

# Real Bugs Real Projects Real Impact

**Andrzej Wąsowski**

joint work with (lexicographically)  
**Iago Abal, Claus Brabrand  
Jonathan Hechtbauer  
Gijs van der Hoorn  
Alexandru F. Iosif Lazar  
Jean Melo, Marcio Ribeiro  
Stefan Stanculescu  
Andre Santos, Chris Timperley**



IT UNIVERSITY OF COPENHAGEN



**MODELS 2018**

**COPENHAGEN**

**October 14<sup>th</sup> -19<sup>th</sup>**

**[www.modelsconference.org](http://www.modelsconference.org)**

**Keynote**



**Silvija Seres**

Independent  
Advisor & Investor

**Keynote**



**Martijn Wisse**

Prof. of Biorobotics  
Delft University of Technology

**Keynote**



**James R. Cordy**

Prof. of Computer Science  
Queen's University at Kingston

**The  
Premier  
Conference on  
Model-Driven Engineering**

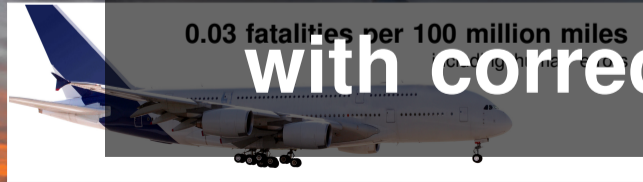
# Ariane V (1996)

A floating point cast bug,  
Throws overflow exception  
A decade of development,  
\$7B development budget,  
\$0.5B lost rocket & cargo,  
but ...

# Ariane V (2013)

A photograph of an Ariane V rocket launch. The rocket is ascending vertically from the bottom center, leaving a thick, dark plume of smoke and fire. The rocket's trajectory curves to the right, leaving a long, thin, white contrail that extends across the upper half of the frame. The background is a clear blue sky with some light clouds. The overall scene is captured during the day, with the sun low on the horizon, creating a warm, orange glow at the bottom of the image.

- **98 launches** since 1996
- **3 crashes** since 1996
- Only the first linked to a software bug  
(Is HW really more reliable?)
- Most recent launch: **Apr 5th, 2018**  
Have you heard about it?
- They never show you this slide ...



0.03 fatalities per 100 million miles

If we are  
Doing so well,  
Why are we still  
SO OBSESSED  
with correctness?



1.27 fatality per 100 million miles

including human failures



0.76 fatality per 100 million miles



# Lesson 1

**Don't drive your research  
by problems that are  
abstract (remote) for you**





- (variability) Bugs in the **Linux** kernel
- Bugs in the **Robot Operating System (ROS)**



# AGENDA

# Bugs are beautiful and fascinating

- **You are not a V&V researcher** if you don't touch real bugs and systems in your work.
- **Real data gives a rich research context**, enables a lot of fascinating work.
- Real **data does not exclude work in clean "lab conditions"** when appropriate. It supports it.



# What is Linux Kernel ?

Incredibly versatile operating system

GNU/Linux runs supercomputers and internet servers

68-98% web servers run on Linux

Android phones tablets smartTVs etc.



Cloud infrastructure



Routers, storage servers, entertainment systems, robots, IoT devices, ...

The most popular OS kernel on the planet!

Device Shipments, 2015



# Linux Kernel is very large

The source code has **700 million characters**, **21 million lines of code**  
(quick measurements on the Raspberry Pi version of Linux)

Boeing 747 has **6 million mechanical parts**, half of them simple fasteners

**Are humans able to understand the entire kernel?**



# Linux Kernel Moves Fast

- **4000 programmers** from **440 companies** contributed to the kernel  
(approximate numbers from 2015 only)
- **10,800** lines of code added, **5,300** removed, **1,875** modified  
**Every. Single. Day.** (on average)
- **Over 8 changes per second**
- **Is any human able to comprehend this evolution speed?**
- Incidentally, this makes it **impossible to verify** with current state of the art
- Nobody has **access to all hardware** on which others work
- Each **potentially breaks things for others**

# Let's indulge! Look! A bug!

Dereferencing uninitialized pointer causes Kernel crash

```
void pts_sb_from_inode(struct inode * inode)
```

```
1 #ifdef CONFIG_DEVPTS_MULTIPLE_INSTANCES
2 if (inode->i_sb->s_magic == ... ) ...
3 #endif
```

```
void pty_close(struct tty_struct * tty)
```

```
4 #ifdef CONFIG_UNIX98_PTYS
5 pts_sb_from_inode (tty->driver_data);
6 #endif
```

```
...
```

```
7 tty = kzalloc(sizeof (*tty), GFP_KERNEL);
8 pty_close (tty)
```

- **Domain knowledge**
- **Data flow**
- **Inter-procedural data-flow**
- **Pointers**
- **Nested structs**
- **[real bug] cross compilation unit and subsystem**
- **[real bug] function pointers (pty\_close)**

Iago Abal, Jean Melo, Stefan Stanculescu, Claus Brabrand, Márcio Ribeiro, Andrzej Wasowski: *Variability Bugs in Highly Configurable Systems: A Qualitative Analysis*. TOSEM 26(3): 10:1-10:34 (2018)

Bug 7acf6cd, see <http://vbdb.itu.dk/#bug/linux/7acf6cd>

# Let's look at another bug

## Control-flow

```
1 extern int preempt_count;
2
3 void tcp_twsk_destructor() {
4     #ifdef CONFIG_TCP_MD5SIG // ENABLED
5     preempt_count--;
6     #endif
7 }
8
9 void inet_twdr_hangman(long data) {
10     void (*fn)(); // function pointer
11     fn = (void (*)( )) data; // cast to funptr
12     fn(); // dynamic invocation
13 }
14
15 void __run_timers() {
16     long data = (long) &tcp_twsk_destructor;
17     int pc = preempt_count; // save
18     inet_twdr_hangman(data);
19     if (pc != preempt_count) BUG(); // check
20 }
```

Annotations for the code:

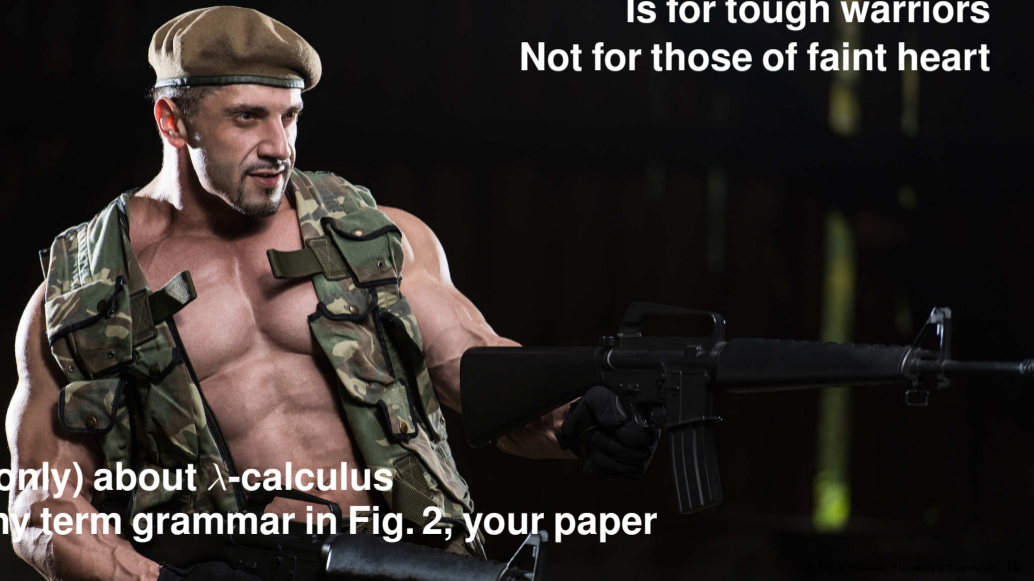
- Line 9: → (5)
- Line 10: (6)
- Line 11: (7)
- Line 12: (8) ~
- Line 15: ⇒ (1)
- Line 16: (2)
- Line 17: (3)
- Line 18: (4) →
- Line 19: → (11) ×

Annotations for the conditional block (lines 4-6):

- Line 4: ~ (9)
- Line 5: ↓
- Line 6: (10) →

- Unsafe casts help **generic programming** of data structures
- Type **casts**, pointers to **ints**; Do not lose **shapes** and **aliasing** info
- Function pointers used heavily (**OO**)
- Inter-procedural data-flow not possible without **control-flow**
- [elsewhere] **conditional struct components** (with **incompatible casts**)

**Lesson 2: Hunting bugs in software  
Is for tough warriors  
Not for those of faint heart**



**It's not (only) about  $\lambda$ -calculus  
or the tiny term grammar in Fig. 2, your paper**





**Warning!**  
**You may get dirty**

# A closer look at a bug



## index : kernel/git/stable/linux-stable.git

Linux kernel stable tree

summary refs log tree **commit** diff stats

author  Peter Hurley <peter@hurleysoftware.com> 2013-01-30 17:43:49 (GMT)  
committer  Greg Kroah-Hartman <gregkh@linuxfoundation.org> 2013-02-04 23:40:28 (GMT)  
commit [7acf6cd80b201f77371a5374a786144153629be8](#) (patch)  
tree [5222e9eca68f3b37ad62d1eb74966705f12d1f96](#)  
parent [16559ae48c76f1ceb970b9719dea62b77eb5d06b](#) (diff)

### pty: Fix BUG(s) when ptx\_open() errors out

If `ptmx_open()` fails to get a slave inode or fails the `pty_open()`, the `tty` is released as part of the error cleanup. As evidenced by the first BUG stacktrace below, `pty_close()` assumes that the linked `pty` has a valid, initialized `inode*` stored in `driver_data`.

Also, as evidenced by the second BUG stacktrace below, `pty_unix98_shutdown()` assumes that the master `pty`'s `driver_data` has been initialized.

- 1) Fix the invalid assumption in `pty_close()`.
- 2) Initialize `driver_data` immediately so proper `devpts fs` cleanup occurs.

Fixes this BUG:

```
[ 815.868844] BUG: unable to handle kernel NULL pointer dereference at 0000000000000028
[ 815.869018] IP: [] devpts_pty_kill+0x1c/0xa0
[ 815.869190] PGD 7c775067 PUD 79deb067 PMD 0
[ 815.869315] Oops: 0000 [#1] PREEMPT SMP
[ 815.869443] Modules linked in: kvm_intel kvm snd_hda_intel snd_hda_codec snd_hwdep snd_pcm snd_seq_midi
```





Linux

## Dereferencing uninitialized pointer causes Kernel crash

[View raw files](#) ▾

During the initialization of a UNIX98 pseudo-terminal by `ptmx_open`, a `tty_struct` structure is allocated. But before its pointer field `link->driver_data` is properly initialized, `ptmx_open` will try to allocate an inode structure for the PTY slave. If this allocation fails, some cleanup code must be executed to free the already allocated resources. Namely, `pty_close` will be called to release the previously opened tty, and this eventually dereferences `tty->link->driver_data`, which is assumed to have been already initialized.

But fixed by commit [7acf6cd80b2](#)

Parent commit tree [here](#)

[Related links](#) ▾

Type	<a href="#">use of variable before initialization (CWE 457)</a>
Config	<a href="#">UNIX98_PTYS</a> && <a href="#">DEVPTS_MULTIPLE_INSTANCES</a> (2nd degree)
C-features	FunctionPointers
Fix-in	<a href="#">code</a>

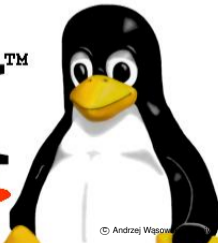
See <http://vbdb.itu.dk/>, and add your own bugs

# Subject Systems

As of December 2015

System	Domain	LOC	#Features	#Commits
Marlin	3D-printer firmware	0.04M	821	2,783
BusyBox	UNIX utilities	0.20M	551	13,878
Apache	Web Server	0.20M	681	27,677
Linux kernel	Operating system	12.70M	14,295	448,314

# Linux<sup>TM</sup>



Iago Abal. Claus Brabrand. Andrzej Wąsowski.

42 variability bugs in the Linux kernel: A qualitative analysis. ASE 2014 + TOSEM'18

# What do we see? Diversity!

## Linux

		<b>CWE ID</b>
<b>15</b>	<b>memory errors</b>	
4	null pointer dereference	476
3	buffer overflow	120
3	read out of bounds	125
2	insufficient memory	-
1	memory leak	401
1	use after free	416
1	write on read only	-
<b>8</b>	<b>compiler warnings</b>	<b>CWE ID</b>
5	uninitialized variable	457
1	unused function (dead code)	598
1	unused variable	563
1	void pointer dereference	-
<b>7</b>	<b>type errors</b>	<b>CWE ID</b>
5	undefined symbol	-
1	undeclared identifier	-
1	wrong number of args to function	-
<b>7</b>	<b>assertion violations</b>	<b>CWE ID</b>
5	fatal assertion violation	617
2	non-fatal assertion violation	617
<b>2</b>	<b>API violations</b>	<b>CWE ID</b>
1	Linux API contract violation	-
1	double lock	764
<b>1</b>	<b>arithmetic errors</b>	<b>CWE ID</b>
1	numeric truncation	197

## BusyBox

		<b>CWE ID</b>
<b>4</b>	<b>memory errors:</b>	
2	null pointer dereference	476
1	memory leak	401
1	use after free	416
<b>6</b>	<b>compiler warnings:</b>	<b>CWE ID</b>
3	unused variable	563
2	uninitialized variable	457
1	incompatible types	843
<b>5</b>	<b>type errors:</b>	<b>CWE ID</b>
3	undefined symbol	-
2	undeclared identifier	-
<b>3</b>	<b>logic errors:</b>	<b>CWE ID</b>
3	behavior violation	440

## **Lesson 3**

**Other researchers will be glad  
if you help them avoiding dirt.**

**You will help research quality  
in your field.**



- (variability) Bugs in the **Linux** kernel
- Bugs in the **Robot Operating System (ROS)**



# AGENDA



# Software engineering for robotics

## Why is it so hard?

- Programs for robots are not **input** → **output mappings**
- Intelligence, planning, mapping, vision, proximity, kinematics
- Operating under **uncertainty** and **lack of predictability**
- Huge diversity: **simple robots to very complex autonomous robots**, difficult to generalize (no one-size fits all)
- HW abstractions easily fall short
- Complex systems made of many components **parallel, distributed**
- **Reliability and safety** requirements
- **Complex vendor market** (OEMs, component providers, integrators, end users)



communication  
middleware  
with uniform  
API

100s  
integrated  
HW drivers  
& SW  
components

separates  
logics and  
algos from  
HW

infrastructure  
for test,  
simulation,  
logging

more tutorials  
than you can  
read; active  
friendly  
community

Linux, Python,  
C++, C, Java



# ROS



... an **open-source**, meta-operating system for your robot. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers.

# Does ROS matter?

Is ROS the OSS platform for robotics?

- In 2016: **1M+ unique page views** a month at [wiki.ros.org](http://wiki.ros.org)
- Yearly interest growth 21%
- Biggest reception in USA and China (comparable share)
- In July 2016: **300K+ visits** to [answers.ros.org](http://answers.ros.org) (your support channel)
- 17 new questions a day, **21K+ questions answered**
- More than **100K+ unique IP addresses** downloading ROS packages in July 2016
- **4400 papers citing** the ROS report





# Yes! Finally a pretty bug from ROS!

Context: argument processing code in main

## Fix indexing beyond end of array #167 New Issue

Merged tfoote merged 1 commit into `ros:indigo-devel` from `eric-wieser:patch-1` on Apr 2, 2016

Conversation 1 Commits 1 Checks 0 Files changed 1

Changes from all commits Jump to... +1 -1

### Fix indexing beyond end of array

melodic-devel (#167) 0.6.2 0.5.14

eric-wieser committed on Apr 2, 2016 commit 1957a441092d0bd90f115924f54831657ac6161f

```
tf2_ros/src/static_transform_broadcaster_program.cpp
```

70	70	{
71	71	if (strcmp(argv[7], argv[8]) == 0)
72	72	{
73	-	ROS_FATAL("target_frame and source frame are the same (%s, %s) this cannot work", argv[8], argv[9]);
73	+	ROS_FATAL("target_frame and source frame are the same (%s, %s) this cannot work", argv[7], argv[8]);
74	74	return 1;
75	75	}
76	76	

This ought to crash the node immediately, shouldn't it?

- Access out of bounds!
- **Wait!** the element outside is zero, so NULL
- A printf-like fun gets NULL, accesses to crash!
- **Wait!** ROS\_FATAL is a macro (call-by-name) → might not access
- Nested macros expand to `::ros::console::print`, several calls reach `vsnprintf` that accesses NULL and crashes
- **Wait!** In glibc `vsnprintf` typesets "(null)" for a NULL string, exits safely
- So no major bug! Just a misprinted error message.

# Above in the same file ...

```
34  int main(int argc, char ** argv)
35  {
36      //Initialize ROS
37      ros::init(argc, argv, "static_transform_publisher", ros::init_options::AnonymousName);
38      tf2_ros::StaticTransformBroadcaster broadcaster;
```

- The first thing a ROS node `main` does is to **initialize the framework** (line 37)
- Includes **processing of framework arguments** (usually quite a few, never zero)
- Arguments `argc` and `argv` are passed **by reference**
- Processed arguments are removed
  - `argv` is **re-sorted** so that only non-framework arguments are in front
  - `argc` is **decreased** accordingly
- What does it mean? **There was never any access out of bounds!**  
(this is just a minor formatting error on a failing execution)



# Lesson 4: Reproduce!



**Andrzej Wąsowski** @AndrzejWasowski · Mar 6



When you are a bug researcher, never claim that you understood what the bug really is, before you managed to actually reproduce the failure and see it yourself. The bug *\*is\** more nuanced than you think it is. [credits to [@zhoulaifu](#)]

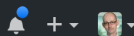




Search or jump to...



[Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)



[robust-rosin](#) / [robust](#)

[Unwatch](#) 5
 [Unstar](#) 7
 [Fork](#) 3

[Code](#)
[Issues](#) 16
 [Pull requests](#) 1
 [Wiki](#)
[Insights](#)

A dataset of 200 bugs in the Robot Operating System for BugZoo

[119](#) commits
 [4](#) branches
 [0](#) releases
 [5](#) contributors

Branch: [master](#)

[New pull request](#)

[Create new file](#)

[Upload files](#)

[Find file](#)

[Clone or download](#)

<a href="#">gavanderhoorn</a> Docs should go in 'doc' dir.	Latest commit <a href="#">f0b8b47</a> 2 days ago
<a href="#">care-o-bot</a>	Updated Dockerfile and BugZoo files to use one-fork-many-branch model (...) 15 days ago
<a href="#">confidential</a>	moved confidential bugs 2 months ago
<a href="#">doc</a>	Docs should go in 'doc' dir. 2 days ago
<a href="#">geometry2</a>	Updated Dockerfile and BugZoo files to use one-fork-many-branch model (...) 15 days ago
<a href="#">kobuki</a>	Added kobuki <a href="#">b166c93</a> (#49) 12 days ago
<a href="#">mavros</a>	Removed L1 and L2, and made L3 the norm ( <a href="#">fixes #17</a> ) a month ago

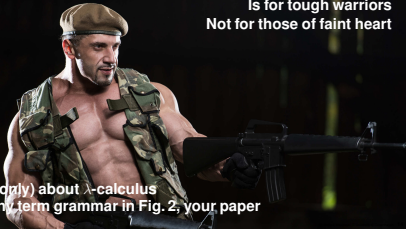
## Lesson 1

Don't drive your research  
by problems that are  
abstract (remote) for you

## Lesson 3

Other researchers will be glad  
if you help them avoiding dirt.  
You will help research quality  
in your field.

Lesson 2: Hunting bugs in software  
Is for tough warriors  
Not for those of faint heart

A muscular man with a beard, wearing a military cap and a tactical vest, is shown from the chest up. He is holding a rifle and looking off to the side with a serious expression. The background is dark and blurry.

It's not (only) about  $\lambda$ -calculus  
or the tiny term grammar in Fig. 2; your paper

## Lesson 4: Reproduce!



Andrzej Wąsowski @AndrzejWasowski · Mar 6

When you are a bug researcher, never claim that you understood what the bug really is, before you managed to actually reproduce the failure and see it yourself. The bug "is" more nuanced than you think it is. [credits to @zhoulailu]



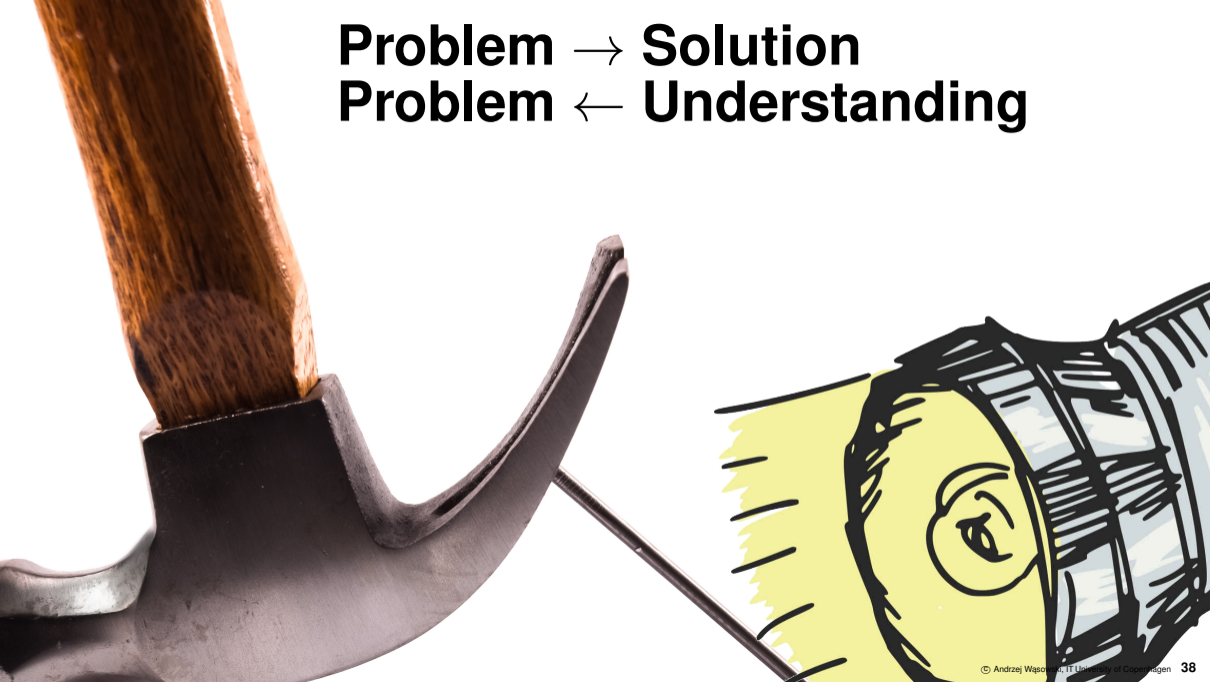


**Part 2: Bug Hunting**  
**Is for tough warriors,**  
**But even warriors need the right weapons**

**Real Problems will take you  
to interesting work**

**Sometimes not  
where you planned to go**

**Problem** → **Solution**  
**Problem** ← **Understanding**



- Problem → solution: **EBA bug finder**
- Problem ← understanding: **Why variability bugs appear?**
- Problem ↔ research methods: **How to collect bug data?**



# AGENDA

# Another Bug: double lock

```
1 void inode_get_rsv_space(struct inode *inode)
2     if (*) return;
3     spin_lock(&inode->i_lock); // 2nd lock
4     spin_unlock(&inode->i_lock);
5 }
6
7 void add_dquot_ref(struct inode *inode) {
8     spin_lock(&inode->i_lock); // 1st lock
9     if (*) {
10         spin_unlock(&inode->i_lock);    7 -
11         return;
12     }
13     inode_get_rsv_space(inode); // call
14     spin_unlock(&inode->i_lock);
15 }
```

```
1 @@
2 expression E;
3 @@
4 *   spin_lock(E);
5     ... when != spin_unlock(E);
6 *   spin_lock(E);
```

- Coccinelle matches patterns over traces
- Supports CPP, efficient,
- **Integrated** into the kernel build system
- Intra procedural, unaware of aliasing

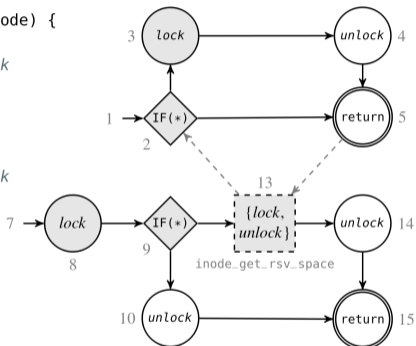




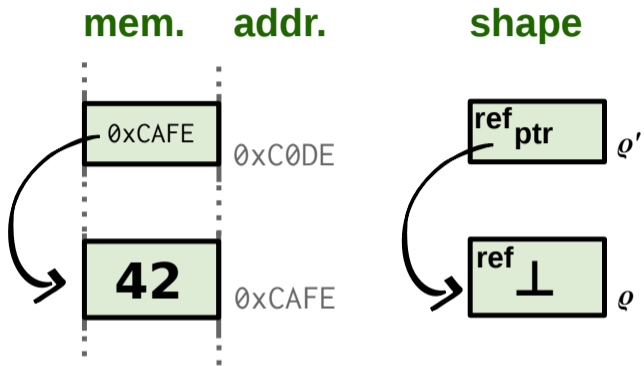
# Welcome to EBA!

## Effect-Based Analyzer

```
1 void inode_get_rsv_space(struct inode *inode) {
2   if (*) return;
3   spin_lock(&inode->i_lock); // 2nd lock
4   spin_unlock(&inode->i_lock);
5 }
6
7 void add_dquot_ref(struct inode *inode) {
8   spin_lock(&inode->i_lock); // 1st lock
9   if (*) {
10    spin_unlock(&inode->i_lock);
11    return;
12  }
13  inode_get_rsv_space(inode); // call
14  spin_unlock(&inode->i_lock);
15 }
```



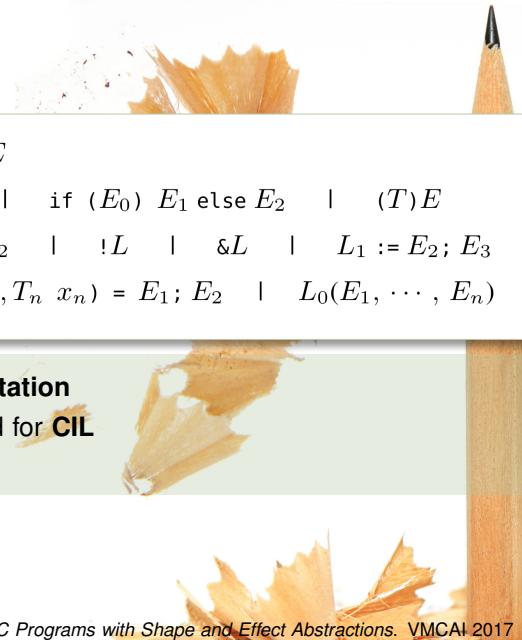
- For this to work we need to know about **effects**
- And about memory objects (to detect **aliasing**)



Shape term:  $\text{ref}_{\rho'} \text{ ptr } \text{ref}_{\rho} \perp$   
**Polymorphic** in regions and shapes (abstraction)

# Syntax

Used to show the type system


$$\begin{aligned} \textit{l-value expressions } L & : x \mid f \mid *E \\ \textit{r-value expressions } E & : n \mid E_1 + E_2 \mid \text{if } (E_0) E_1 \text{ else } E_2 \mid (T)E \\ & \mid \text{new } x : T = E_1; E_2 \mid !L \mid \&L \mid L_1 := E_2; E_3 \\ & \mid \text{fun } T f(T_1 x_1, \dots, T_n x_n) = E_1; E_2 \mid L_0(E_1, \dots, E_n) \end{aligned}$$

- We use a specialized language only for **presentation**
- The type system is formalized and implemented for **CIL**
- Entire C is **translatable** to CIL

# Inference of Shapes & Effects

$\vdash : \text{ENV} \times \text{EXP} \times \text{SHAPE} \times \text{EFFECT}$

$$\text{[FETCH]} \frac{\Gamma \vdash_L L : \text{ref}_\rho Z \ \& \ \varphi}{\Gamma \vdash_E !L : Z \ \& \ \varphi \cup \{\text{read}_\rho\}}$$

$$\text{[ASSIGN]} \frac{\Gamma \vdash_L L : \text{ref}_\rho Z \ \& \ \varphi_1 \quad \Gamma \vdash_E E_1 : Z \ \& \ \varphi_2 \quad \Gamma \vdash_E E_2 : Z' \ \& \ \varphi_3}{\Gamma \vdash_E L := E_1; E_2 : Z' \ \& \ \varphi_1 \cup \varphi_2 \cup \{\text{write}_\rho\} \cup \varphi_3}$$

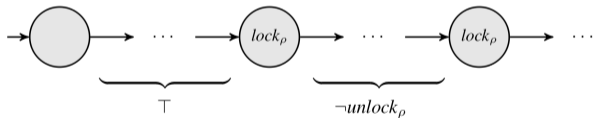
- Formalized and implemented for entire C
- Including spec. of selected kernel functions, e.g:

$$\begin{array}{ll} \text{spin\_lock} : & \text{ref}_{\rho_1} \text{ ptr ref}_{\rho_2} \zeta \xrightarrow{\text{lock}_{\rho_2}} \perp \\ \text{spin\_unlock} : & \text{ref}_{\rho_1} \text{ ptr ref}_{\rho_2} \zeta \xrightarrow{\text{unlock}_{\rho_2}} \perp \end{array}$$

# Bug Pattern Definitions

- Formalize **bug patterns in CTL** with nominals over effects
- A simple **reachability checker** finds paths matching a formula
- **E.g. double lock** = lock, then take same lock again without unlocking

$$\top \text{ EU } (lock_\rho \wedge \text{ EX } (\neg unlock_\rho \text{ EU } lock_\rho))$$



double free

$$\top \text{ EU } (free_\rho \wedge \text{ EX } (\neg alloc_\rho \text{ EU } free_\rho))$$

memory leak

$$\top \text{ EU } (alloc_\rho \wedge \text{ EX EG } \neg free_\rho)$$

use before initialization

$$\neg init_\rho \text{ EU } use_\rho$$

# Does this work?

## Precision (false positives), new bugs

- **Nine thousand files** in drivers analyzed (you do get dirty!)
- **9 reports for 9K lines** is not a lot of noise
- Each reported **bug classified** as either a true or a false positive.
- Still a lot of work to **filter out false positives** (you get dirty!)
- You talk to devs: they want you to **fix bugs!** (you may get dirty!)
- We right now have **14 new bugs** and **>=5 fixed** in the Linux kernel project (some in the main tree already)

	EBA	SMATCH	COCCINELLE
Bugs found	<b>4</b>	0	0
False positives	<b>5</b>	8	6
TIME (minutes)	23	16	<b>2</b>



# Does this work?

Experimental Evaluation, **time** in seconds, **recall** on historical bugs



hash ID	depth	E	S	C
1173ff0	0	<b>0.6</b>	<b>1.3</b>	<b>0.1</b>
149a051	0	<del>0.7</del>	<del>0.6</del>	<del>0.3</del>
16da4b1	0	<b>0.4</b>	<b>0.8</b>	<b>0.1</b>
344e3c7	0	<b>0.7</b>	<b>1.3</b>	<del>0.1</del>
2904207	0	<b>5.8</b>	<b>2.0</b>	<del>2.8</del>
59a1264	0	<b>0.2</b>	<b>0.6</b>	<b>0.1</b>
5ad8b7d	0	<b>0.6</b>	<b>3.4</b>	<b>0.1</b>
8860168	0	<b>0.7</b>	<b>1.0</b>	<b>0.1</b>
a7eef88	0	<b>0.6</b>	<b>1.2</b>	<b>0.2</b>
b838396	0	<b>3.3</b>	<b>2.8</b>	<b>1.1</b>
ca9fe15	0	<b>0.4</b>	<del>0.7</del>	<b>1.8</b>
e1db4ce	0	<b>0.4</b>	<b>1.1</b>	<b>0.2</b>
e50fb58	0	<b>0.5</b>	<b>0.9</b>	<b>0.1</b>
023160b	0	<b>1.0</b>	<b>2.6</b>	<del>0.1</del>
09dc3cf	0	<b>1.2</b>	<del>1.4</del>	<del>0.1</del>
0adb237	0	<b>1.1</b>	<b>1.5</b>	<b>0.2</b>
0e6f989	0	<b>0.4</b>	<b>1.0</b>	<b>0.3</b>

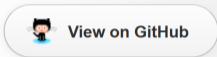
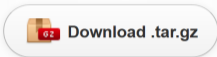
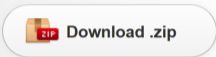
(17 historical bugs, intra-proc, double-lock, in Linux kernel, biased against EBA)

Bug	depth	E	S	C
00dfff7	2	<b>5.0</b>	1.5	<del>0.1</del>
5c51543	2	<b>2.3</b>	1.5	<del>0.3</del>
b383141	2	<del>6.1</del>	<del>2.9</del>	<del>0.3</del>
1c81557	1	<b>5.0</b>	1.9	<del>0.6</del>
328be39	1	<b>8.9</b>	1.7	<del>0.2</del>
5a276fa	1	<del>0.9</del>	1.2	<del>0.2</del>
80edb72	1	<del>6.3</del>	2.1	<del>0.7</del>
872c782	1	<b>1.7</b>	2.8	1.9
d7e9711	1	<b>21</b>	1.3	2.7

(9 historical bugs, inter-proc, double-lock, in kernel, biased against EBA)

# EBA

An effective bug finder for C



EBA is a prototype tool to find non-trivial resource manipulation bugs in C programs, at compile-time, and super-fast.

In its few months of existence, EBA has found several double-lock bugs in Linux 4.7–4.10 releases (i.e. in code that has passed code reviews). All the following bugs are caught by EBA in a matter of *seconds*:

- [HSI: cmt\\_speech: Fix double spin\\_lock](#)
- [usb: gadget: pch\\_udc: reorder spin\\_\[un\]lock to avoid deadlock](#)
- [ath10k: fix deadlock while processing rx\\_in\\_ord\\_ind \[1\]](#)
- [net: ethernet: ti: cpdma: fix lockup in cpdma\\_ctlr\\_destroy\(\) \[2\]](#)
- [libceph: ceph\\_build\\_auth\(\) doesn't need ceph\\_auth\\_build\\_hello\(\)](#)
- [\[PATCH\] Fix: scsi: megaraid: reduce the scope of pending-list lock to avoid double lock](#)
- [iommu/vt-d: Fix dead-locks in disable\\_dmar\\_iommu\(\) path \[3\]](#)
- [Re: Potential double-lock BUG in drivers/tty/serial/sh-sci.c \(Linux 4.9\)](#)
- [Potential deadlock BUG in drivers/net/wireless/st/cw1200/sta.c \(Linux 4.9\) \[4\]](#)
- [Potential deadlock BUG in Linux 4.9 drivers/dma/coh901318.c \[4\]](#)
- [\[PATCH\] \[media\] pctv452e: fix double lock bug \[4\]](#)
- [Potential double-lock BUG in drivers/infiniband/core/umem\\_odp.c \(Linux 4.9-rc7\) \[4\]](#)
- [dmaengine: pl330: fix double lock](#)
- [cros\\_ec: Fix deadlock when EC is not responsive at probe \[3\]](#)



Advertisement

**We are hiring!**

**After all, lambdas, types,  
and model checking  
are useful for  
solving real problems.**

**(but we wouldn't know without  
trying on real problems)**

- Problem → solution: **EBA bug finder**
- Problem ← understanding: **Why variability bugs appear?**
- Problem ↔ research methods: **How to collect bug data?**



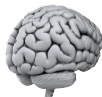
# AGENDA

# What do we find in the variability bugs ?

## A quick extract

- Bugs appear in **unanticipated configurations**
  - The programmer did not think about other configurations
  - Essentially all 98 VBDB bugs appear to be such ...
- Bugs may **involve non-locally defined features** (defined in other subsystems)
  - 30 out of 43 Linux bugs in VBDB have this feature
- The interviewed developer: cross-cutting features are a frequent source of problems; Developers are often **experts only in a particular subsystem.**

*“Code cluttered with ifdefs is difficult to read and maintain. Don’t do it. Instead put your ifdefs in a header, and conditionally define ‘static inline’ functions or macros, which are used in the code.” [submitting patches]*



# An interesting pattern with negative conditions

(a presence conditions for a bug to trigger)

structure of the sufficient presence condition for the bug  
pattern frequency in our bug collection

## 49 some-enabled

21	$a$
21	$a \wedge b$
6	$a \wedge b \wedge c$
1	$a \wedge b \wedge c \wedge d \wedge e$

## 45 some-enabled-one-disabled

20	$\neg a$
20	$a \wedge \neg b$ <i>incl.: <math>(a \vee a') \wedge \neg b</math></i>
4	$a \wedge b \wedge \neg c$
1	$a \wedge b \wedge c \wedge d \wedge \neg e$

## 4 other configurations

1	$\neg a \wedge \neg b$
1	$a \wedge \neg b \wedge \neg c$
2	$a \wedge \neg b \wedge \neg c \wedge \neg d \wedge \neg e$

- In Linux commit 60e233a5660 function `add_uevent_var` with `HOTPLUG` disabled overflows a buffer
- Originally we spinned this for testing and sampling
- **Second thought:** isn't this a symptom of forgetting to "mentally" enable/disable a feature?
- The kernel developer: you hardly can think of more than 5 involved configs (features) when coding, debugging or profiling.



**What are  
the cognitive  
challenges?**

This program contains a simple coding bug

How would you debug it?

Where is the bug?

What configurations contain the bug?

```
1 import java.util.Random;
2
3 public class Http {
4     String subject = null;
5     int totalLength = 600;
6     final int HTTP_UNAUTHORIZED = 401;
7     final int HTTP_NOT_IMPLEMENTED = 501;
8     String REQUEST_GET = "GET";
9
10    public void sendHeaders(int responseNum) {
11        int buf = 0;
12        buf = totalLength - responseNum;
13        subject = "response header";
14
15        if (subject.isEmpty())
16            subject = "Void response";
17
18        System.out.println("Done");
19    }
20
21    private void handleIncoming(String requestType) {
22        boolean http_unauthorized = new Random().nextBoolean();
23        if (http_unauthorized)
24            sendHeaders(HTTP_UNAUTHORIZED);
25
26        if (!requestType.equals(REQUEST_GET))
27            sendHeaders(HTTP_NOT_IMPLEMENTED);
28    }
29
30    public static void main(String[] args) {
31        Http http = new Http();
32        http.handleIncoming("POST");
33    }
34 }
```

# Controlled Experiment I

## RQs

How does the **degree of variability** affect ...

- ... the **time** of bug finding?
- ... the **accuracy** of bug finding?

## Cross-over Design

- Programs:** 3 programs from 3 systems
  - Linux, open source, 12MLOC/13K features
  - Busy Box, open source, 204KLOC/600 features
  - Best Lap, Commercial game, 15 KLOC
- Subjects:** N=69 [31 × Msc+32 × Phd+6 × post-doc]
- Task:** find the bug
- Metrics:** time and accuracy
- Small, fit on screen—**no scrolling** (25-35 LOC)
- Bugs:** uninitialized var, null ptr deref, assert violation
- Deactivate features in NO/LO version, keep the bug



P1



P2



P3

NO ( $|F| = 0$ )

LO ( $|F| = 1$ )

HI ( $|F| = 3$ )

<pre>public class Msc1 {     boolean use_mail = false;     boolean SPIN_SUPPORT = true;     public String handleMessage() {         String m = Message;         boolean type = true;         if (use_mail)             type = false;         if (type)             m = "Destination address required";         return handleMessage();     }     private void handleMessage(String requestType) {         use_mail = true;         SPIN_SUPPORT = false;         return "done";     }     public static void main(String[] args) {         Message m = new Message();         handleMessage(m);     } }</pre>	<pre>public class Msc1 {     boolean use_mail = false;     boolean SPIN_SUPPORT = true;     public String handleMessage() {         String m = Message;         boolean type = true;         if (use_mail)             type = false;         if (type)             m = "Destination address required";         return handleMessage();     }     private void handleMessage(String requestType) {         use_mail = true;         SPIN_SUPPORT = false;         return "done";     }     public static void main(String[] args) {         Message m = new Message();         handleMessage(m);     } }</pre>	<pre>public class Msc1 {     boolean use_mail = false;     boolean SPIN_SUPPORT = true;     public String handleMessage() {         String m = Message;         boolean type = true;         if (use_mail)             type = false;         if (type)             m = "Destination address required";         return handleMessage();     }     private void handleMessage(String requestType) {         use_mail = true;         SPIN_SUPPORT = false;         return "done";     }     public static void main(String[] args) {         Message m = new Message();         handleMessage(m);     } }</pre>
<pre>public class HTTP {     String subject = null;     int statusCode = 200;     boolean use_cookies = false;     boolean use_authentication = true;     boolean use_https = false;     boolean use_https_certificate = true;     boolean use_https_certificate_key = true;     private void sendHttpRequest(String requestType) {         int status = 200;         if (use_authentication)             subject = "Authentication";         if (statusCode != 200)             subject = "Response";         System.out.println(status);     }     private void handleHttpRequest(String requestType) {         boolean use_authentication = true;         boolean use_https_certificate = true;         boolean use_https_certificate_key = true;         if (requestType.equals("REQUEST_GET"))             sendHttpRequest("GET /index.html");         if (requestType.equals("REQUEST_POST"))             sendHttpRequest("POST");     }     public static void main(String[] args) {         HTTP http = new HTTP();         http.handleHttpRequest("POST");     } }</pre>	<pre>public class HTTP {     String subject = null;     int statusCode = 200;     boolean use_cookies = false;     boolean use_authentication = true;     boolean use_https = false;     boolean use_https_certificate = true;     boolean use_https_certificate_key = true;     private void sendHttpRequest(String requestType) {         int status = 200;         if (use_authentication)             subject = "Authentication";         if (statusCode != 200)             subject = "Response";         System.out.println(status);     }     private void handleHttpRequest(String requestType) {         boolean use_authentication = true;         boolean use_https_certificate = true;         boolean use_https_certificate_key = true;         if (requestType.equals("REQUEST_GET"))             sendHttpRequest("GET /index.html");         if (requestType.equals("REQUEST_POST"))             sendHttpRequest("POST");     }     public static void main(String[] args) {         HTTP http = new HTTP();         http.handleHttpRequest("POST");     } }</pre>	<pre>public class HTTP {     String subject = null;     int statusCode = 200;     boolean use_cookies = false;     boolean use_authentication = true;     boolean use_https = false;     boolean use_https_certificate = true;     boolean use_https_certificate_key = true;     private void sendHttpRequest(String requestType) {         int status = 200;         if (use_authentication)             subject = "Authentication";         if (statusCode != 200)             subject = "Response";         System.out.println(status);     }     private void handleHttpRequest(String requestType) {         boolean use_authentication = true;         boolean use_https_certificate = true;         boolean use_https_certificate_key = true;         if (requestType.equals("REQUEST_GET"))             sendHttpRequest("GET /index.html");         if (requestType.equals("REQUEST_POST"))             sendHttpRequest("POST");     }     public static void main(String[] args) {         HTTP http = new HTTP();         http.handleHttpRequest("POST");     } }</pre>
<pre>public class GameOver {     private int totalScore = 0;     private int time = 0;     private float time_min = 2;     private float perfect_score = 1;     private void startGame() {         if (score &gt; 0) {             totalScore = score;         } else {             totalScore = 0;         }     }     private void resetGame() {         totalScore = 0;         time = 0;         perfect_score = 1;         totalScore = 0;     }     public static void main(String[] args) {         GameOver go = new GameOver();         go.startGame();     } }</pre>	<pre>public class GameOver {     private int totalScore = 0;     private int time = 0;     private float time_min = 2;     private float perfect_score = 1;     private void startGame() {         if (score &gt; 0) {             totalScore = score;         } else {             totalScore = 0;         }     }     private void resetGame() {         totalScore = 0;         time = 0;         perfect_score = 1;         totalScore = 0;     }     public static void main(String[] args) {         GameOver go = new GameOver();         go.startGame();     } }</pre>	<pre>public class GameOver {     private int totalScore = 0;     private int time = 0;     private float time_min = 2;     private float perfect_score = 1;     private void startGame() {         if (score &gt; 0) {             totalScore = score;         } else {             totalScore = 0;         }     }     private void resetGame() {         totalScore = 0;         time = 0;         perfect_score = 1;         totalScore = 0;     }     public static void main(String[] args) {         GameOver go = new GameOver();         go.startGame();     } }</pre>



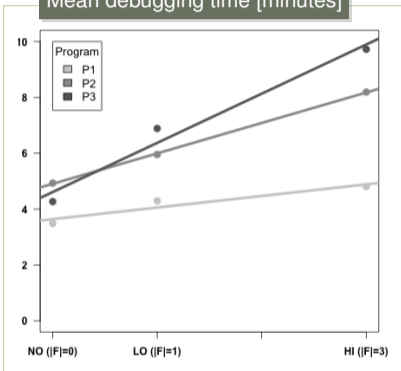
**Your guess?**

**How does the time change  
from P-NO via LO to HI?**

# Bug Finding Time Increases Linearly with $|F|$

Variance of the time is amplified by variability

Mean debugging time [minutes]

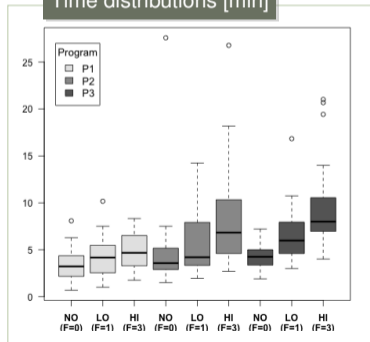


- Variability makes debugging **harder** but **not terribly so**
- Might explain why FOSD/SPLE works !!!
- Humans “reason **family-based**” (at least until  $|F|=3$ )
- We’ve got some innate ability for **meta-programming** !

*I tried to keep all different paths in mind, but it was especially difficult with multiple colors [HI]*

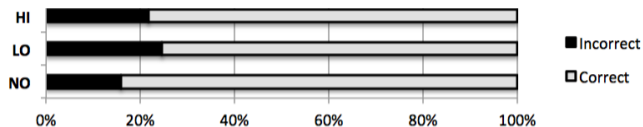
- Obvious **consequence of the former**, but meaningful
- Variability **amplifies** differences in **bug finding competences**

Time distributions [min]

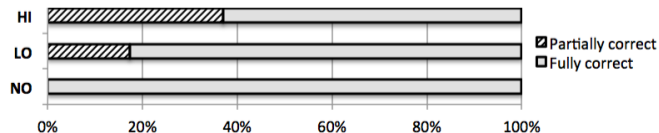


# Finding Variability Bugs is Easy

Linking them to configurations is harder



- Most developers **correctly identify bugs** regardless of the variability degree
- Many **fail to identify the of erroneous configurations**; give too few or too many (!)
- Precision **decreases with increasing variability** degree
- We can expect this to be harder in presence of **constraints** (feature models)
- Speculation: devs **don't think about configurability all the time**; Afterthought, **reversed staging**
- We likely make the **same mistakes when coding** (vbdb!) as when debugging;





**What's  
in your head  
when you  
work with  
variability?**

# Tobii



Jean Melo. Fabricio Batista Narcizo. Dan Witzner Hansen. Claus Brabrand. Andrzej Wąsowski.  
*Variability through the eyes of the programmer.* ICPC 2017

# Controlled Experiment II

## ■ 2 buggy programs

- 1 derived from Busy Box, 1 derived from Best Lap
- Same programs as before, but using #ifdefs not colors
- We wanted to see whether people look at #ifdefs

## ■ Same two bugs

- Null pointer dereference, assertion violation
- Both bugs in HI (3 features) and NO (0 features) versions

## ■ N=20 subjects, 7 BSc, 1 MSc, 7 PhD, 5 post-doc

## ■ Task: What is the bug? where is the bug? and in which configurations it appears? while we track your gaze

## ■ Latin square: subject solves two tasks order on different programs (randomized order and assignments)

## ■ No time limit (effectively 4–12 minutes per task)

```
#ifdef HI
public class HI {
    ...
}
#endif

#ifdef NO
public class NO {
    ...
}
#endif

private void handleIncoming(String requestType) {
    ...
}

public static void main(String[] args) {
    ...
}
```

```
public class GameScreen {
    private int totalScore = 0;
    private int penalty;
    private final int PERFECT_CURVE_AND_STRAIGHT = 7;
    private final int TIMM Bonus = 1;

    private void setScore(int score) {
        totalScore = (score * 0.7 + score);
    }

    private void setPenalty(int penalty) {
        this.penalty = penalty;
    }

    private void computeLevelScore() {
        assert totalScore > 0;
        totalScore = PERFECT_CURVE_AND_STRAIGHT;
        totalScore += TIMM Bonus;
        totalScore -= penalty;
        setScore(totalScore);
        assert totalScore < 8;
    }

    public static void main(String[] args) {
        GameScreen game = new GameScreen();
        game.setPenalty(1);
        game.computeLevelScore();
    }
}
```

# Analyzing the Eye Tracker Data

Eye tracker gives a **fixation sequence**: triples  $(x, y, t)$  of locations and time stamps

Connect-the-dots diagram

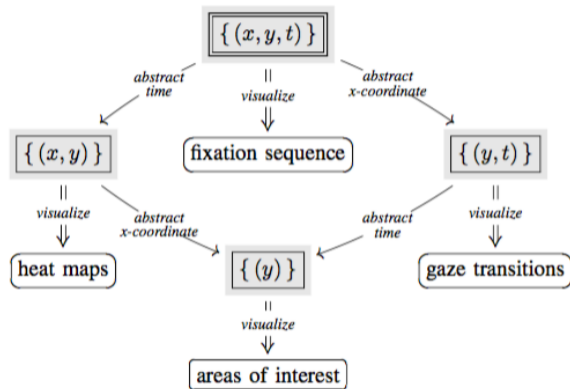
```
import java.util.Random;

public class Http {
    String object = null;
    int callDelay = 500;
    final int HTTP_UNAUTHORIZED = 401;
    final int HTTP_NOT_IMPLEMENTED = 501;
    #ifdef CONFIG_FEATURE_HTTP_CGI
    String REQUEST = "CGI";
    #endif

    public void sendHeaders(int responseNum) {
        #ifdef CONFIG_FEATURE_HTTP_CGI
        int bufLength = responseNum;
        StringBuffer responseHeader =
            new StringBuffer(bufLength);
        responseHeader.append("Content-type: text/html");
        System.out.println(responseHeader);
        #endif
    }

    private void handleRequest(String requestType) {
        #ifdef CONFIG_FEATURE_HTTP_BASIC_AUTH
        boolean httpUnauthorized = new Random().nextBoolean();
        if (httpUnauthorized) {
            sendHeader(HTTP_UNAUTHORIZED);
        }
        #ifdef CONFIG_FEATURE_HTTP_CGI
        if (requestType.equals("CGI")) {
            sendHeader(HTTP_NOT_IMPLEMENTED);
        }
        #endif
    }

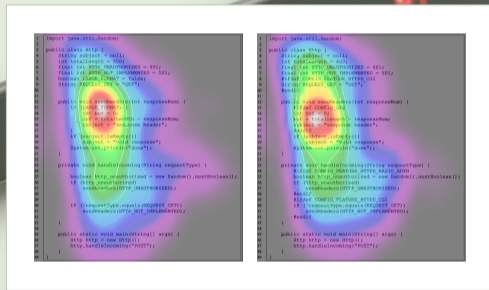
    public static void main(String[] args) {
        Http http = new Http();
        http.handleRequest("POST");
    }
}
```



# Variability Attracts Attention (or Confusion)

- **Observation:** Variability appears to increase debugging time of the areas of the program that contain variability.
  - **Time doubles** from no to hi for both programs
  - Consistent with the previous study, but now for **#ifdefs not colors**
  - **Heatmaps similar** (KL divergence), but there is a small shift

area of interest		variability		increase factor
lines	area	without	with	
4-9	fields	26 s	58 s	2.2 x
12-21	sendHeaders	63 s	120 s	1.9 x
23-33	handleIncoming	56 s	98 s	1.8 x
35-38	main	8.2 s	5.3 s	0.7 x
Σ	<i>all four areas</i>	153 s	281 s	1.8 x



**Observation:** Time increases for fragments without variability in proximity of code fragments that do contain variability

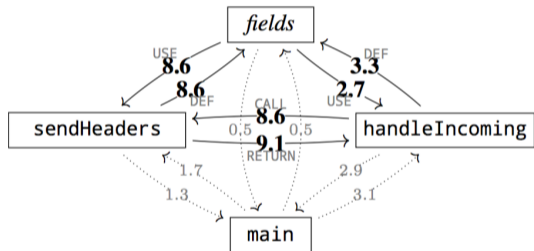


# Variability Intensifies Eye Movements (or Confusion)

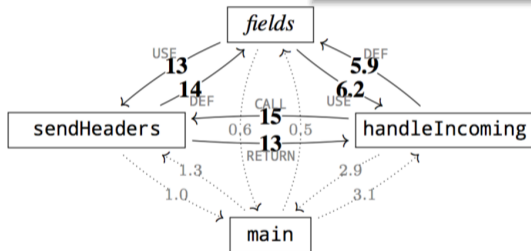
## Observation

Variability appears to **increase the number of gaze transitions** between definitions-usages for fields and call-returns for methods.

Could alternative code organization help?



(a) Without variability.



(b) With variability.

Fig. 10: Average number of gaze transitions (eye switches) between the different elements of program P.

- Problem → solution: **EBA bug finder**
- Problem ← understanding: **Why variability bugs appear?**
- Problem ↔ research methods: **How to collect bug data?**



# AGENDA

# We need realistic bug benchmarks

## Rule 1

- Most interesting questions about programs are **undecidable**.
- More theoretical misbehaviours than **what occurs in real systems**
- Bugs not caused by meticulously **contrived** computations and circumstances.
- But by simple misconceptions: omissions, misspellings, confusion, miscommunication, misunderstandings, misuse of a library, or simply lack of information about the intended behavior of the system
- Human **cognition functions that determine the errors**
- Historical bugs **approximate** problems introduced by human cognition
- Need benchmarks reflecting real problems to **guide research**

# Selection of bugs should be unbiased

## Rule 2

- Avoid **sampling bias**.
- You can limit the **bug category** (ROS bugs, Linux bugs, variability bugs, concurrency bugs, etc.)
- But do **question correctness of sampling within the category**
- Do the bugs collected represent **anything more than your collection**
- Using a particular **tool introduces bias**

# Reproducible bugs, reproducible benchmarks!

## Rule 3

- It should be possible for another researcher **to recreate a reasonably similar benchmark** by following your method
- **Each bug** should be reproducible
- Recall what are the **risks** of misunderstanding the bug
- **Hard** to achieve for flaky non-testable bugs (concurrency!), bugs relying on hardware that you don't have etc. For instance:
  - Robotics is diverse: actuators, sensors, control, distribution, communication, planning, simulation and visualization, diagnostics, perception (incl. object and collision detection), HRI (human-robot interaction), SLAM (Simultaneous Localization and Mapping), artificial intelligence
  - Specialist skills required!
  - Robotics software depends on hardware and a physical environment.
  - hardware might be unavailable, intermittent environmental conditions irreproducible
- Consequently, **not all** historical bugs will be reproducible

# Restoration of buggy version of the system code

## Rule 4

- Get it from the code repository (**git**)
- Use the **time of reporting or fixing** the bug (travel in git history)
  - The bug has likely been fixed since it was reported which means that it can be **only reproduced on an older snapshot**
  - ROS is a **moving target** with ever changing properties which means that the newest ROS version is likely to prevent reproduction in a **repeatable** manner.
- The problem is however not only getting the file with the bug at the right time. **You need the entire system source code.**
- in Robust, we obtain the **entire source distribution** of ROS from a given point in time for each bug (compute only dependencies of the buggy package; credits: rosinstall generator)
- Really irritating to hit **more than one bug** in this snapshot (for instance a build problem prevents reproducing a dynamic problem)

# Restoration of historical development ecosystem

## Rule 5

- Compilers and interpreters (**for all the languages**), runtime library, build system, operating system and all dependent distribution libraries.
- In ROBUST we use **docker containers** (one per bug) in which the environment for a bug is re-established.
- We use **bugzoo**\* to manage the containers uniformly.
- Code **repositories and branches disappear**
- In ROBUST we **fork all involved repositories** (for the source of buggy packages).  
**We store all dependencies in a docker container** and store the container on dockerhub.
- We keep a **redundant copy** at the university.

# Facilitate automatic test reproduction

## Rule 6

- In the fork we develop a test case (bug witness, **regression test**)
- Make available **both** in the textbfixed and the **broken** branch

<p><b><u>buggy code:</u></b> <math>C</math> (code <math>C</math>, with the bug)</p>	<p><b><u>test case, <math>\varphi</math>:</u></b> <math>C \not\models \varphi</math> (code <math>C</math> <i>fails</i> test case <math>\varphi</math>)</p>
<p><b><u>fixed code:</u></b> <math>C' = \text{fix}(C')</math> (code <math>C'</math>, without the bug)</p>	<p><b><u>test case, <math>\varphi</math>:</u></b> <math>C' \models \varphi</math> (code <math>C'</math> <i>passes</i> test case <math>\varphi</math>)</p>

- We provide **scripts to manipulate the container state**: build, test, fix/unfix
- The test case has to be **non-intrusive**



# Add the test-case non-invasively

## Rule 7

- The test case **contaminates** the original historic source
- Because we require realism, the **invasion shall be minimal**
- Modifying existing code should be avoided (but often impossible)
- The legacy code often exhibits **bugs outside the testable surfaces**
- We use a number of **patterns** to minimize the invasion:
  - Inject short **assertions** (if the property cannot be tested on an output of a function),
  - **Determinize** control-flow (if the bug is not reproducible with decent probability),
  - **Mock hardware** components with software, etc.


# Document context meta-data for researchers

## Rule 8

- Bugs are deeply embedded in **intricate** functionality, architecture, and other idiosyncratic aspects of the subject system.
- This creates a very **high entry barrier for researchers** and **inhibits usefulness** of a benchmark.
- Meta-data lets the users **understand the problem fast**
- ROBUST and VBDB record detailed meta-data as **human-readable descriptions** to facilitate this usage

# Benchmark design rules

- 1 We need **realistic benchmarks**
- 2 Selection of bugs for a benchmark should be **unbiased**
- 3 Make **benchmarks reproducible**, and **reproduce bugs** in them
- 4 Restore historical **system source code**
- 5 Restore historical **development ecosystem**
- 6 Facilitate **automatic test reproduction**
- 7 Add the regression test possibly **non-invasively**
- 8 Document context **meta-data**

- 
- Problem → solution: **EBA bug finder**
  - Problem ← understanding: **Why variability bugs appear?**
  - Problem ↔ research methods: **How to collect bug data?**



# AGENDA

# Pasteur's Quadrant

Post-Conclusion

