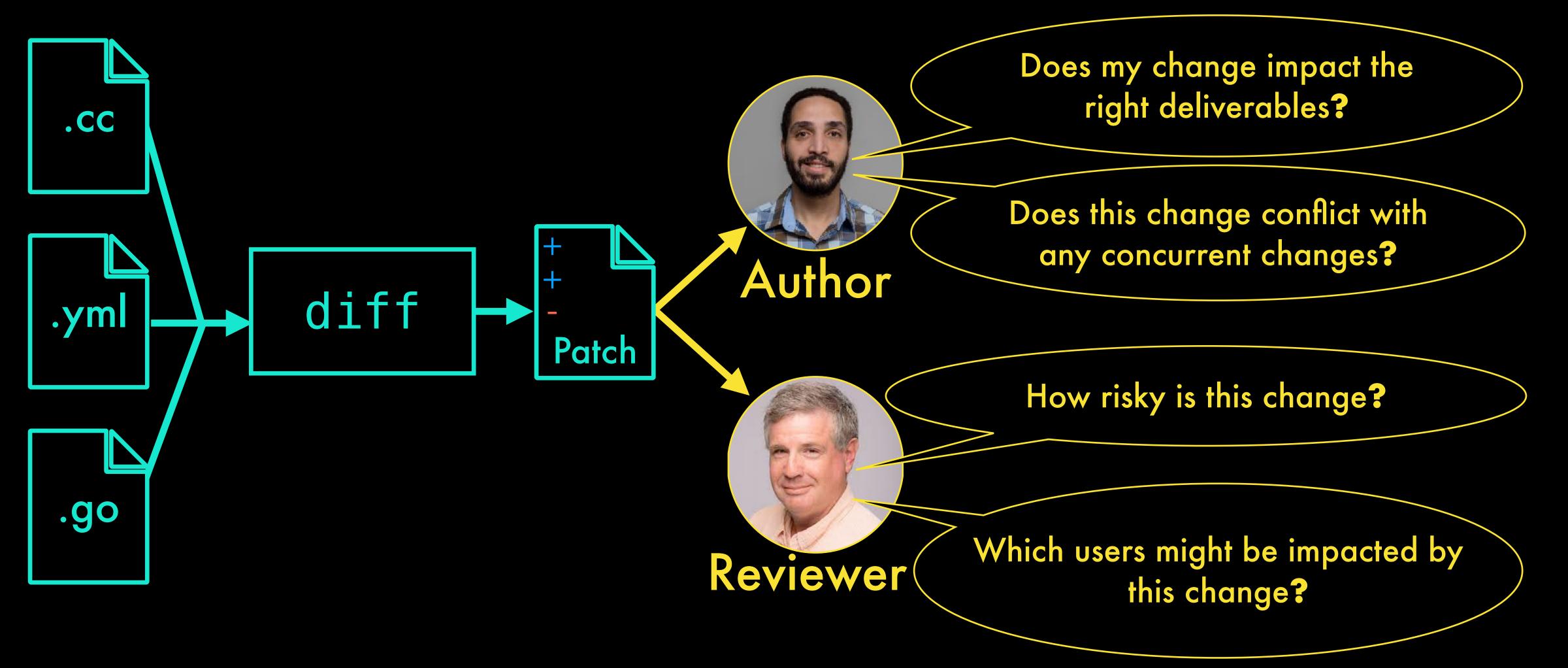






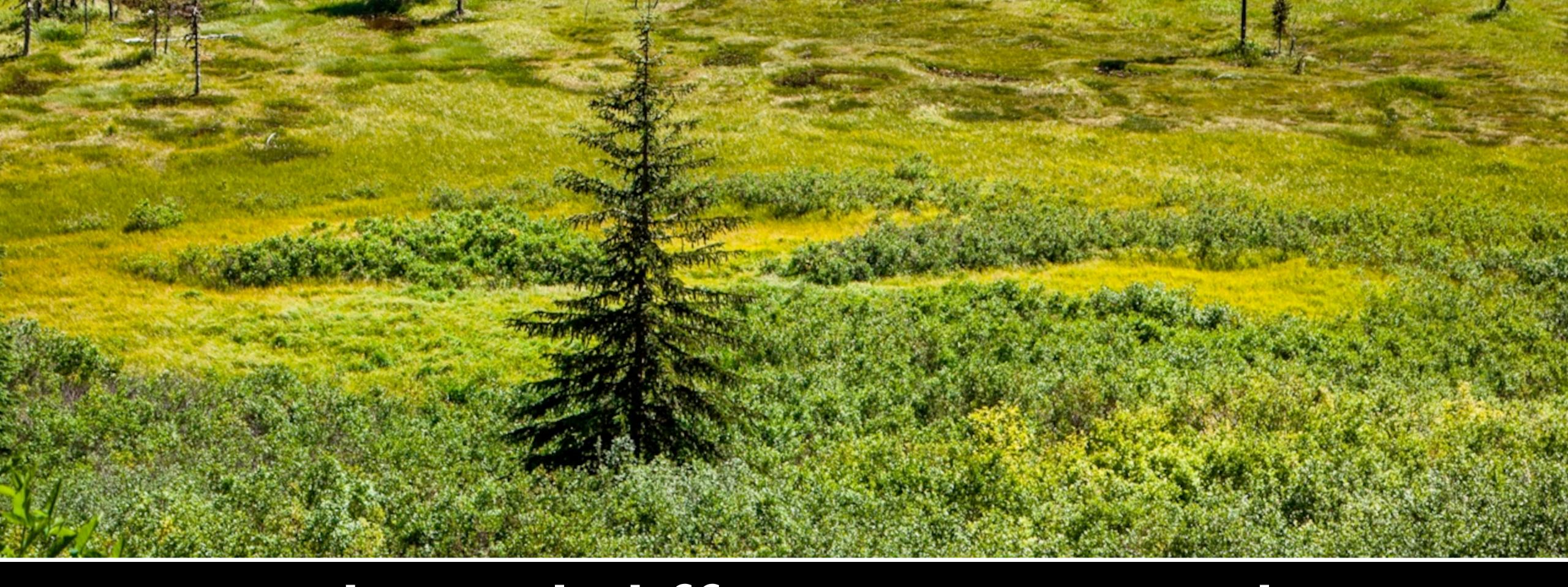












Text-based differences provide a fine-grained perspective





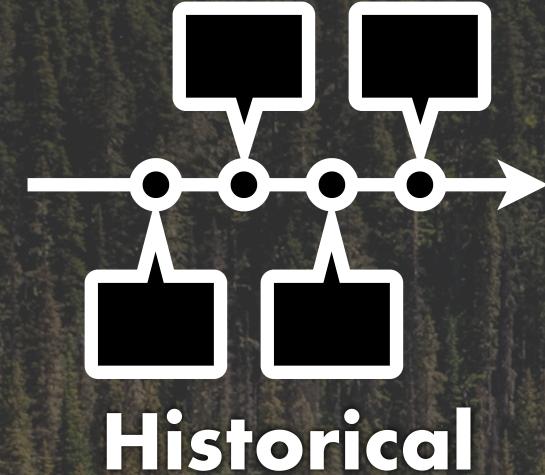




Can't see the forest for the trees Historical



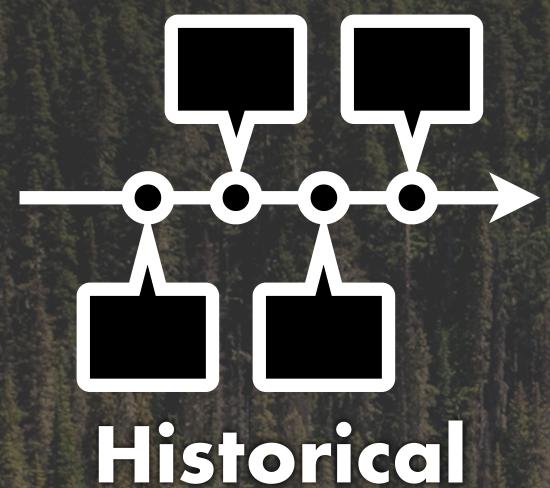




Deliverable









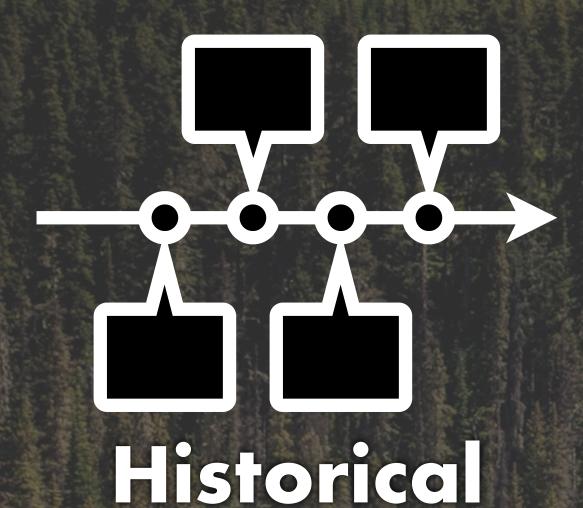


Operational



@SoftwareREBELs







Operational



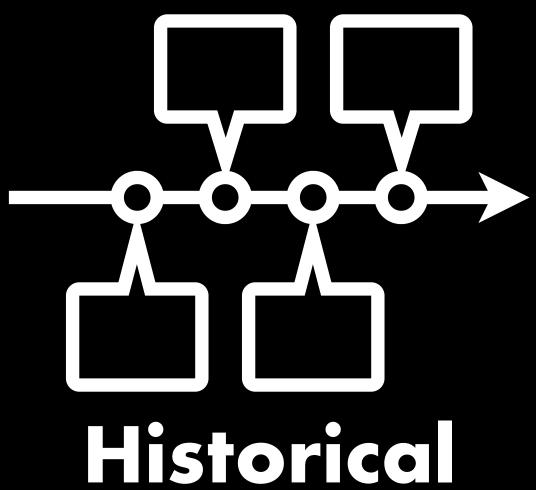


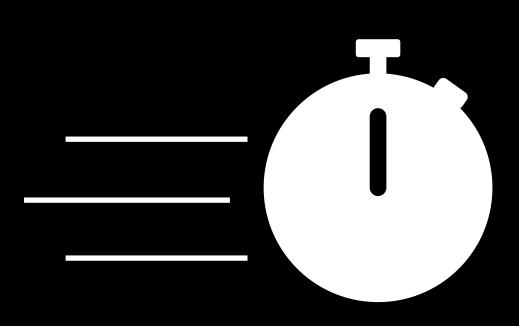


Concurrent



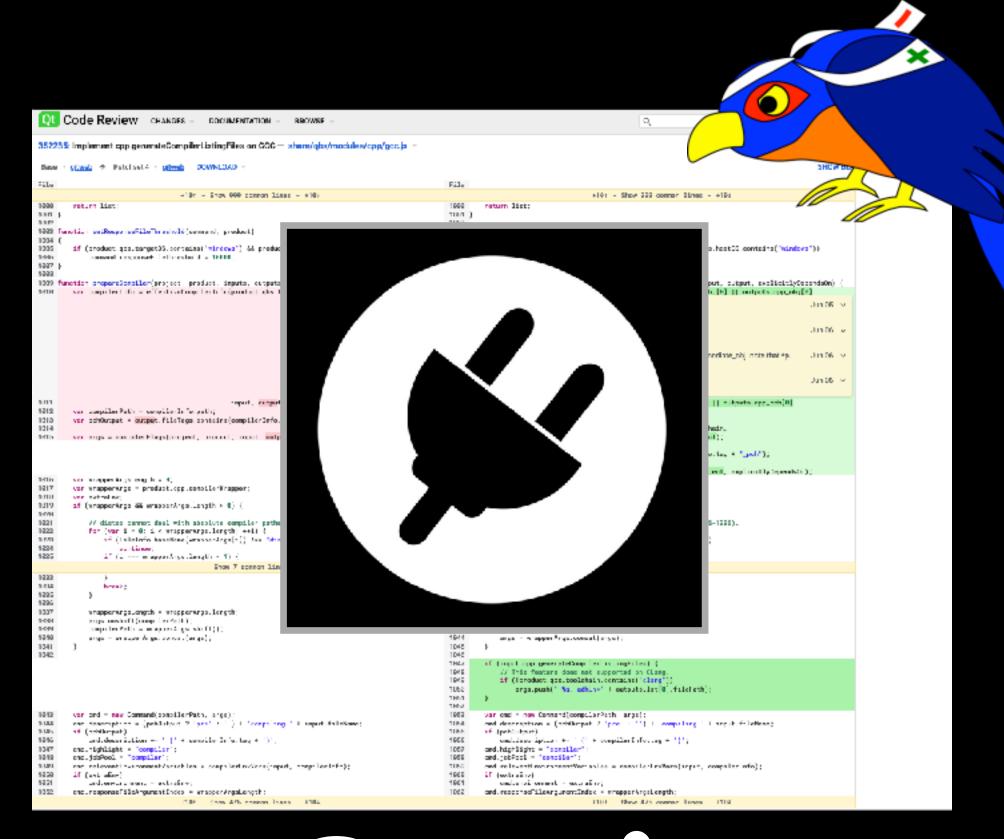
Promoting Situational Awareness in Code Review Platforms





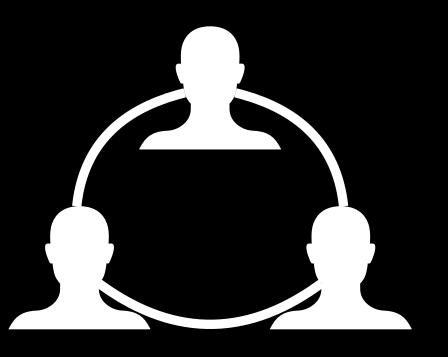






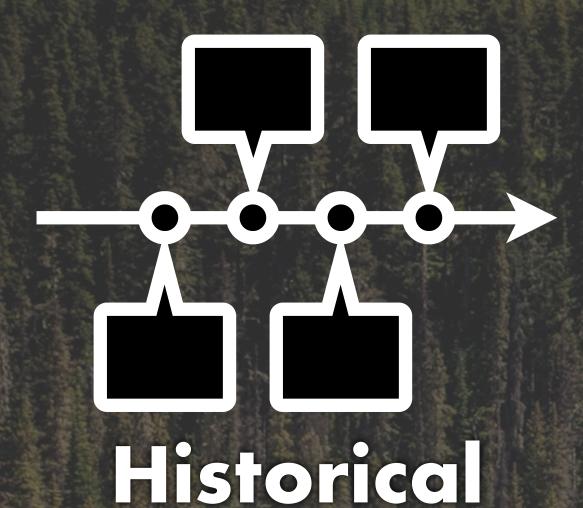
Gerrit
Code Review





Concurrent







Operational

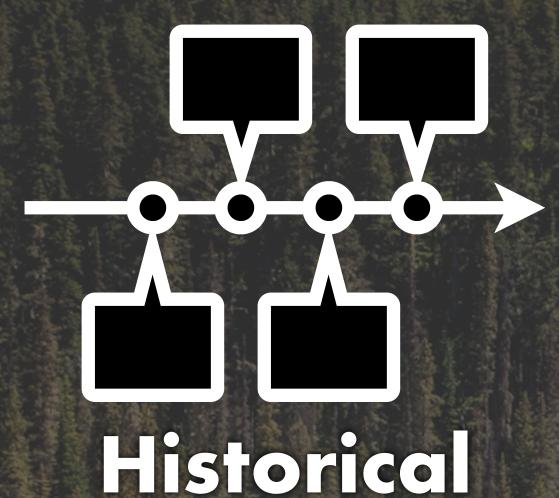






Concurrent











Operational



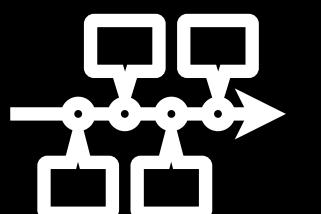


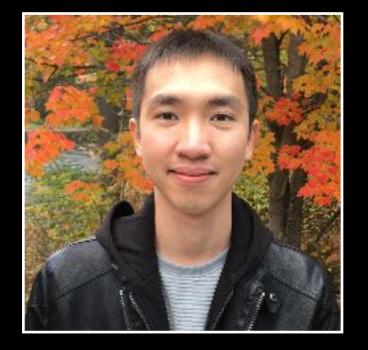






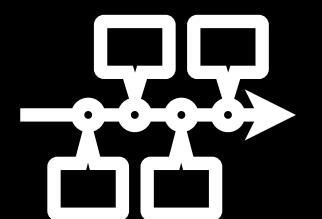








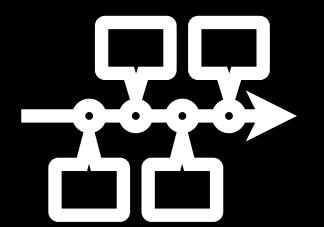








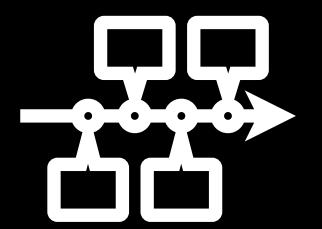
Gerrit





```
if (err != 0)
   goto fail;
   goto fail;
                /* MISTAKE! */
```







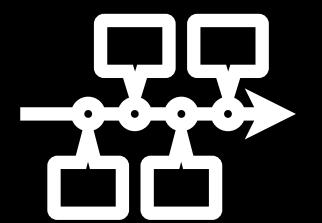
```
sslClientServerHandshake(...)
.....
+ if (err != 0)
+    goto fail;
+    goto fail; /* MISTAKE! */
.....
```

Ship it!











```
+ if (err != 0)
+ goto fail;
+ goto fail; /* MISTAKE! */
```

sslClientServerHandshake(...)

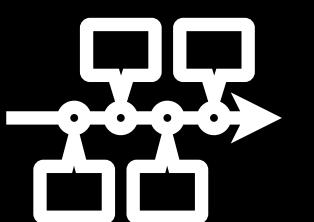


Ship it!





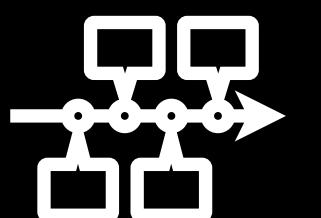




Team Lead





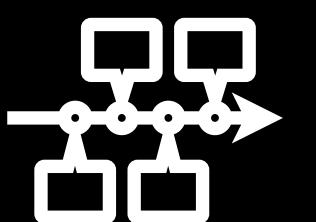


Team Lead



Argh! 10 review requests pending for me this morning!





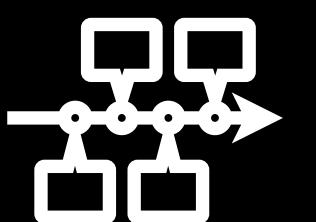
Team Lead



Argh! 10 review requests pending for me this morning!

I should focus my effort on the risky patches





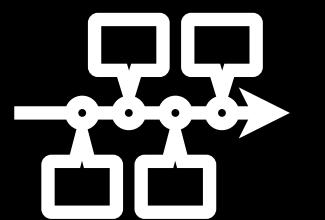
Team Lead



Argh! 10 review requests pending for me this morning!

I should focus my effort on the risky patches





Team Lead

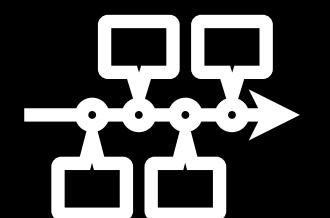


Argh! 10 review requests pending for me this morning!

I should focus my effort on the risky patches

SINCE LINES OF CODE (LOC) == RISK . . .









Argh! 10 review requests pending for me this morning!

I should focus my effort on the risky patches

@SoftwareREBELs

Priority Queue Largest Patch

17

REVIEW REQUEST #1

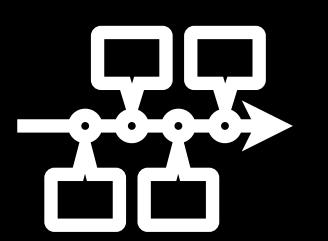
REVIEW REQUEST #2

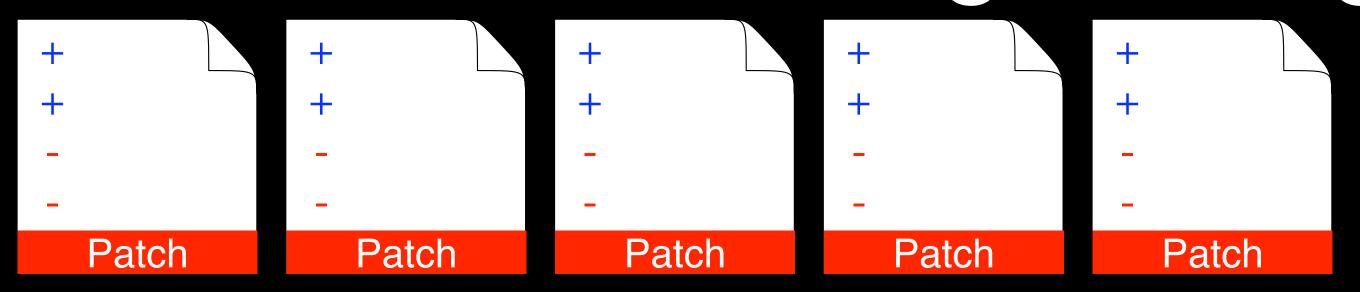
REVIEW REQUEST #10

Smallest Patch



JIT models are trained to predict fix-inducing changes





	True	False	True	False	False
Churn	12	3	21	5	7
Past bugs		5	2	4	5

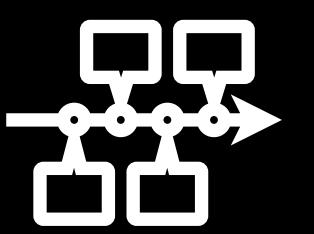
Predicting Risk of Software Changes Mockus and Weiss

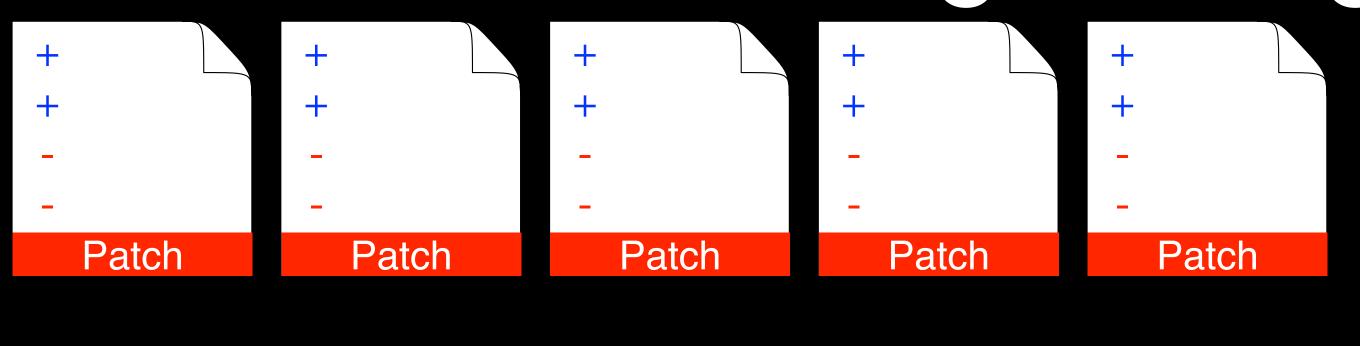
[Bell Labs Tech. Journal 2000]





JIT models are trained to predict fix-inducing changes







Churn

Past bugs

Predicting Risk of Software Changes

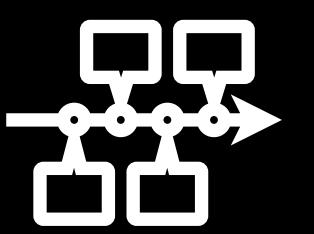
Mockus and Weiss

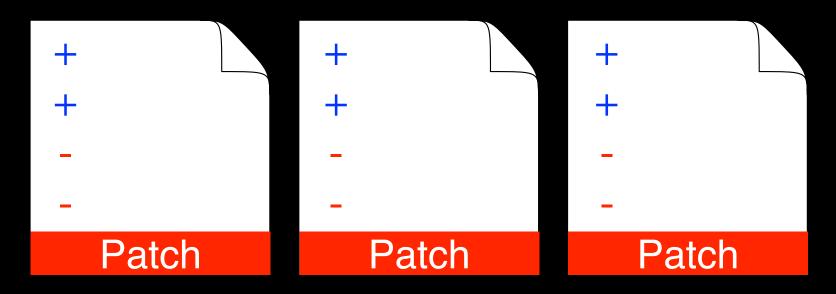
[Bell Labs Tech. Journal 2000]





Fix-inducing changes can be predicted in an online fashion









Churn 2 11 4

Past bugs 9 9

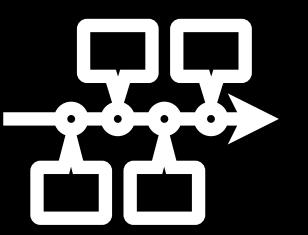
Predicting Risk of Software Changes

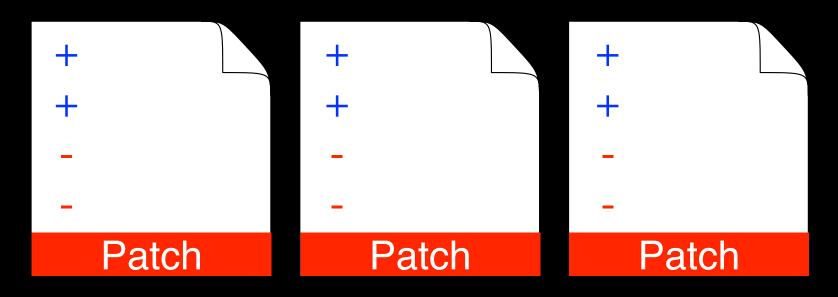
Mockus and Weiss
[Bell Labs Tech. Journal 2000]





Fix-inducing changes can be predicted in an online fashion









Churn 2 11 4

Past bugs 9 9

Predicting Risk of Software Changes

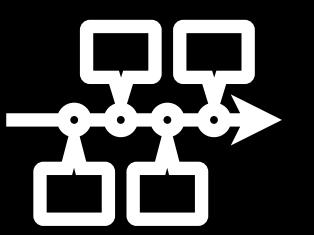
Mockus and Weiss

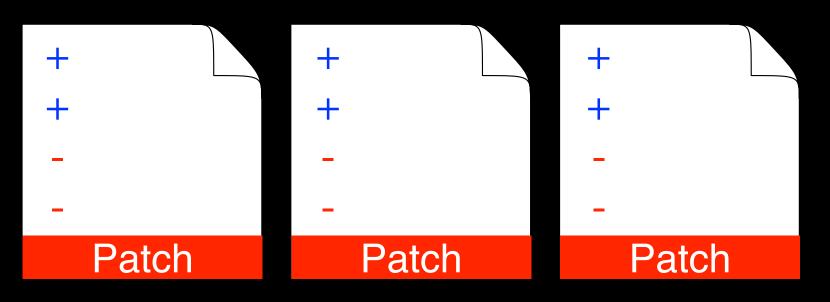
[Bell Labs Tech. Journal 2000]





Fix-inducing changes can be predicted in an online fashion









Churn 2 11 4

Past bugs 9 9

Predicting Risk of Software Changes

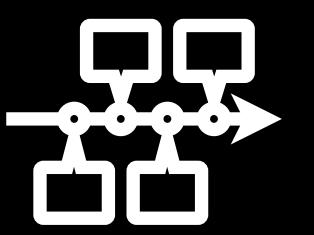
Mockus and Weiss

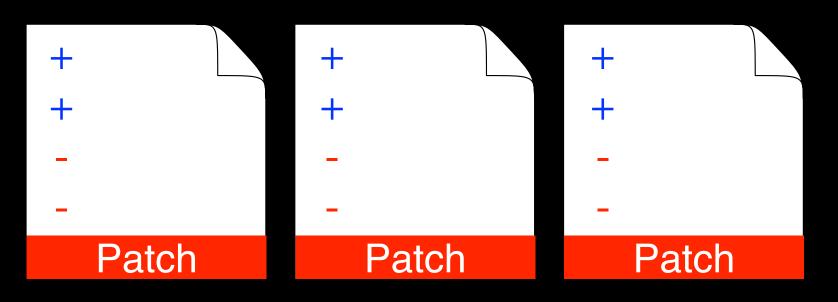
[Bell Labs Tech. Journal 2000]





Fix-inducing changes can be predicted in an online fashion









False?

True?

False?

Churn

Past bugs 9

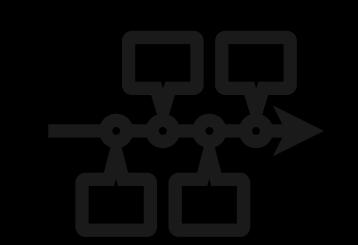
Predicting Risk of Software Changes

Mockus and Weiss [Bell Labs Tech. Journal 2000]





Fix-inducing changes can be n an online fashion





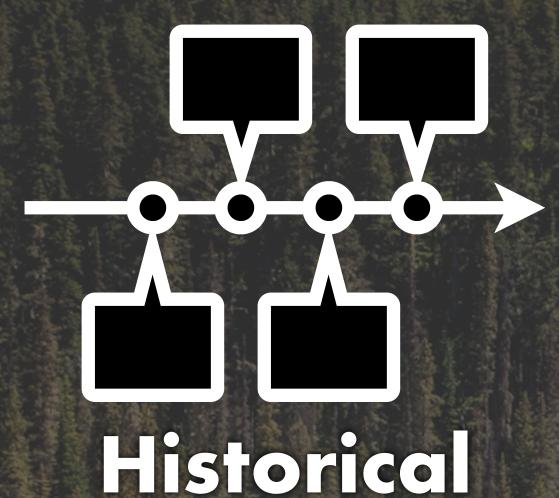


Prioritize this review!

[Bell Labs Tech. Journal 2000]











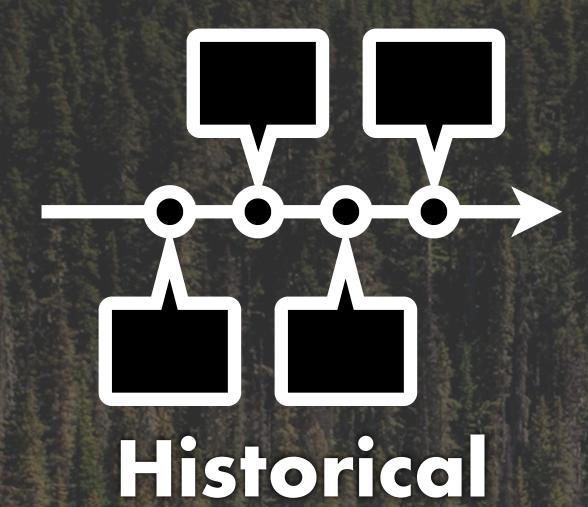


Operational









Exploit historical data for risk assessment





Concurrent

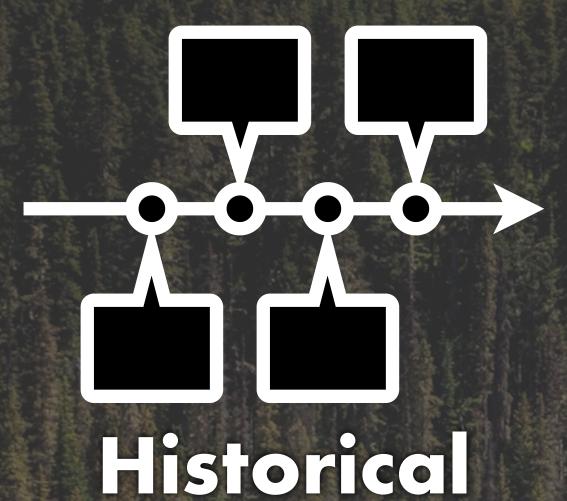


rebels.cs.uwaterloo.ca



Operational





Exploit historical data for risk assessment





Operational



@Software REBELs

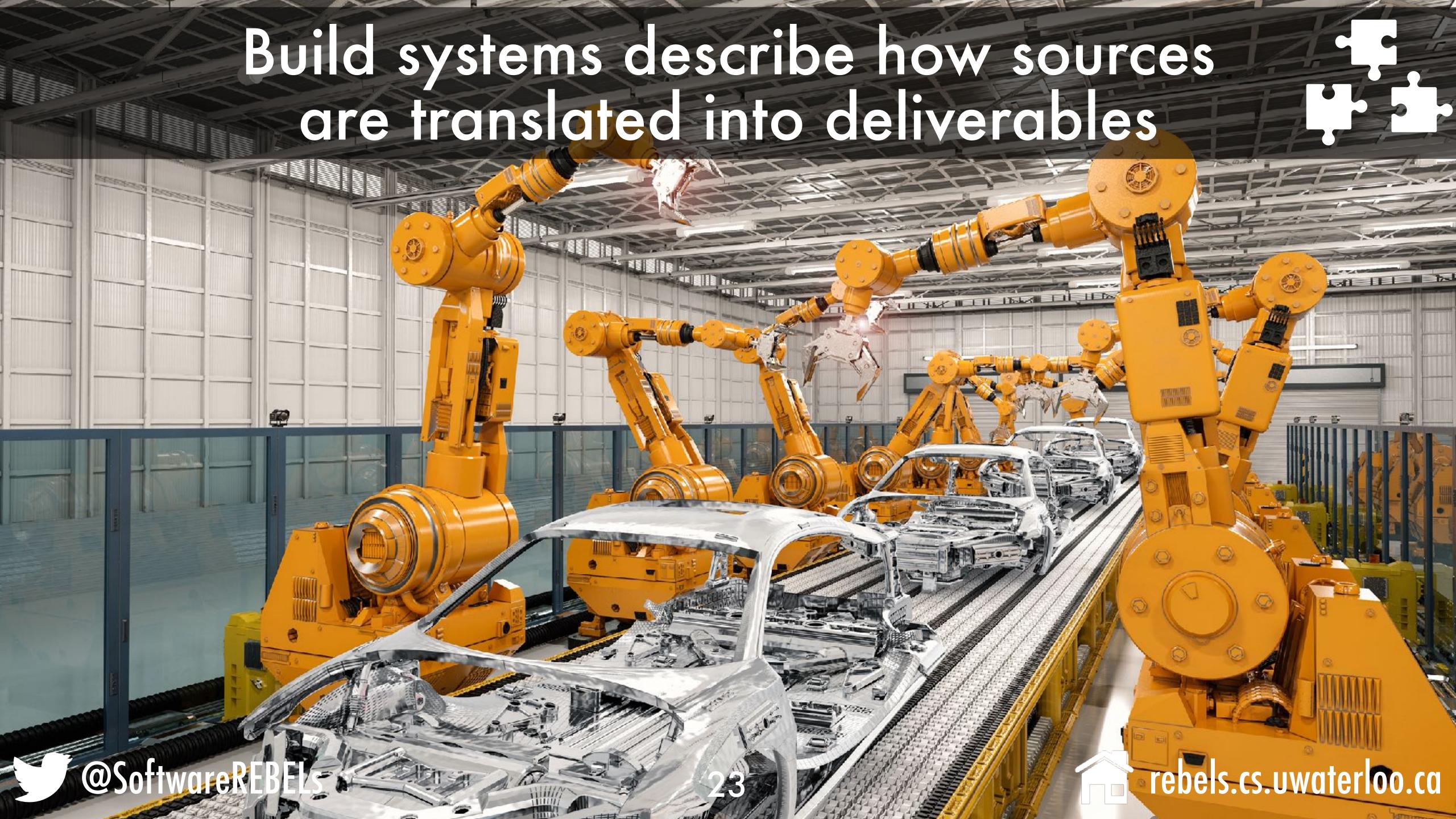


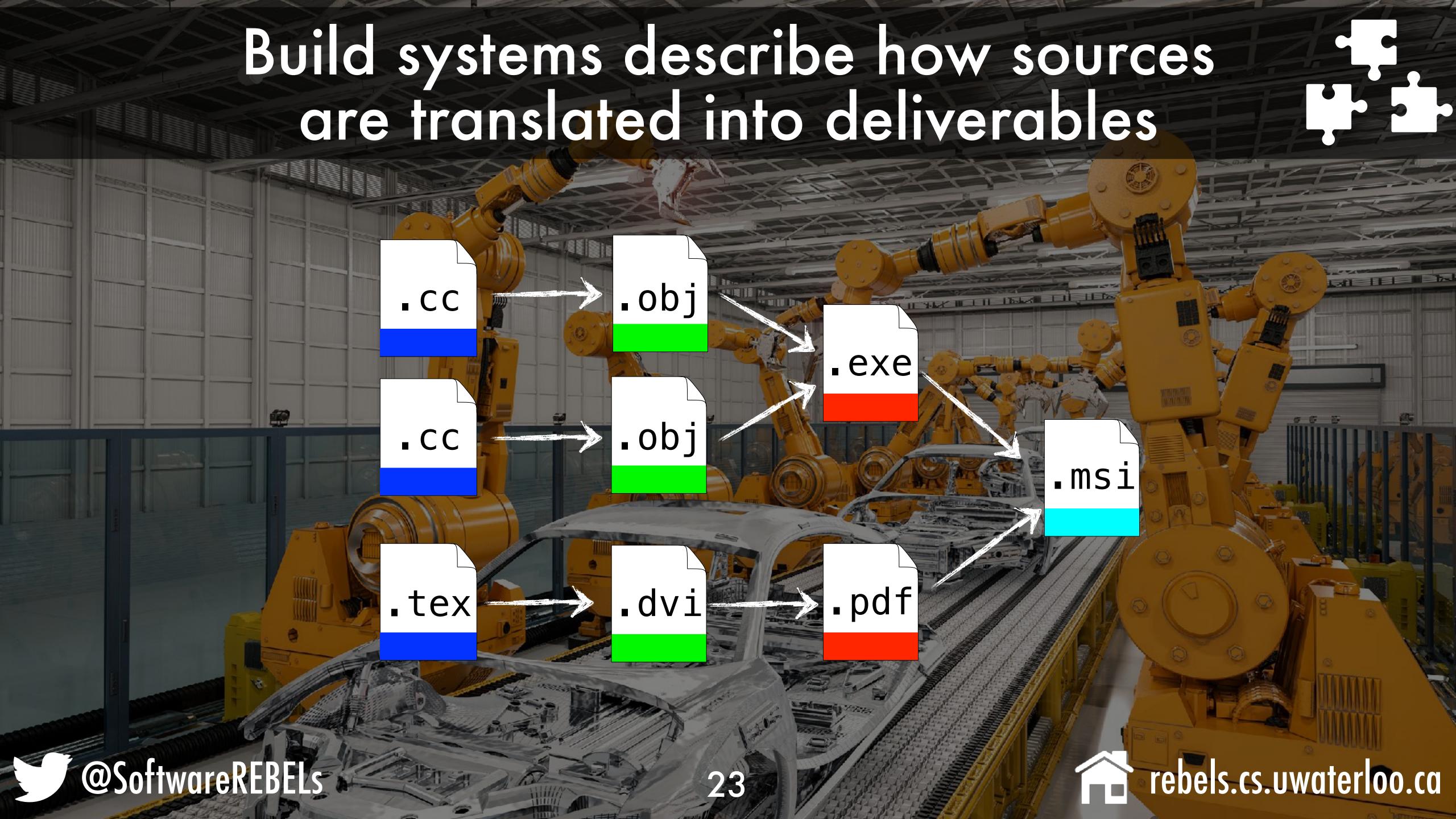
Concurrent



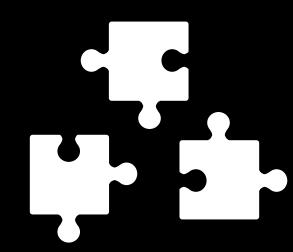


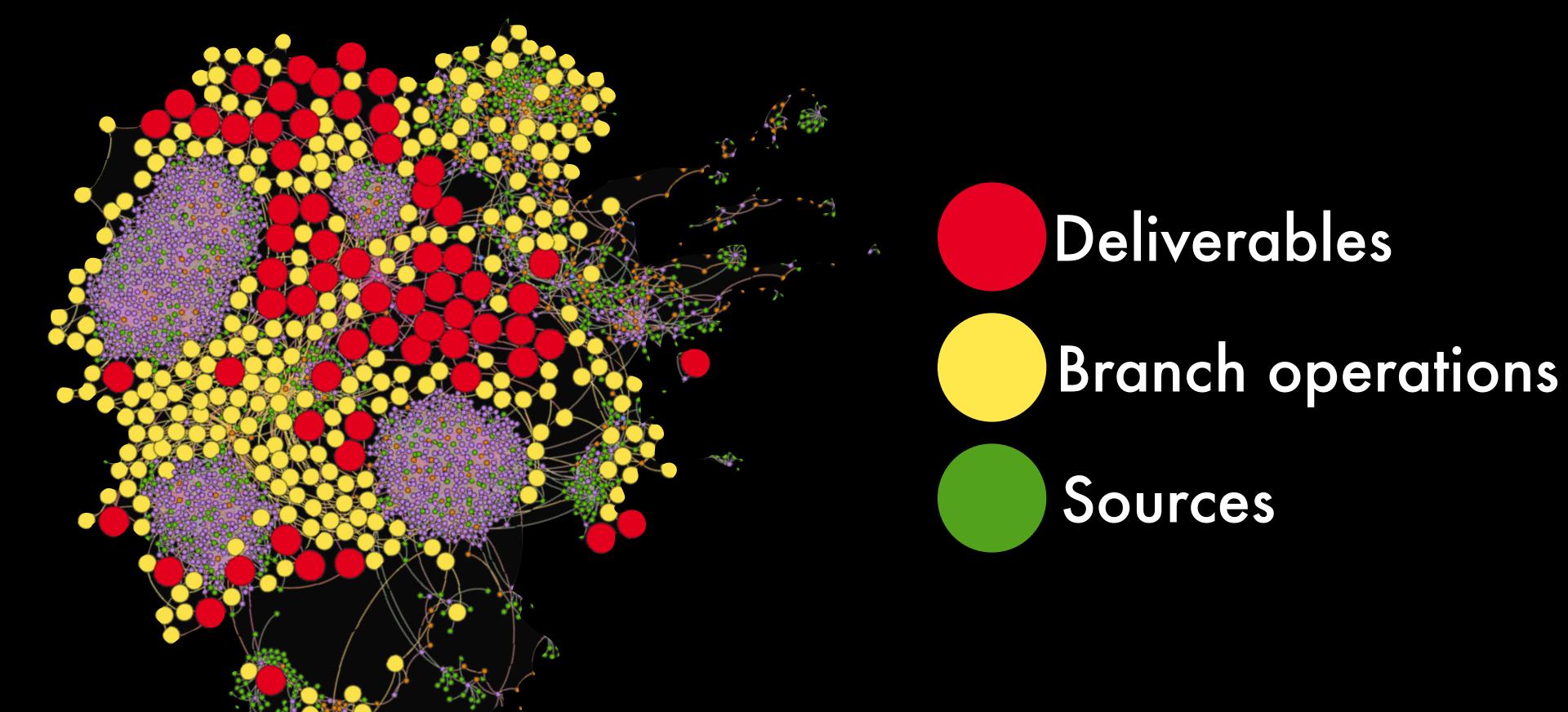






In real software systems, dependency graphs are extremely rich and complex







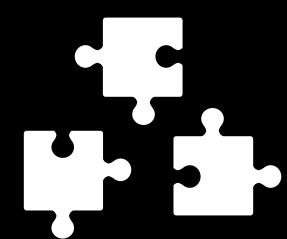


In real software systems, dependency graphs are extremely rich and complex

Current review interfaces miss this important perspective







Step 1: Identify changed files

All files in a sample system:

BLIMP Tracer: Integrating Build Impact Analysis with **Code Review**

Wen et al. [ICSME 2018]

eg1.c

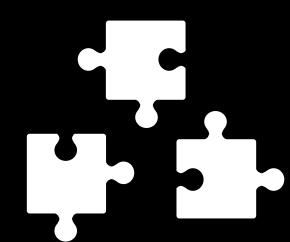
eg2.c

eg3.c

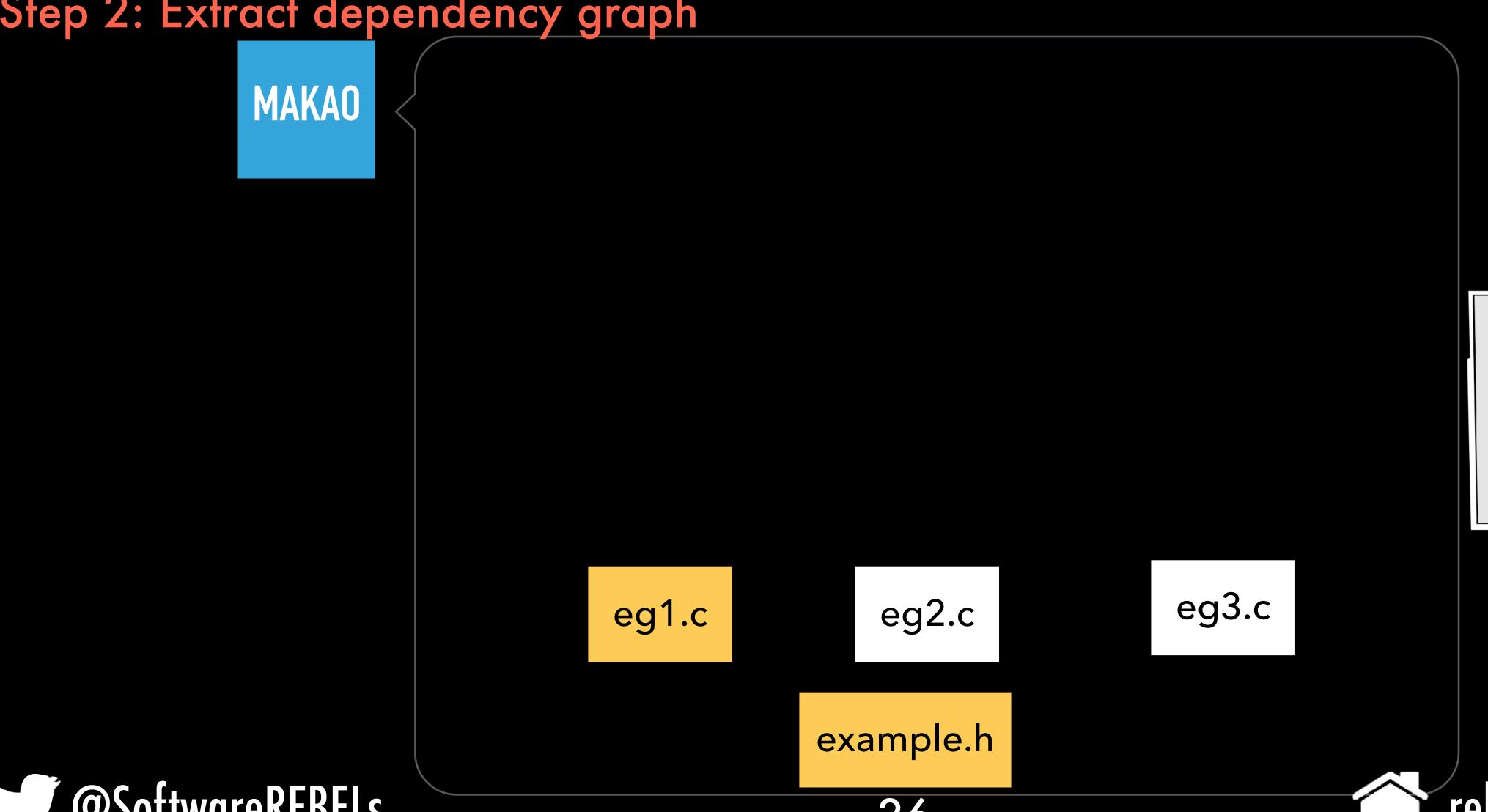
example.h





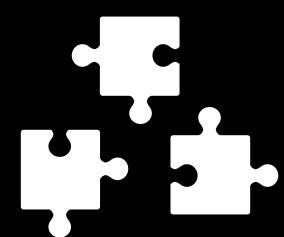


Step 2: Extract dependency graph



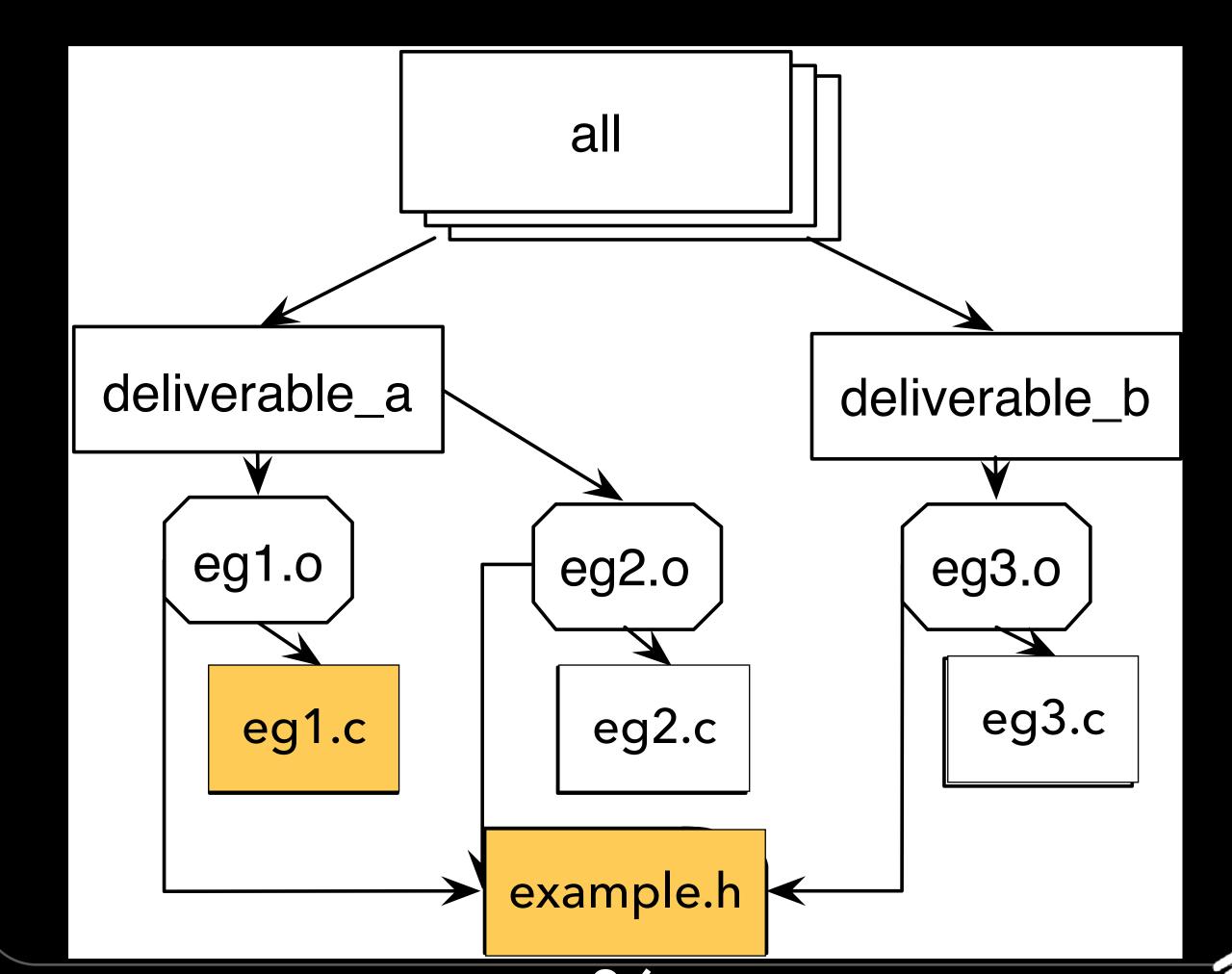
BLIMP Tracer: Integrating Build Impact Analysis with **Code Review** Wen et al.

[ICSME 2018]



Step 2: Extract dependency graph

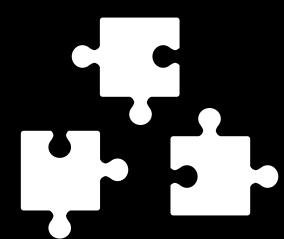
MAKAO



BLIMP Tracer:
Integrating Build
Impact Analysis with
Code Review

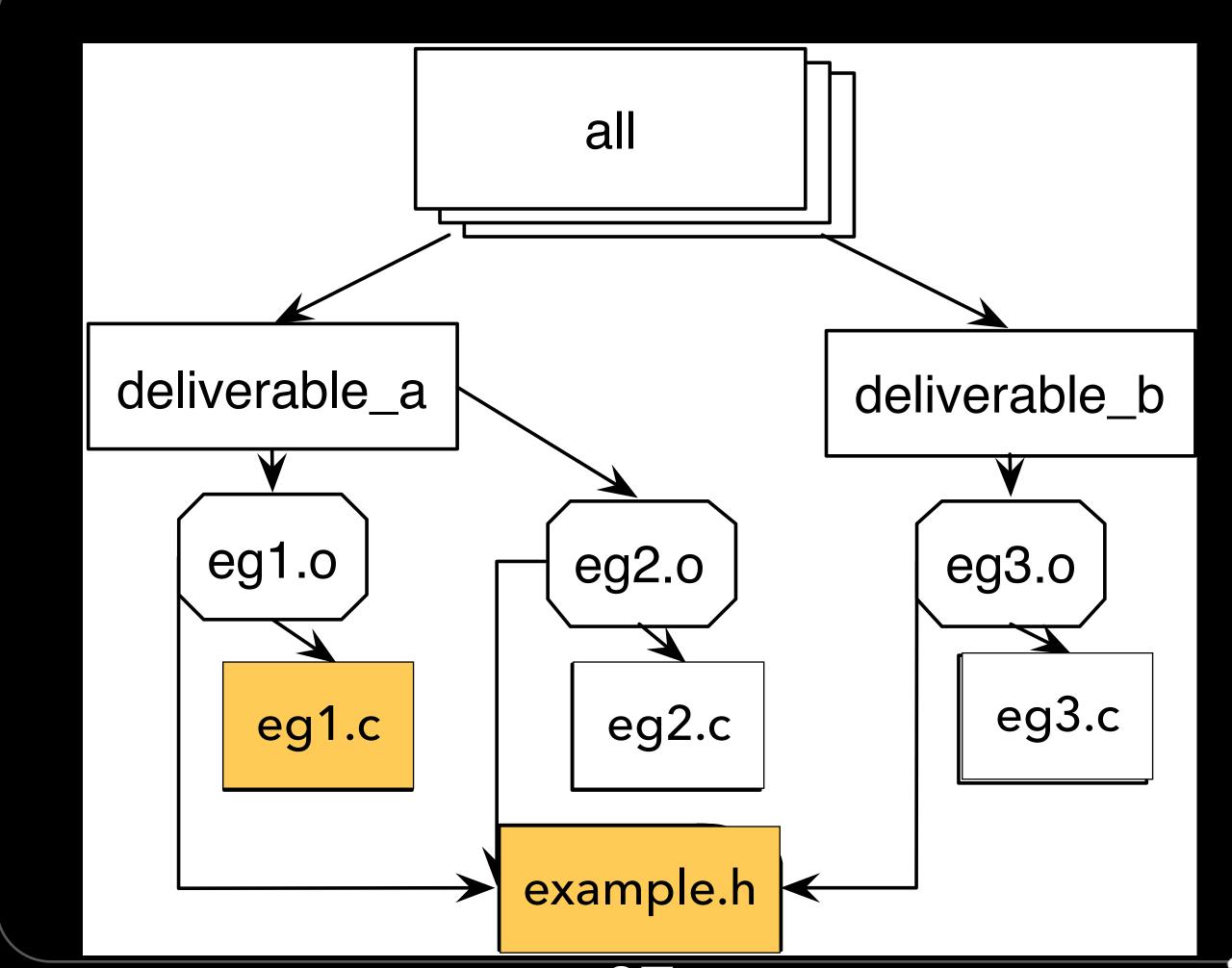
Wen et al. [ICSME 2018]





Step 3: Graph traversal and filtering

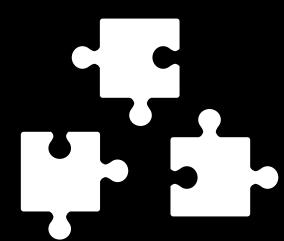
MAKAO



BLIMP Tracer:
Integrating Build
Impact Analysis with
Code Review
Wen et al.

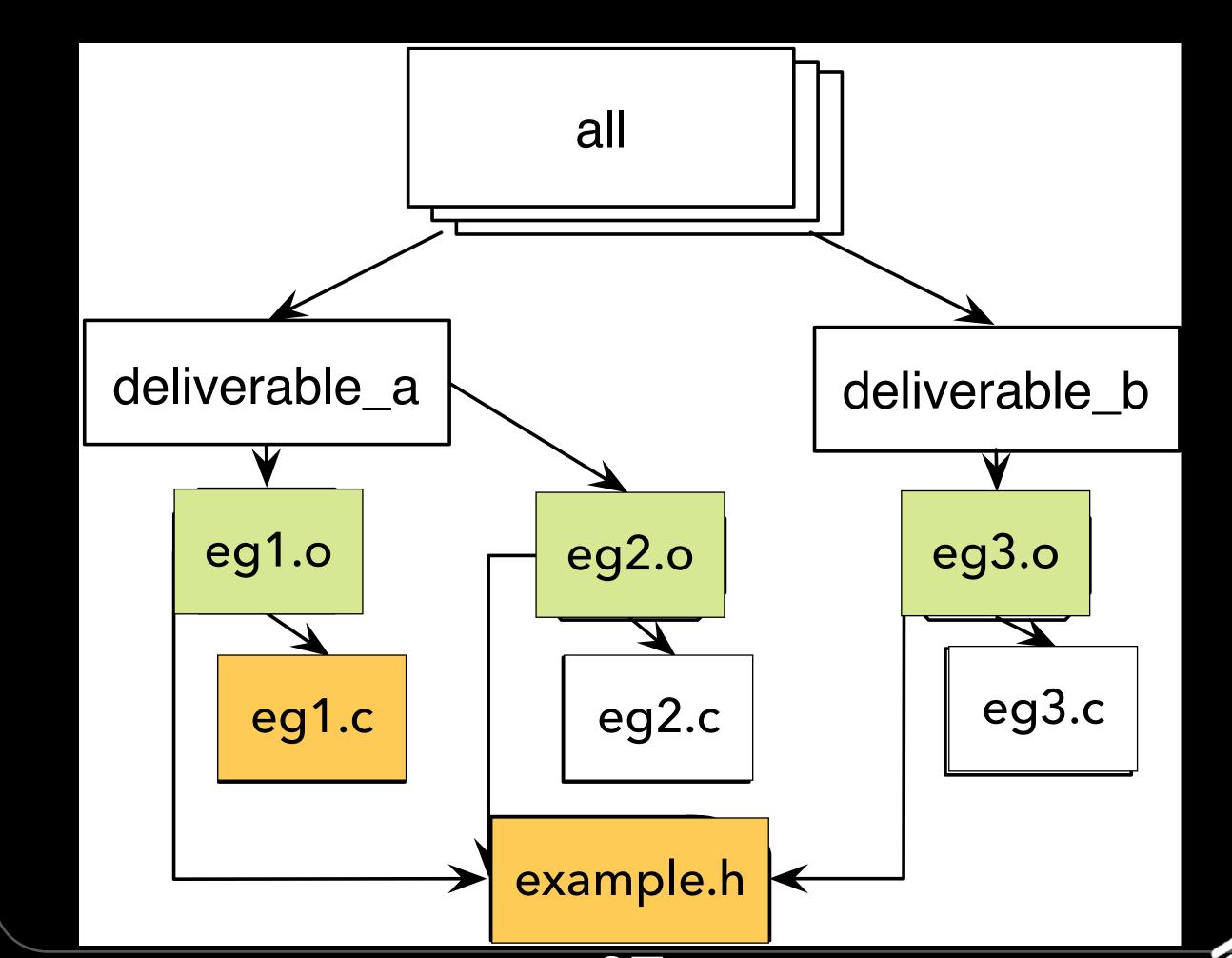
[ICSME 2018]





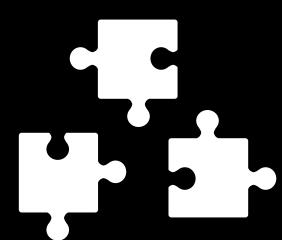
Step 3: Graph traversal and filtering

MAKAO



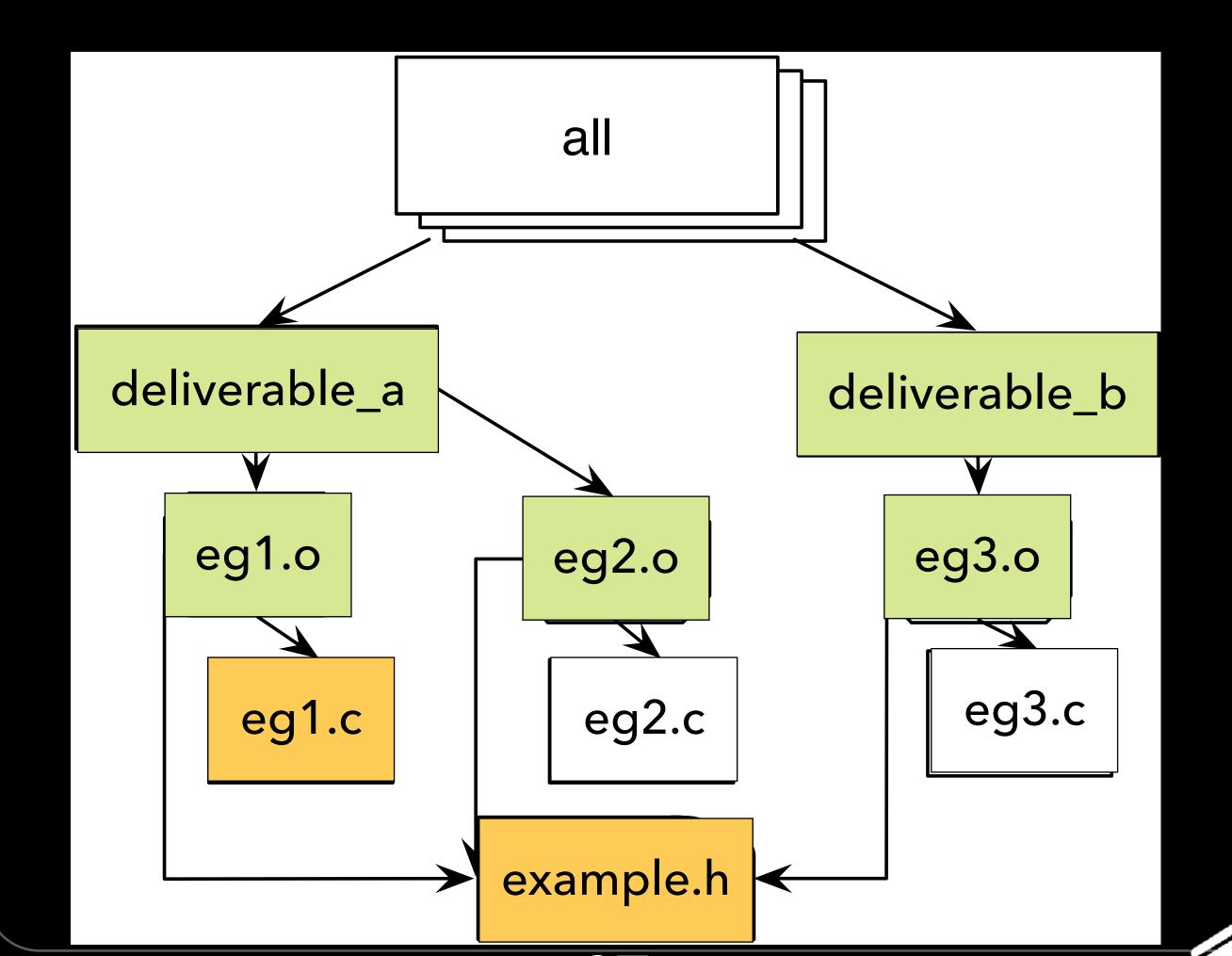
BLIMP Tracer:
Integrating Build
Impact Analysis with
Code Review
Wen et al.
[ICSME 2018]





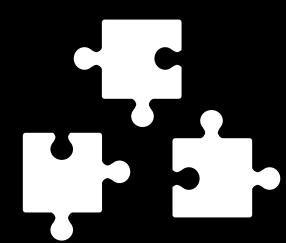
Step 3: Graph traversal and filtering

MAKAO



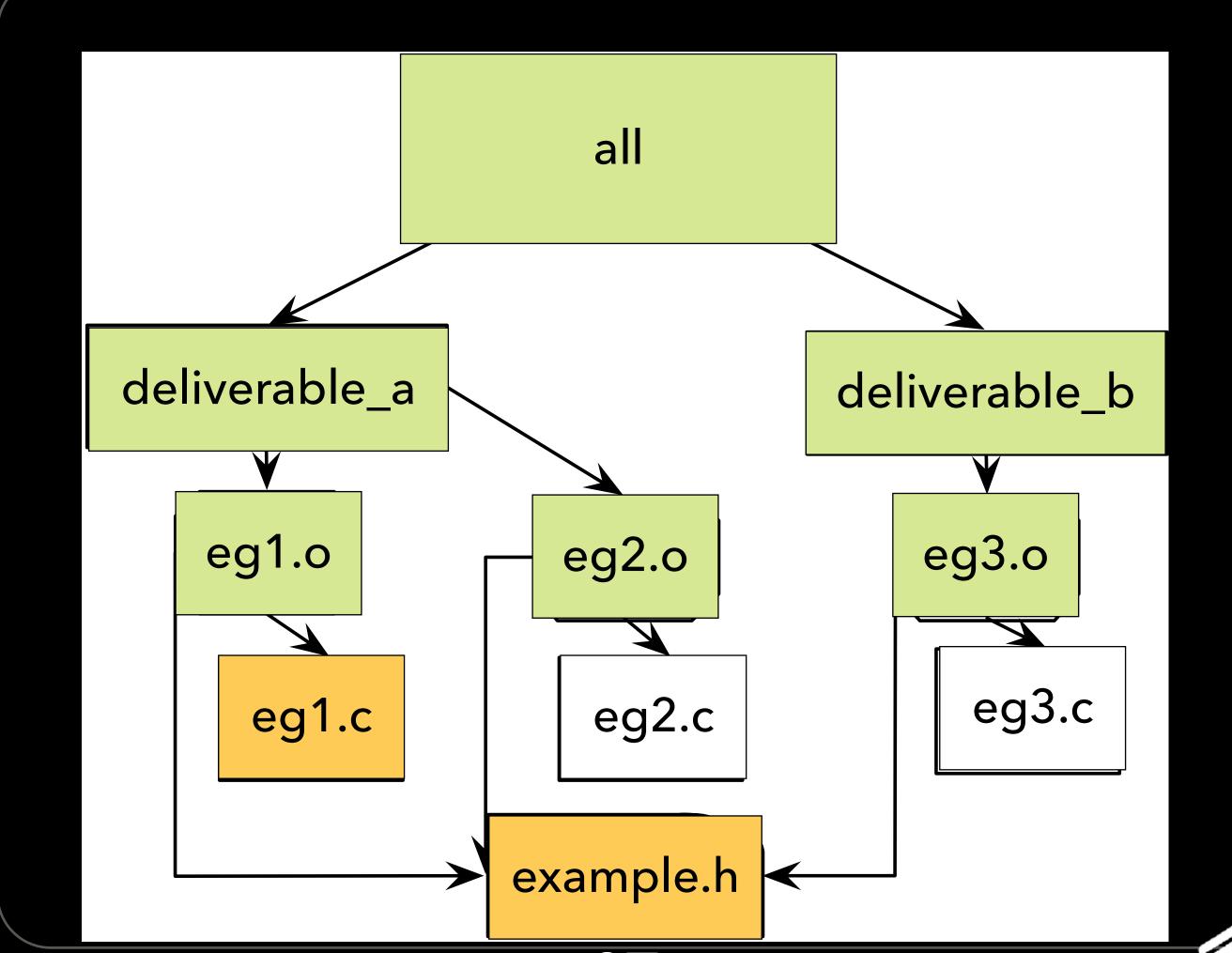
BLIMP Tracer:
Integrating Build
Impact Analysis with
Code Review
Wen et al.
[ICSME 2018]





Step 3: Graph traversal and filtering

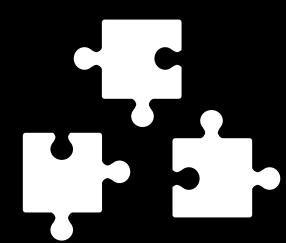
MAKAO



BLIMP Tracer:
Integrating Build
Impact Analysis with
Code Review
Wen et al.

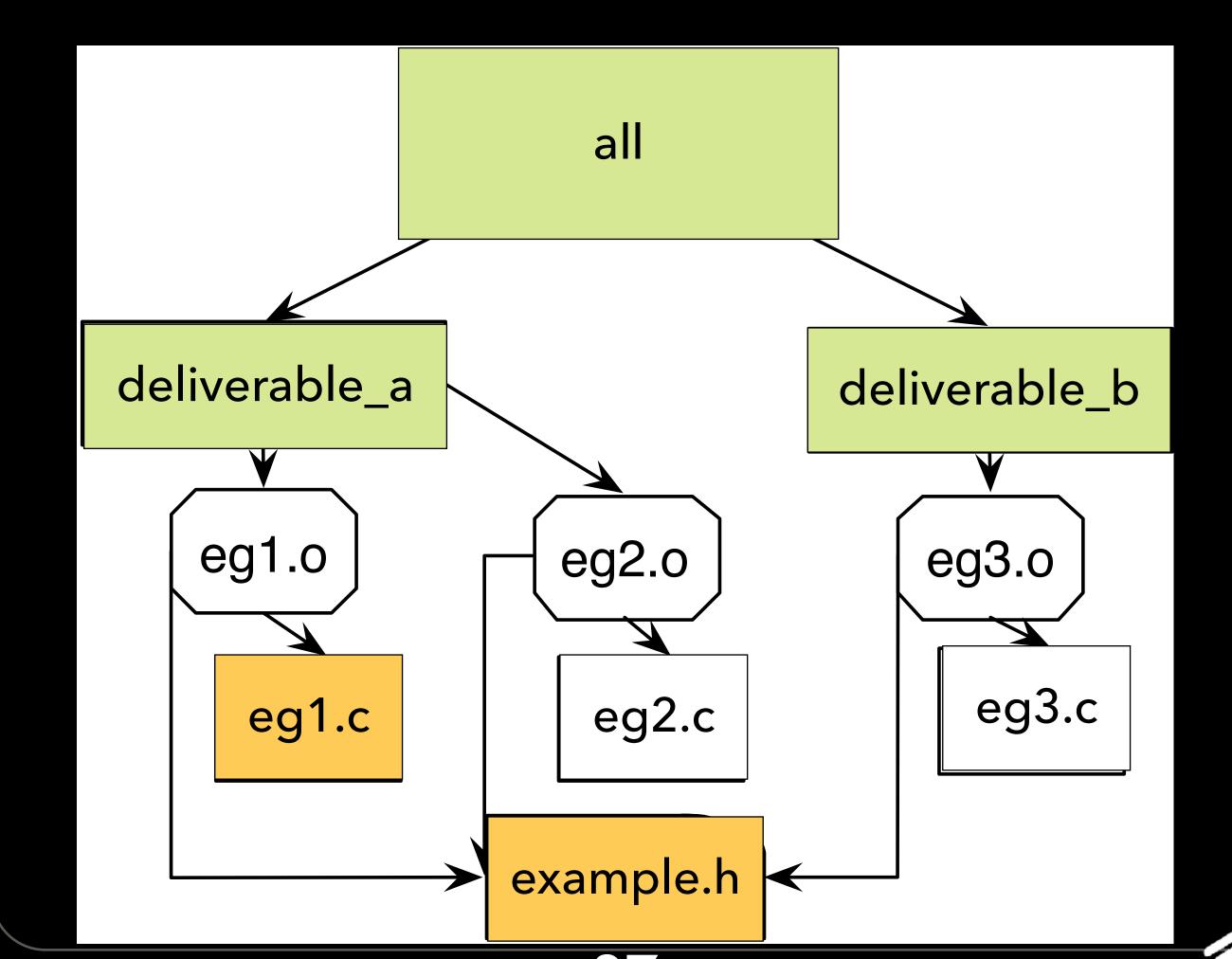
[ICSME 2018]





Step 3: Graph traversal and filtering

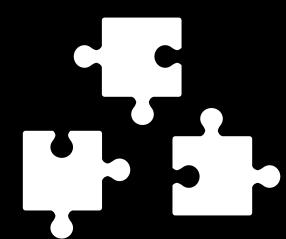
MAKAO



BLIMP Tracer:
Integrating Build
Impact Analysis with
Code Review

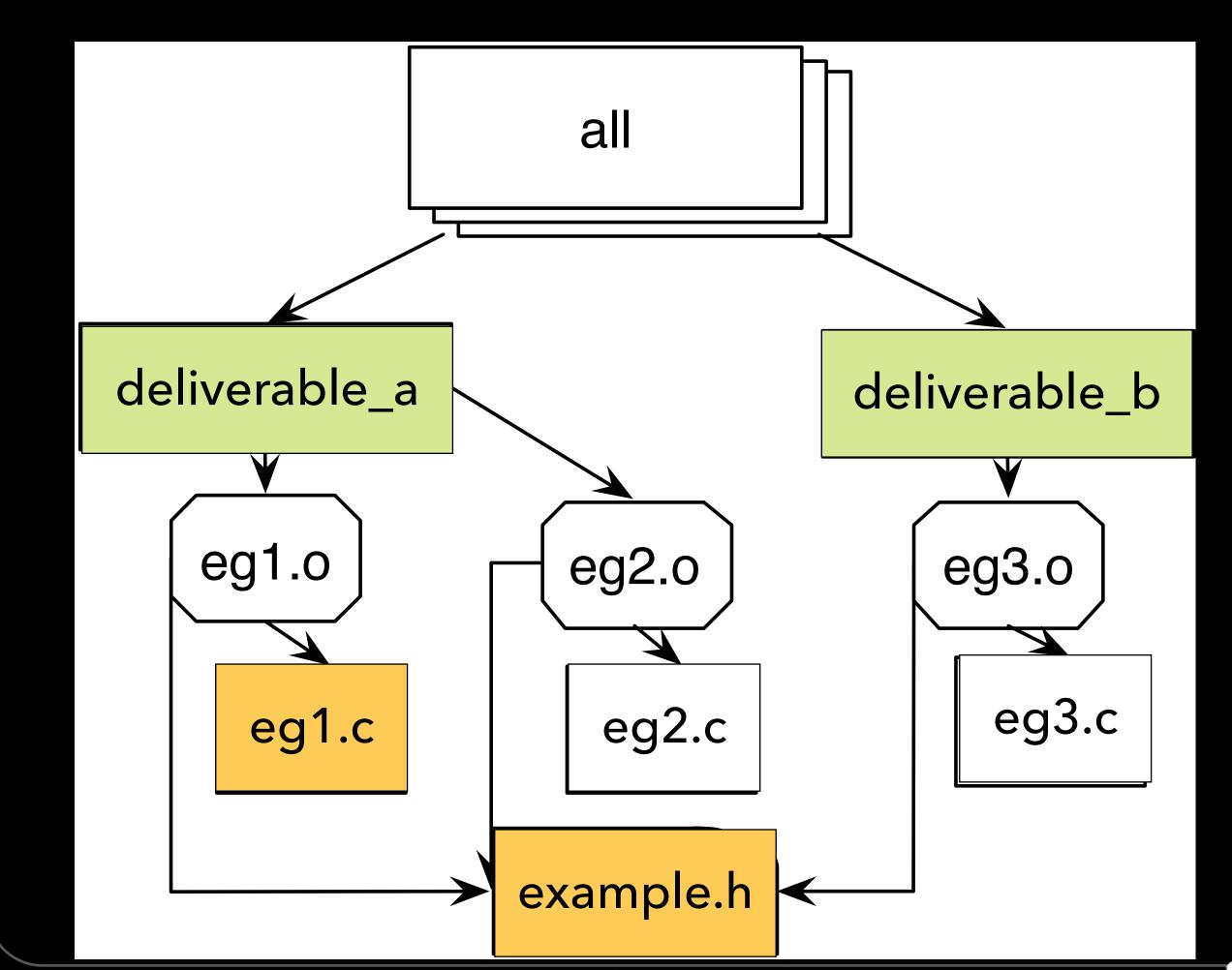
Wen et al. [ICSME 2018]





Step 3: Graph traversal and filtering

MAKAO

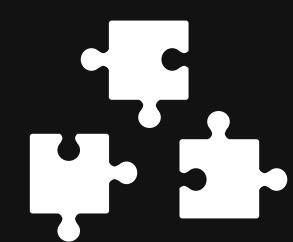


BLIMP Tracer:
Integrating Build
Impact Analysis with
Code Review
Wen et al.

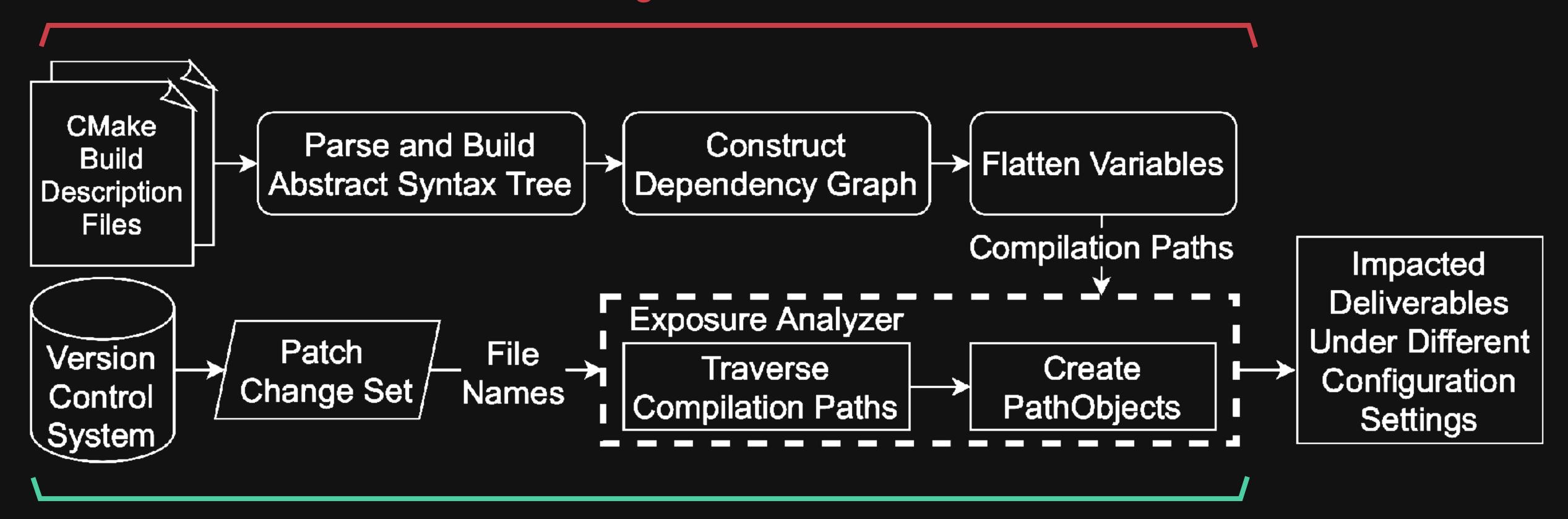
[ICSME 2018]



More recently, we started to develop the DiPiDi approach



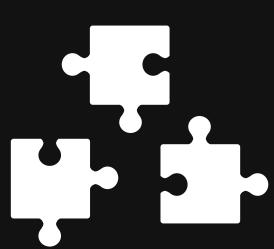
Indexing Phase

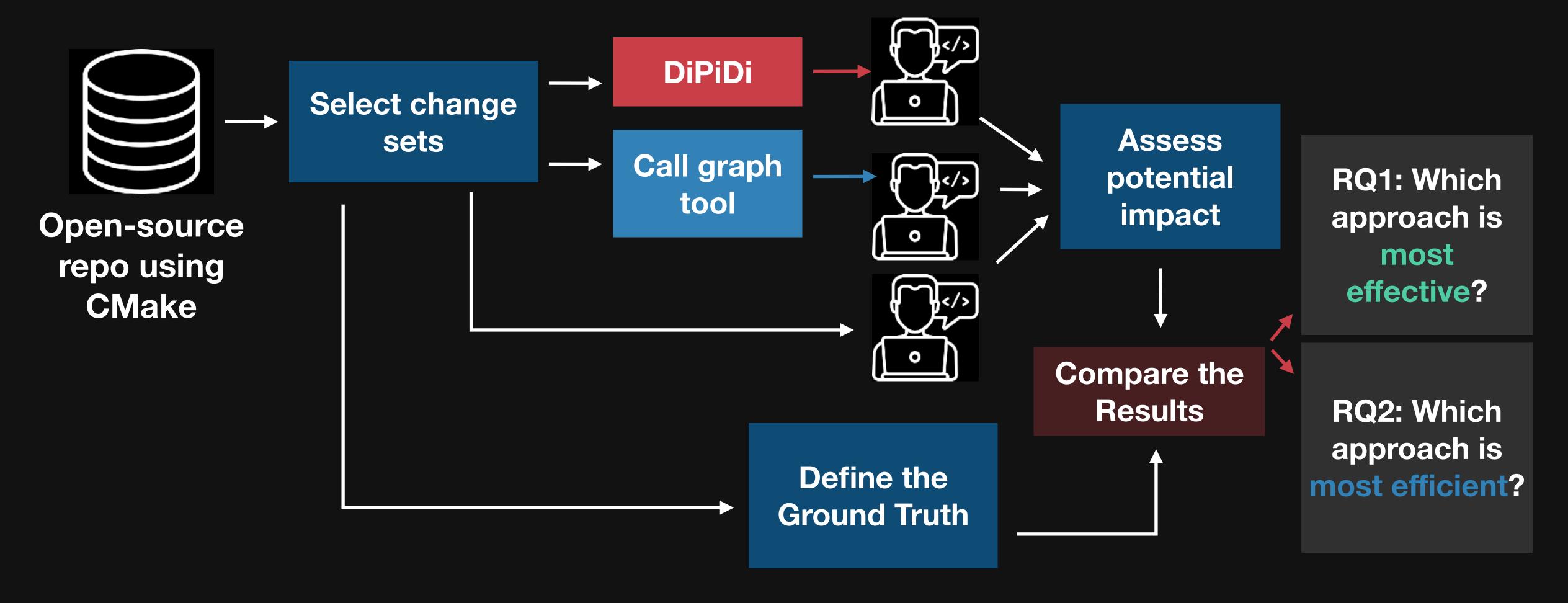






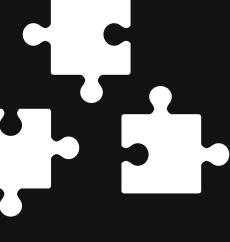
We are conducting controlled trials to evaluate whether DiPiDi actually helps









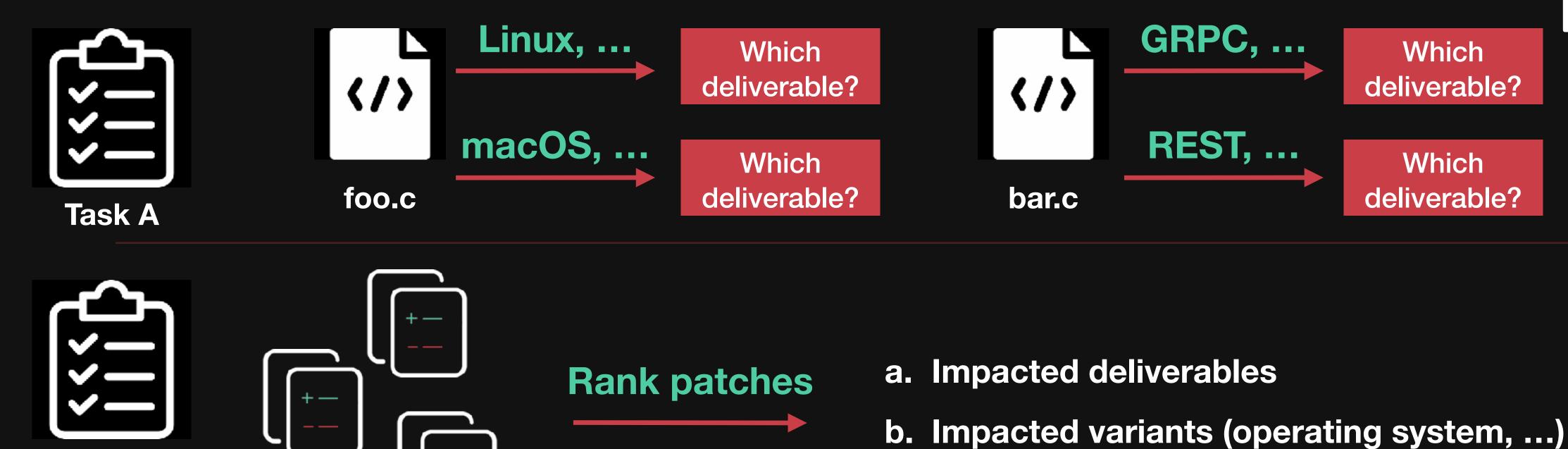








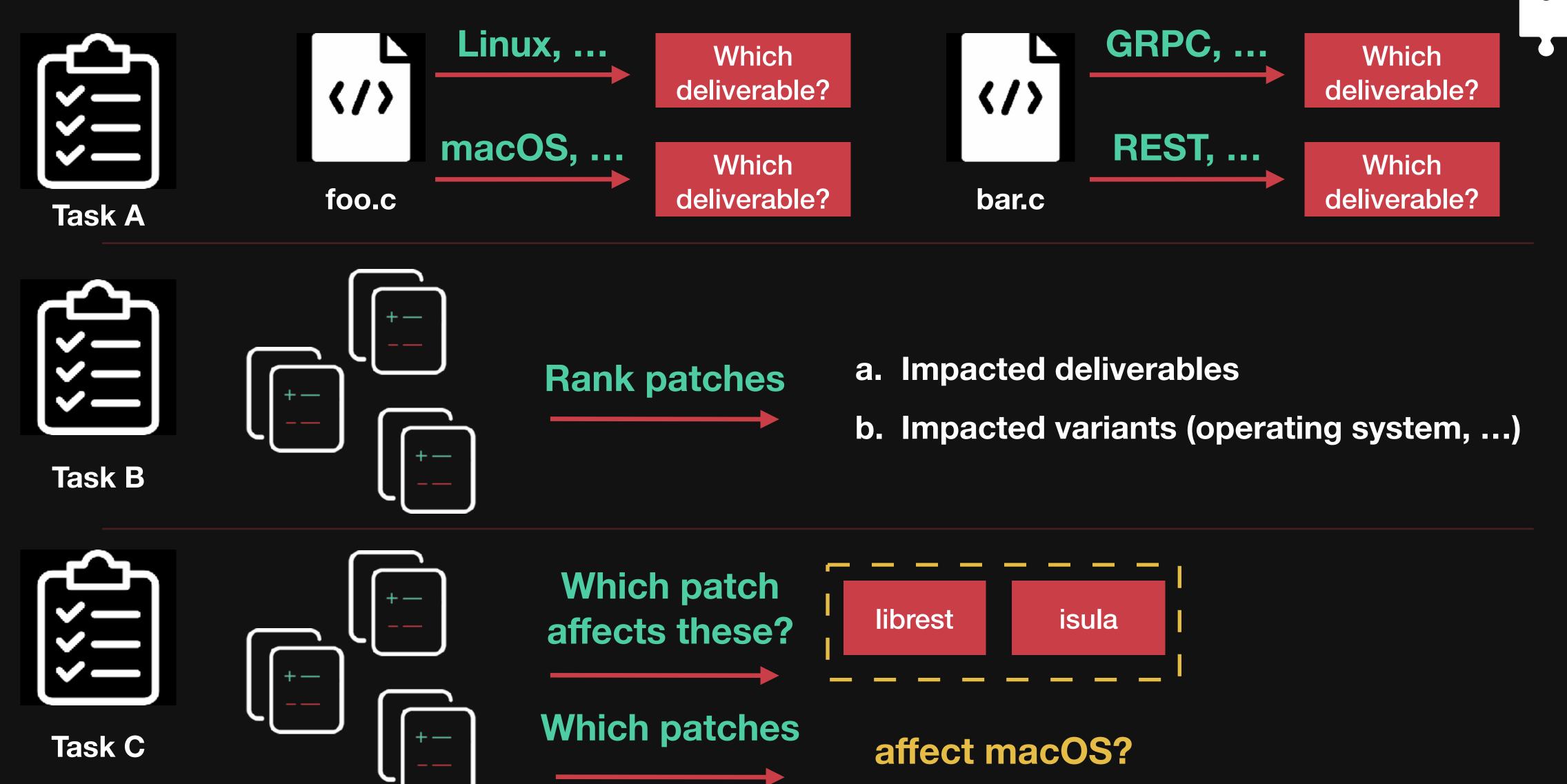






Task B



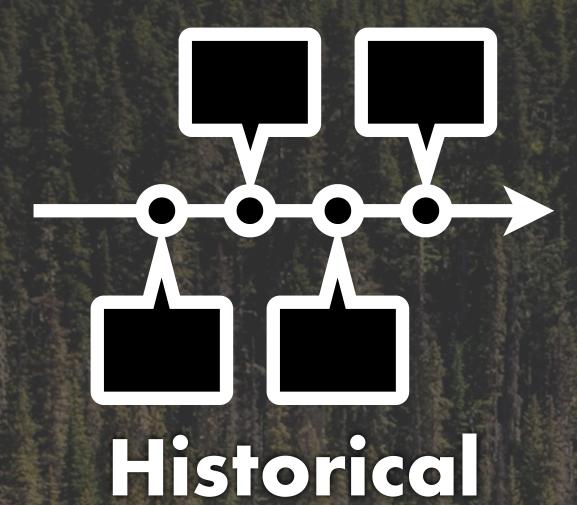






We need your help!

Participant sign-up: https://forms.gle/spmQEy5qxZShZyZ47



Exploit historical data for risk assessment





Operational

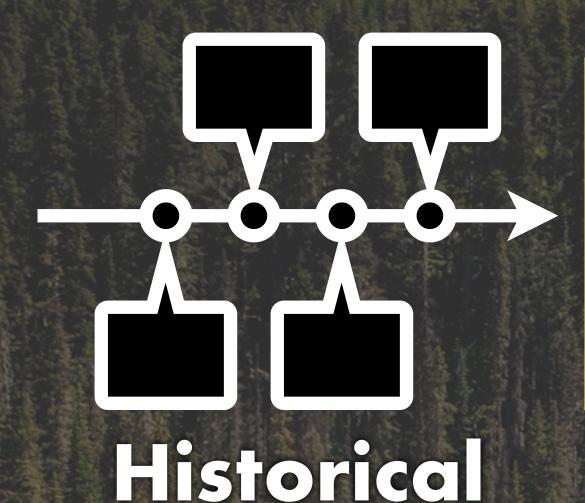


@Software REBELs



Concurrent





Exploit historical data for risk assessment



Mine dep. graphs for impact assessment



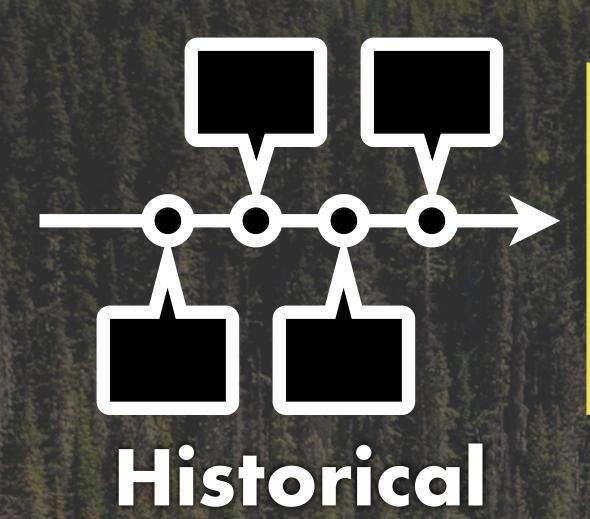
Operational





Concurrent





Exploit historical data for risk assessment



Mine dep. graphs for impact assessment



Operational

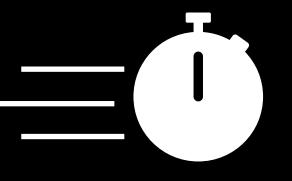


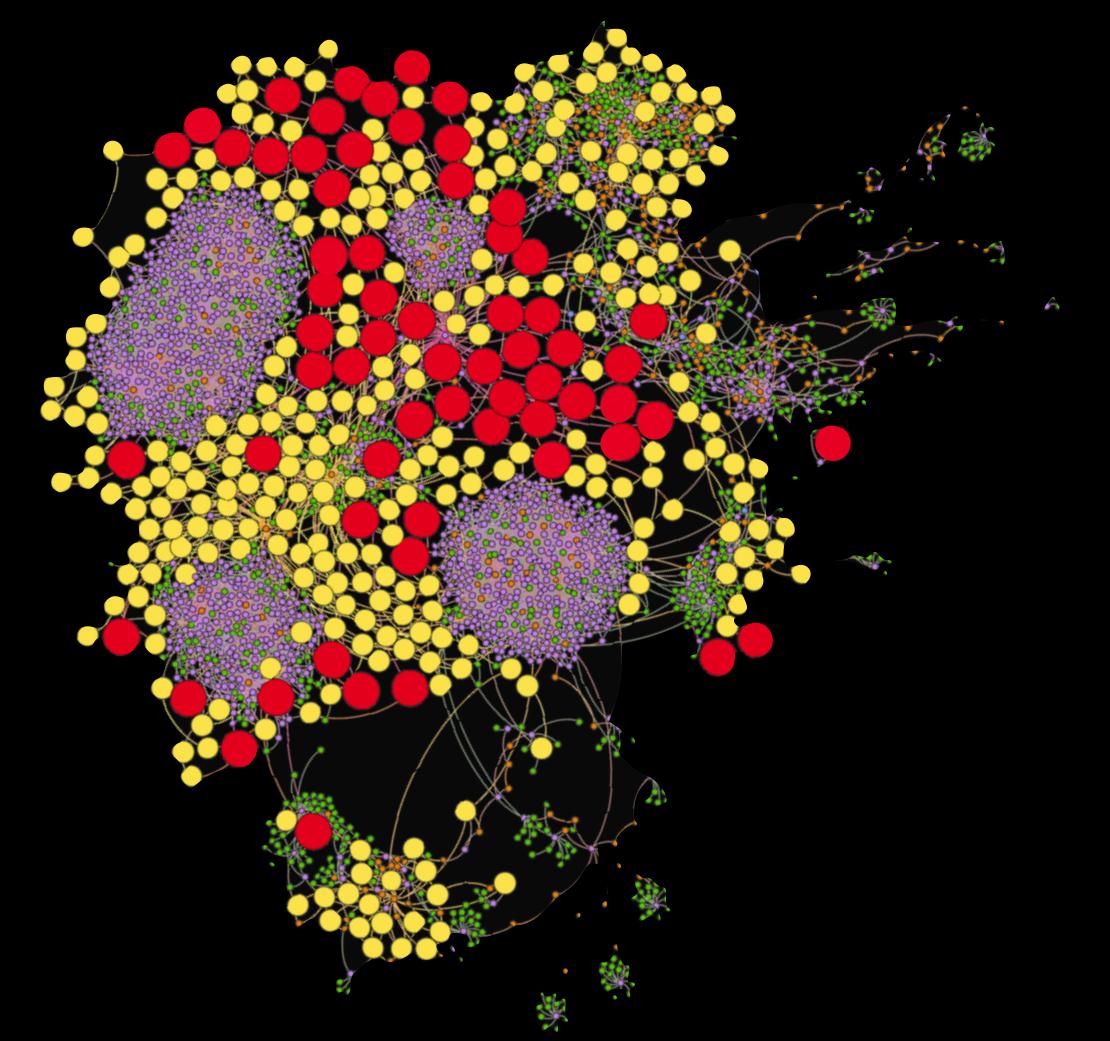


Concurrent



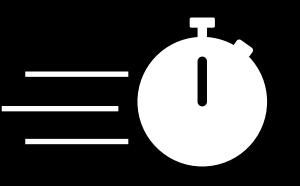
The build dependency graph misses the perspective of application usage tendencies





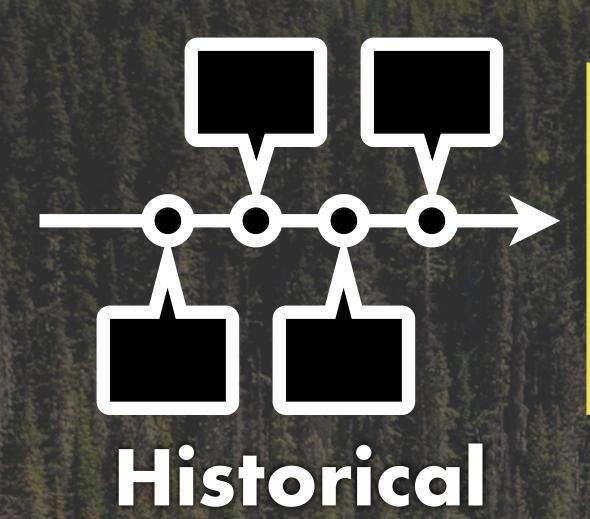


The build dependency graph misses the perspective of application usage tendencies









Exploit historical data for risk assessment



Mine dep. graphs for impact assessment



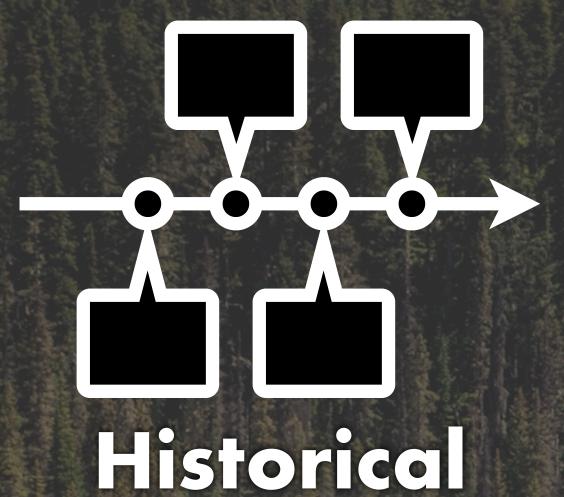
Operational





Concurrent





Exploit historical data for risk assessment



Operational



Leverage execution logs to add usage context

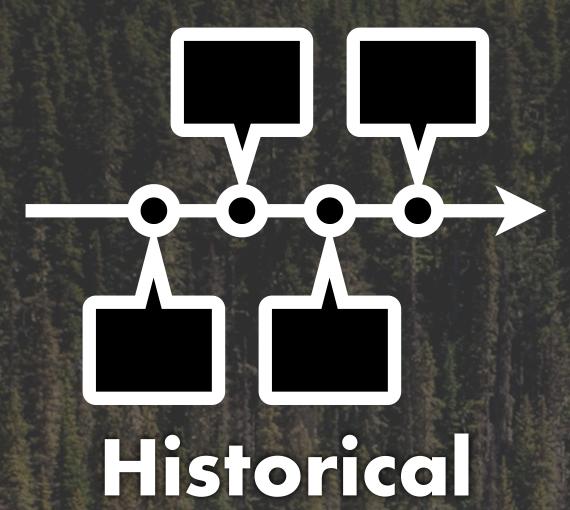


Mine dep. graphs for impact assessment



Concurrent





Exploit historical data for risk assessment



Operational

Leverage execution logs to add usage context



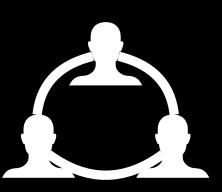
Mine dep. graphs for impact assessment

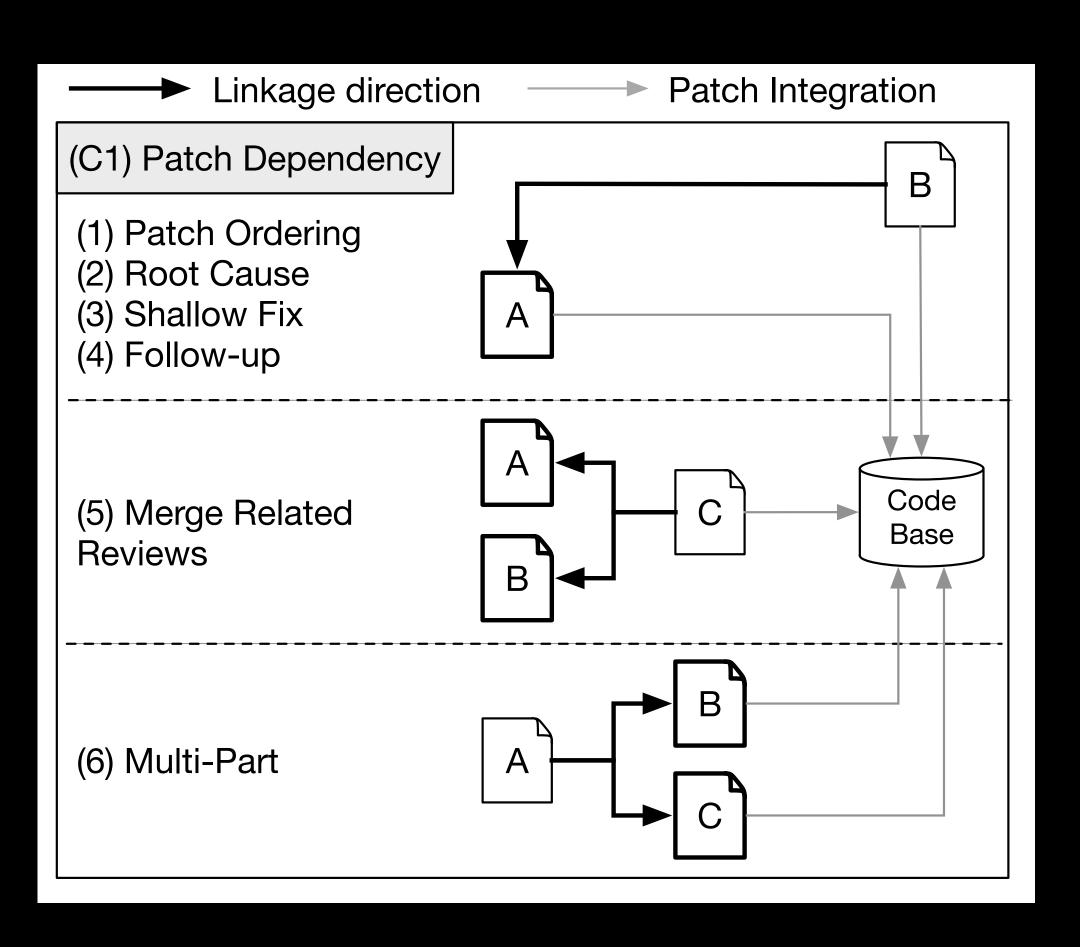


Concurrent

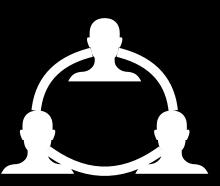


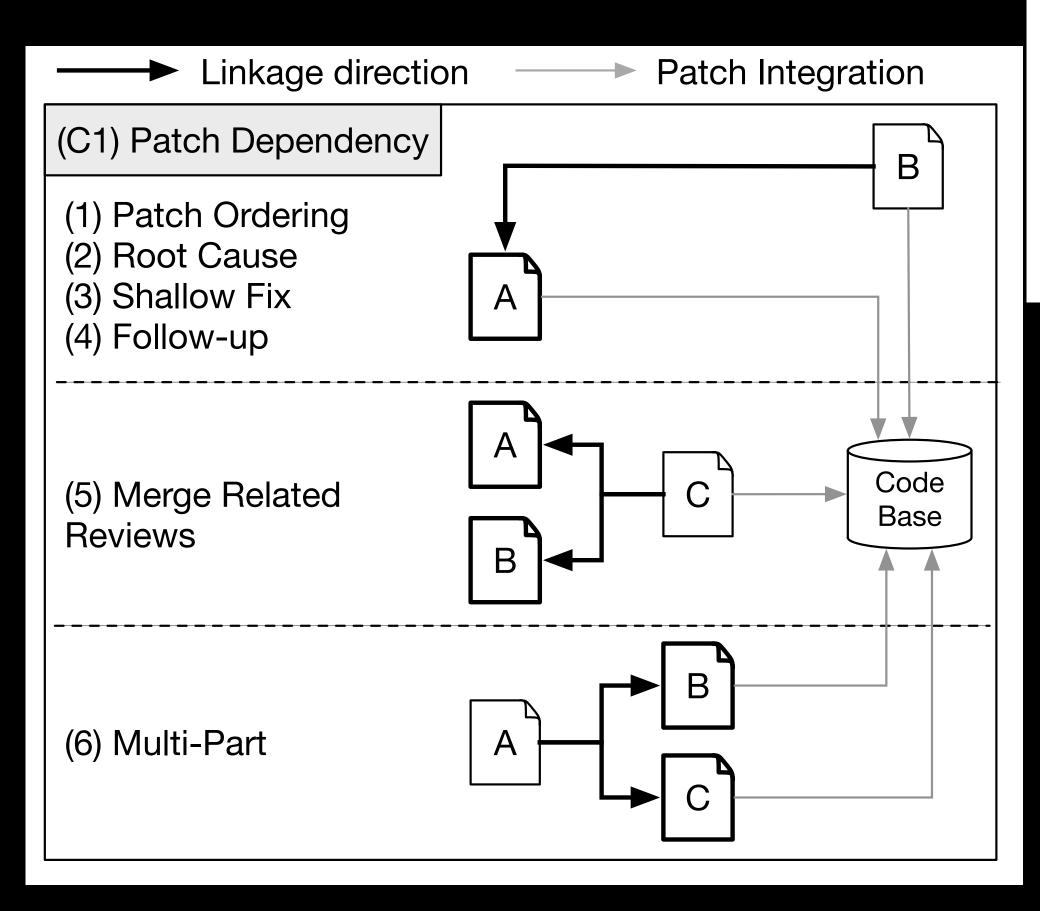


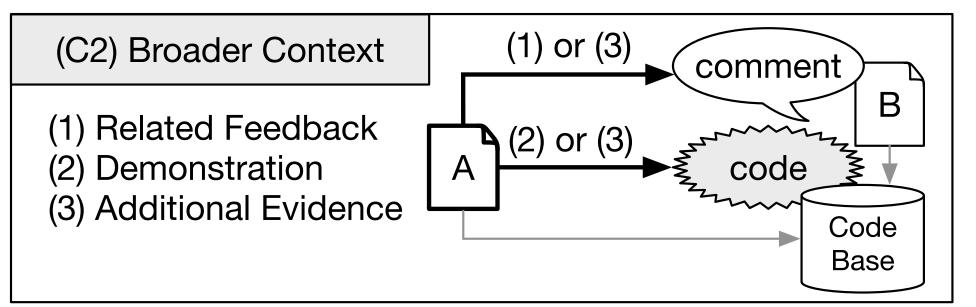




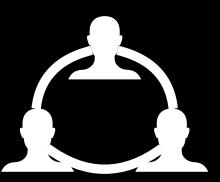


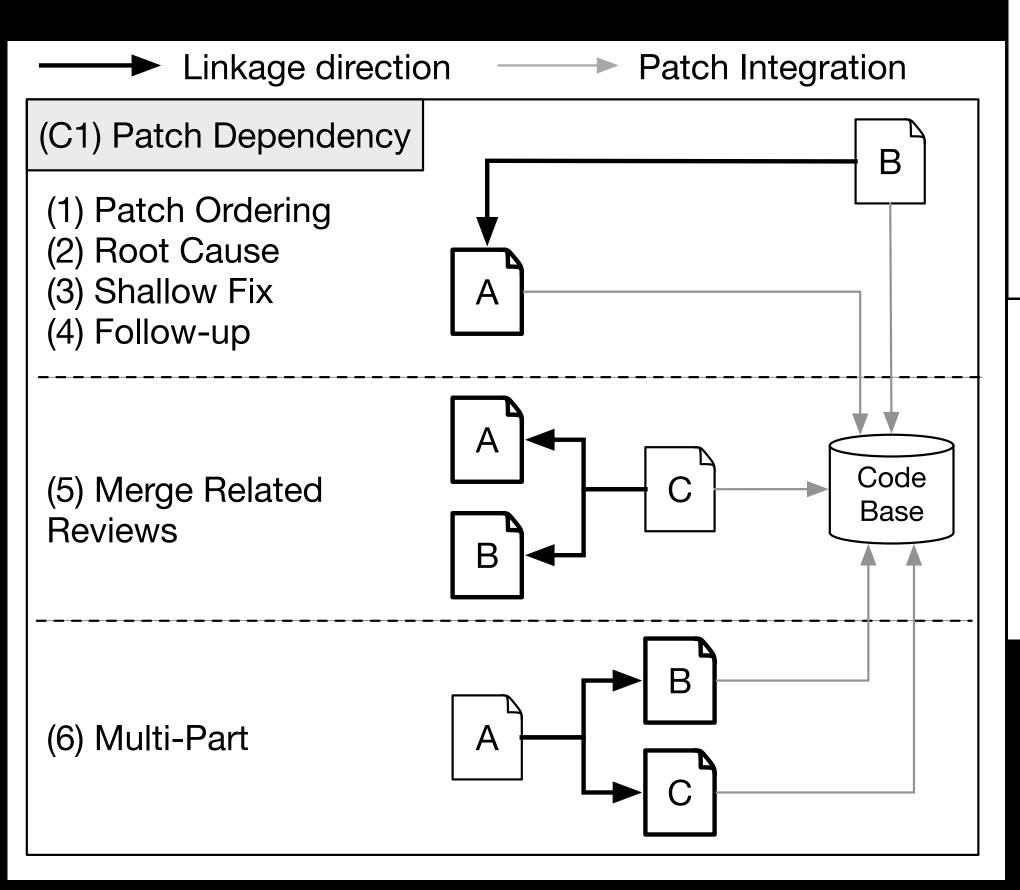


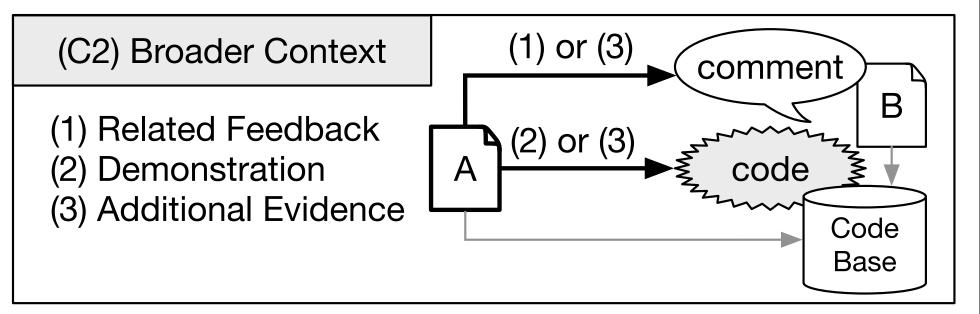


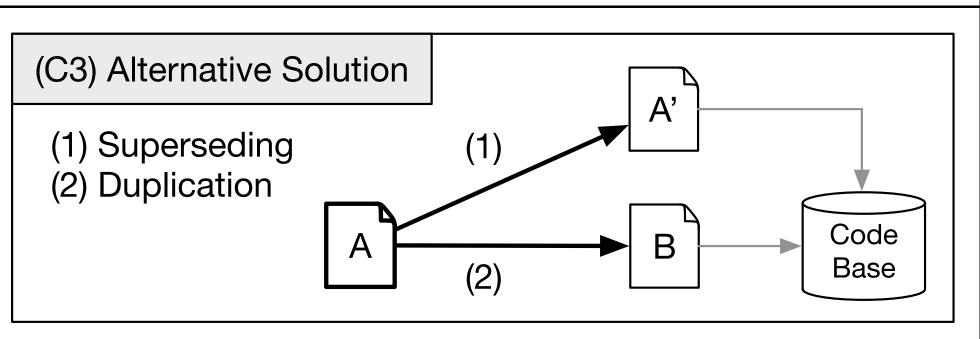




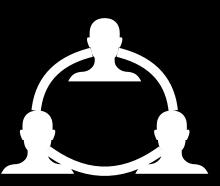


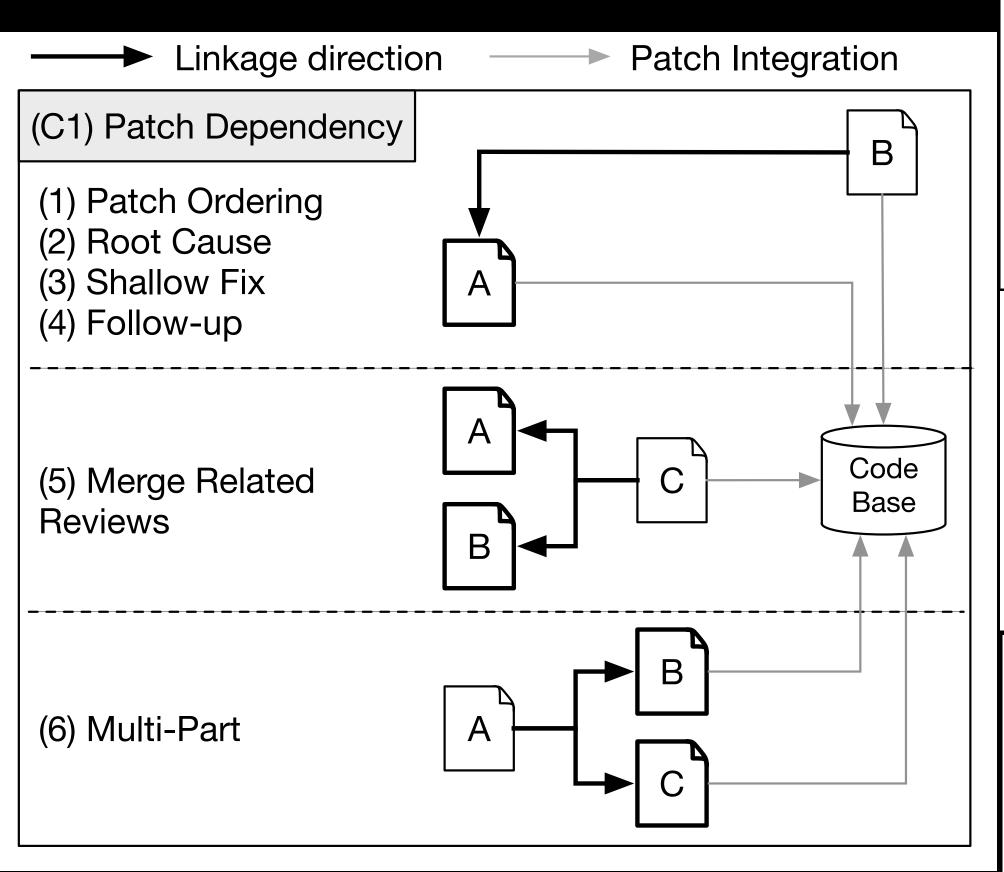


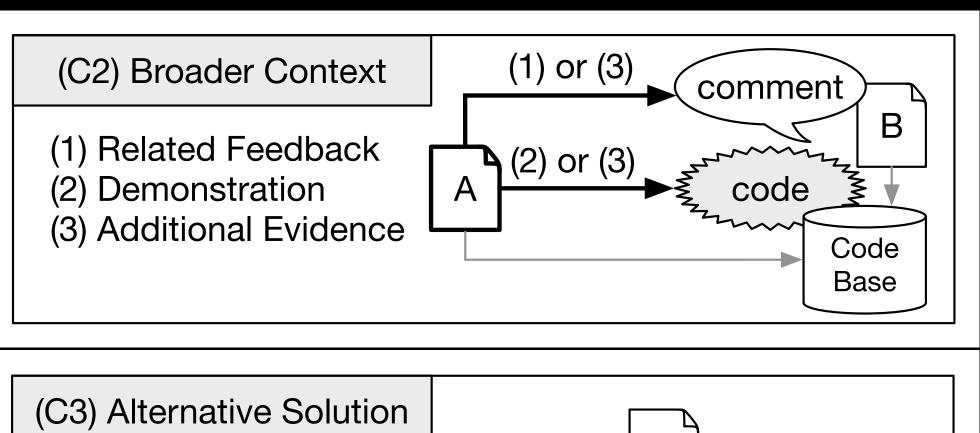


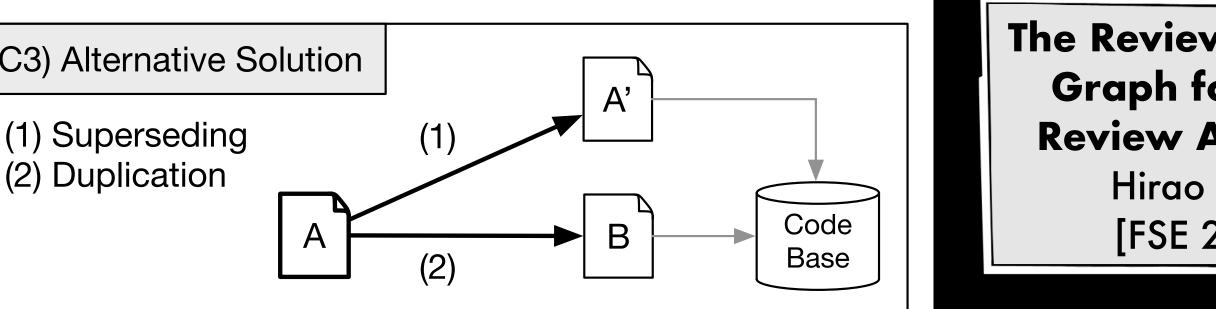


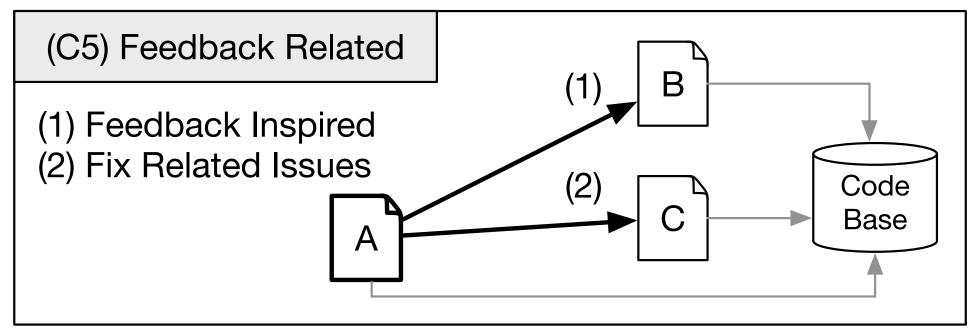














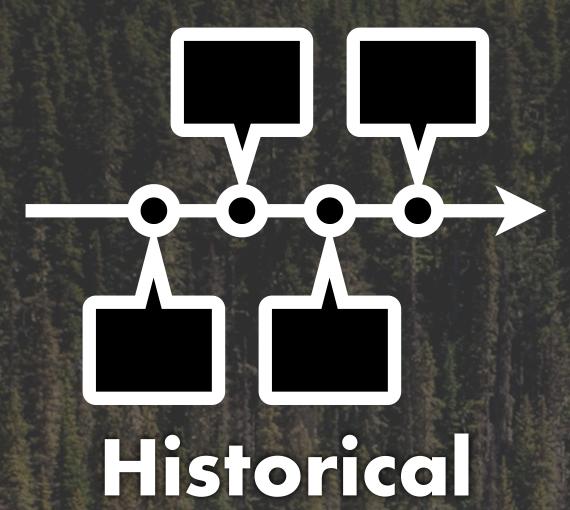












Exploit historical data for risk assessment



Operational

Leverage execution logs to add usage context



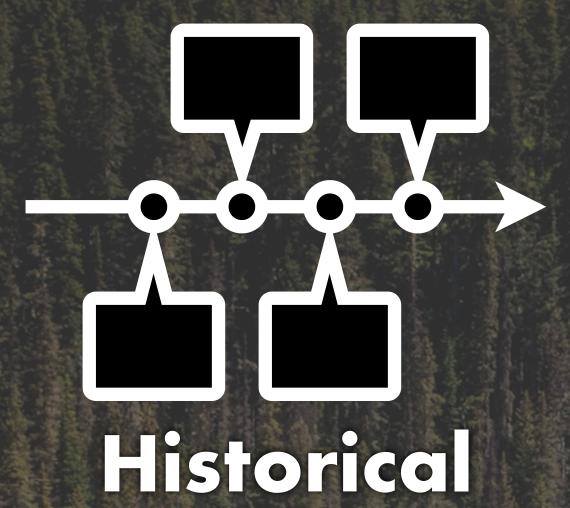
Mine dep. graphs for impact assessment



Concurrent







Exploit historical data for risk assessment



Operational

Leverage execution logs to add usage context



Mine dep. graphs for impact assessment



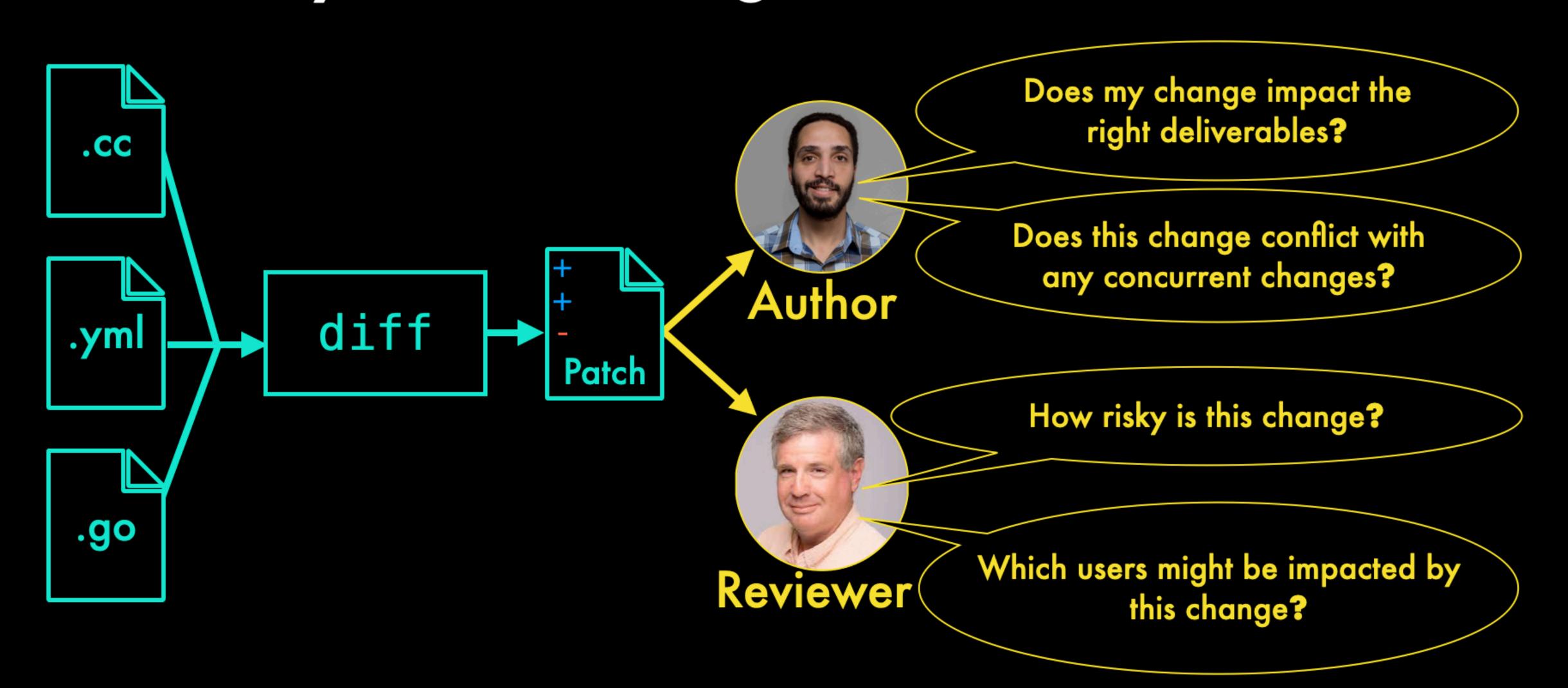
Incorporate associated reviews in the review interface

Concurrent

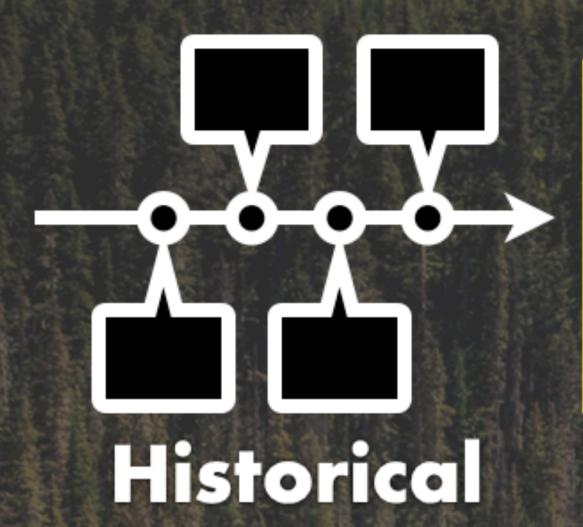




Text-based differences are pragmatic, but they are missing the broader context







Exploit historical data for risk assessment



Operational

Leverage execution logs to add usage context



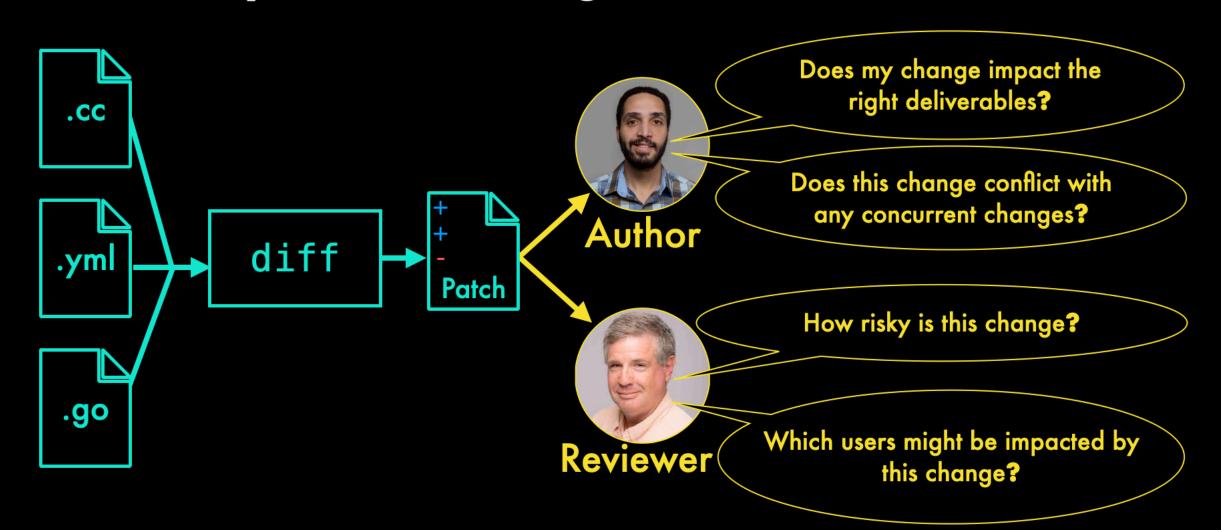
Mine dep. graphs for impact assessment



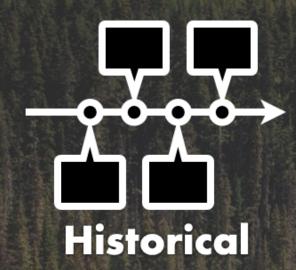
Concurrent

Incorporate associated reviews in the review interface

Text-based differences are pragmatic, but they are missing the broader context



Can't see the forest for the trees



historical data for risk assessment



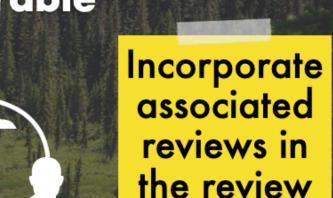
Operational

Exploit

Leverage execution logs to add usage context



Deliverable



Concurrent

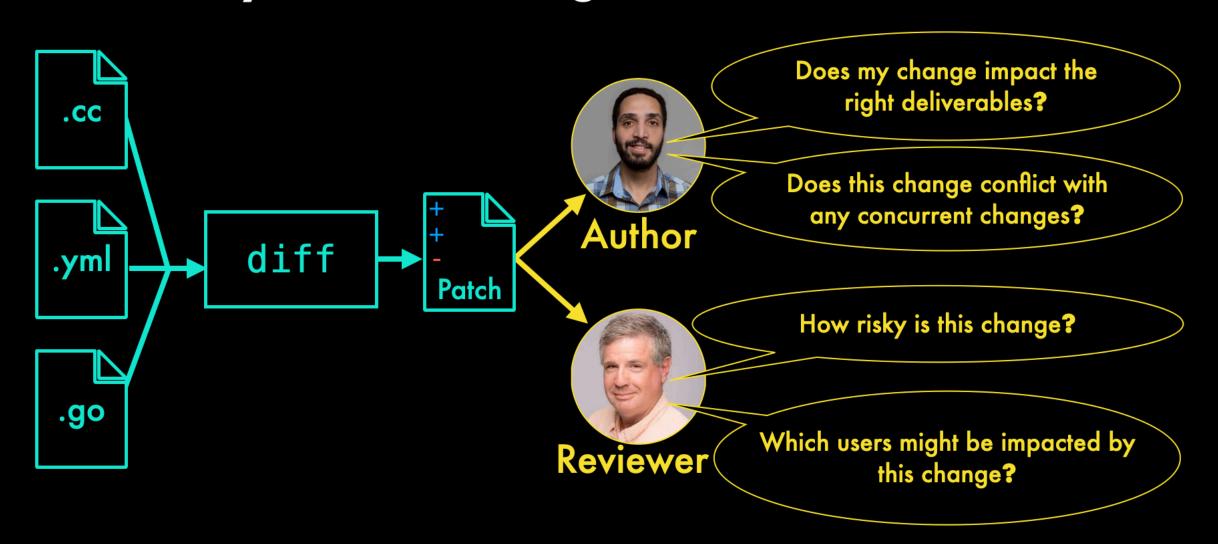
impact assessment

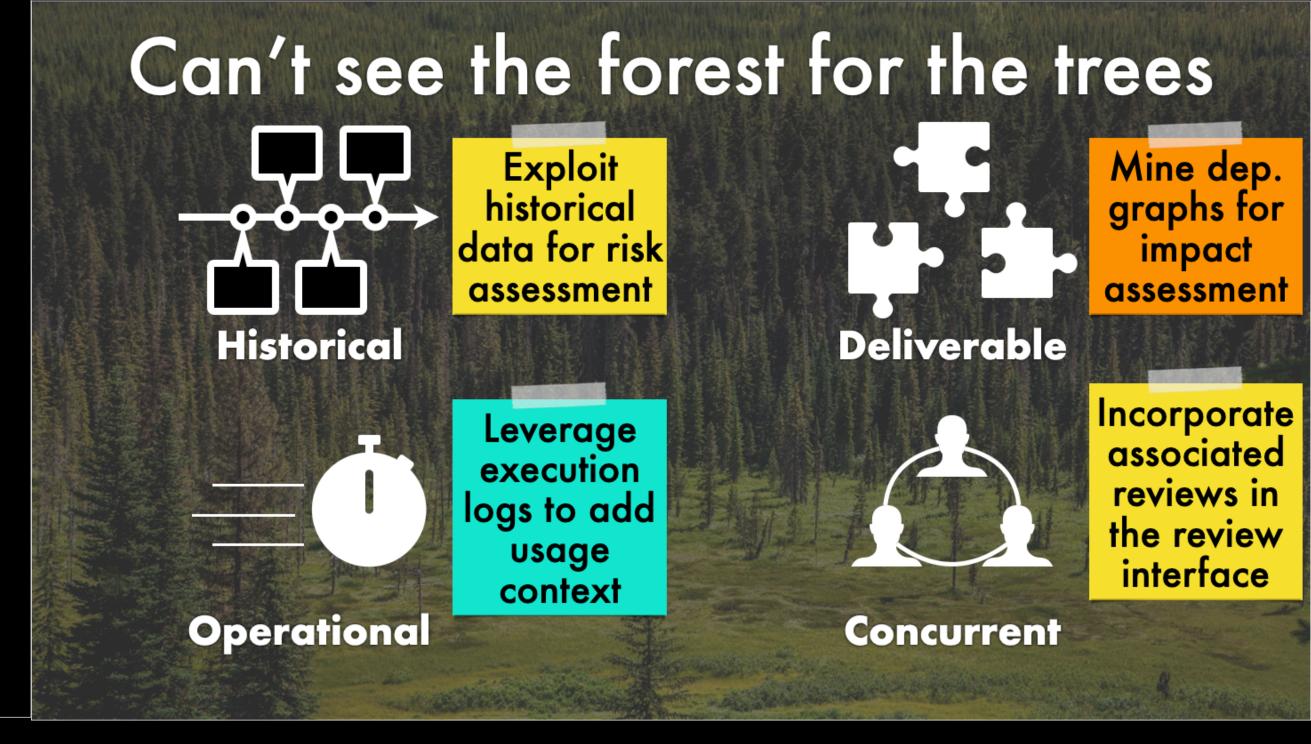
Mine dep.

graphs for

the review interface

Text-based differences are pragmatic, but they are missing the broader context









Participant sign-up: https://forms.gle/spmQEy5qxZShZyZ47