Debugging for math lover

LambdaDays2022

Michał J. Gajda https://www.migamake.com

2022-07-28

Plan

- How to model debugging?
- How to find error?
- Time to fix
- Search space
- Cognitive limits

Types of errors

- Does not compile

Errors higher on this page are faster to fix

Types of errors

- Does not compile
- Type error

Errors higher on this page are faster to fix

Types of errors

- Does not compile
- Type error
- Gives wrong result

Errors higher on this page are faster to fix

Types of errors

- Does not compile
- Type error
- Gives wrong result
- Performance is too slow

Errors higher on this page are faster to fix

Types of errors

- Does not compile
- Type error
- Gives wrong result
- Performance is too slow
- Wrong concept was used to model reality

Errors higher on this page are faster to fix

Types of errors

- Does not compile
- Type error
- Gives wrong result
- Performance is too slow
- Wrong concept was used to model reality
- User experience is frustrating

Errors higher on this page are faster to fix

Types of errors

- Does not compile
- Type error
- Gives wrong result
- Performance is too slow
- Wrong concept was used to model reality
- User experience is frustrating
- Specification does not match user expectations

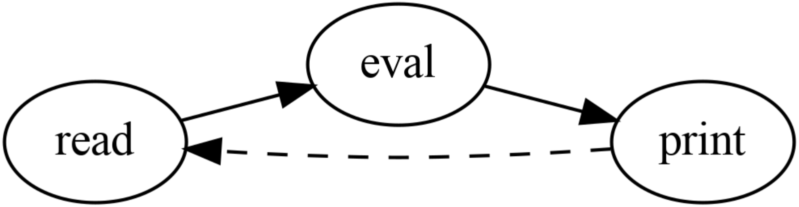
Errors higher on this page are faster to fix

Types of errors (2)

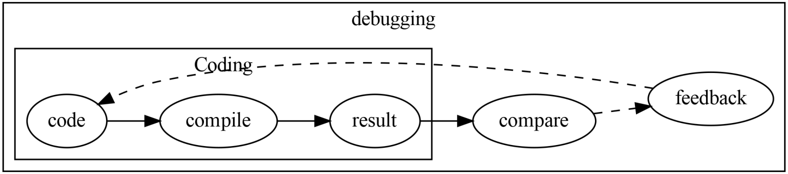
- Does not compile
- Type error
- Gives wrong result
- Performance is too slow
- Wrong concept was used to model reality
- User experience is frustrating
- Specification does not match user expectations

Error is a difference between what we want, and what we got.

Read-eval-print loop



Discrepancy model



Feedback cycle

Greenhorn student model

- Student randomly changes a single character of code to get error.

Greenhorn student model

- Student randomly changes a single character of code to get error.
- Complexity: $|alphabet|^{\#characters}$

Greenhorn student model

- Student randomly changes a single character of code to get error.
- Complexity: $|alphabet|^{\#characters}$
- *Can write small programs, under 50 characters*

Syntax-aware student model

- Student randomly changes a *syntactic token*

Syntax-aware student model

- Student randomly changes a *syntactic token*
- `while` $\rightarrow$ `for`

Syntax-aware student model

- Student randomly changes a *syntactic token*
- `while` $\rightarrow$ `for`
- `(` $\rightarrow$ `;`

Syntax-aware student model

- Student randomly changes a *syntactic token*
- `while` $\rightarrow$ `for`
- `(` $\rightarrow$ `;`
- Complexity: $|\textit{token types}^{\#\textit{tokens}}$

Syntax-aware student model

- Student randomly changes a *syntactic token*
- `while` $\rightarrow$ `for`
- `(` $\rightarrow$ `;`
- Complexity: $|\textit{token types}^{\#\textit{tokens}}$
- *Can write small programs, under 30 characters*

Student reads compiler errors

- Compiler gives line of the syntax error

Student reads compiler errors

- Compiler gives line of the syntax error
- Depending on language syntax, we have high probabilty this is correct

Student reads compiler errors

- Compiler gives line of the syntax error
- Depending on language syntax, we have high probabilty this is correct
- With small probability we devolve to searching neighbourhood

Time-to-fix

Compare different kinds of errors:

- Lexer error – editor changes color $t > 1s$

Time-to-fix

Compare different kinds of errors:

- Lexer error – editor changes color $t > 1s$
- Syntax error – compiler parses $t > 10s$

Time-to-fix

Compare different kinds of errors:

- Lexer error – editor changes color $t > 1s$
- Syntax error – compiler parses $t > 10s$
- Processing error – after program is complete $(1min < t < 1h)$

Time-to-fix

Compare different kinds of errors:

- Lexer error – editor changes color $t > 1s$
- Syntax error – compiler parses $t > 10s$
- Processing error – after program is complete $(1min < t < 1h)$
- Latent misbehaviour – after program sees new input in production $(1mo < t < 1y)$

Time-to-fix

Compare different kinds of errors:

- Lexer error – editor changes color $t > 1s$
- Syntax error – compiler parses $t > 10s$
- Processing error – after program is complete $(1min < t < 1h)$
- Latent misbehaviour – after program sees new input in production $(1mo < t < 1y)$
- Cognitive research: time to learn from errors $\sim \frac{1}{t^2}$

Time-to-fix

Compare different kinds of errors:

- Lexer error – editor changes color $t > 1s$
- Syntax error – compiler parses $t > 10s$
- Processing error – after program is complete $(1min < t < 1h)$
- Latent misbehaviour – after program sees new input in production $(1mo < t < 1y)$
- Cognitive research: time to learn from errors $\sim \frac{1}{t^2}$
- $\rightarrow$ decrease *latency*!

Fixing function

What is easier to debug:

```
capitalize     (w:ws) =
        toUpper w:
    map toLower   ws
```

OR:

```
capitalize = zipWith f [1..]
  where
    f 1 a = toUpper a
    f _ a = toLower a
```

Fixing function (2)

- nested conditions are hard to understand

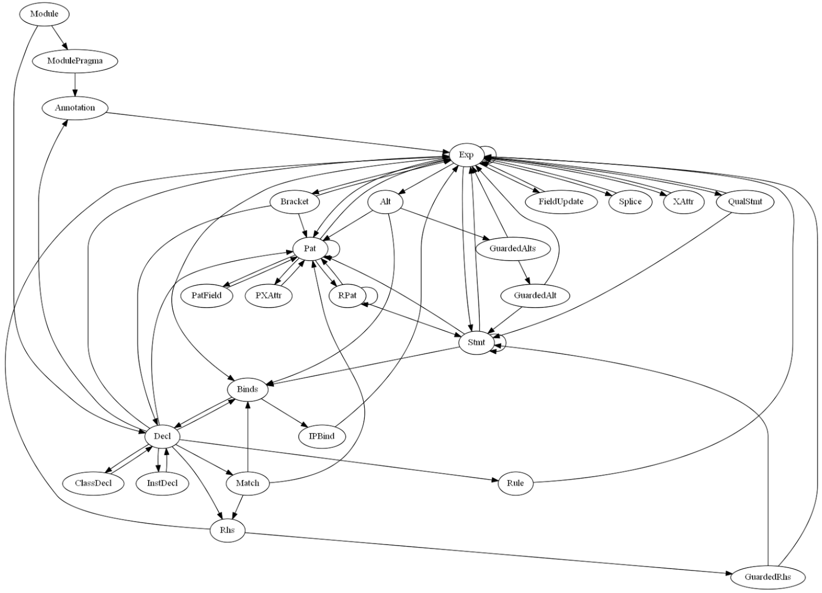
Fixing function (2)

- nested conditions are hard to understand
- avoid nesting conditions, loops etc.

Fixing function (2)

- nested conditions are hard to understand
- avoid nesting conditions, loops etc.
- cyclomatic complexity

Fixing a library



Neil Mitchell *Haskell type graphs with Uniplate and Haskell-src-exts*

Fixing library

Library function of 3 lines has a bug. . .

- How long does it take to fix it?

Fixing library

Library function of 3 lines has a bug. . .

- How long does it take to fix it?
- May depends of different use cases

Fixing library

Library function of 3 lines has a bug...

- How long does it take to fix it?
- May depends of different use cases
- May depend on completenss of test coverage

Fixing library

Library function of 3 lines has a bug...

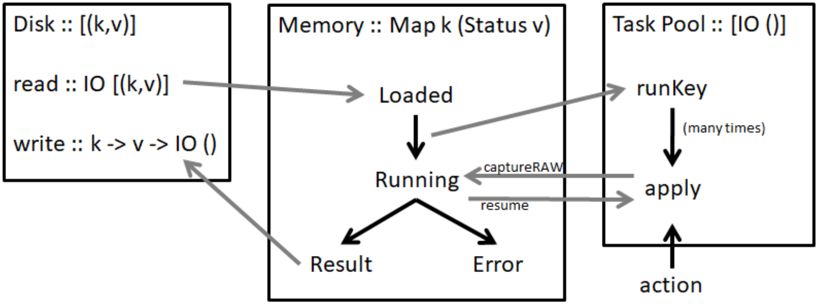
- How long does it take to fix it?
- May depends of different use cases
- May depend on completenss of test coverage
- May depend on the number of different users

Fixing library

Library function of 3 lines has a bug...

- How long does it take to fix it?
- May depends of different use cases
- May depend on completenss of test coverage
- May depend on the number of different users
- ...depends of a number of interactions with other functions.

Fixing library



Neil Mitchell - Shake from 10,000ft

Fixing library (2)

- McCabe complexity measures number of different operators, functions and operands.

Fixing library (2)

- McCabe complexity measures number of different operators, functions and operands.
- number of common subexpressions

Fixing library (2)

- McCabe complexity measures number of different operators, functions and operands.
- number of common subexpressions
- using high-level type classes decreases number of different expressions

Fixing library (2)

- McCabe complexity measures number of different operators, functions and operands.
- number of common subexpressions
- using high-level type classes decreases number of different expressions
- high-level abstraction $\rightarrow$ common interface $\rightarrow$ decreased complexity

Ultimate ways of reducing complexity

- decrease size of the problem (function, module. . .)

Ultimate ways of reducing complexity

- decrease size of the problem (function, module. . .)
- decrease latency to comparison (editor hints, type errors. . .)

Ultimate ways of reducing complexity

- decrease size of the problem (function, module. . .)
- decrease latency to comparison (editor hints, type errors. . .)
- reduce interaction (function arguments, module interface. . .)

Ultimate ways of reducing complexity

- decrease size of the problem (function, module. . .)
- decrease latency to comparison (editor hints, type errors. . .)
- reduce interaction (function arguments, module interface. . .)
- reuse abstraction (monad, applicative, mathematics. . .)

Ultimate ways of reducing complexity

- decrease size of the problem (function, module. . .)
- decrease latency to comparison (editor hints, type errors. . .)
- reduce interaction (function arguments, module interface. . .)
- reuse abstraction (monad, applicative, mathematics. . .)
- use `Homplexity` to find hot spots in your Haskell code!