

CS 4530

Software Engineering

Lecture 12 - Debugging II + Code Review

Jonathan Bell, John Boyland, Mitch Wand
Khoury College of Computer Sciences

Zoom Mechanics

- Recording: This meeting is being recorded
- If you feel comfortable having your camera on, please do so! If not: a photo?
- I can see the zoom chat while lecturing, slack while you're in breakout rooms
- If you have a question or comment, please either:
 - “Raise hand” - I will call on you
 - Write “Q: <my question>” in chat - I will answer your question, and might mention your name and ask you a follow-up to make sure your question is addressed
 - Write “SQ: <my question>” in chat - I will answer your question, and not mention your name or expect you to respond verbally



Today's Agenda

Administrative:

HW3 due tomorrow!

Today's session:

Debugging

Code Review

What are your pro debugging strategies?

Domain-specific debugging

Valgrind: C/C++ Memory Errors

```
$ ./main  
Segmentation fault (core dumped)
```

```
$ valgrind ./main  
==8515== Memcheck, a memory error detector  
==8515== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.  
==8515== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info  
==8515== Command: ./main  
==8515==  
==8515== Conditional jump or move depends on uninitialised value(s)  
==8515==    at 0x400813: fail() (main.cpp:7)  
==8515==    by 0x40083F: main (main.cpp:13)  
==8515==  
==8515== Invalid read of size 4  
==8515==    at 0x400819: fail() (main.cpp:8)  
==8515==    by 0x40083F: main (main.cpp:13)  
==8515== Address 0x0 is not stack'd, malloc'd or (recently) free'd  
==8515==  
==8515==  
==8515== Process terminating with default action of signal 11 (SIGSEGV): dumping core  
==8515== Access not within mapped region at address 0x0  
==8515==    at 0x400819: fail() (main.cpp:8)  
==8515==    by 0x40083F: main (main.cpp:13)  
==8515== If you believe this happened as a result of a stack  
==8515== overflow in your program's main thread (unlikely but  
==8515== possible), you can try to increase the size of the  
==8515== main thread stack using the --main-stacksize= flag.  
==8515== The main thread stack size used in this run was 8388608.  
==8515==  
==8515== HEAP SUMMARY:  
==8515==    in use at exit: 72,704 bytes in 1 blocks  
==8515== total heap usage: 1 allocs, 0 frees, 72,704 bytes allocated  
==8515==  
==8515== LEAK SUMMARY:  
==8515==    definitely lost: 0 bytes in 0 blocks  
==8515==    indirectly lost: 0 bytes in 0 blocks  
==8515==    possibly lost: 0 bytes in 0 blocks  
==8515==    still reachable: 72,704 bytes in 1 blocks  
==8515==    suppressed: 0 bytes in 0 blocks  
==8515== Rerun with --leak-check=full to see details of leaked memory  
==8515==  
==8515== For counts of detected and suppressed errors, rerun with: -v  
==8515== Use --track-origins=yes to see where uninitialised values come from  
==8515== ERROR SUMMARY: 2 errors from 2 contexts (suppressed: 0 from 0)
```


Domain-specific debugging

TSan: Data race detector for C/C++/Go

```
$ cat simple_race.cc
#include <pthread.h>
#include <stdio.h>

int Global;

void *Thread1(void *x) {
    Global++;
    return NULL;
}

void *Thread2(void *x) {
    Global--;
    return NULL;
}

int main() {
    pthread_t t[2];
    pthread_create(&t[0], NULL, Thread1, NULL);
    pthread_create(&t[1], NULL, Thread2, NULL);
    pthread_join(t[0], NULL);
    pthread_join(t[1], NULL);
}
```

```
$ clang++ simple_race.cc -fsanitize=thread -fPIE -pie -g
$ ./a.out
=====
WARNING: ThreadSanitizer: data race (pid=26327)
  Write of size 4 at 0x7f89554701d0 by thread T1:
    #0 Thread1(void*) simple_race.cc:8 (exe+0x000000006e66)

  Previous write of size 4 at 0x7f89554701d0 by thread T2:
    #0 Thread2(void*) simple_race.cc:13 (exe+0x000000006ed6)

  Thread T1 (tid=26328, running) created at:
    #0 pthread_create tsan_interceptors.cc:683 (exe+0x00000001108b)
    #1 main simple_race.cc:19 (exe+0x000000006f39)

  Thread T2 (tid=26329, running) created at:
    #0 pthread_create tsan_interceptors.cc:683 (exe+0x00000001108b)
    #1 main simple_race.cc:20 (exe+0x000000006f63)
=====
ThreadSanitizer: reported 1 warnings
```

Domain-specific debugging

RVPredict: Data race detector (Java)

```
package examples;

public class SimpleRace {
    static int sharedVar;

    public static void main(String[] args) {
        new ThreadRunner() {
            @Override
            public void thread1() {
                sharedVar++;
            }

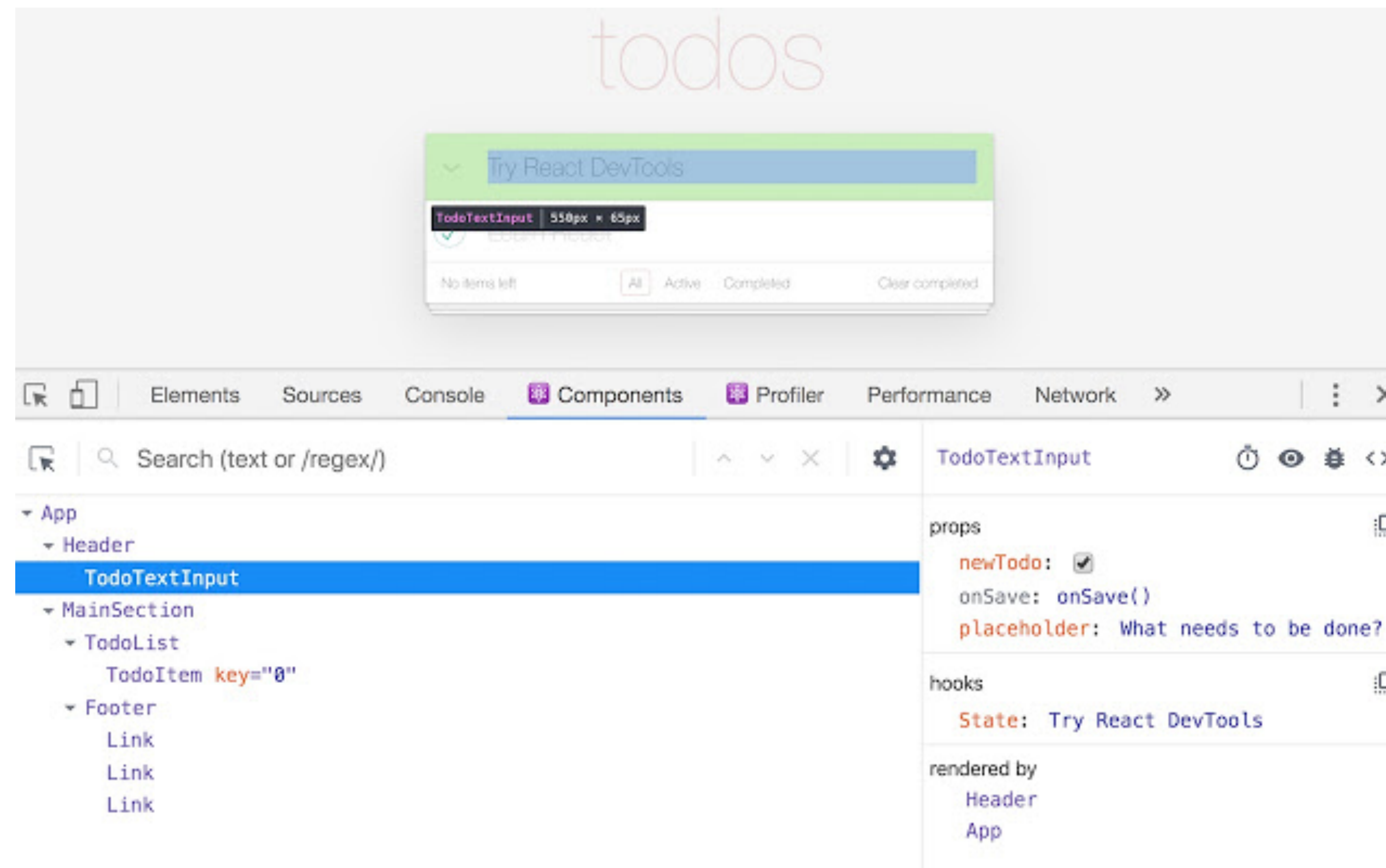
            @Override
            public void thread2() {
                sharedVar++;
            }
        };
    }
}
```

```
Data race on field examples.SimpleRace.sharedVar: {{{
  Concurrent write in thread T10 (locks held: {})
  ----> at examples.SimpleRace$1.thread1(SimpleRace.java:11)
         at examples.ThreadRunner$1.run(ThreadRunner.java:17)
  T10 is created by T1
         at examples.ThreadRunner.<init>(ThreadRunner.java:26)

  Concurrent read in thread T11 (locks held: {})
  ----> at examples.SimpleRace$1.thread2(SimpleRace.java:16)
         at examples.ThreadRunner$2.run(ThreadRunner.java:23)
  T11 is created by T1
         at examples.ThreadRunner.<init>(ThreadRunner.java:27)
  }}}
}
```

Domain-specific debugging


React Developer Tools




Activity: Debugging (Continued)

<https://neu-se.github.io/CS4530-CS5500-Spring-2021/Activities/activity7-1>


What are code reviews?

...re-api/src/main/java/org/apache/maven/surefire/booter/CommandReader.java  Hide resolved


```
case BYE_ACK:
    //After SHUTDOWN no more commands can come. Hence, do NOT go back to blocking in IO
    callListeners( command );
    return;
default:
    callListeners( command );
```




Tibor17 on Nov 12, 2019 Contributor

 ...


The listeners are called here. But we can put IF condition:
IF BYE_ACK -> return at the end of the default case.




Tibor17 on Nov 12, 2019 Contributor

 ...


Instead of calling the return we can make softer exit with `CommandReader.this.state.set(TERMINATED)`.




eolivelli on Dec 17, 2019 Contributor

 ...


Yes, I came to this same conclusion, change the state to TERMINATED.



jon-bell on Dec 19, 2019 Author Contributor

 ...

Changed.



Darko

Linus' Law

“Many eyes make all bugs shallow”

```
public disconnect(session : PlayerSession) : void {  
    console.log(`Disconnect: ${session.sessionToken}`);  
    this._sessions = this._sessions.filter(s => s !== session);  
    this._listeners.map((l) => l.onPlayerDisconnected(session.player));  
}
```

Formal “Code Inspections”

The origins of Code Reviews

- Formal process for reading through code as a group
- Applied to all project documents
- 3-5 person team reads the code aloud and explains what is being done
- Generally a 60 minute meeting
- Less efficient (defects/cost) than modern review processes

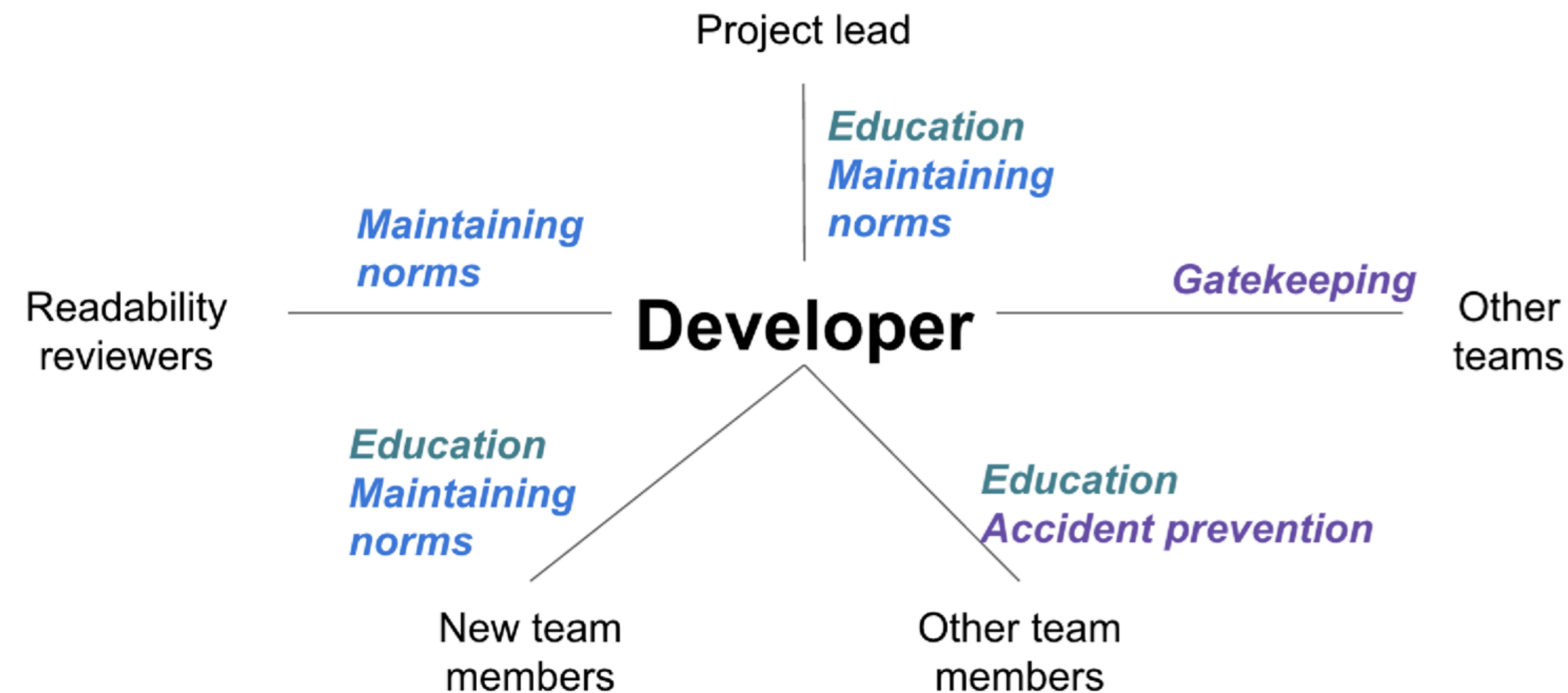
Why do we perform code reviews?

Survey internal to Google

- Original motivation: “force developers to write code that other developers could understand”
- Three key benefits found:
 - Ensure consistent style/design standards followed
 - Ensure adequate tests
 - Provides a security control (gatekeeping, especially for critical code)

Why do we perform code reviews?

Different team members have different motivations



Why do we perform code reviews?

Survey internal to Microsoft

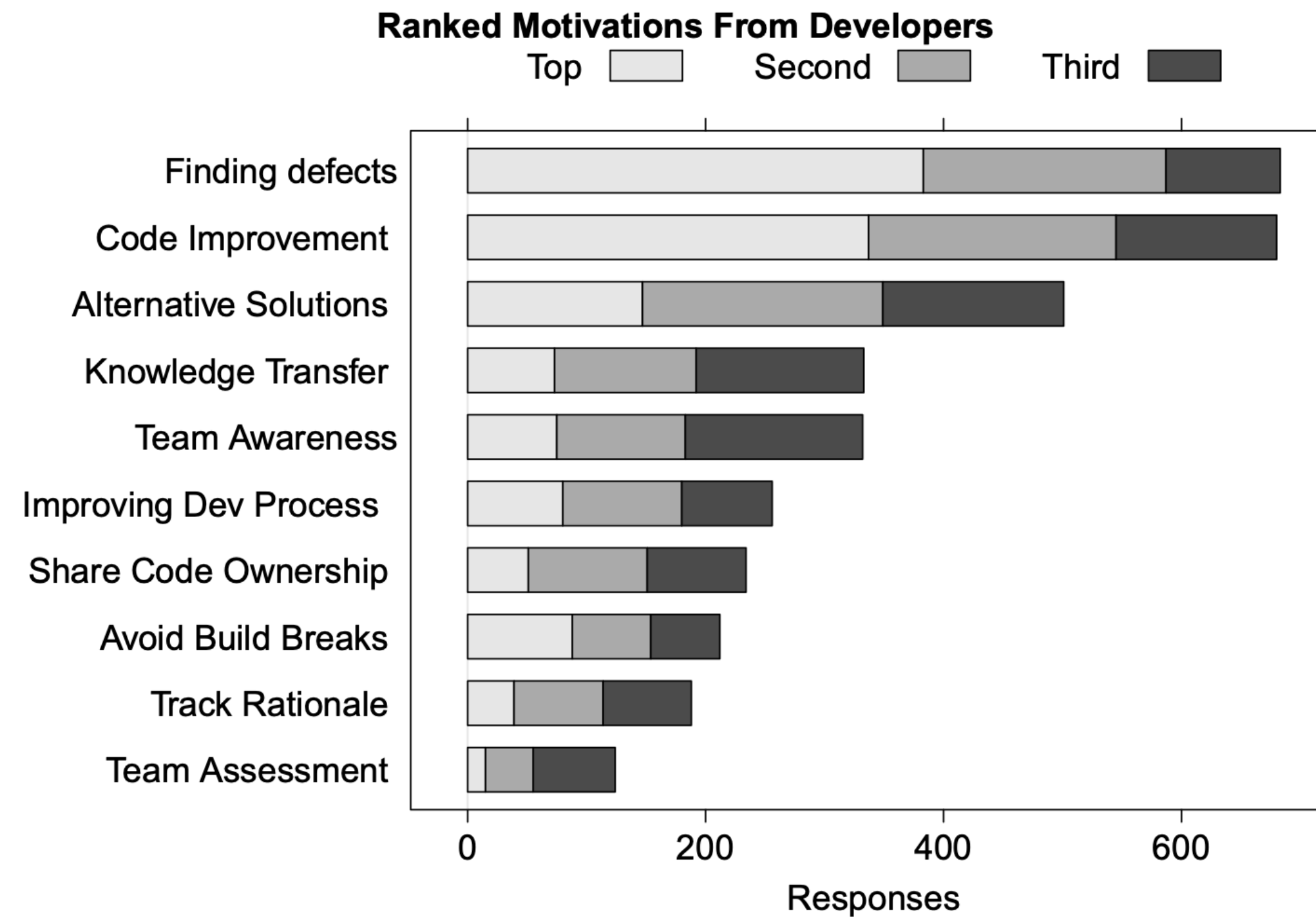
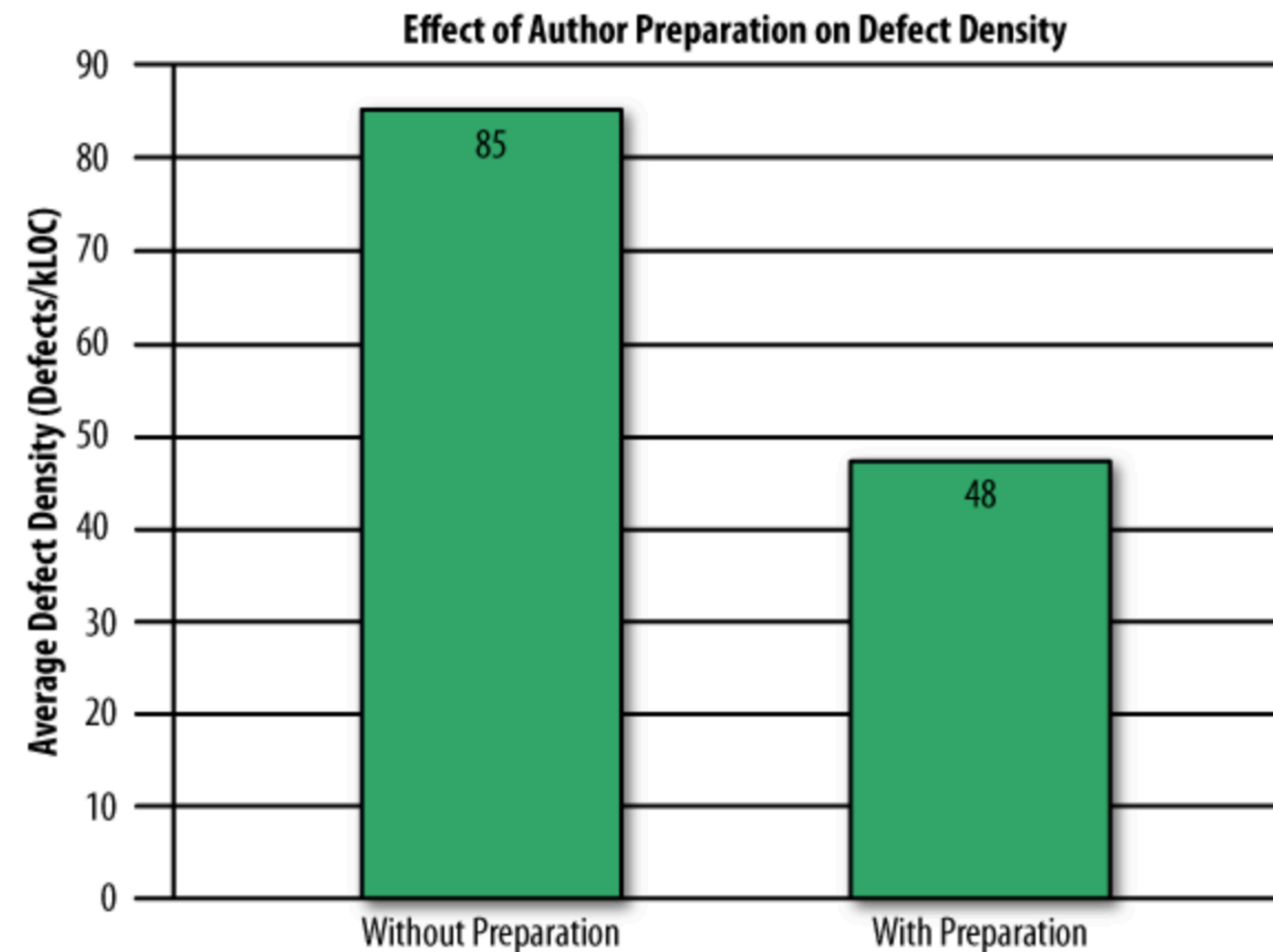


Fig. 3. Developers' motivations for code review.

Studies show self-review is less effective

300 reviews at Cisco in 2006



Even if developers pre-review their code, many defects still found in peer review

“Best Kept Secrets of Peer Code Review”, Jason Cohen, SmartBear Software, 2006

How do we perform code reviews?

Who reviews what?

- Don't review your own code
- Ideal: reviewer has different background, different experience
- Come in with no preconceived idea of “correctness”
- Don't be biased by “what was intended”

Code Review: Proposed Patch

“Prevents more than 5 users from joining the room”

```
export async function roomJoinHandler(requestData: TownJoinRequest): Promise<ResponseEnvelope<TownJoinResponse>> {
  const room = CoveyRoom.findInstance(requestData.coveyTownID);
  if (!room) {
    return {
      isOK: false,
      message: 'Error: No such room',
    };
  }
  const newPlayer = new Player(requestData.userName);
  const newSession = await room.registerPlayer(newPlayer);
  assert(newSession.videoToken);
  if (room.occupancy > 5) {
    return {
      isOK: false,
      message: 'The room is full!',
    };
  }
  return {
    isOK: true,
    response: {
      coveyUserID: newPlayer.id,
      coveySessionToken: newSession.sessionToken,
      providerVideoToken: newSession.videoToken,
      currentPlayers: room.getPlayers(), // TODO check
      friendlyName: room.friendlyName,
      isPubliclyListed: room.isPubliclyListed,
    },
  };
};
```


How do we perform code reviews?

What are we looking for?



AIRBUS A340
FLIGHT CREW OPERATING MANUAL

NORMAL CHECK LIST

BEFORE START	
COCBIT PREP.....	COMPLETE
GEAR PINS and COVERS.....	REMOVED
SIGNS.....	ON
ADIRS.....	NAV
FUEL QUANTITY.....	KG
TO DATA.....	SET
BARO REF.....	SET
WINDOWS/DOORS.....	CLOSED
BEACON.....	ON
THR LEVERS.....	IDLE
PARKING BRAKE.....	AS RORD
AFTER START	
ANTI ICE.....	AS RORD
ECAM STATUS.....	CHECKED
THR LEVERS.....	IDLE
PARKING BRAKE.....	AS RORD
BEFORE TAKEOFF	
FLIGHT CONTROLS.....	CHECKED
FLIGHT INST.....	CHECKED
BRIEFING.....	CONFIRMED
FLAPS SETTING.....	CONF - SET
V1, VR, V2 / FLEX TEMP.....	SET
ATC.....	SET
ECAM MEMO.....	TO ALL GREEN
• SIGNS ON • AUTO BRK MAX • CABIN READY • TO CONFG NORM • SPLRS ARM • FLAPS TO	
CABIN CREW.....	ADVISED
ENG MODE SEL.....	AS RORD
PACKS.....	AS RORD
AFTER TAKEOFF / CLIMB	
LDG GEAR.....	UP
FLAPS.....	RETRACTED
PACKS.....	ON
BARO REF.....	STD

APPROACH	
BRIEFING.....	CONFIRMED
ECAM STATUS.....	CHECKED
SEAT BELTS.....	ON
BARO REF.....	SET
MDA / DH.....	SET
ENG MODE SEL.....	AS RORD
LANDING	
CABIN CREW.....	ADVISED
A/THR.....	SPEED / OFF
ECAM MEMO.....	LDG ALL GREEN
• SIGNS ON • FLAPS SET • CABIN READY • SPLRS ARM • LOG GEAR DN	
AFTER LANDING	
FLAPS.....	RETRACTED
SPOILERS.....	DISARMED
APU.....	START
PADAR.....	OFF / STBY
PARKING	
APU BLEED.....	ON
ENGINES.....	OFF
SEAT BELTS.....	OFF
EXT LT.....	AS RORD
FUEL PUMPS.....	OFF
PARKING BRAKE and CHOCKS.....	AS RORD
SECURING THE AIRCRAFT	
ADIRS.....	OFF
OXYGEN.....	OFF
APU BLEED.....	OFF
EMER EXIT LIGHTS.....	OFF
NO SMOKING.....	OFF
APU and BAT.....	OFF

ON GROUND EMER EVACUATION

- AIRCRAFT / PARKING BRK..... STOP/ON
- ATC (VHF 1)..... NOTIFY
- ΔP (only if MAN CAB PR has been used)..... CHECK ZERO
- if not zero, MODE SEL on MAN and V/S CTL FULL UP
- ENG MASTER 1 and 2..... OFF
- CABIN CREW (PA)..... NOTIFY
- FIRE P/Bs (ENG and APU)..... PUSH
- AGENTS (ENG and APU)..... AS RORD
- EVACUATION..... INITIATE

A340 TAKEOFF CG 18 20 22 24 26 28 30 32 34 36 38 40

TRIM POS 7 6 5 4 3 2 1

NOSE UP

Code review checklists

Some common ideas to get your checklist started

- Am I able to understand the code easily?
- Does the code follow our style guidelines?
- Is the same code duplicated more than once?
- Does this code meet our non-functional requirements?
- Is this code maintainable? Does it have **tests**?
- Does this code have unintended side-effects?
- Can include issues previously detected in the past

Code Review: Proposed Patch

“Prevents more than 5 users from joining the room”

```
it('Does not allow more than 5 users to join a room', async () => {  
  const newRoom = await apiClient.createRoom({isPubliclyListed: true, friendlyName: 'test'});  
  const promisesShouldBeAccepted = [];  
  for (let i = 0; i < 5; i += 1) {  
    promisesShouldBeAccepted.push(apiClient.joinRoom({userName: 'test', coveyTownID: newRoom.coveyTownID}));  
  }  
  await Promise.all(promisesShouldBeAccepted);  
});
```

Code Review: Proposed Patch

“Prevents more than 5 users from joining the room”

```
it('Does not allow more than maximum users to join a room', async () => {
  const newRoom = await apiClient.createRoom({isPubliclyListed: true, friendlyName: 'test'});
  const rooms = await apiClient.listRooms();
  const createdRoomInfo = rooms.towns.find(room => room.coveyTownID === newRoom.coveyTownID);
  assert(createdRoomInfo);
  const promisesShouldBeAccepted = [];
  for (let i = 0; i < createdRoomInfo.maximumOccupancy; i += 1) {
    promisesShouldBeAccepted.push(apiClient.joinRoom({userName: 'test', coveyTownID: newRoom.coveyTownID}));
  }
  await Promise.all(promisesShouldBeAccepted);
  await expect(apiClient
    .joinRoom({coveyTownID: newRoom.coveyTownID, userName: 'test'}))
    .rejects.toThrowError();

});
```

Code Review: Proposed Patch

“Prevents more than 5 users from joining the room”

```
it('Does not allow more than maximum users to join a room', async () => {
  const newRoom = await apiClient.createRoom({isPubliclyListed: true, friendlyName: 'test'});
  const rooms = await apiClient.listRooms();
  const createdRoomInfo = rooms.towns.find(room => room.coveyTownID === newRoom.coveyTownID);
  assert(createdRoomInfo);
  const promisesShouldBeAccepted = [];
  for (let i = 0; i < createdRoomInfo.maximumOccupancy; i += 1) {
    promisesShouldBeAccepted.push(apiClient.joinRoom({userName: 'test', coveyTownID: newRoom.coveyTownID}));
  }
  await Promise.all(promisesShouldBeAccepted);
  await expect(apiClient
    .joinRoom({coveyTownID: newRoom.coveyTownID, userName: 'test'}))
    .rejects.toThrowError();

  // Now list rooms
  const roomsAfterJoining = await apiClient.listRooms();
  const updatedRoomInfo = roomsAfterJoining.towns.find(r => r.coveyTownID === newRoom.coveyTownID);
  assert(updatedRoomInfo);
  expect(updatedRoomInfo.currentOccupancy).toBeLessThanOrEqual(updatedRoomInfo.maximumOccupancy);
});
```

When do we perform code reviews?

In modern development environments (large OSS + companies)

- For **every** change that gets merged upstream!
- If you are asked to review something, you **must do it soon** (but don't interrupt current task)
- Include the entire context of a change, not just a `diff`
- Google's entire process is publicly documented: <https://google.github.io/eng-practices/review/>

This work is licensed under a Creative Commons Attribution-ShareAlike license

- This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/4.0/>
- You are free to:
 - Share — copy and redistribute the material in any medium or format
 - Adapt — remix, transform, and build upon the material
 - for any purpose, even commercially.
- Under the following terms:
 - Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
 - ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.
 - No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.