Emotional Code

Kate Gregory

kate@gregcons.com

@gregcons

https://www.includecpp.org

What Are Emotions?

- Out of band interrupts
- Deliver a conclusion without all the supporting evidence clearly listed
- Some of us act on them, some less so
- Emotional Intelligence is a skill that varies among people
- People who lack it often distrust emotions
- Some consider using them to be lazy, non-rigorous, or cutting corners

No Emotions Allowed

OH THE HUMANITY!!!
AIRLANDER
makeameme.org

Here’s a Little Logic

- Programmers are human beings
- Human beings have emotions
- Therefore…
- Programmers have emotions
- Emotions are not for the weak: emotions are for **people**

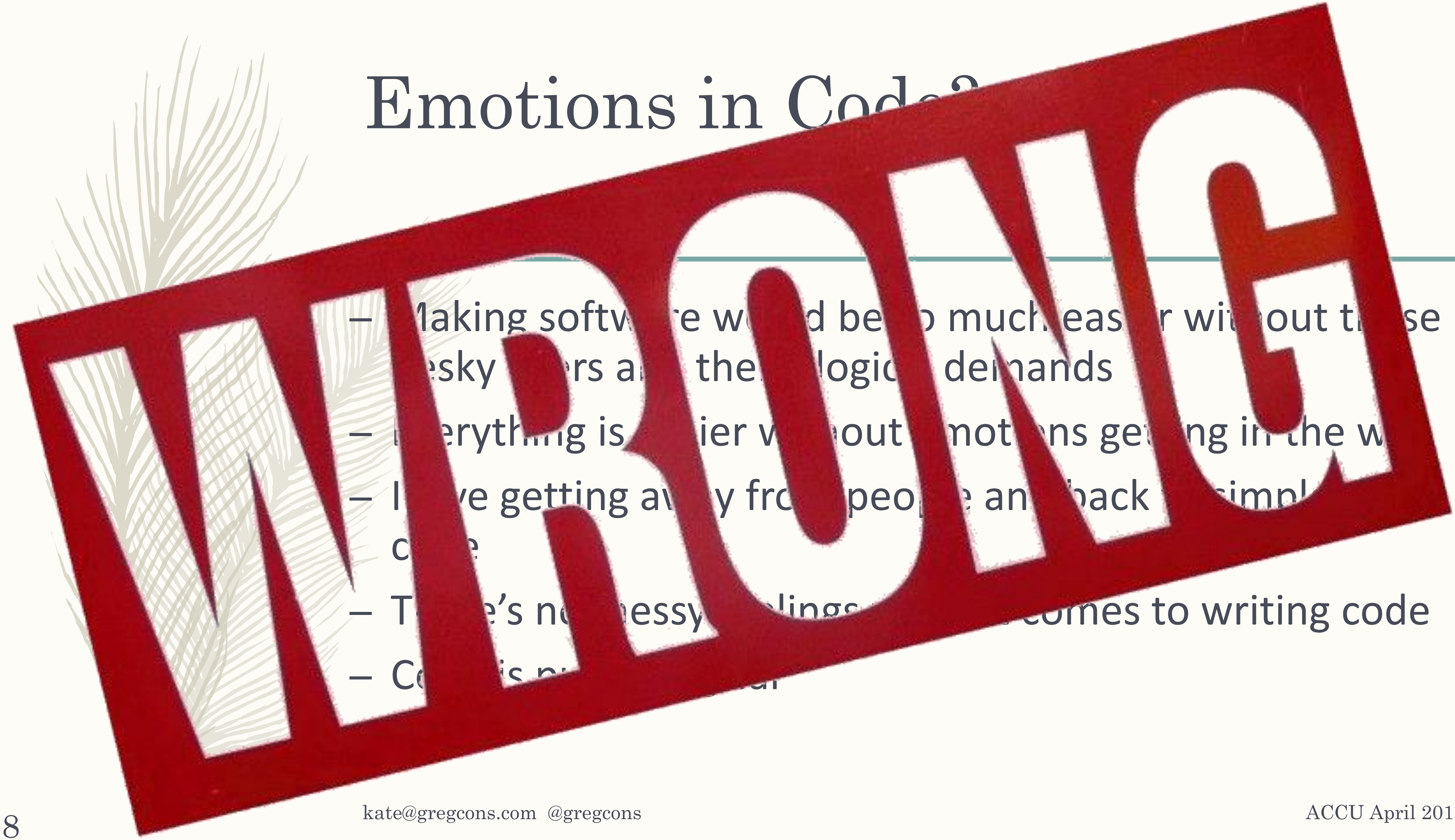
Emotions in Software Development

- – Persuading people to do things your way
- – Listening to what people want and why
- – Being seen as helpful and valuable
- – Winning the meeting
- – Trusting your team to help you
- – Being someone your team can trust
- – Standing up for your values

Emotions in Code?

- Making software would be so much easier without these pesky users and their illogical demands
- Everything is easier without emotions getting in the way
- I love getting away from people and back to simple pure code
- There’s no messy feelings when it comes to writing code
- Code is purely logical

Emotions in Code?



- [illegible]aking softw[illegible]re wo[illegible]d be [illegible]o much eas[illegible]r wit[illegible]out th[illegible]se [illegible]esky [illegible]ers a[illegible] the[illegible]logic[illegible] demands
- [illegible]erything is [illegible]ier w[illegible]out [illegible]mot[illegible]ns get[illegible]ng in the w[illegible]
- I[illegible]ve getting aw[illegible]y fro[illegible] peop[illegible]e an[illegible] back [illegible]simpl[illegible] c[illegible]e
- T[illegible]e's n[illegible]essy [illegible]lings [illegible] comes to writing code
- Co[illegible]is n[illegible]

There Are No Emotions In Code

– There are

– I can see them

- Commented out code
  - I might not be doing this right; I might need this
- Comments with who told you to change this
  - Don't blame me if this does the wrong thing
- Unused variables and code not removed
  - How can I be sure we won't need it?
- No time taken to clean up
  - I'm on a knife edge as it is, I can't take time for that
- Follow the same bad patterns that were there
  - I can't stand up for doing it differently or better

```
//if (m_nCurrentX != g_nCurrentX
//   || m_nCurrentABC != g_nCurrentABC) {
//}
```

- Comments with who told you to change this
  - Don’t blame me if this does the wrong thing
- Unused variables and code not removed
  - How can I be sure we won’t need it?
- No time taken to clean up
  - I’m on a knife edge as it is, I can’t take time for that
- Follow the same bad patterns that were there
  - I can’t stand up for doing it differently or better

- Commented out code
  - I might not be doing this right; I might need this
- Comments with who told you to change this

```
// int nData;  3/22/03 uninitialized catch by VC7
int nData = 0;
```

- Unused variables and code not removed
  - How can I be sure we won't need it?
- No time taken to clean up
  - I'm on a knife edge as it is, I can't take time for that
- Follow the same bad patterns that were there
  - I can't stand up for doing it differently or better

- Commented out code
  - I might not be doing this right; I might need this
- Comments with who told you to change this
  - Don’t blame me if this does the wrong thing
- Unused variables and code not removed
  - How can I be sure we won’t need it?
- No time taken to clear
  - I’m on a knife edge as
- Follow the same bad p
  - I can’t stand up for do

```
int c,n;
int r1,r2,r3,r4;
double factor;
double pct1,pct2,pct3,v1,v2,v3,v4,v5;
double d1,d2,d3;
```

- Commented out code
  - I might not be doing this right; I might need this
- Comments with who told you to change this
  - Don't blame me if this does the wrong thing
- Unused variables and code not removed
  - How can I be sure we won't need it?
- No time taken to clean up
  - I'm on a knife edge as it is, I can't take time for that
- Follow the same bad patterns that were there
  - I can't stand up for doing it differently or better

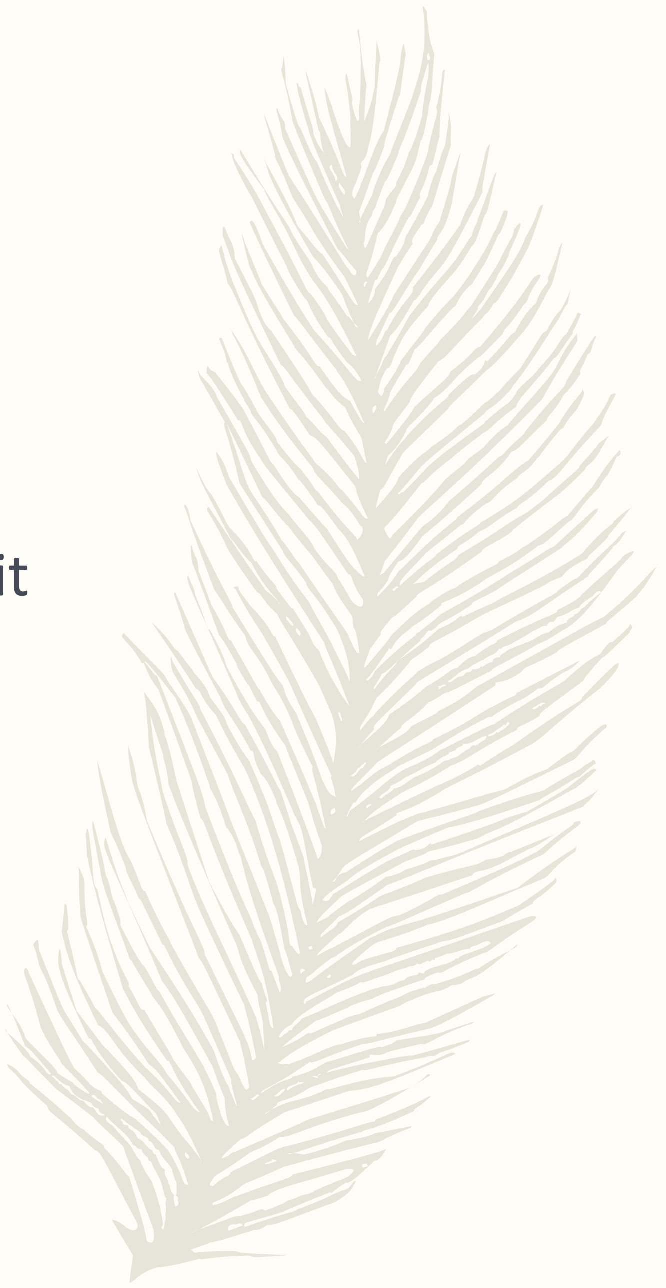
Fear

- Checking what doesn’t need to be checked
  - I can’t be sure I’ll be looked after
- Checking again and again
  - I can’t remember if I did or not, I can’t count on it
  - That was in a team-mate’s code, they might have changed it without telling me
- Doing everything by hand
  - I need to see it, step through it
  - I can’t trust anyone else’s code
  - I’ve been hurt before

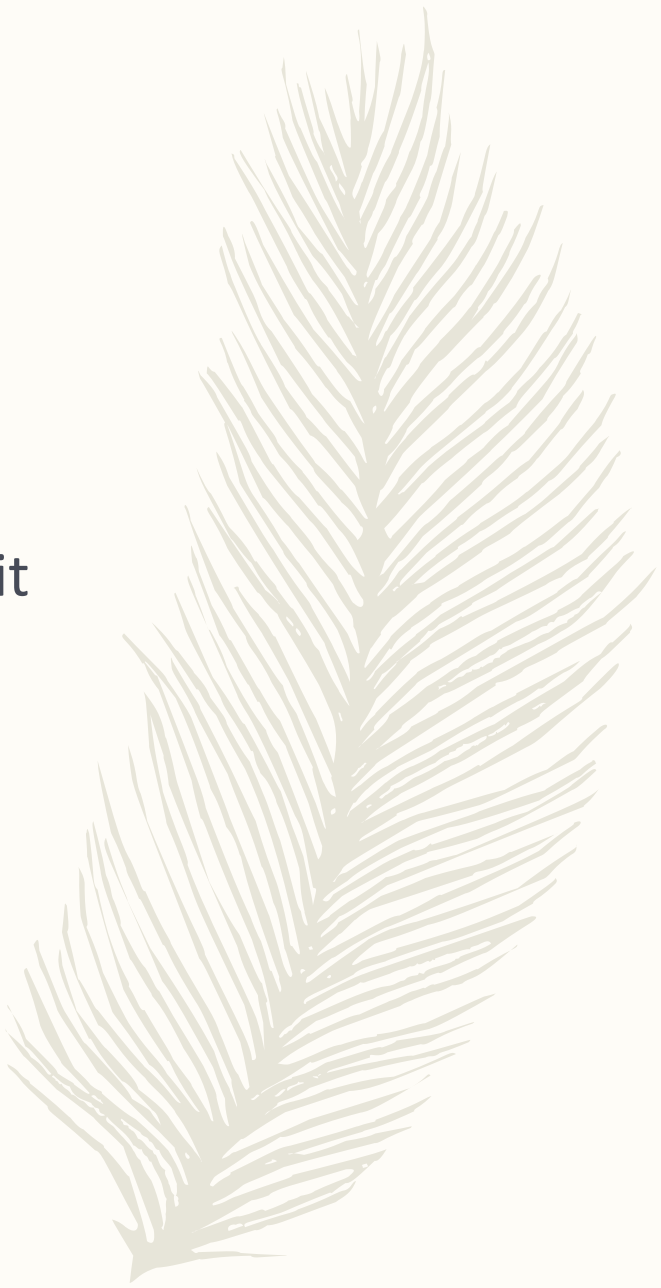
- Checking what doesn't need to be checked

```
if (pPolicy) { delete pPolicy;}
```

- Checking again and again
  - I can't remember if I did or not, I can't count on it
  - That was in a team-mate's code, they might have changed it without telling me
- Doing everything by hand
  - I need to see it, step through it
  - I can't trust anyone else's code
  - I've been hurt before



- Checking what doesn't need to be checked
  - I can't be sure I'll be looked after
- Checking again and again
  - I can't remember if I did or not, I can't count on it
  - That was in a team-mate's code, they might have changed it without telling me
- Doing everything by hand
  - I need to see it, step through it
  - I can't trust anyone else's code
  - I've been hurt before



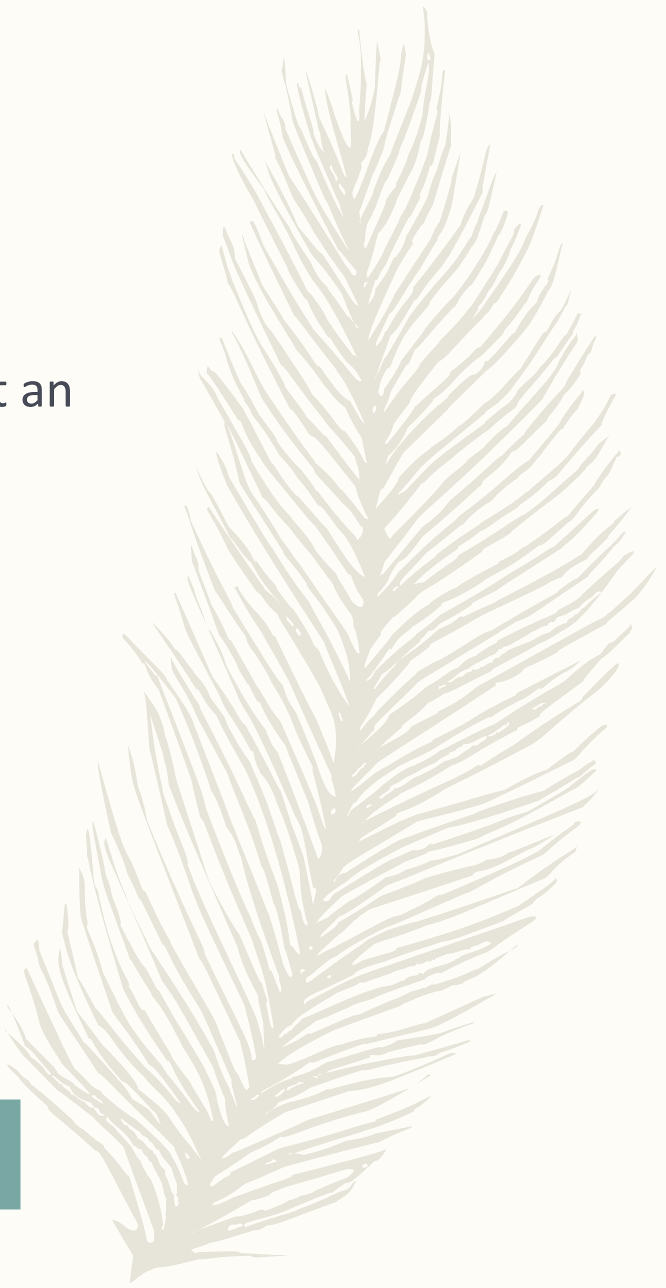
Fear

- Tiny variable names
  - Aren’t you smart enough to figure out what these are?
- Obscure function names
  - Why should I explain myself to people who can’t understand it without an explanation?
- Deliberately opaque names
  - foo and bar considered harmful
  - f(), g(), etc not much better
- Raw loops, own containers, own algorithms
  - In most cases
  - Perhaps “it ain’t bragging if you can do it” applies
- Sneering comments and names
  - If you say lusers, pebcak, and rtfm in slack, you say it in your code too

– Tiny variable names
  – Aren’t you smart enough to figure out what these are?
– Obscure function names
  – Why should I explain myself to people who can’t understand it without an explanation?
– Deliberately opaque names
  – foo and bar considered harmful
  – f(), g(), etc not much better
– Raw loops, own containers, own algorithms
  – In most cases
  – Perhaps “it ain’t bragging if you can do it” applies
– Sneering comments and names

```
void UndoStevesNonsense();
```

  – If you say lusers, pebcak, and rtfm in slack, you say it in your code too



- Tiny variable names
  - Aren't you smart enough to figure out what these are?
- Obscure function names
  - Why should I explain myself to people who can't understand it without an explanation?
- Deliberately opaque names
  - foo and bar considered harmful
  - f(), g(), etc not much better
- Raw loops, own containers, own algorithms
  - In most cases
  - Perhaps "it ain't bragging if you can do it" applies
- Sneering comments and names
  - If you say lusers, pebcak, and rtfm in slack, you say it in your code too

Arrogance

- No time taken to clean up: refactor, rearrange, rename
  - Why should I spend my time making things easy for you?
- Short and opaque names, magic numbers
  - I’m being measured here, and I’ve got tickets to close
- Side effects and consequences everywhere
  - Public variables because it’s quicker
  - Mutable global state because it’s quicker
- Information Hoarding
  - My job is safe if nobody else can do this

Selfishness

- Whatever, it works
  - Mostly, enough anyway
- No STL, no libraries to speak of
  - I can’t be learning new stuff, I have code to write
- No testing, no build automation, no scripts
  - If you think that matters, you do it
- Copy-paste-edit
  - Abstraction? Sounds like work to me!
- No commitment to the future

Laziness

Code Shows Emotions

- Fear
- Arrogance
- Selfishness
- Laziness

One single-letter variable name does not a psychopath make

Why Does This Matter?

- Empathy as you read and fix that legacy code
- Does your team or workplace need to change?
- Are your management practices causing runtime performance issues?
- A lodestar for yourself as you write new code or tidy old

But Can’t Some Code Be Neutral?

- Shopping lists can be neutral
- Love letters can’t
  - If they’re not actively warm and loving, they’re cold and disappointing
- Letters of recommendation can’t
  - If they just confirm facts, they scream “bad hire”!
- Code can’t either
  - No in between
  - And even if there was, why aim for that?

No Neutrality

Choose to Be	Instead Of
– Confident and capable	– Insecure and afraid
– Reassuring and obvious	– Secretive or slapdash
– Open and transparent	– Information hoarding
– Humble	– Arrogant
– Generous and empathetic	– Selfish

Look Where You Want to Go

- Delete things you don’t need
  - I have source control and work notes
- Take time to clean up
  - It might help me, it might help someone else
- Comments and names explain thinking
  - I know I’m right, let me show you
- Obsolete or handrolled things replaced
  - I’m brave enough to stand up for doing things the right way

Confidence

- Use libraries
  - Include a link to the doc if it’s not just cppreference.com
- Gentle comments
  - Where things aren’t obvious, leave some help for the next person
- Helpful names
  - For functions, variables, everything
- I know you’re as good as me and will understand it if I explain it
  - you’re worth explaining this to

- Use libraries
  - Include a link to the doc if it’s not just cppreference.com

```
//Set page size to standard 8.5 x 11 (96 is DPI for WPF)
page->Height = 8.5 * 96;
page->Width = 11 * 96;
```

- Helpful names
  - For functions, variables, everything
- I know you’re as good as me and will understand it if I explain it
  - you’re worth explaining this to

- Use libraries
  - Include a link to the doc if it’s not just cppreference.com
- Gentle comments
  - Where things aren’t obvious, leave some help for the next person
- Helpful names
  - For functions, variables, everything
- I know you’re as good as me and will understand it if I explain it
  - you’re worth explaining this to

Humility

- Clean engineering to make next time easier
  - Well thought out encapsulation
  - Appropriate level of abstraction
- Again, take time to clean up: refactor, rearrange, rename
- Information sharing
  - My job is safe if we can all do this

Generosity

Let's Talk About Names

– Naming is hard

– We're famously bad at it

– Why?

– It requires empathy

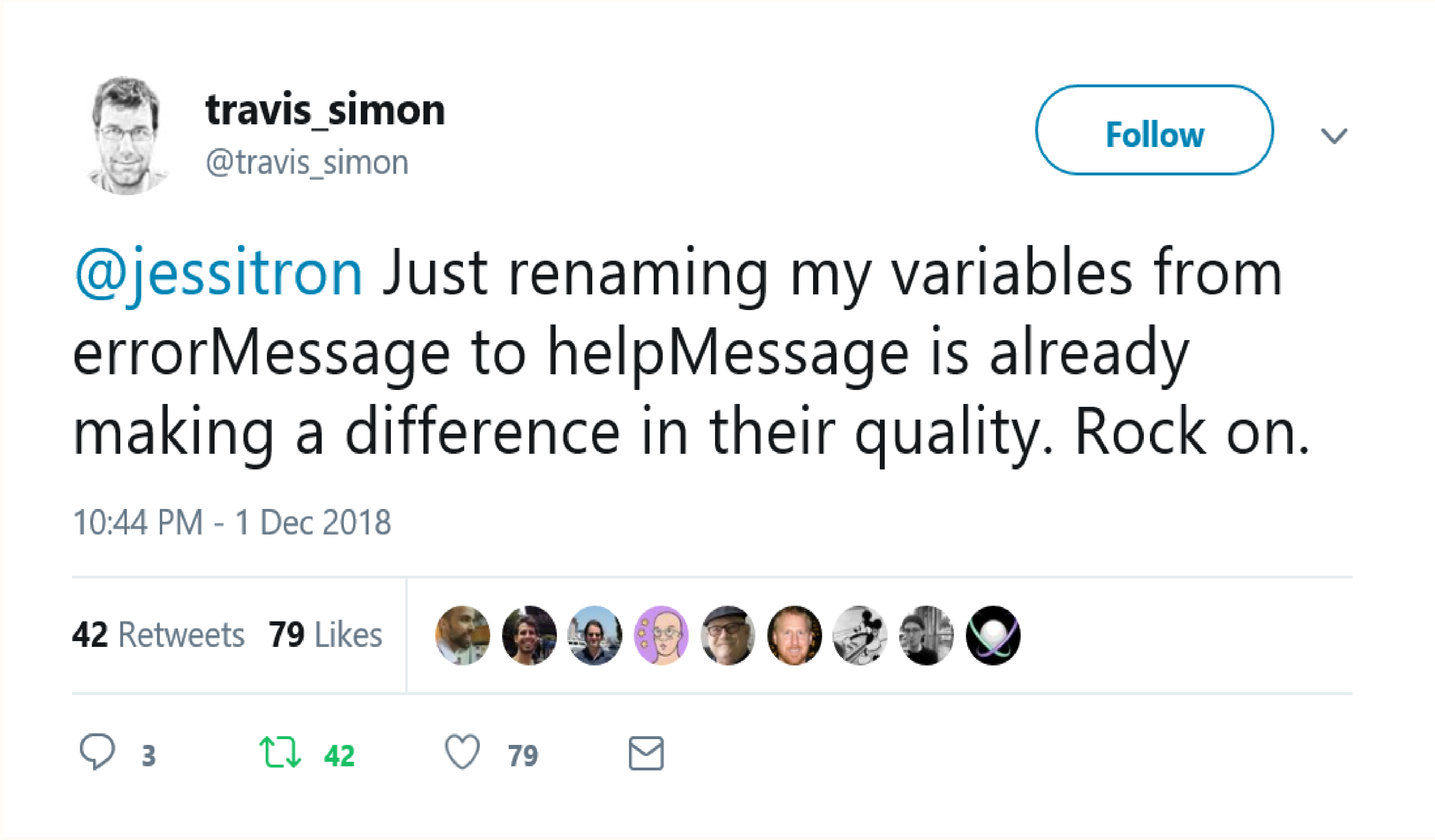
An Algorithm Story

– sort()

– partial_sort()

– partial_sort_copy()

– top_n()



travis_simon
@travis_simon

Follow

@jessitron Just renaming my variables from errorMessage to helpMessage is already making a difference in their quality. Rock on.

10:44 PM - 1 Dec 2018

42 Retweets 79 Likes

3 42 79

Hard Working

- It compiles, links, runs, and passes the tests
  - No warnings, no "you get one exception on startup, just hit Continue", no stray files left behind
  - Tests are complete and well documented
  - I don't have to be asked to do it right
- It uses modern constructs or libraries or tools
  - I'm always learning; my code gets the benefit
  - But not tools for the sake of tools or for fun
- Modern practices
  - Not just churning out code
- Commitment to the future
  - My own ease
  - The team's success

Choose to Show Positive Emotions

- Your code will be easier to read and maintain
- You will enjoy reading and maintaining it more
- Your reputation will improve
- Even if the code isn’t better
  - But it probably will be

Call to action

- Care about those who wrote the code you maintain
- Show your confidence
- Be generous and empathetic
- You are going to show emotions in your code