



# Delphi Summit

June 2024 | Amsterdam

# Evidence Based Delphi Engineering:

How to ignore “best practices”  
and so called “experts.”

Jim McKeeth  
jim@gdksoftware.com  
gdksoftware.com

The *Global* **Delphi** Summit

*Organized by*



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

# About Jim McKeeth

- Director of GDK Software, USA & Delphi MVP
- Previously Embarcadero Chief Dev Advocate
- Over 30 years experience in software development
- Worked professionally in many languages and platforms, including Delphi, C#, C/C++, Java, JavaScript, and Python
- Multiple technology patents & book contributions
- Spoken on software development all around the world
- Professional improv comedy performer for 10 years



**Delphi Summit**

June 2024 | Amsterdam



# Delphi Summit

June 2024 | Amsterdam

# BEGIN

# INTERFACE

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

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

[github.com/jimmckeeth/  
Evidence-Based-Software-Engineering](https://github.com/jimmckeeth/Evidence-Based-Software-Engineering)



**Delphi Summit**

June 2024 | Amsterdam



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."*



**Delphi Summit**

June 2024 | Amsterdam

[https://en.wikipedia.org/wiki/Grace\\_Hopper](https://en.wikipedia.org/wiki/Grace_Hopper)

# Margaret Hamilton

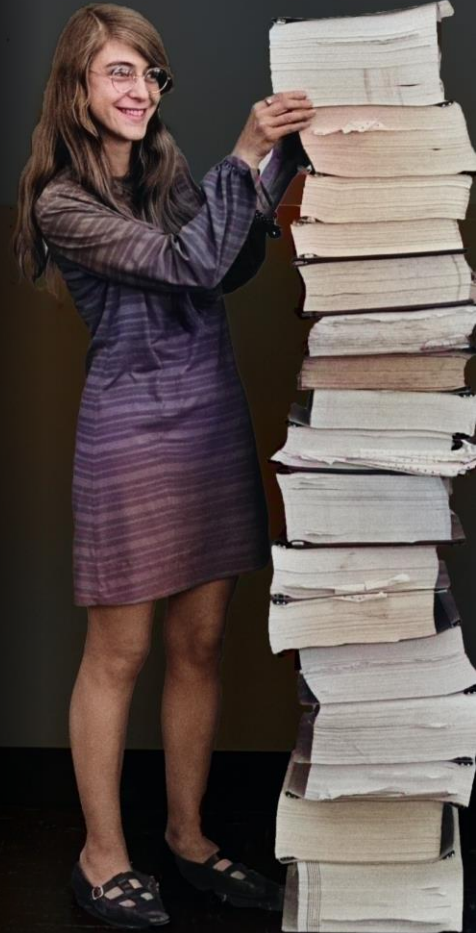
- Director of the Software Engineering Division of the MIT Instrumentation Laboratory
- Developed flight software for the Apollo program
- 2016 – Presidential Medal of Freedom from Obama
- Published over 130 papers
- Invented the term “software engineering”
  - “...to distinguish it from hardware and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



**Delphi Summit**

June 2024 | Amsterdam

[https://en.wikipedia.org/wiki/Margaret\\_Hamilton\\_\(software\\_engineer\)](https://en.wikipedia.org/wiki/Margaret_Hamilton_(software_engineer))





# Women of NASA

## LEGO Set # 21312



**Delphi Summit**

June 2024 | Amsterdam







# Delphi Summit

June 2024 | Amsterdam

# IMPLEMENTATION

Don't believe  
**anything** I tell you!



**Delphi Summit**

June 2024 | Amsterdam



***TRAVEL  
BACK  
IN  
TIME...***

Circa 2007



# 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...



**Delphi Summit**

June 2024 | Amsterdam



# Superstitions

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



**Delphi Summit**

June 2024 | Amsterdam

The question  
I'm always asked

What is the *best* ...



**Delphi Summit**

June 2024 | Amsterdam

# 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



**Delphi Summit**

June 2024 | Amsterdam

# 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....



**Delphi Summit**

June 2024 | Amsterdam



The answer  
is always

It depends ...

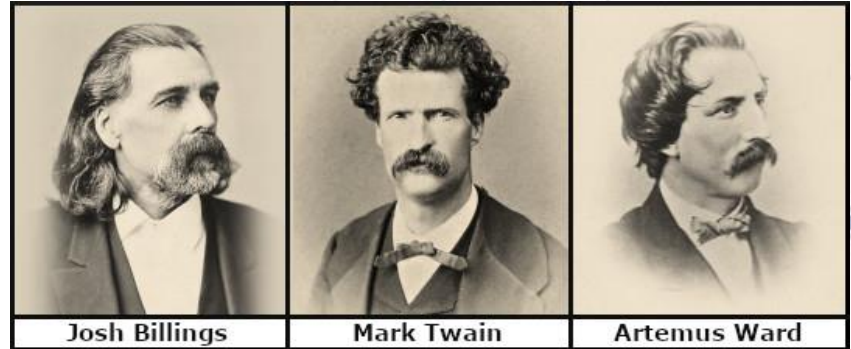


**Delphi Summit**

June 2024 | Amsterdam

”It ain’t what you don’t know  
that gets you into trouble.

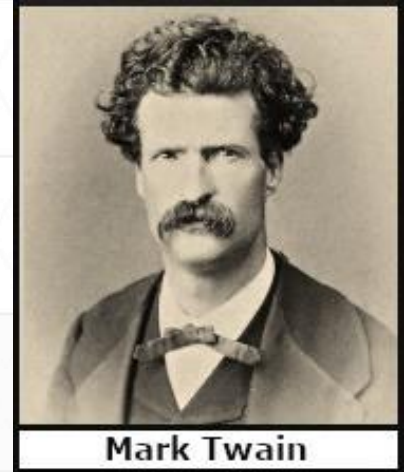
It’s what you know for sure  
that just ain’t so.



**Delphi Summit**

June 2024 | Amsterdam

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



**Delphi Summit**

June 2024 | Amsterdam

# Unfounded confidence gets us in trouble



**Delphi Summit**

June 2024 | Amsterdam



# 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*



**Delphi Summit**

June 2024 | Amsterdam

"Programming is an ART  
that fights back."

-Chad "Kudzu" Howard

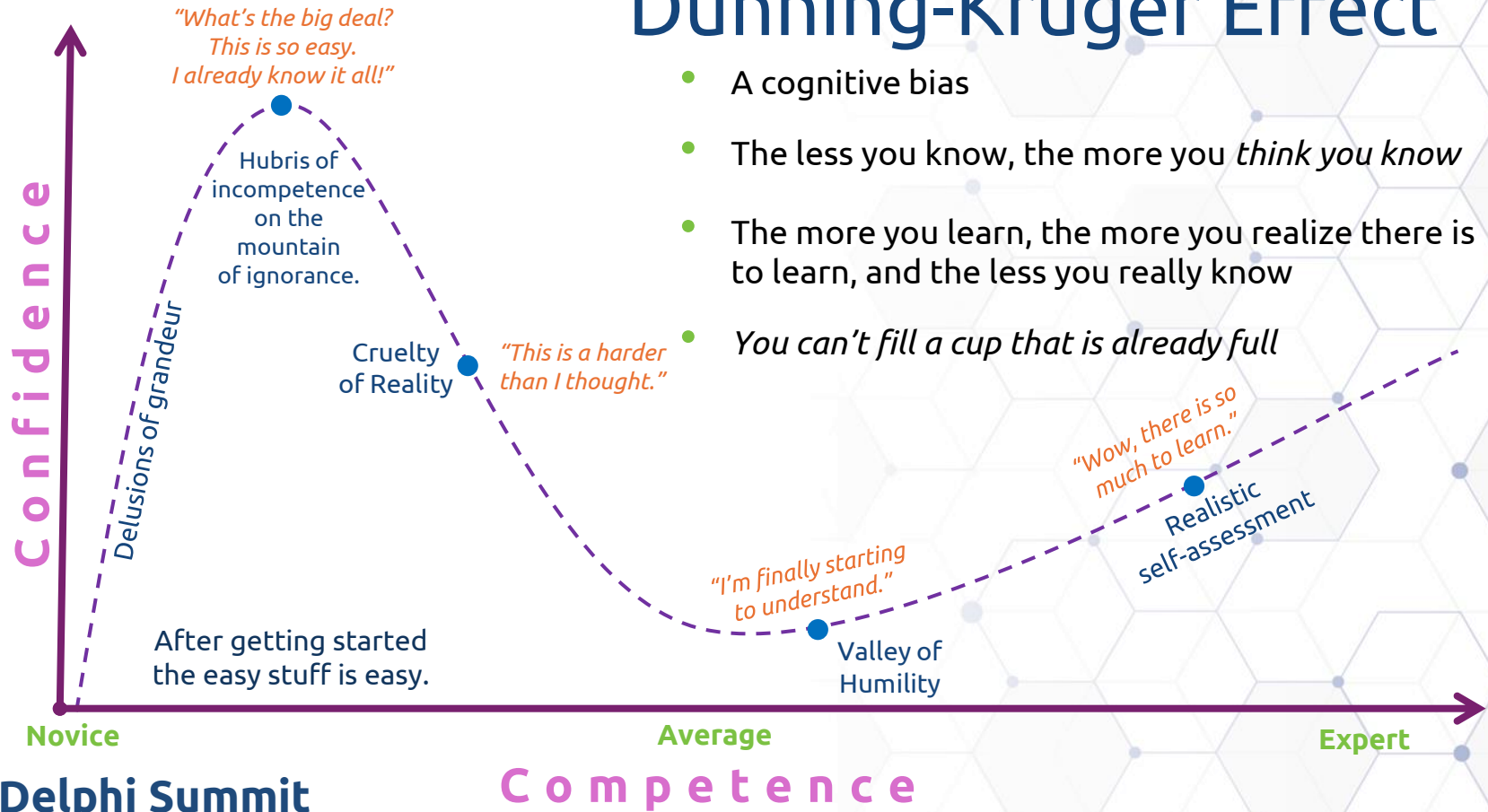


**Delphi Summit**

June 2024 | Amsterdam

# Dunning-Kruger Effect

- A cognitive bias
- The less you know, the more you *think you know*
- The more you learn, the more you realize there is to learn, and the less you really know
- *You can't fill a cup that is already full*



**Delphi Summit**

June 2024 | Amsterdam

# Software Developer Levels

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



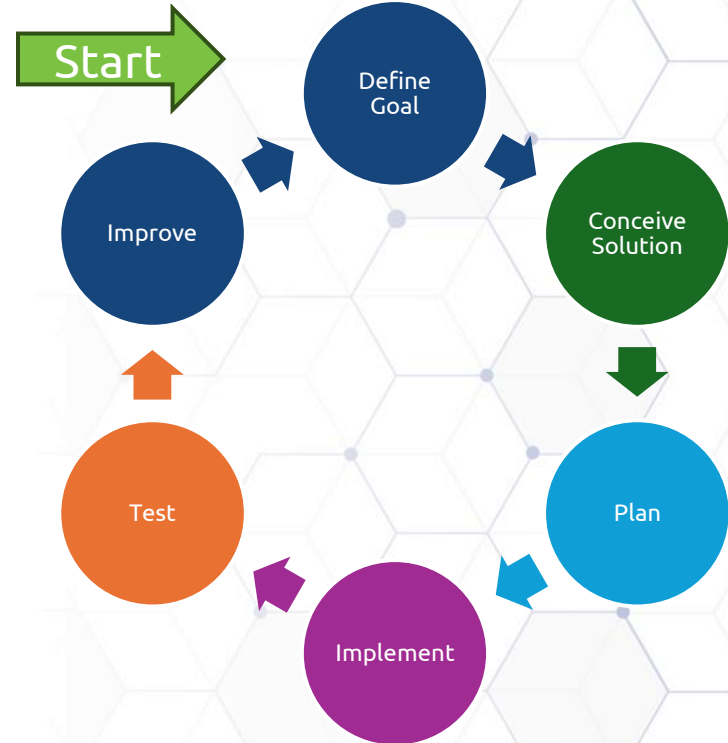
**Delphi Summit**

June 2024 | Amsterdam



# 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



# The Math of Bugs and Fixes

- The source of bugs is writing code
- 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 number of bugs found

Features

Create

Bugs

Require

Time

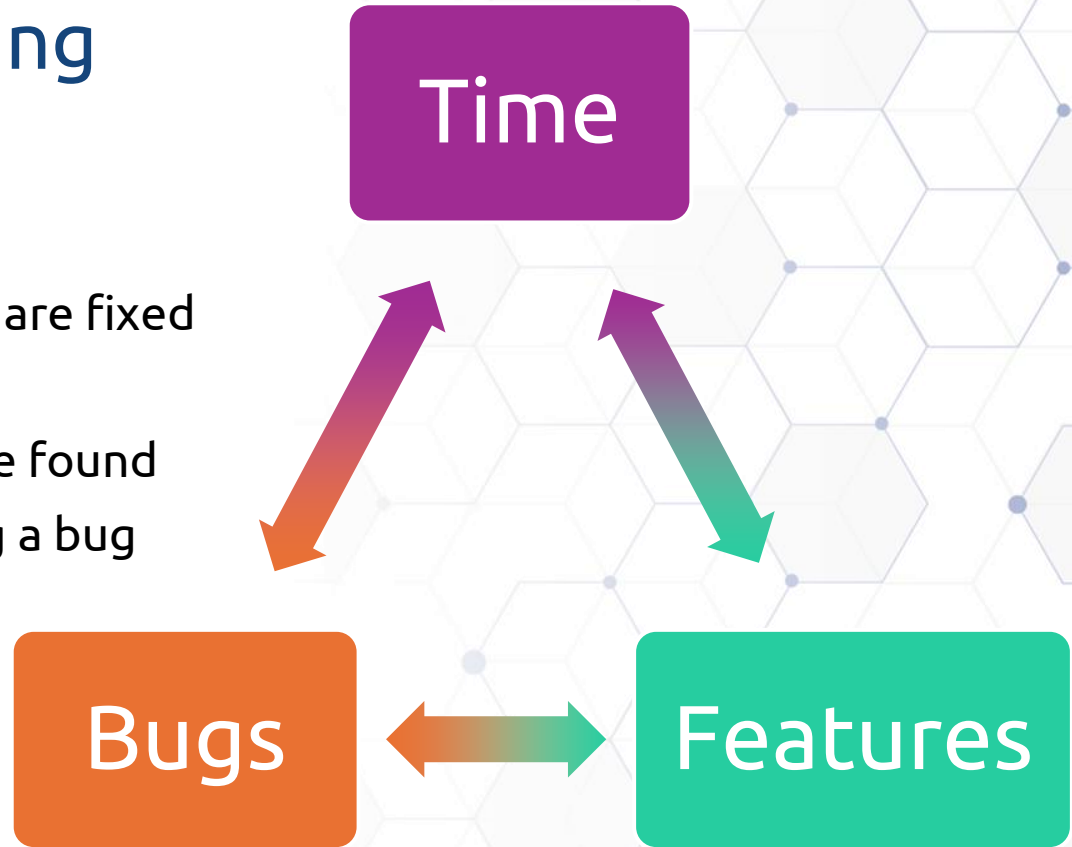


Delphi Summit

June 2024 | Amsterdam

# Software Engineering

- It is a 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 need to be maintained

### Other options

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



Deleting >1000 lines of code after finding a framework that does it better

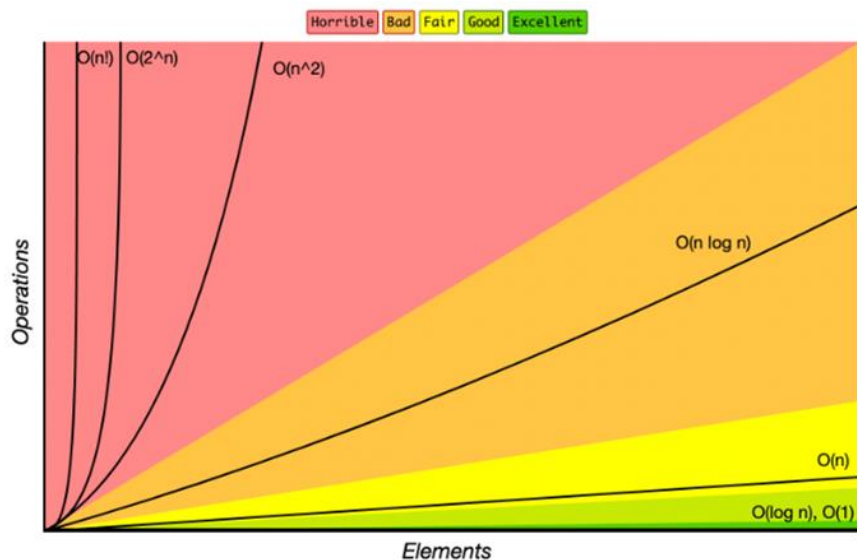


**Delphi Summit**

June 2024 | Amsterdam

# 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

- Constant:  $O(1)$ 
  - Same time for any data size
- Linear time:  $O(n)$ 
  - Single loop
- Logarithmic time:  $O(n \log n)$ 
  - Recursion
- Quadratic time:  $O(n^2)$ 
  - Nested loops
- Exponential time:  $O(2^n)$ 
  - Doubles with each item
- Factorial time:  $O(n!)$ 
  - GAH!

[freecodecamp.org/news/big-o-cheat-sheet-time-complexity-chart/](https://freecodecamp.org/news/big-o-cheat-sheet-time-complexity-chart/)  
[en.wikipedia.org/wiki/Big\\_O\\_notation](https://en.wikipedia.org/wiki/Big_O_notation)  
[khanacademy.org/computing/computer-science/algorithms/asymptotic-notation/a/big-o-notation](https://khanacademy.org/computing/computer-science/algorithms/asymptotic-notation/a/big-o-notation)



# Putting Theory into Practice

## Extremism vs. Pragmatism



**Delphi Summit**

June 2024 | Amsterdam

# Different Types of Code

## Application Code

- The majority of development
- Applications that solve problems for end users
- Very focused on productivity
- Performance is less important
- *Underengineered*

## Library Code

- Used in applications
- Focused on correctness, robustness, and reusability
- Testing is more important than shipping
- Performance is very important
- *Overengineered*

Both still require good, clean code



**Delphi Summit**

June 2024 | Amsterdam

# The FizzBuzz Example

For numbers 1 through 100,

- if the number is divisible by 3 print Fizz;
- if the number is divisible by 5 print Buzz;
- if the number is divisible by 3 and 5 (15) print FizzBuzz;
- else, print the number.

1

2

Fizz

4

Buzz

Fizz

7

8

Fizz

Buzz

11

Fizz

13

14

FizzBuzz

...



**Delphi Summit**

June 2024 | Amsterdam

# FizzBuzz : Application Solution

```
procedure RunFizzBuzz;  
begin  
  for var i := 1 to 100 do  
    begin  
      if (i mod 3 = 0) and (i mod 5 = 0) then  
        WriteLn('FizzBuzz')  
      else if i mod 3 = 0 then  
        WriteLn('Fizz')  
      else if i mod 5 = 0 then  
        WriteLn('Buzz')  
      else  
        WriteLn(i);  
    end;  
  end;  
end;
```

1  
2  
Fizz  
4  
Buzz  
Fizz  
7  
8  
Fizz  
Buzz  
11  
Fizz  
13  
14  
FizzBuzz  
...



**Delphi Summit**

June 2024 | Amsterdam



# The “Enterprise” Solution

- Heavy use of Factory pattern, Dependency Injection, and the Strategy pattern
- Everything has an object, every object has an interface
- A processor to handle the rules
- Modularity to add or remove rules as needed
- 200+ line example in Delphi:

[github.com/jimmckeeth/FizzBuzzEnterpriseEdition-Delphi](https://github.com/jimmckeeth/FizzBuzzEnterpriseEdition-Delphi)

- *This is a joke to illustrate a point.*
- See also:
  - [github.com/jongee1/FizzBuzzEnterpriseEdition-CSharp](https://github.com/jongee1/FizzBuzzEnterpriseEdition-CSharp) 48 C# files, over 6 projects
  - [github.com/EnterpriseQualityCoding/FizzBuzzEnterpriseEdition](https://github.com/EnterpriseQualityCoding/FizzBuzzEnterpriseEdition) 89 Java files

1  
2  
Fizz  
4  
Buzz  
Fizz  
7  
8  
Fizz  
Buzz  
11  
Fizz  
13  
14  
FizzBuzz  
...



**Delphi Summit**

June 2024 | Amsterdam

# So What?

- Most of the time we are writing application code
- ***Don't over engineer it***
- ***Avoid premature optimization***
- Use safeguards
- Still write clean code
- Still use tests and best practices
- ***Get it done and ship the application***

1

2

Fizz

4

Buzz

Fizz

7

8

Fizz

Buzz

11

Fizz

13

14

FizzBuzz

...



**Delphi Summit**

June 2024 | Amsterdam

# Premature Optimization

- Avoid it
- Without data any optimization:
  - May be counter productive
  - Produce little impact
  - Not worth the effort
- Don't waste your time, *profile first*



# Tools

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



**Delphi Summit**

June 2024 | Amsterdam



# Profilers

- [delphitools.info/samplingprofiler](https://delphitools.info/samplingprofiler)
- [prodelphi.de](https://prodelphi.de)
- [smartbear.com/product/aqtime-pro](https://smartbear.com/product/aqtime-pro)
- [yavfast.github.io/dbg-spider](https://yavfast.github.io/dbg-spider)
- [github.com/ase379/gpprofile2017](https://github.com/ase379/gpprofile2017)

## CPU Specific

- [Intel Vtune](#) + [map2pdb](#)
- [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*



**Delphi Summit**

June 2024 | Amsterdam

# Unit Testing

- Dunit

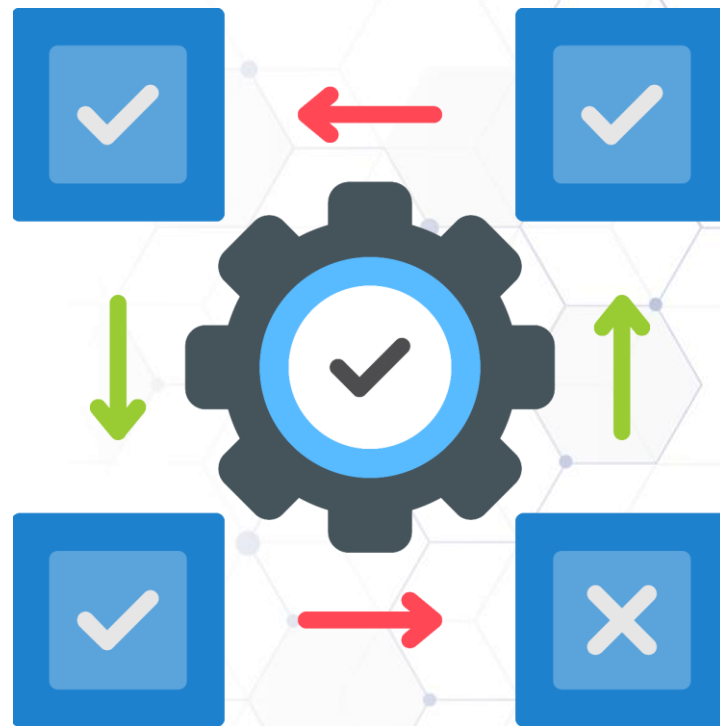
[docwiki/RADStudio/en/DUnit Overview](https://docwiki/RADStudio/en/DUnit%20Overview)

- DunitX 🙌

[github.com/VSoftTechnologies/DUnitX](https://github.com/VSoftTechnologies/DUnitX)

- TestInsight 🙌

[bitbucket.org/sglienke/testinsight/wiki](https://bitbucket.org/sglienke/testinsight/wiki)



**Delphi Summit**

June 2024 | Amsterdam

# Logging

## Logging

- [raize.com/codesite](https://raize.com/codesite) *with Method Tracer!*
- [code-partners.com/offerings/smartinspect](https://code-partners.com/offerings/smartinspect)
- [github.com/grijjy/GrijjyCloudLogger](https://github.com/grijjy/GrijjyCloudLogger)
- [madexcept.com](https://madexcept.com)
- [eurekalog.com](https://eurekalog.com)



**Delphi Summit**

June 2024 | Amsterdam

# Code Coverage

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



**Delphi Summit**

June 2024 | Amsterdam



# Static Code Analysis

- [www.tmssoftware.com/site/fixinsight.asp](http://www.tmssoftware.com/site/fixinsight.asp)
- [github.com/Embarcadero/SonarDelphi](https://github.com/Embarcadero/SonarDelphi)
- [peganza.com](http://peganza.com) Pascal Analyzer & Pascal Expert
- [derscanner.com](http://derscanner.com)

## DIY

- [github.com/RomanYankovsky/DelphiAST](https://github.com/RomanYankovsky/DelphiAST)

## Outdated

- [github.com/SourceMonitor/SM-Info](https://github.com/SourceMonitor/SM-Info)
- [socksoftware.com/codehealer.php](http://socksoftware.com/codehealer.php)
- [github.com/Mikhailzvekov/DelphiSCA](https://github.com/Mikhailzvekov/DelphiSCA)



**Delphi Summit**

June 2024 | Amsterdam

# FINALIZATION

- Don't believe what you “know”
- Test
- Collect data
- Use the source & tools
- [github.com/jimmckeeth/Evidence-Based-Software-Engineering](https://github.com/jimmckeeth/Evidence-Based-Software-Engineering)
- [jim@gdksoftware.com](mailto:jim@gdksoftware.com)





# Delphi Summit

June 2024 | Amsterdam

# END.

[github.com/jimmckeeth/Evidence-Based-Software-Engineering](https://github.com/jimmckeeth/Evidence-Based-Software-Engineering)

[jim@gdksoftware.com](mailto:jim@gdksoftware.com)