How I learned to stop fixing code

. . . over and over again

INSTITUT FÜR ANGEWANDTE INFORMATIK

Zuverlässigkeit:	Die durchgehende Funktionserbringung eines Systems für einen festen Zeitraum
Security:	Schutz des Systems vor Angreifern
Safety:	Schutz von Personen und Dingen vor dem System

KIT

Kernforschungszentrum Karlsruhe
Nuclear Research Center Karlsruhe

I

Bugs

What is a Bug?

A piece of Code . . .

. . . exhibiting behaviour not intended by the developer

. . . not always exhibiting the intended behaviour

. . . which can quickly turn into the above

. . . that is so convoluted that it's basically the above

Static Analyses

- detect problematic patterns
- detect common bugs
- enforce code style
- never tired, lazy, overworked

History

- Ada (1977-1983)
- Lint (1977-1979)
- MISRA (1998-*now*)
- Rust (2009-2015)
- Clippy (2014-*now*)

Why. . .

. . . should you write static analyses?

(instead of leaving it to *“the experts”*)

Once upon a time. . .

. . . in a fairly recent present

. . . there was

The. Best^{TM}. Programmer.

Why?

You never hack alone.

1. Your future self
   - will have forgotten what you mean
2. Your past self
   - wrote bad code
3. Your new teammates
   - need help frequently
   - with easy issues

Why? (continued)

Issue $\rightarrow$ Permanent solution

Why? (continued)

Issue → Permanent solution

- C → Lint: ~10 years
- Ada → SPARK: 6 years
- ISO C → MISRA: 8 years
- C99 → MISRA: 14 years

II

Let's do something about it

What do we want?

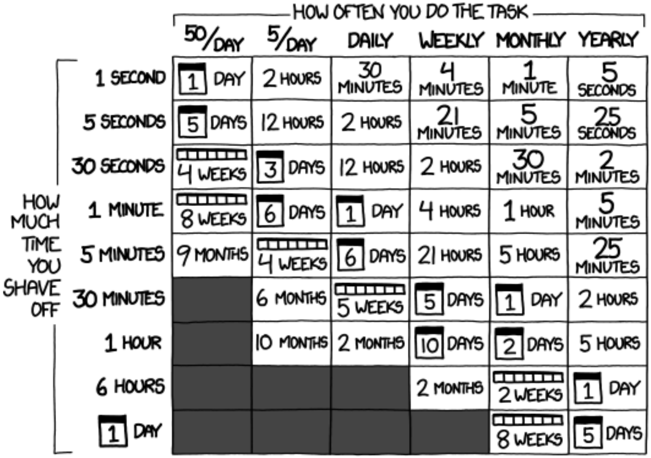
short time from issue to permanent fix

- easy integration
- easy development
- easy sharing
- useful diagnostics

Easy integration

- single setup
- automatically run
- no usability difference from compiler errors

Easy development



HOW OFTEN YOU DO THE TASK

HOW MUCH TIME YOU SHAVE OFF

	50/DAY	5/DAY	DAILY	WEEKLY	MONTHLY	YEARLY
1 SECOND	1 DAY	2 HOURS	30 MINUTES	4 MINUTES	1 MINUTE	5 SECONDS
5 SECONDS	5 DAYS	12 HOURS	2 HOURS	21 MINUTES	5 MINUTES	25 SECONDS
30 SECONDS	4 WEEKS	3 DAYS	12 HOURS	2 HOURS	30 MINUTES	2 MINUTES
1 MINUTE	8 WEEKS	6 DAYS	1 DAY	4 HOURS	1 HOUR	5 MINUTES
5 MINUTES	9 MONTHS	4 WEEKS	6 DAYS	21 HOURS	5 HOURS	25 MINUTES
30 MINUTES	▨	6 MONTHS	5 WEEKS	5 DAYS	1 DAY	2 HOURS
1 HOUR	▨	10 MONTHS	2 MONTHS	10 DAYS	2 DAYS	5 HOURS
6 HOURS	▨	▨	▨	2 MONTHS	2 WEEKS	1 DAY
1 DAY	▨	▨	▨	▨	8 WEEKS	5 DAYS

Easy development

- share code with compiler
  - gcc, clang, rustc, ghc, scala, rebar3
- tools to analyze the bug
- convenience functions
- test driven development

Easy sharing

- sharing is caring
- updating to new compiler versions
- get new analyses

Useful diagnostics

- no false positives
- specialized error messages
- suggestions

III

Questions?

IV

Workshop: Fixing bugs forever

Lints are unstable

- break around every second week
- get fixed fast if part of clippy
- require *the latest* nightly compiler

Lints share code

- clippy's `util` module
- grouping similarly operating lints

Boilerplate 1

```
#![feature(plugin_registrar, box_syntax, rustc_private)]

extern crate syntax;
#[macro_use] extern crate rustc;

use rustc::lint;
use syntax::ast;
```

Boilerplate 2

```
extern crate rustc_plugin;
use rustc_plugin::Registry;

#[plugin_registrar]
fn plugin_registrar(reg: &mut Registry) {
    reg.register_early_lint_pass(box Pass);
}
```

Boilerplate 3

```
declare_lint!(TEST_LINT, Warn, "Warn about items named 'lintme'");

struct Pass;

impl lint::LintPass for Pass {
    fn get_lints(&self) -> lint::LintArray {
        lint_array!(TEST_LINT)
    }
}

impl lint::EarlyLintPass for Pass {
    fn check_item(&mut self, cx: &lint::EarlyContext, it: &ast::Item) {
        if it.ident.name.as_str() == "lintme" {
            cx.span_lint(TEST_LINT, it.span, "item is named 'lintme'");
        }
    }
}
```

Quick guide

setup instructions at

`https://github.com/Manishearth/rust-clippy/tree/rust_belt_rust`

1. open `tests/compile-fail/rust_belt_rust.rs`
2. write a piece of code you dislike
3. Or have a look at clippy issues labeled `E-easy`
4. Develop your lint in `clippy_lints/src/rust_belt_rust.rs`
5. run `cargo test`
6. Repeat **4.** until the tests pass
7. Write your lint info into *the list*
8. Create a pull request to the clippy repository