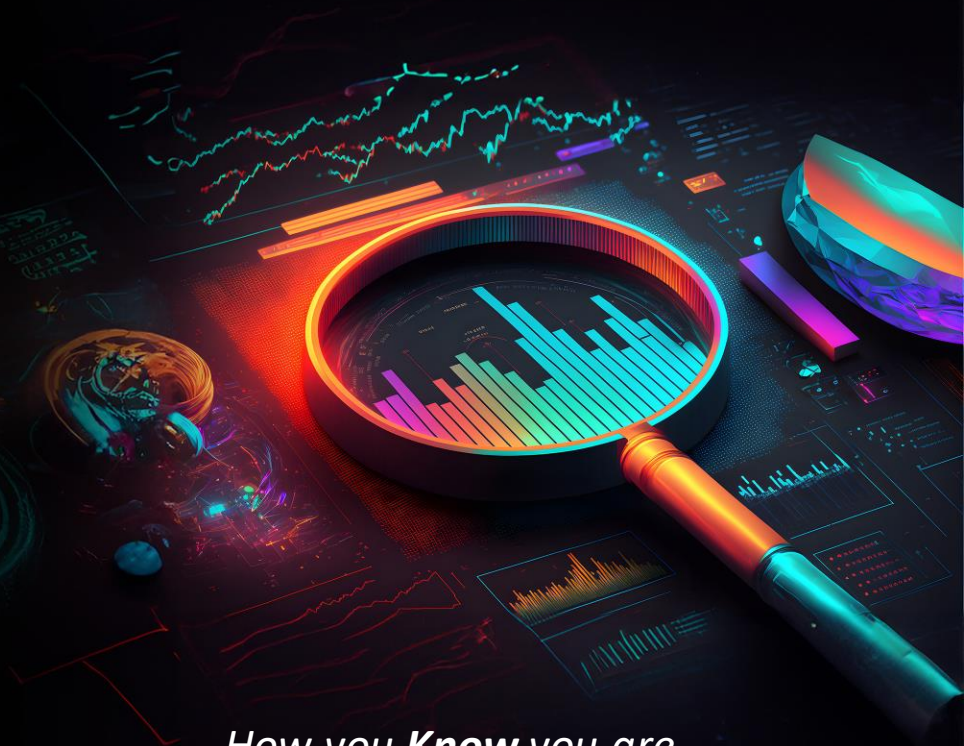
This is a work in progress and will continue to evolve over time.

https://github.com/jimmckeeth/Evidence-Based-Software-Engineering

Beyond Best Practices:

Cultivating Evidence-Based Software Engineering



Jim McKeeth
jim@gdksoftware.com
gdksoftware.com

*How you **Know** you are Writing the RIGHT code*

Agenda

- Premise
- What is software engineering?
- Finding the Evidence
- Understanding Runtime: Big O
- Premature Optimization
- Use the source
- In the IDE
- External Tools
- More Resources

Why do **you** write the **CODE** you write?

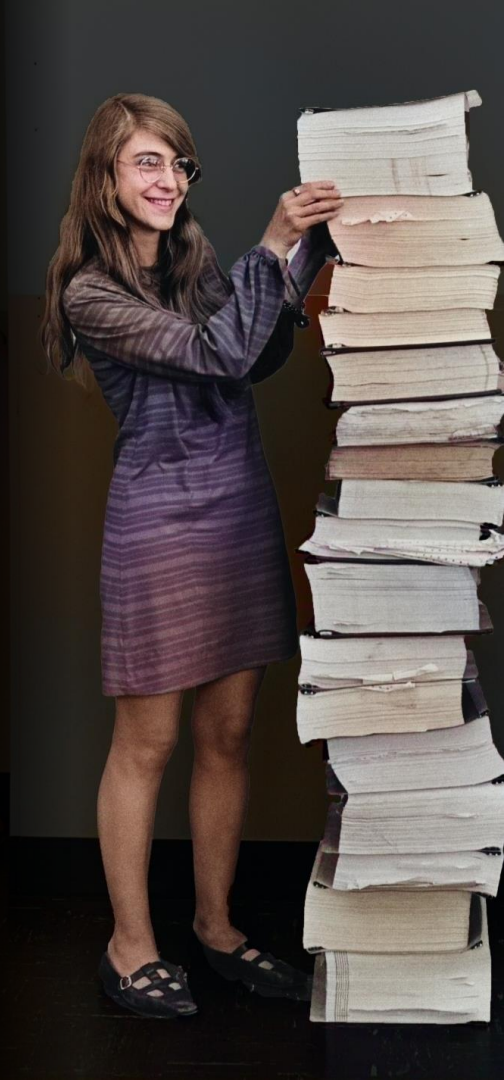
Grace Hopper, RDML

- Wrote first computer manual
- Machine-independent programming languages
  - FLOW-MATIC & COBOL
- Carried around light nanoseconds
  - 29.97 cm of wire
- The most dangerous phrase, *“We've always done it this way.”*

https://en.wikipedia.org/wiki/Grace_Hopper

Margaret Hamilton

- Director of the Software Engineering Division of the MIT Instrumentation Laboratory
- Developed flight software for the Apollo program
- 2016 – Presidential Medal of Freedom from Obama
- Published over 130 papers
- Invented the term “software engineering”
    - *“…to distinguish it from hardware and other kinds of engineering, yet … as part of the overall **systems engineering process**.”*



https://en.wikipedia.org/wiki/Margaret_Hamilton_(software_engineer)

Women of NASA

LEGO Set # 21312

Don't believe
anything I tell you!

TRAVEL BACK IN TIME...

BOISE CODE CAMP 2007

with Rich Hundhausen

Consider this code

```
sl := TStringList.Create;
try
   // use TStringList
except
  raise;
end;
sl.free;
```

Justification…

The **try/except** guarantees the code after the **end** is executed…

Raise allows the exception handler further up the stack to handle it…

Unfortunately…

Raise in the **except** prevents the code after the **end** from running when an exception occurs….

The exception code path was never tested, so this pattern was wide spread in production and multiple projects.

The developer read this in a book and accepted it as **truth**, never questioning the *way* he used it…

Superstitions

- “We've always done it that way”
- Right way
- Best practices

Clarke's three laws

1. When a distinguished but elderly scientist states that something is possible, he is almost certainly right. When he states that something is impossible, he is very probably wrong.

2. The only way of discovering the limits of the possible is to venture a little way past them into the impossible.

3. Any sufficiently advanced technology is indistinguishable from magic.

Arthur C. Clarke

The question I'm always asked

What is the ***best*** …

What is the *best* …

- Component set
- Database access framework
- Grid component or library for …
- Way to handle exceptions
- LiveBindings vs Data Aware vs manual
- Memory manager
- OOP vs Procedural vs Functional
- Database (NoSQL vs RDMBS)
- Development methodology (SCRUM, Agile, etc.)
- Programming language

Now you try

You need to loop through some **`TDataSet`** records. For each record you need to examine multiple fields:

- What is *best* solution and why?
- Fastest?
- Uses least memory?
- Easiest to maintain?
- Simplest to explain?

How confident are you with your answer?

Possible Solutions:

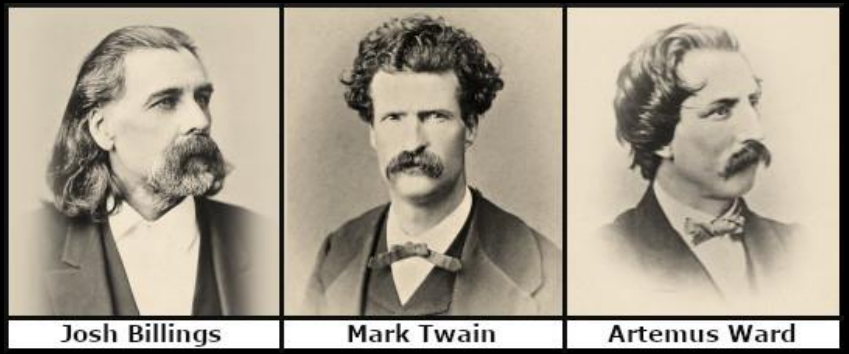
1. FieldByName
2. FieldByNumber
3. Hard coded field names
4. Local references to each field
5. Custom SQL for each
6. Don't use TDataSet descendent
7. Switch to NoSQL from RDBMS
8. Something else….

The answer is always

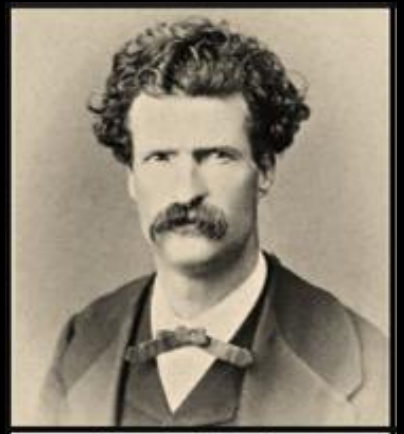
It depends …

"It ain't what you don't know
that gets you into trouble.

It's what you know for sure
that just ain't so.



“The trouble with old men is they remember so many things that ain’t so.”



Mark Twain

Unfounded confidence gets us in trouble

Programming Changes

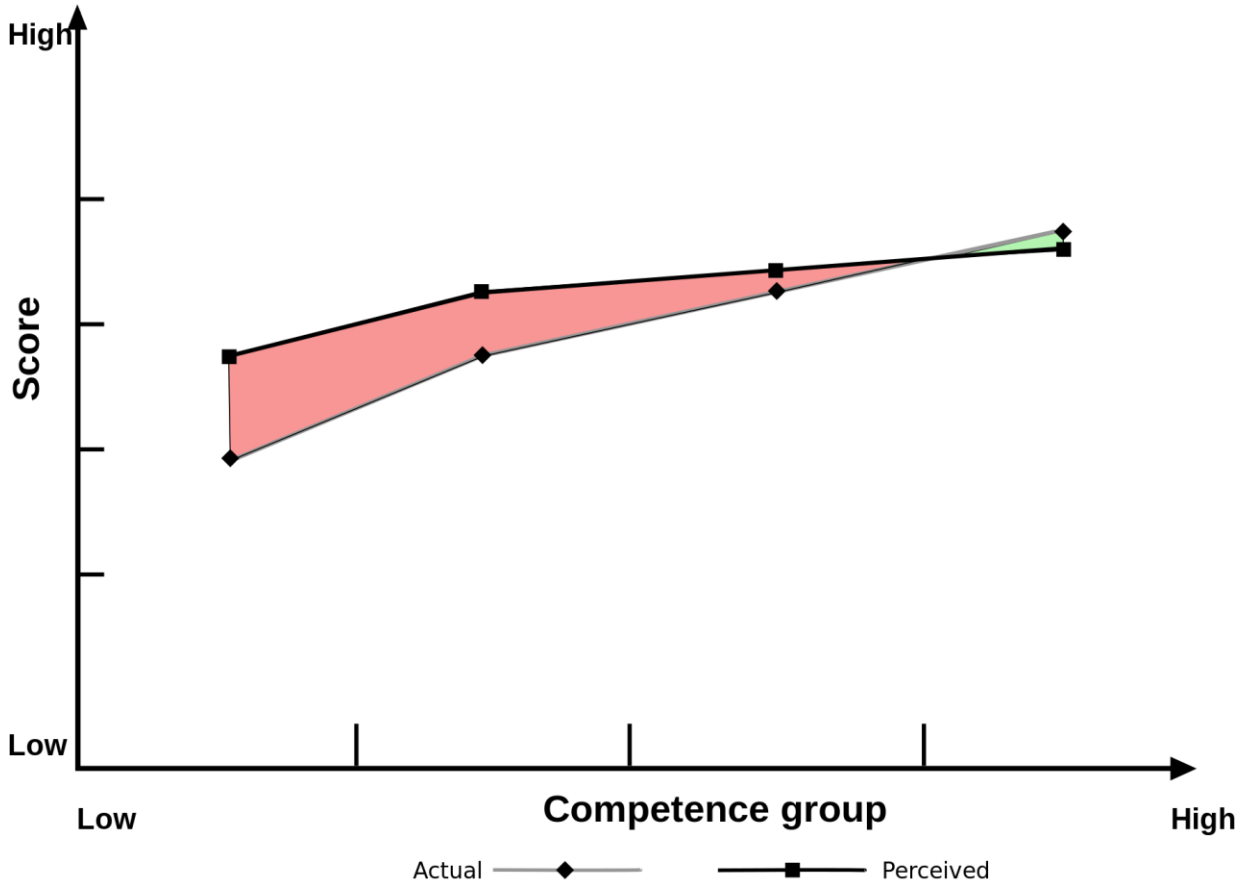
- New versions of the Delphi compiler
- Changes to the RTL
- Different database access frameworks
- Database backend changes
- API changes
- New versions of Windows
- Different operating systems
  (Android, MacOS, Linux, etc.)
- Multicore CPUs
- New CPU instructions
- *We learn more*

"Programming is an ART
that fights back."

-Chad "Kudzu" Howard

- Cognitive bias
- The less you know,
  the more you think you know
- The more you learn,
  the less you realize you know

Dunning–Kruger Effect

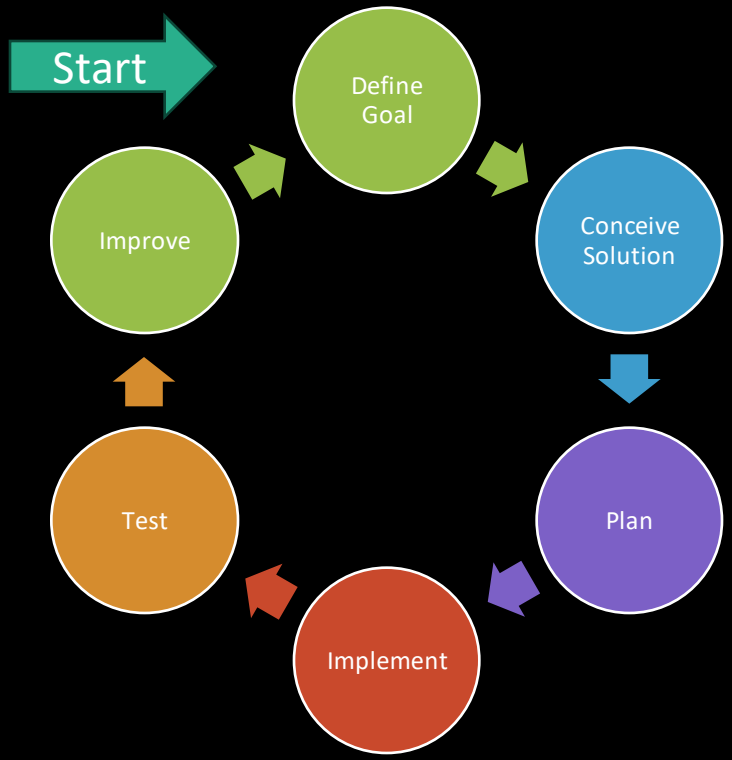


Software Developer Levels

- Junior Developer
  - *Learning best practices*
- Intermediate Developer
  - *Follows best practices*
- Senior Developer
  - *Knows when to not follow best practices*

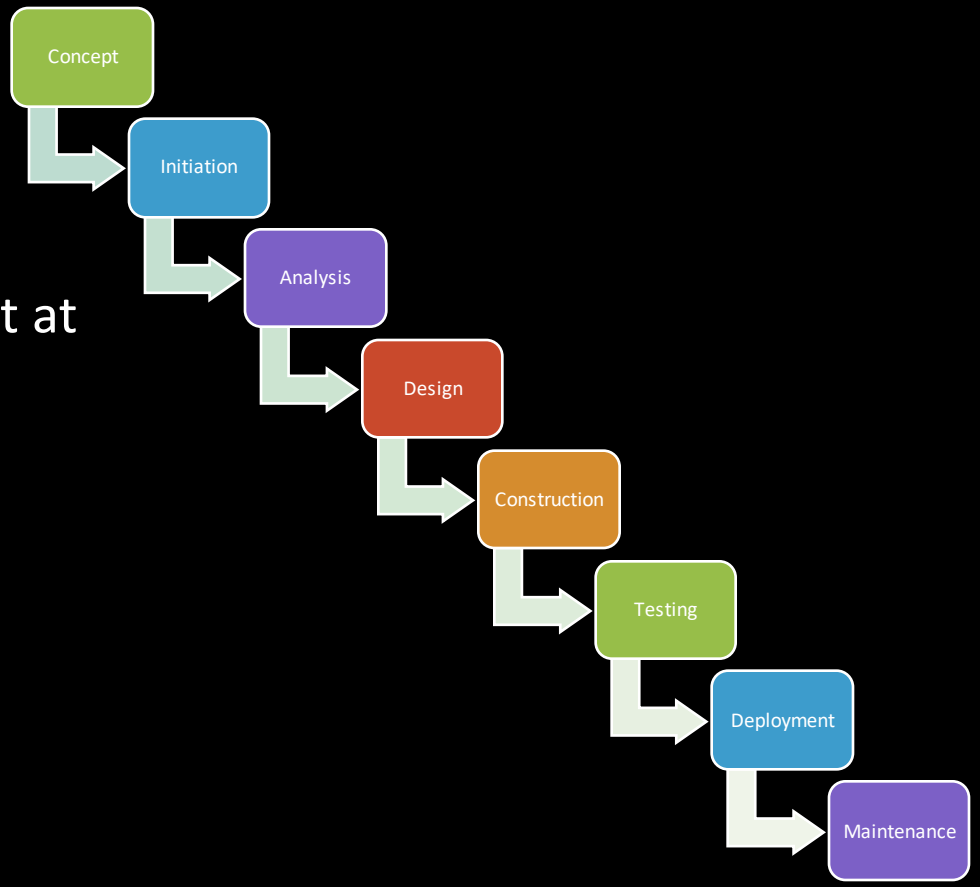
Scientific Method or Engineering Process

- Engineering brings a formal process
- A very important step is to “test”
- The results of the test provide the data
- During planning it is important to set an expectation and define success
- This is a cycle!
  - Use data to inform future solutions
  - The goals will change and evolve



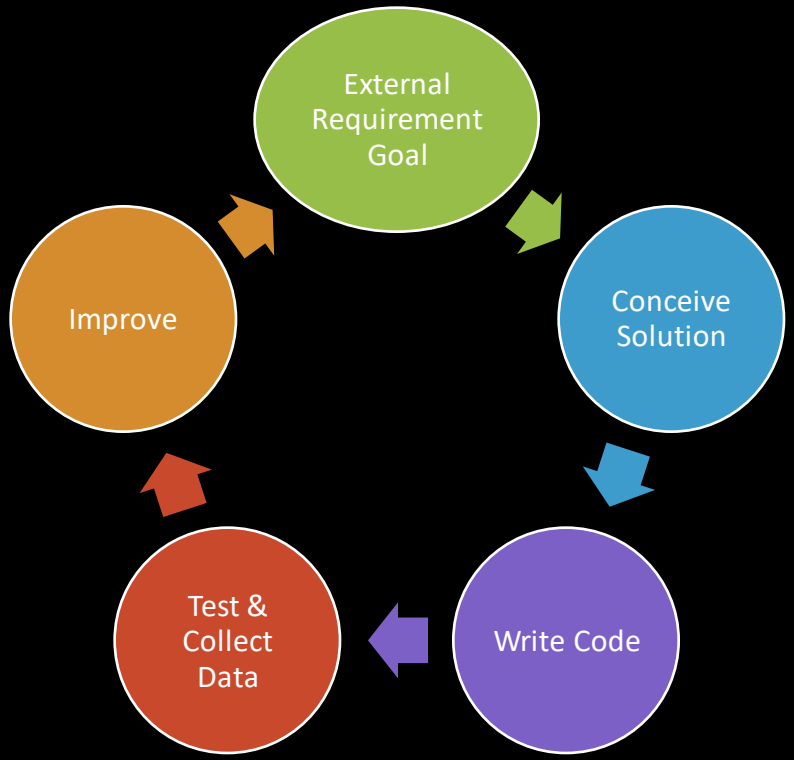
Waterfall Model

- A linier, non-cyclic model
- Unrealistic
- Requires all knowledge to exist at the beginning



Microscale Cyclic Process

- Even if the requirements are external
- Still look at your own process as cyclic
- Repeat the cycle until you meet the requirements
- Use your data to meet future requirements better
- Always test and collect data

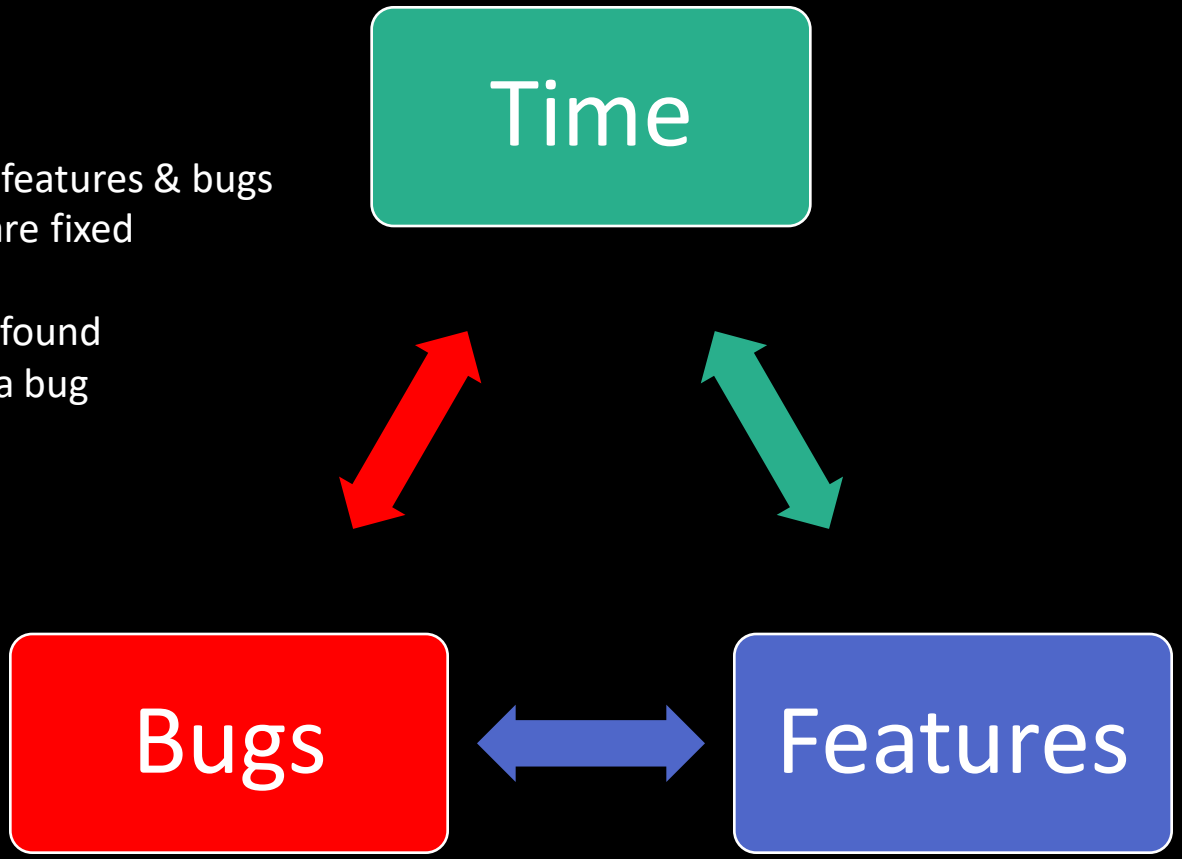


The Math of Bugs and Fixes

- Adding features produces bugs
- Measure feature work as churn
- Collect data on your team
- How many bugs per unit of churn?
- Keep testing and fixing until expected bugs found

Engineering

- The trade off between time, features & bugs
- Given enough time all bugs are fixed
- Shipping is also a feature
- Reality is not all bugs will be found
- What is the cost of shipping a bug

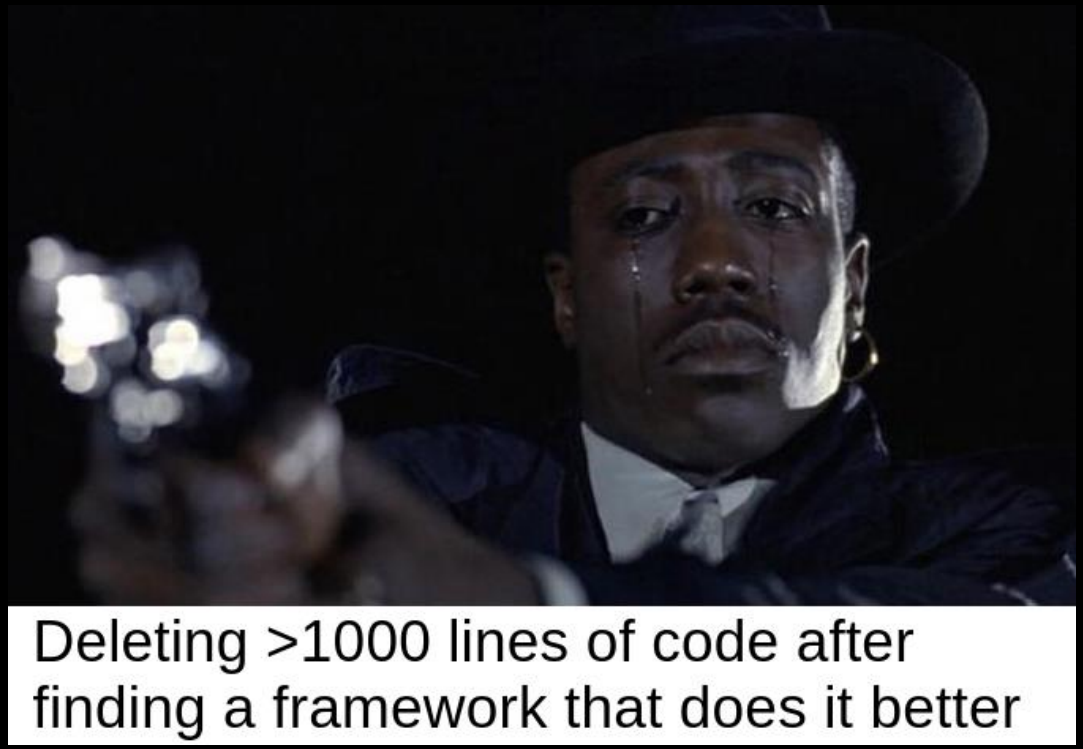


Think Outside The Box

- Writing code isn't always the answer
- Removed code doesn’t requireme maintenance

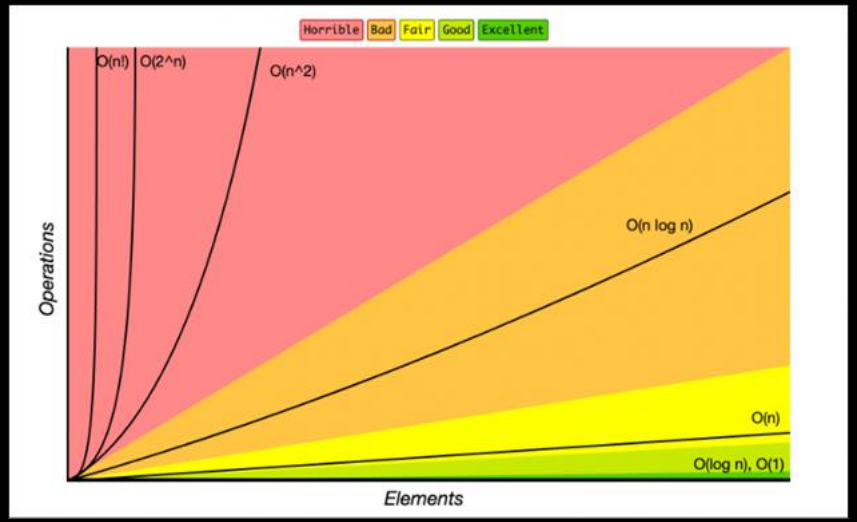
Other options

- Change system requirements
- Update the environment
- Use an external library
- Change the back-end
- Add hardware
- What about a RAM drive?



Big O Notation

- Measures worst-case time and space complexity based on input size
- Useful to consider performance of different algorithms



Six types/levels of complexity

1. Constant: O(1)
2. Linear time: O(n)
   - Single loop
3. Logarithmic time: O(n log n)
   - Recursion
4. Quadratic time: O(n^2)
   - Nested loops
5. Exponential time: O(2^n)
   - Doubles with each item
6. Factorial time: O(n!)
   - GAH!

https://www.freecodecamp.org/news/big-o-cheat-sheet-time-complexity-chart/
https://en.wikipedia.org/wiki/Big_O_notation
https://www.khanacademy.org/computing/computer-science/algorithms/asymptotic-notation/a/big-o-notation

Extremism

vs

Theory

vs

Pragmatism

Premature Optimization

- Avoid it
- Without data, an optimization may be wasted where it produces little results
- Don’t waste your time, profile first

Tools

- Profilers
- Unit Testing
- Code Coverage
- Static Code Analysis
- Logging
- What other tools give you evidence?

Profilers

- www.delphitools.info/samplingprofiler/
- www.prodelphi.de
- smartbear.com/product/aqtime-pro/
- yavfast.github.io/dbg-spider/
- github.com/ase379/gpprofile2017

CPU Specific

- Intel Vtune
- AMD µProf

- Apple Instruments
- List of Performance Analysis Tools

See what code is spending the most time, both per execution, and total number of executions

Find out where to optimize

Focus on slowest code with most executions and impacting most users

The xz Utils backdoor was discovered via micro-benchmarking

Unit Testing

- Dunit

  docwiki/RADStudio/en/DUnit_Overview
- DunitX

  github.com/VSoftTechnologies/DUnitX
- TestInsight

  bitbucket.org/sglienke/testinsight/wiki/Home

Logging

- raize.com/codesite
- code-partners.com/offerings/smartinspect
- github.com/grijjy/GrijjyCloudLogger
- www.madexcept.com
- www.eurekalog.com
- mORMot SynLog

Logging and Instrumentation

Code Coverage

- github.com/DelphiCodeCoverage/DelphiCodeCoverage
- github.com/MHumm/delphi-code-coverage-wizard-plus
- https://sourceforge.net/projects/discoverd/
- AQTime
- SmartInspect
- CodeInsite

Static Code Analysis

- www.tmssoftware.com/site/fixinsight.asp
- github.com/Embarcadero/SonarDelphi
- Peganza www.peganza.com
  - Pascal Analyzer
  - Pascal Expert
- derscanner.com
- github.com/SourceMonitor/SM-Info
- socksoftware.com/codehealer.php
- github.com/MikhailIzvekov/DelphiSCA
- github.com/RomanYankovsky/DelphiAST