University of Sheffield

Legacy Systems

Software Reengineering
(COM3523 / COM6523)

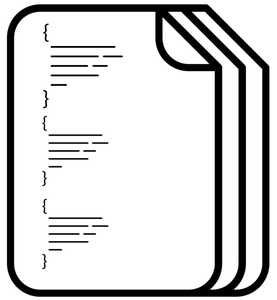
The University of Sheffield

Software systems are multifaceted and complex.

Further reading in "Software Systems Architecture (Second Edition)" by Nick Rozanski and Eoin Woods

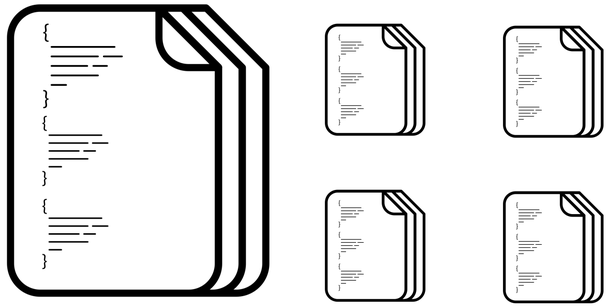
Methods, functions

Methods, functions



Modules, packages

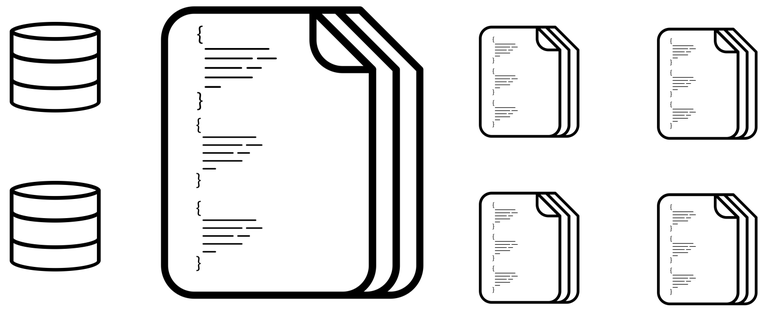
Methods, functions



Modules, packages

Data storage

Methods, functions

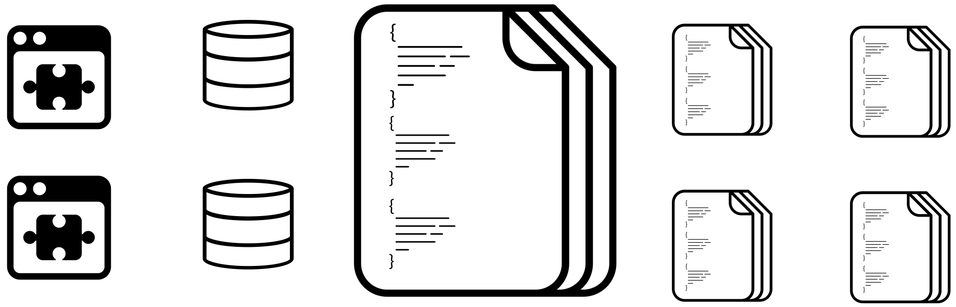


Modules, packages

Libraries, components

Data storage

Methods, functions



Modules, packages

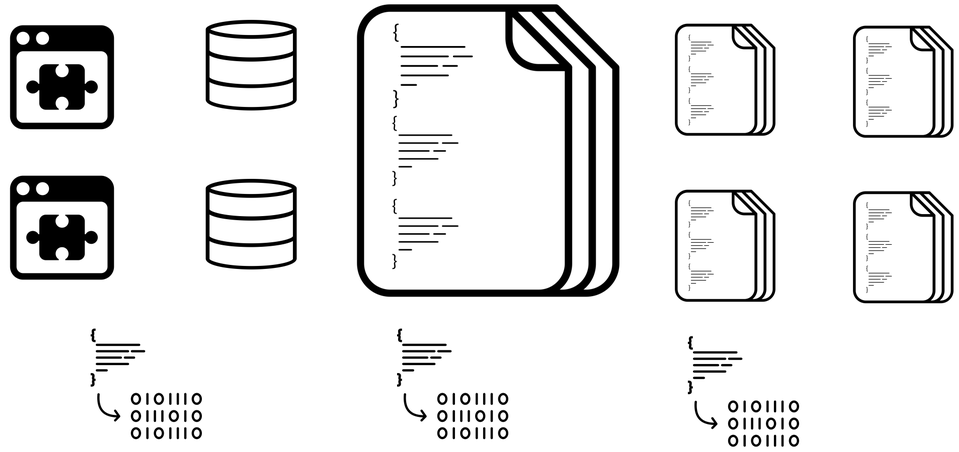
Libraries, components

Data storage

Methods, functions

Compilers, interpreters

Virtual machines



Modules, packages

Libraries, components

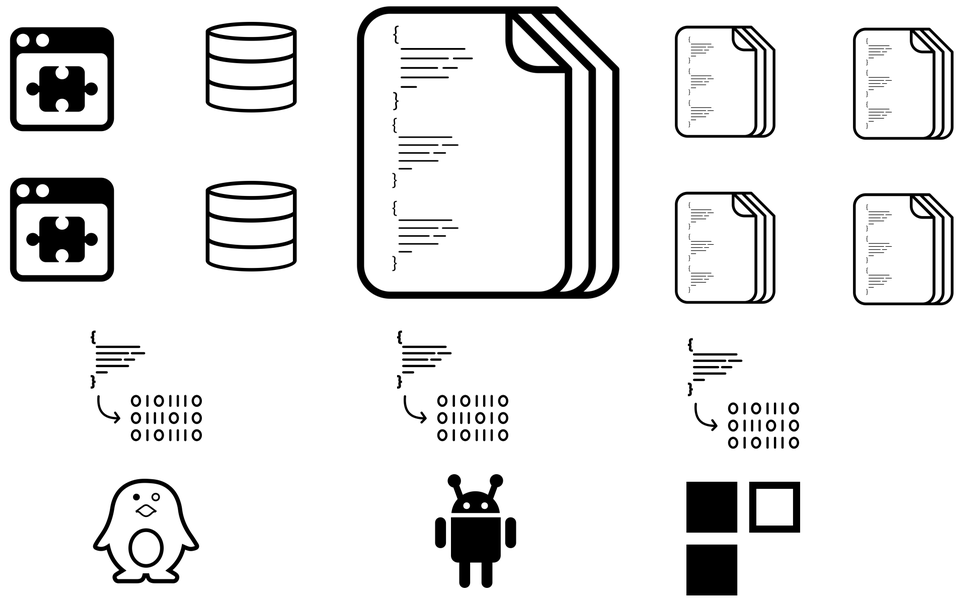
Data storage

Methods, functions

Compilers, interpreters

Virtual machines

Operating Systems



Modules, packages

Libraries, components

Data storage

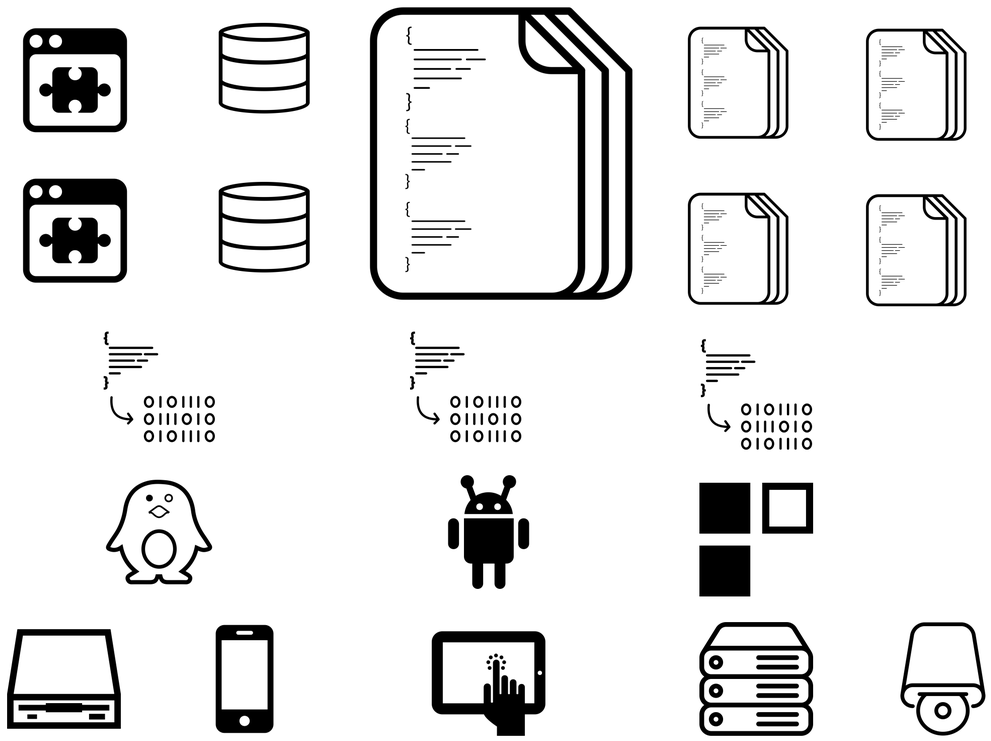
Methods, functions

Compilers, interpreters

Virtual machines

Operating Systems

Hardware platforms & devices



Frontend / interfaces
Web / Windows / CLI

Modules, packages

Libraries, components

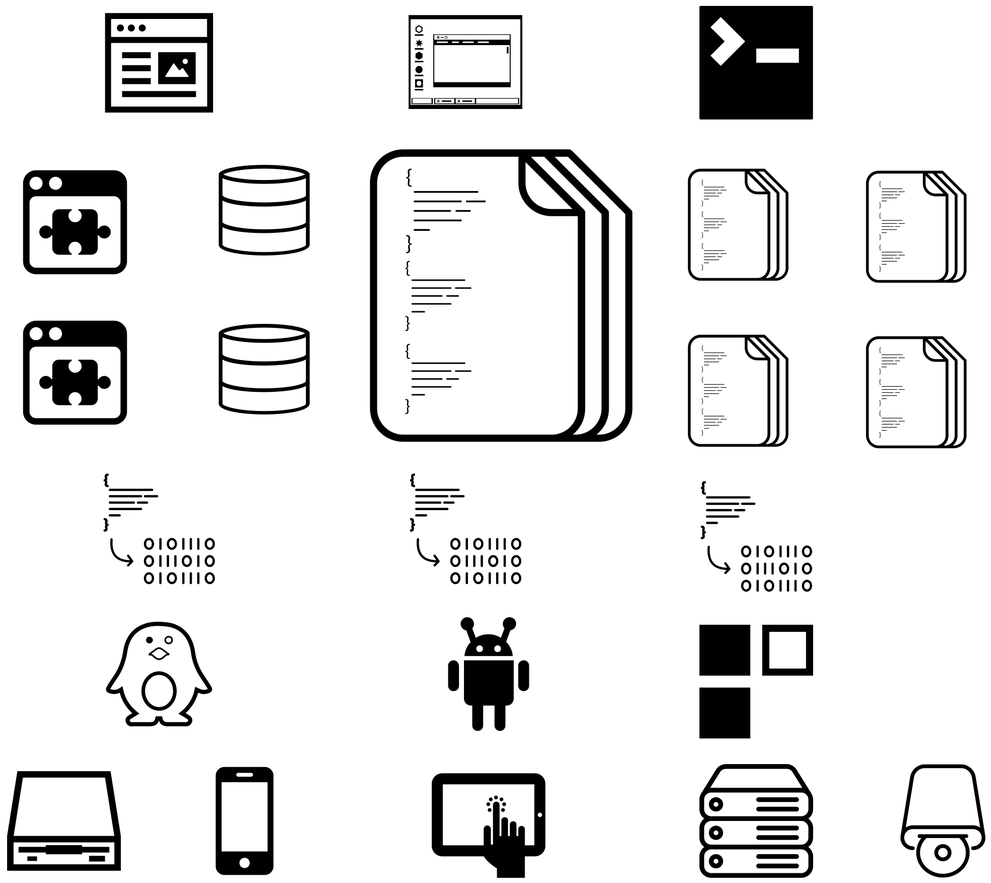
Data storage

Methods, functions

Compilers, interpreters
Virtual machines

Operating Systems

Hardware platforms
& devices



Users, misusers, developers, managers

Frontend / interfaces
Web / Windows / CLI

Modules, packages

Libraries, components

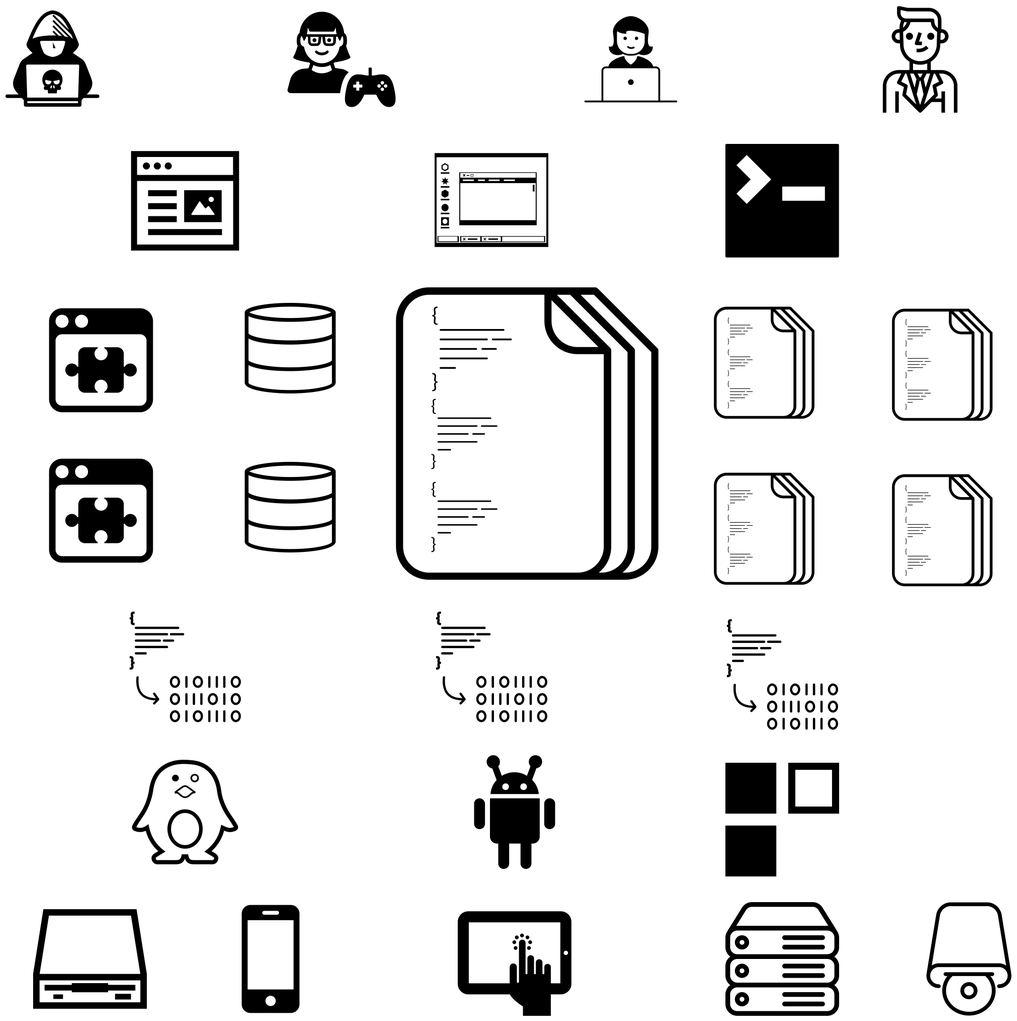
Data storage

Methods, functions

Compilers, interpreters
Virtual machines

Operating Systems

Hardware platforms & devices



Processes / workflows, Organisational structures

Users, misusers, developers, managers

Frontend / interfaces Web / Windows / CLI

Modules, packages

Libraries, components

Data storage

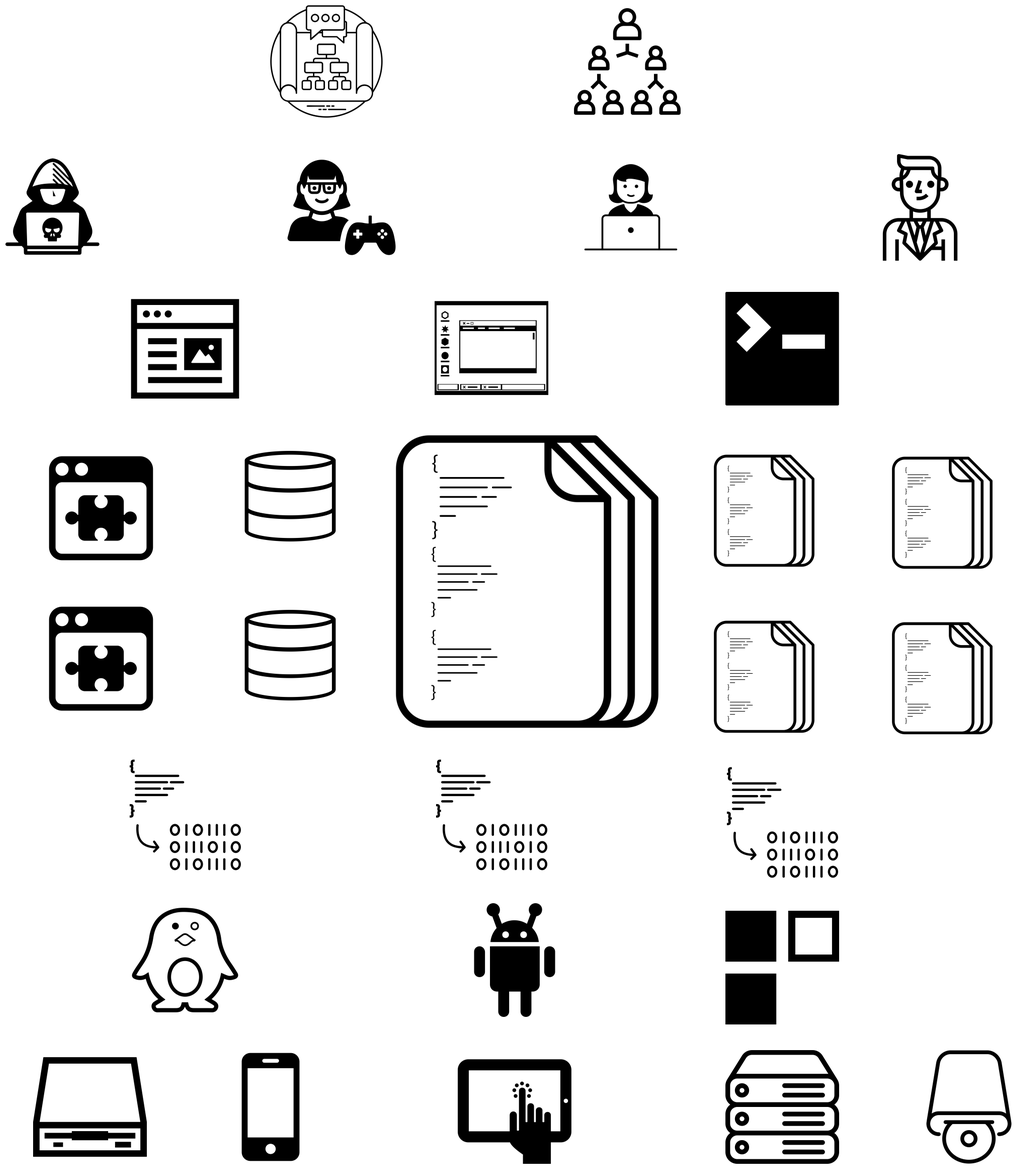
Methods, functions

Compilers, interpreters

Virtual machines

Operating Systems

Hardware platforms & devices



Legal requirements, Industry standards, market forces

Processes / workflows, Organisational structures

Users, misusers, developers, managers

Frontend / interfaces Web / Windows / CLI

Modules, packages

Libraries, components

Data storage

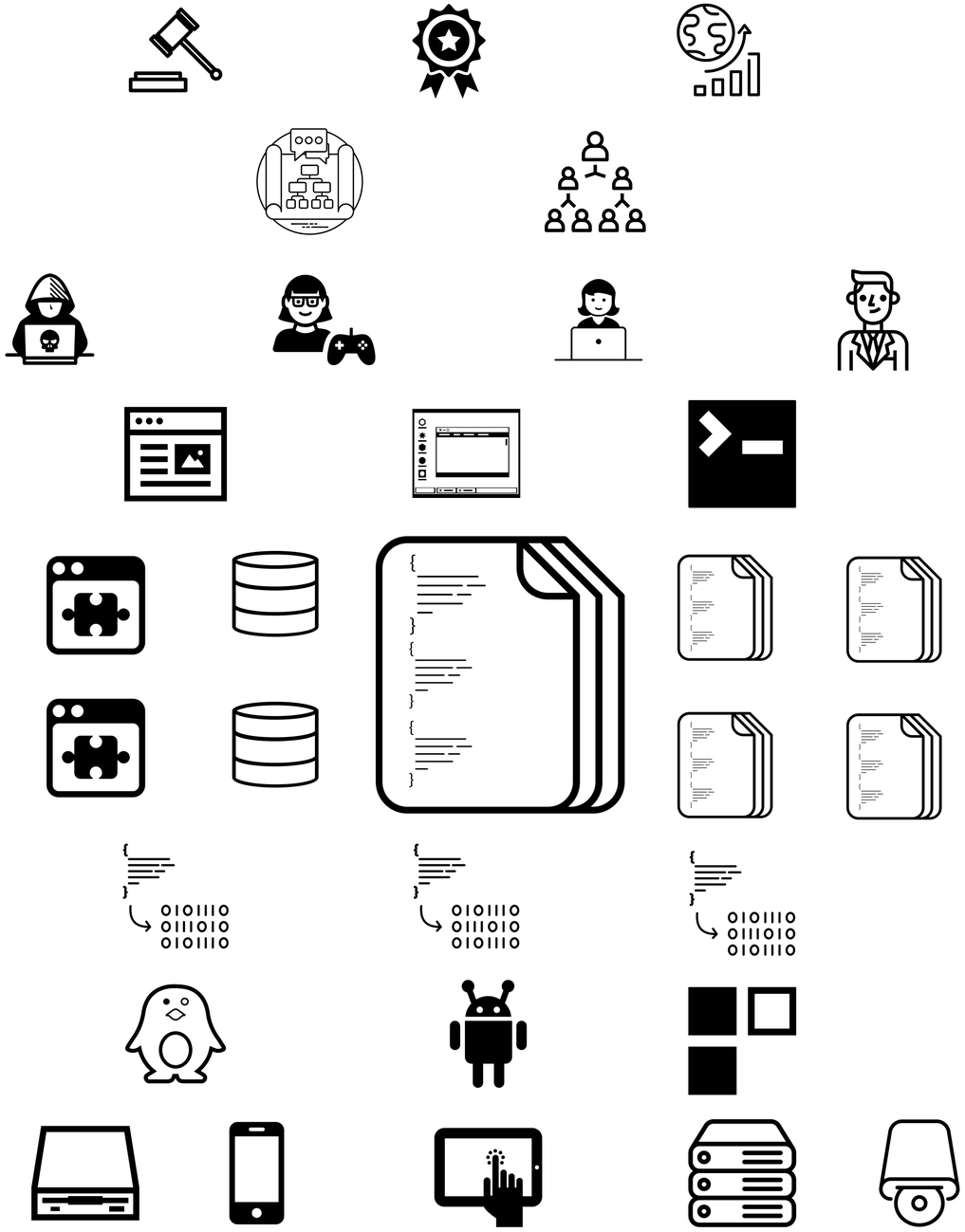
Methods, functions

Compilers, interpreters

Virtual machines

Operating Systems

Hardware platforms & devices



Legal requirements, Industry standards, market forces

Processes / workflows, Organisational structures

Users, misusers, developers, managers

Frontend / interfaces Web / Windows / CLI

Modules, packages

Libraries, components

Data storage

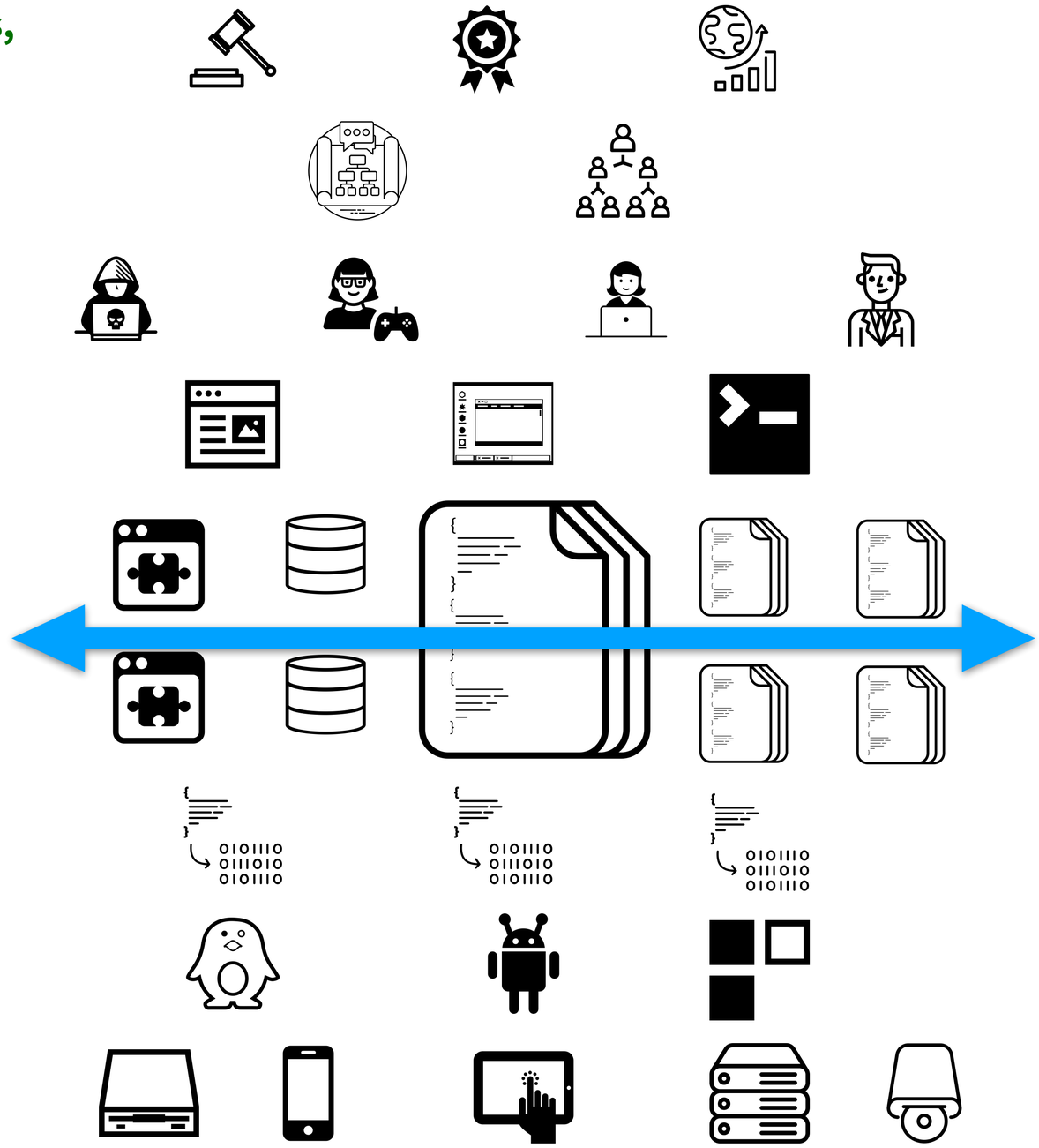
Methods, functions

Compilers, interpreters

Virtual machines

Operating Systems

Hardware platforms & devices



Legal requirements, Industry standards, market forces

Processes / workflows, Organisational structures

Users, misusers, developers, managers

Frontend / interfaces Web / Windows / CLI

Modules, packages

Libraries, components

Data storage

Classes, files

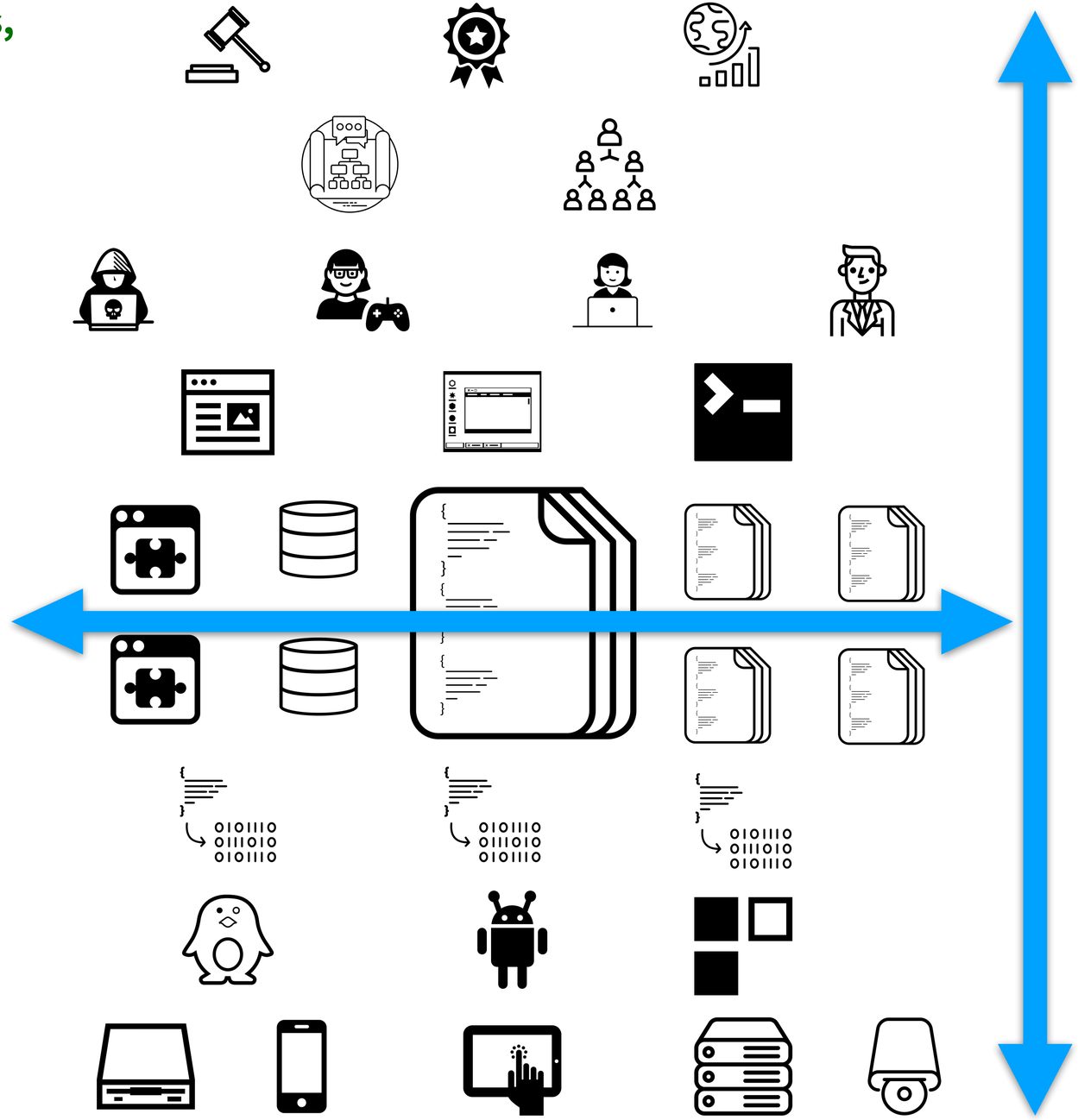
Methods, functions

Compilers, interpreters

Virtual machines

Operating Systems

Hardware platforms & devices



Viewpoints

Viewpoint	Description

Collection of patterns, templates, and conventions for constructing one type of view.

Defines the relevant stakeholders, and the relevant guidelines, principles and template models for constructing the views.

Viewpoints

Viewpoint	Description
Context	Describes the relationships, dependencies, and interactions between the system and its environment.

Collection of patterns, templates, and conventions for constructing one type of view.

Defines the relevant stakeholders, and the relevant guidelines, principles and template models for constructing the views.

Viewpoints

Viewpoint	Description
Context	Describes the relationships, dependencies, and interactions between the system and its environment.
Functional	Functional elements, their responsibilities and interactions. Traditionally the starting point for an architectural description.

Defines the relevant stakeholders, and the relevant guidelines, principles and template models for constructing the views.

Viewpoints

Viewpoint	Description
Context	Describes the relationships, dependencies, and interactions between the system and its environment.
Functional	Functional elements, their responsibilities and interactions. Traditionally the starting point for an architectural description.
Information	How the architecture stores, manipulates, manages and distributes information. Content, structure, ownership, latency, etc.

Viewpoints

Viewpoint	Description
Context	Describes the relationships, dependencies, and interactions between the system and its environment.
Functional	Functional elements, their responsibilities and interactions. Traditionally the starting point for an architectural description.
Information	How the architecture stores, manipulates, manages and distributes information. Content, structure, ownership, latency, etc.
Concurrency	Maps functional elements to concurrency units to clearly identify which parts of the system are synchronous or asynchronous.

Viewpoints

Viewpoint	Description
Context	Describes the relationships, dependencies, and interactions between the system and its environment.
Functional	Functional elements, their responsibilities and interactions. Traditionally the starting point for an architectural description.
Information	How the architecture stores, manipulates, manages and distributes information. Content, structure, ownership, latency, etc.
Concurrency	Maps functional elements to concurrency units to clearly identify which parts of the system are synchronous or asynchronous.
Development	How does the architecture support development - building, testing, maintaining, and enhancing the system.

Viewpoints

Viewpoint	Description
Context	Describes the relationships, dependencies, and interactions between the system and its environment.
Functional	Functional elements, their responsibilities and interactions. Traditionally the starting point for an architectural description.
Information	How the architecture stores, manipulates, manages and distributes information. Content, structure, ownership, latency, etc.
Concurrency	Maps functional elements to concurrency units to clearly identify which parts of the system are synchronous or asynchronous.
Development	How does the architecture support development - building, testing, maintaining, and enhancing the system.
Deployment	What environment will the system be deployed to? Dependencies on the runtime environment, network connections, etc.

Viewpoints

Viewpoint	Description
Context	Describes the relationships, dependencies, and interactions between the system and its environment.
Functional	Functional elements, their responsibilities and interactions. Traditionally the starting point for an architectural description.
Information	How the architecture stores, manipulates, manages and distributes information. Content, structure, ownership, latency, etc.
Concurrency	Maps functional elements to concurrency units to clearly identify which parts of the system are synchronous or asynchronous.
Development	How does the architecture support development - building, testing, maintaining, and enhancing the system.
Deployment	What environment will the system be deployed to? Dependencies on the runtime environment, network connections, etc.
Operational	How the system will be operated, administered, etc.

Perspectives

Perspective	Description
Accessibility	The ability of the system to be used by people with disabilities
Availability & Resilience	Ability to be operational as and when required, and to handle failures that could affect availability.
Development Resource	Ability to be designed, build, deployed, and operated within constraints around people, budget, time, materials.
Evolution	Ability to be flexible in the face of inevitable change, balanced against costs of providing this flexibility.
Internationalisation	Ability to be independent from any particular language, country or cultural group.
Location	Ability to overcome problems brought about by the location of its elements and distances between them.
Performance & Scalability	Ability to predictably execute within mandated performance profile, and handle increasing volumes.
Regulation	Ability to conform to local and international laws, quasi-legal regulations, company policies, etc.
Security	Ability to reliably control, monitor, and audit who can perform what mechanisms on what resources.
Usability	Ease with which people can interact with the system and work effectively.

Successful systems change. A lot.

Hardware

Hardware is constantly improving.

- E.g. Rise of GPUs for parallel computing.
- Moore's law: ICs double transistors every two years

Difficult to replace.

- Can be physically difficult to extract.
- E.g. circuitry that is built into aircraft.
- Embedded components can include hidden logic.
- Difficult to reverse-engineer if not documented.

Expensive!

- Once you have purchased it, you have to make it work.

Difficult to test.

The US nuclear arsenal relied on 8" floppy disks until 2019!

A 5MB (>1 ton) hard disk being loaded into a plane in 1956.

https://www.nytimes.com/2019/10/24/us/nuclear-weapons-floppy-disks.html

Operating Systems

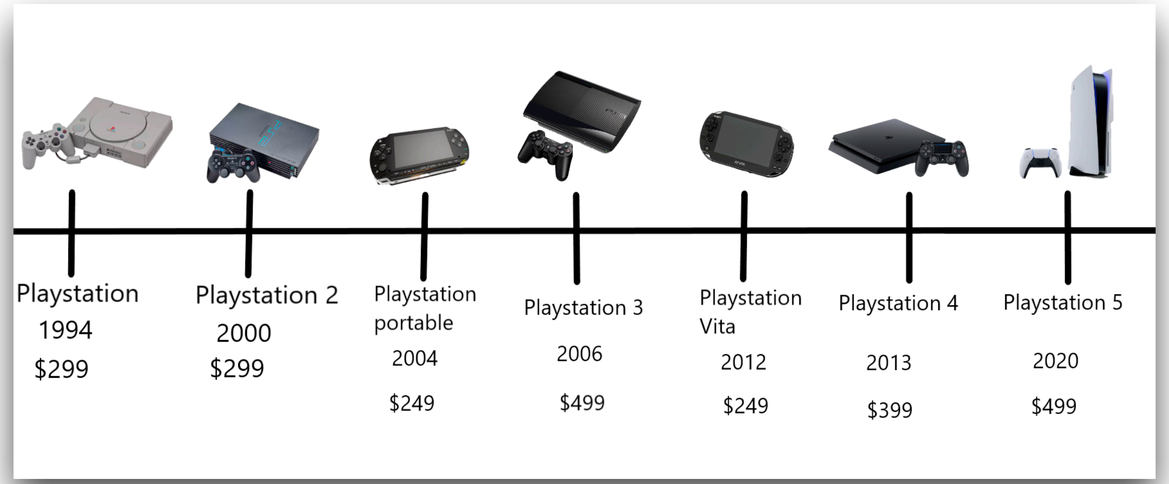
Constantly evolving to incorporate changes in hardware.

- Vendors eventually remove support for older versions.
- Can force the issue when OS support is relied upon.

Can be challenging to adapt legacy code to.

- New filesystems
- Changes in security mechanisms.
- Removal of support for particular hardware.

OS/2 Warp (1995) remains one of the leading ATM operating systems.



The Sony Playstation product line

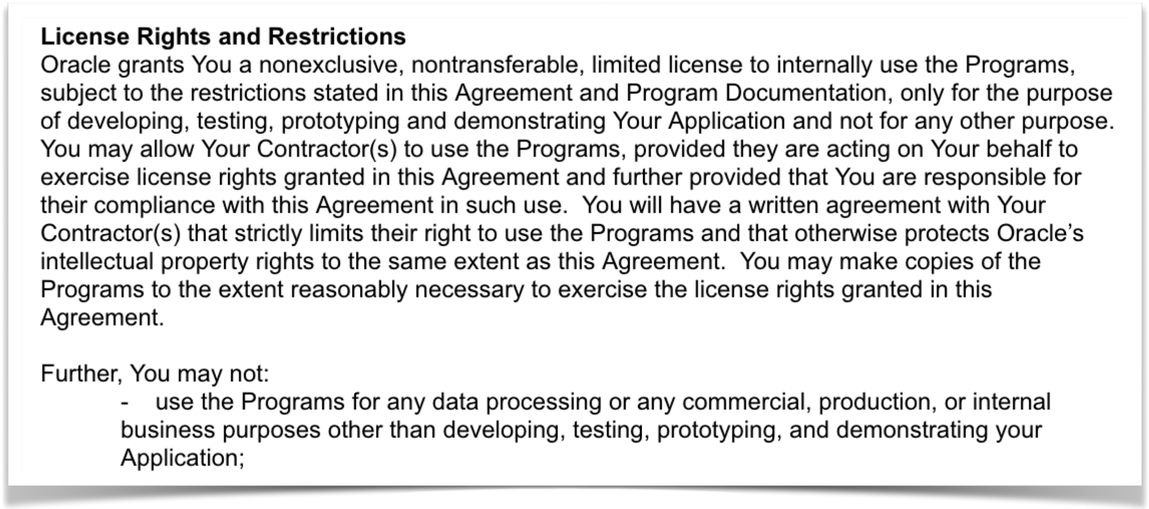
Compilers, Virtual Machines, Interpreters

Often tailored to hardware and OS combinations.

Often associated with development kits.

Can require change for many reasons:

- Deprecation of language features.
- Removal of support.
- License changes.
- Changes in application needs.
  - E.g. Older compilers cannot handle Unicode text.



License Rights and Restrictions
Oracle grants You a nonexclusive, nontransferable, limited license to internally use the Programs, subject to the restrictions stated in this Agreement and Program Documentation, only for the purpose of developing, testing, prototyping and demonstrating Your Application and not for any other purpose. You may allow Your Contractor(s) to use the Programs, provided they are acting on Your behalf to exercise license rights granted in this Agreement and further provided that You are responsible for their compliance with this Agreement in such use. You will have a written agreement with Your Contractor(s) that strictly limits their right to use the Programs and that otherwise protects Oracle's intellectual property rights to the same extent as this Agreement. You may make copies of the Programs to the extent reasonably necessary to exercise the license rights granted in this Agreement.

Further, You may not:

- use the Programs for any data processing or any commercial, production, or internal business purposes other than developing, testing, prototyping, and demonstrating your Application;

In 2011 Oracle changed the Java compiler license, so that the free version could no longer be used for commercial software.

Source code

Lots of reasons for source code to change.

- Bug fixes, feature additions, maintenance, refactoring, modularisation,

In many ways the most “malleable” part of a system.

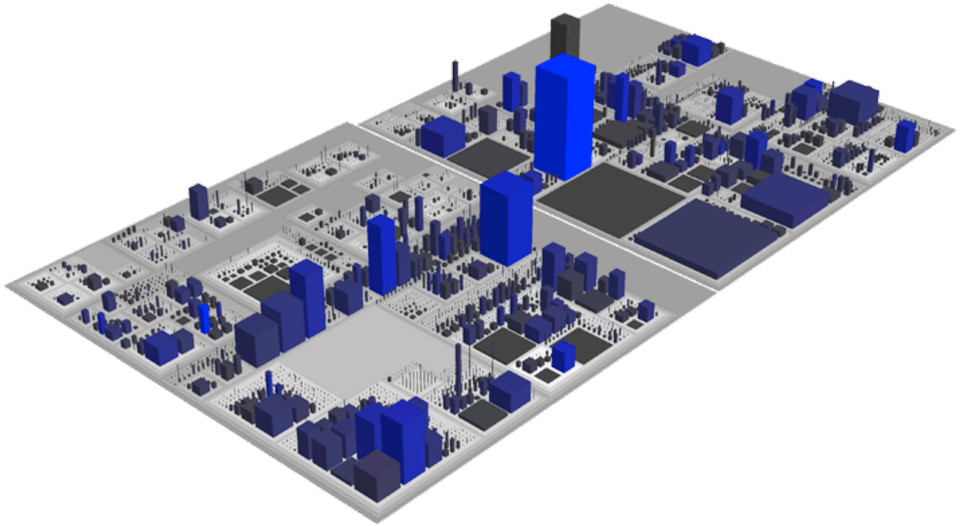
- Changes to source code can work-around changes further down the stack.

Successful code can be subject to continuous change.

- E.g. the Linux kernel.

Invariable increase in scale and complexity.

- Some of which is **essential** to implement necessary features.
- Some of which is **accidental** - to work around poor design choices.



CodeCity visualisation of a software system. Each block is a code file. Height depends on LOC.

Fred Brooks, “No Silver Bullet: Essence and Accidents in Software Engineering”, Computer (20),4, 1987

Data storage

Many drivers for change

- Change in legislation.
  - E.g. Change to rules about what types of data can be stored, for how long.
- Change in database system.
- Differences in reliability, efficiency, ease of use, compatibility, etc.

Change in schemas, storage formats

Migration can be extremely challenging.

- Data often always needs to be available.
- So do the systems that access and manipulate it.

Front end / UI

Several drivers for change

- Accessibility improvements
- Usability / UX

New modes of interaction

- AI-powered chat interfaces
- Touch gestures
- Motion sensing devices
- Voice activation

Changes in Browser / GUI technology

Users

Subject to market forces

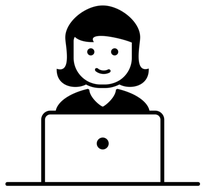
Exciting new products come to market with innovative features.

- Broader eco-system might change.
- Might demand compatibility with other applications and services.

Habits can change (fitness trackers, social media, etc.).

Privacy or security concerns.

Might seek to misuse the system - identify loop-holes, security vulnerabilities.



Business process

Changes to workflow.

Often brought about by sudden changes in legislation.

E.g. Sudden impact of Brexit or Covid on travel.

Systems need to suddenly adapt to new requirements.

Business structure

Conway's Law:

"Organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations."

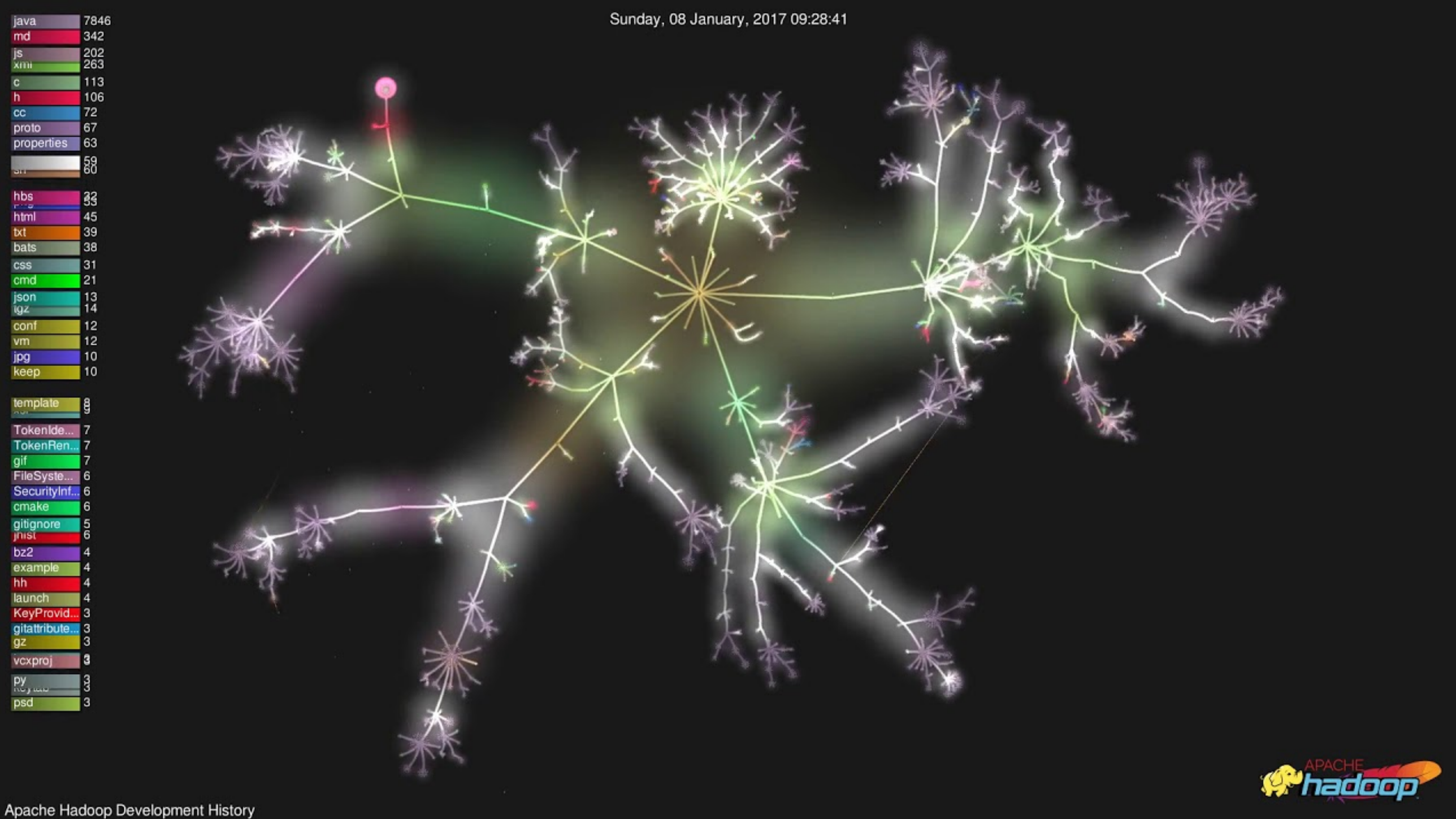
Melvin Conway, "How do Committees Invent?", Datamation, 14 (5): 28–31, 1968

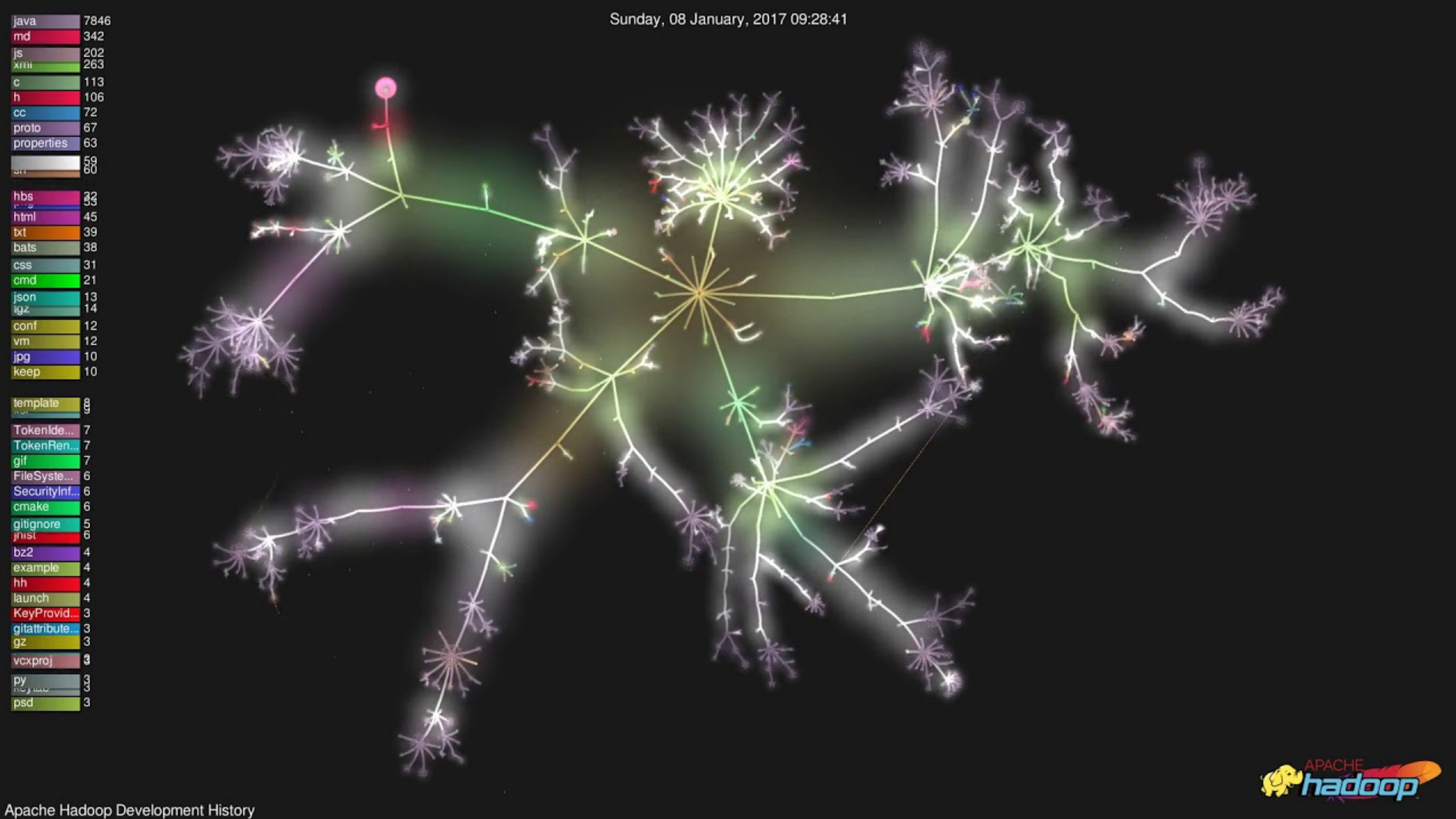
Organisational changes tend to lead to changes in enterprise software.

Supported by an empirical study at Microsoft

Demonstrated relationship between organisational and software metrics.

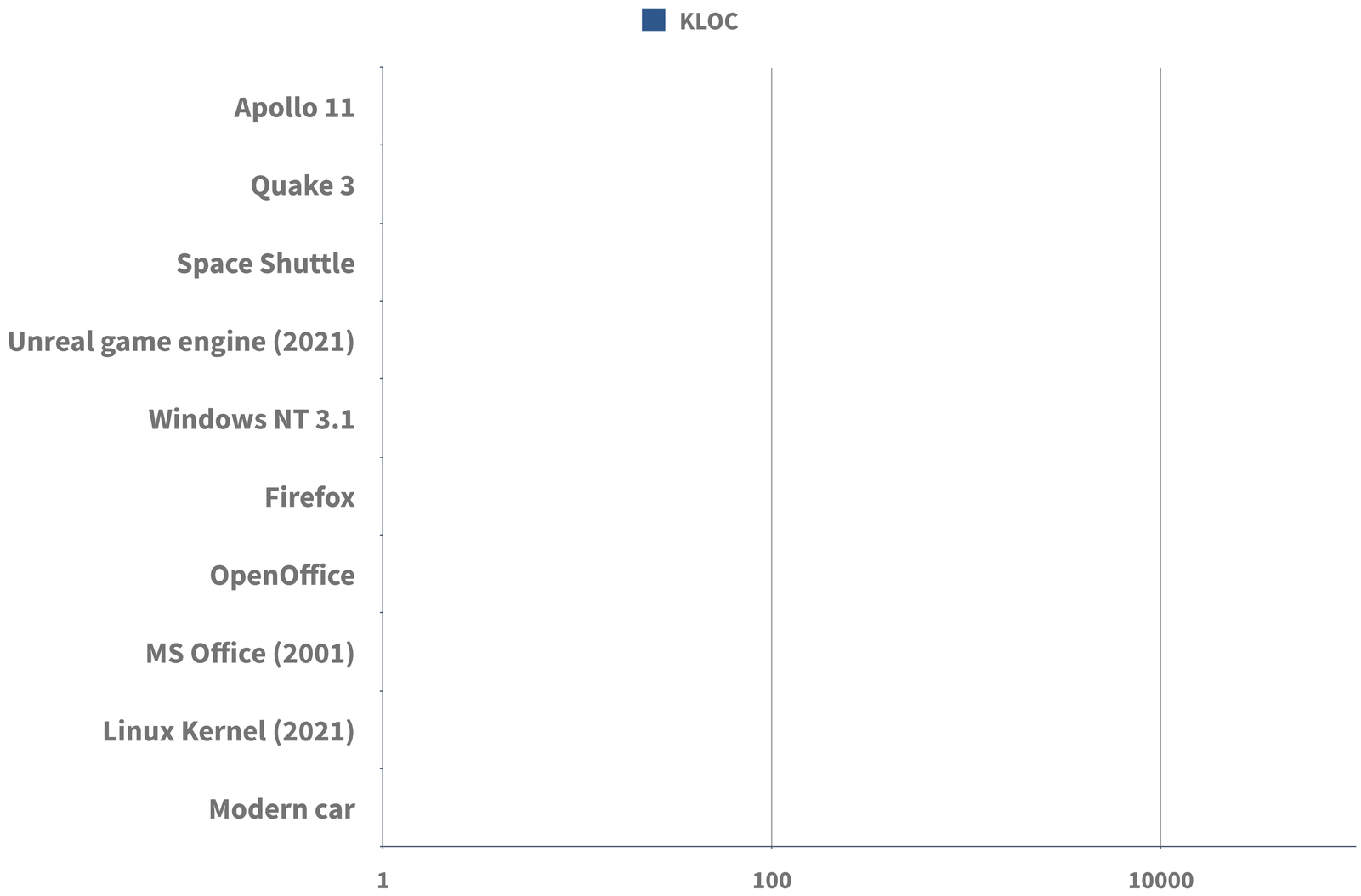
Nagappan, Murphy, Basili. "The influence of organizational structure on software quality.", International Conference on Software Engineering (ICSE), 2008.


java 7846
md 342
js 202
xml 263
c 113
h 106
cc 72
proto 67
properties 63
hbs 32
html 45
txt 39
bats 38
css 31
cmd 21
json 13
conf 12
vm 12
jpg 10
keep 10
template 8
TokenIde... 7
TokenRen... 7
gif 7
FileSyste... 6
SecurityInf... 6
cmake 6
gitignore 5
bz2 4
example 4
hh 4
launch 4
KeyProvid... 3
gitattribute... 3
gz 3
vcxproj 3
py 3
psd 3
Sunday, 08 January, 2017 09:28:41
APACHE hadoop
Apache Hadoop Development History


java 7846
md 342
js 202
xml 263
c 113
h 106
cc 72
proto 67
properties 63
hbs 32
html 45
txt 39
bats 38
css 31
cmd 21
json 13
conf 12
vm 12
jpg 10
keep 10
template 8
TokenIde... 7
TokenRen... 7
gif 7
FileSyste... 6
SecurityInf... 6
cmake 6
gitignore 5
bz2 4
example 4
hh 4
launch 4
KeyProvid... 3
gitattribute... 3
gz 3
vcxproj 3
py 3
psd 3
Sunday, 08 January, 2017 09:28:41
APACHE hadoop
Apache Hadoop Development History

Why is reengineering hard?

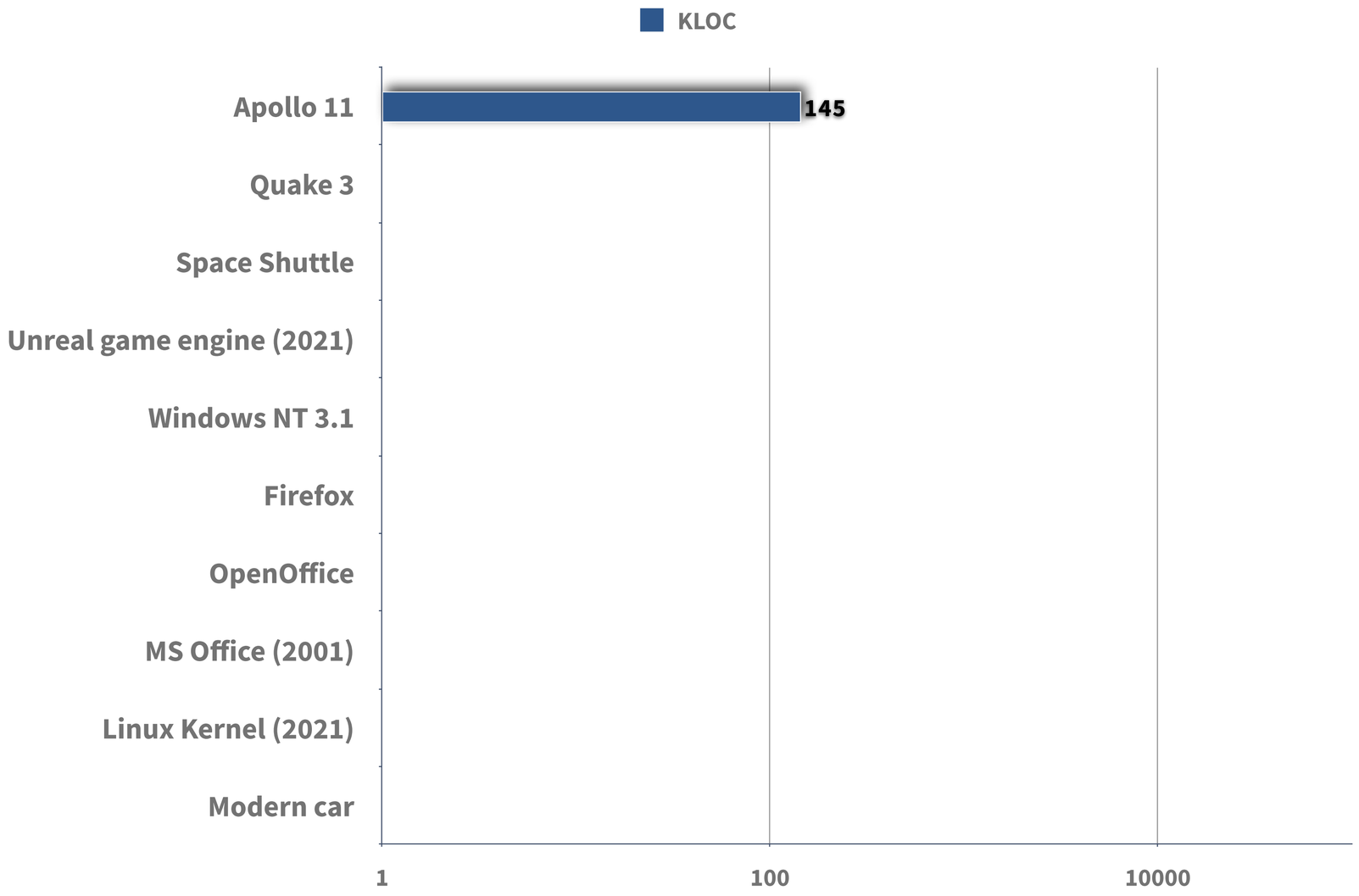
Scale



https://github.com/chrislgarry/Apollo-11

Margaret Hamilton next to the 1969 Apollo 11 moon lander code.

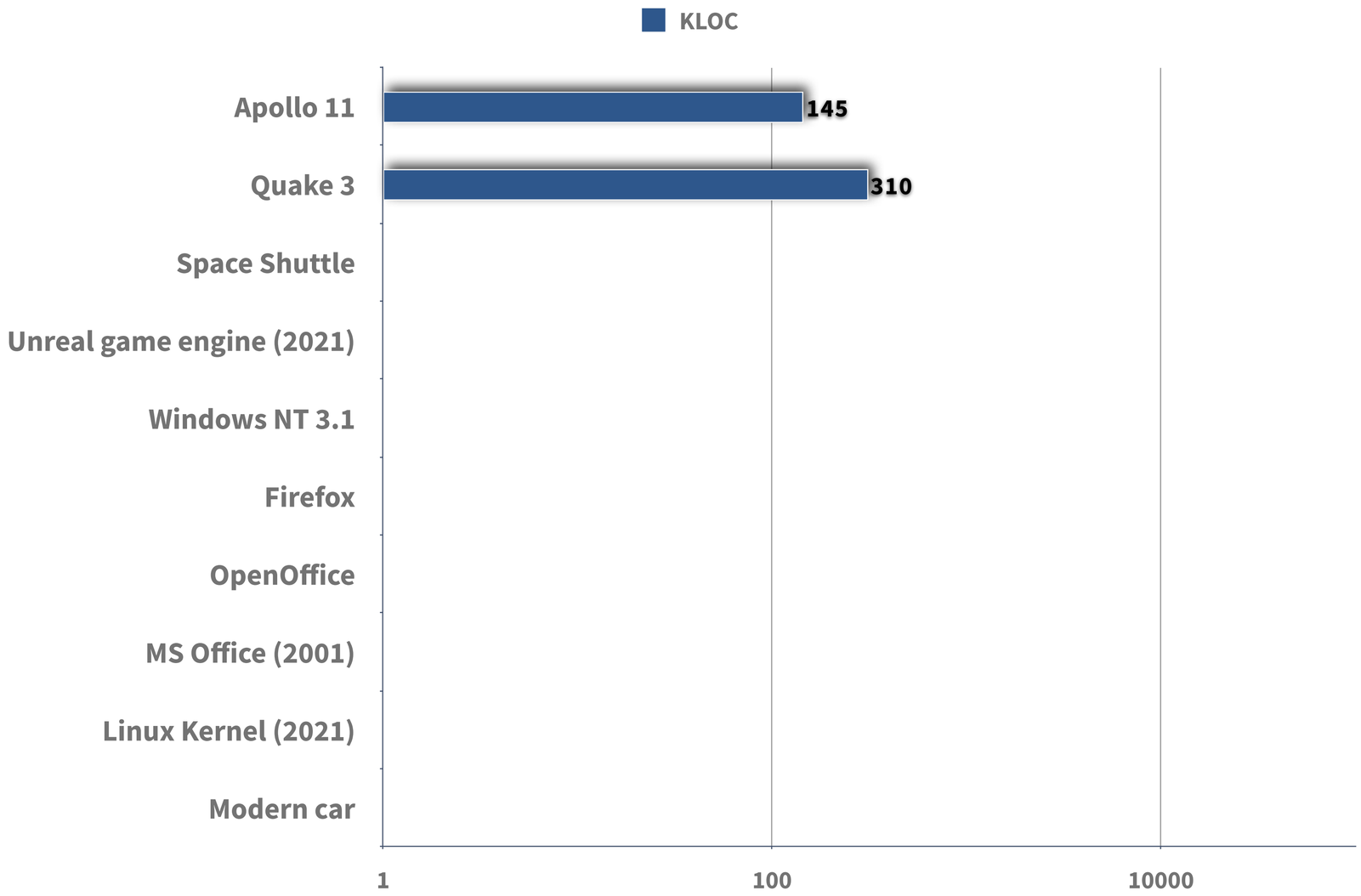
Scale



https://github.com/chrislgarry/Apollo-11

Margaret Hamilton next to the 1969 Apollo 11 moon lander code.

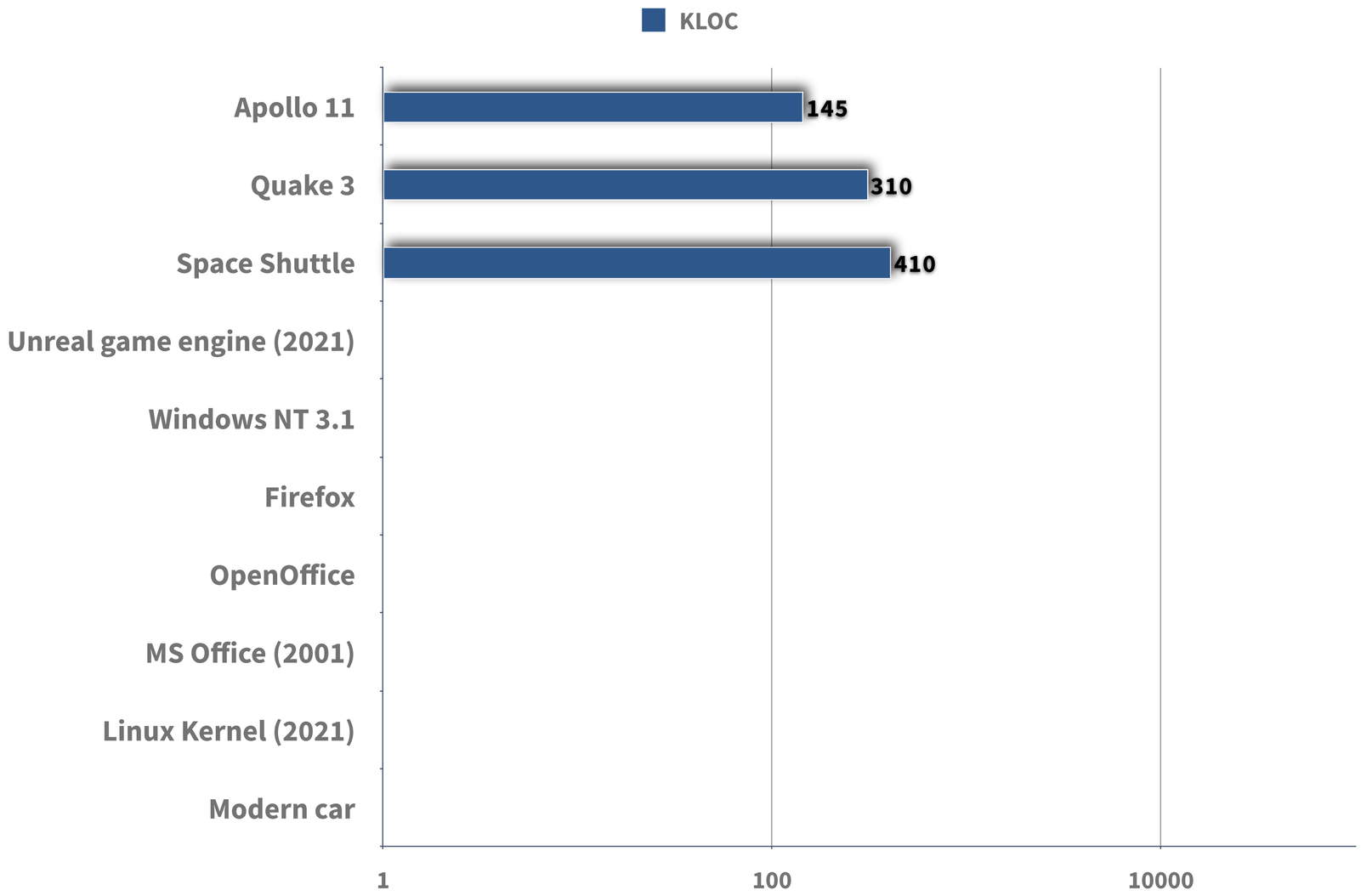
Scale



https://github.com/chrislgarry/Apollo-11

Margaret Hamilton next to the 1969 Apollo 11 moon lander code.

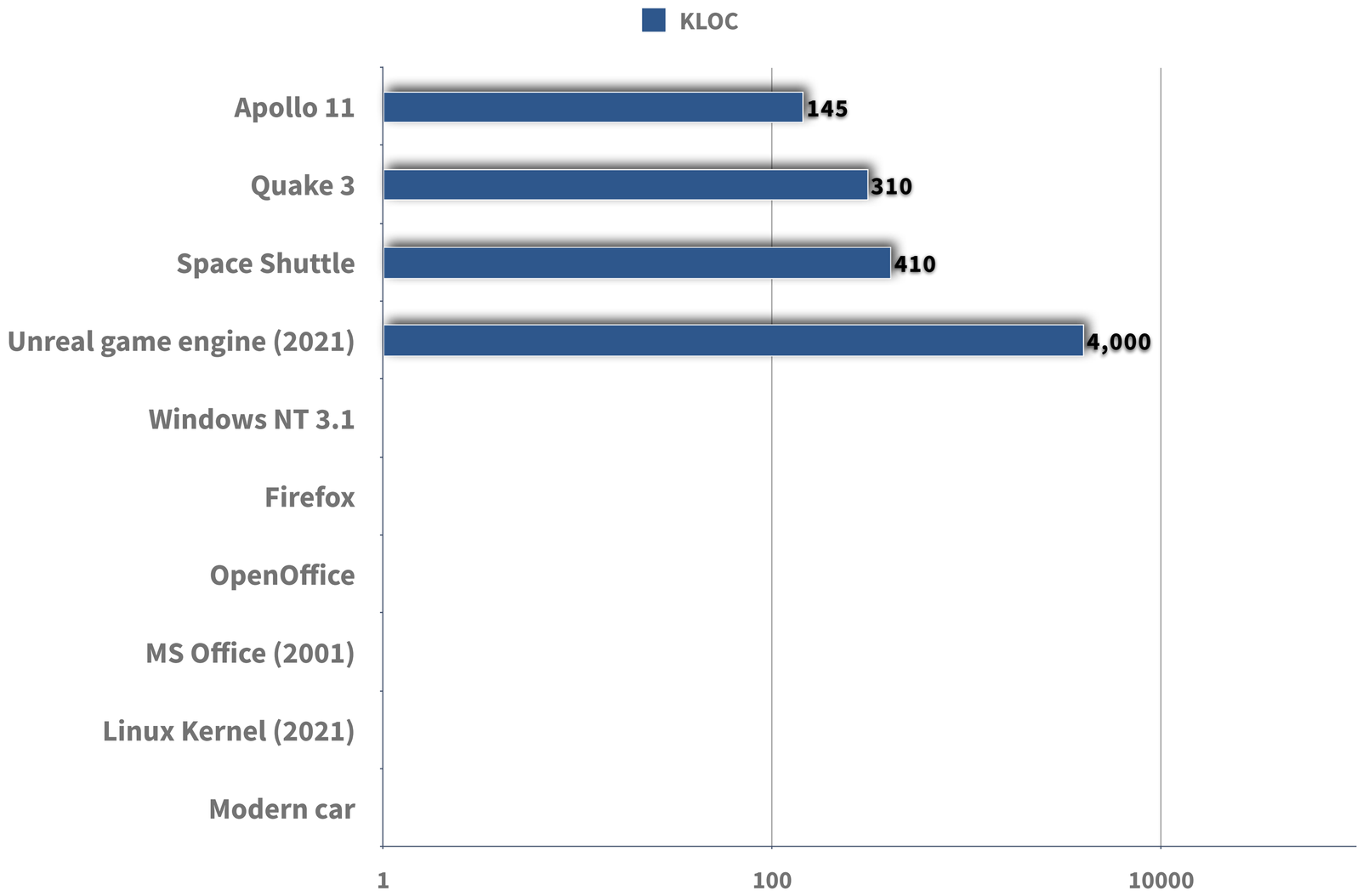
Scale



https://github.com/chrislgarry/Apollo-11

Margaret Hamilton next to the 1969 Apollo 11 moon lander code.

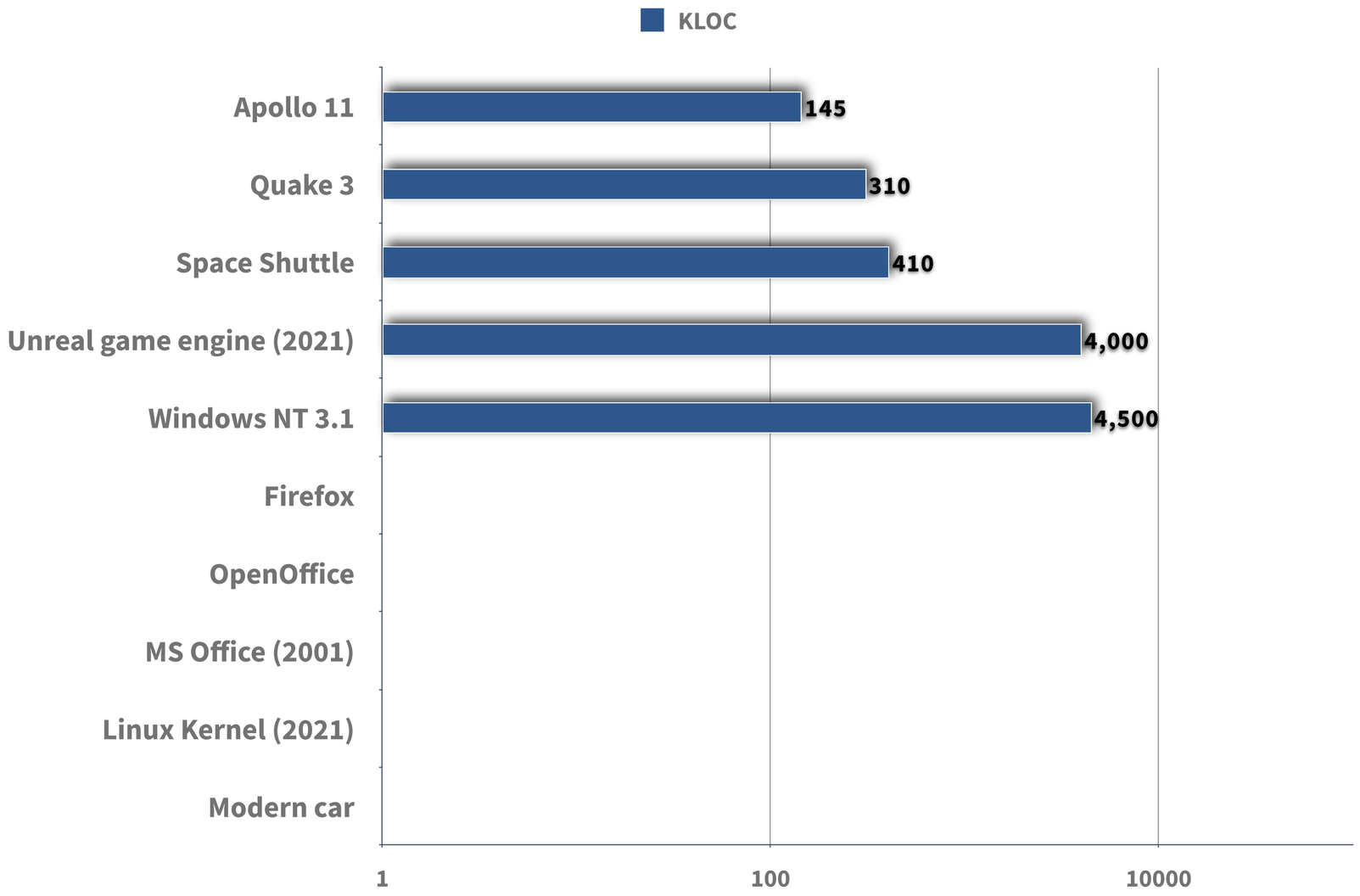
Scale



https://github.com/chrislgarry/Apollo-11

Margaret Hamilton next to the 1969 Apollo 11 moon lander code.

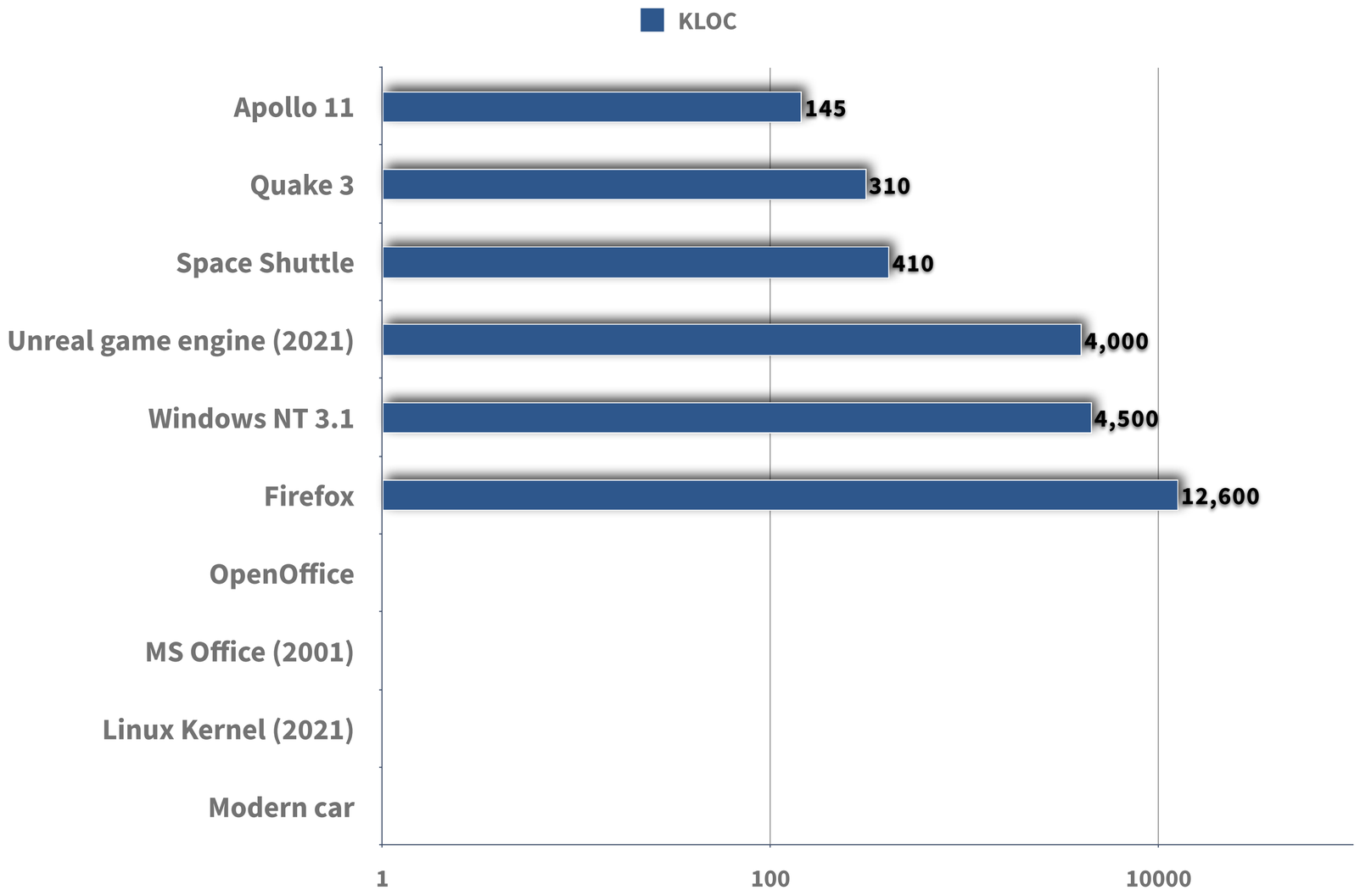
Scale



https://github.com/chrislgarry/Apollo-11

Margaret Hamilton next to the 1969 Apollo 11 moon lander code.

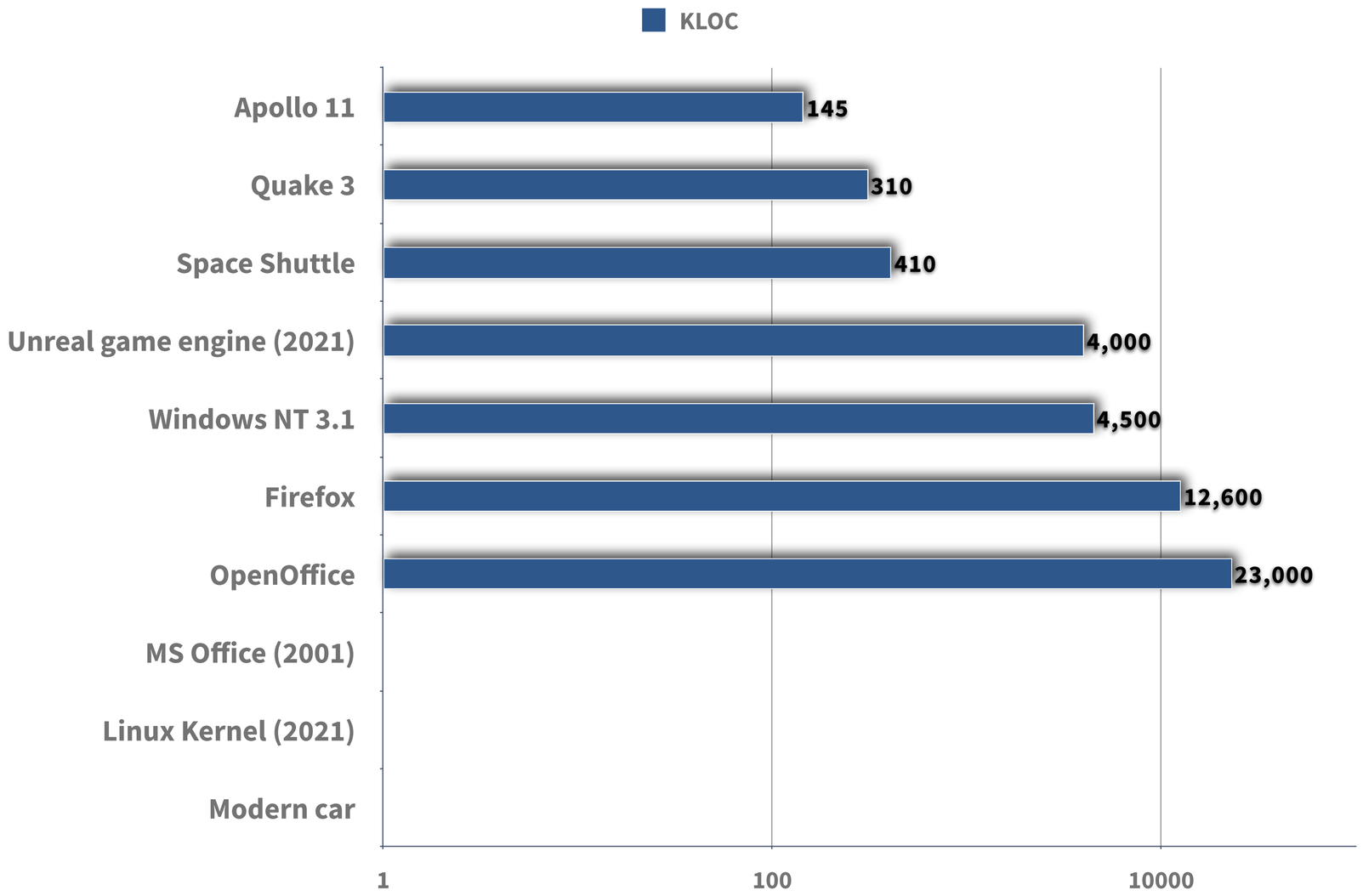
Scale



https://github.com/chrislgarry/Apollo-11

Margaret Hamilton next to the 1969 Apollo 11 moon lander code.

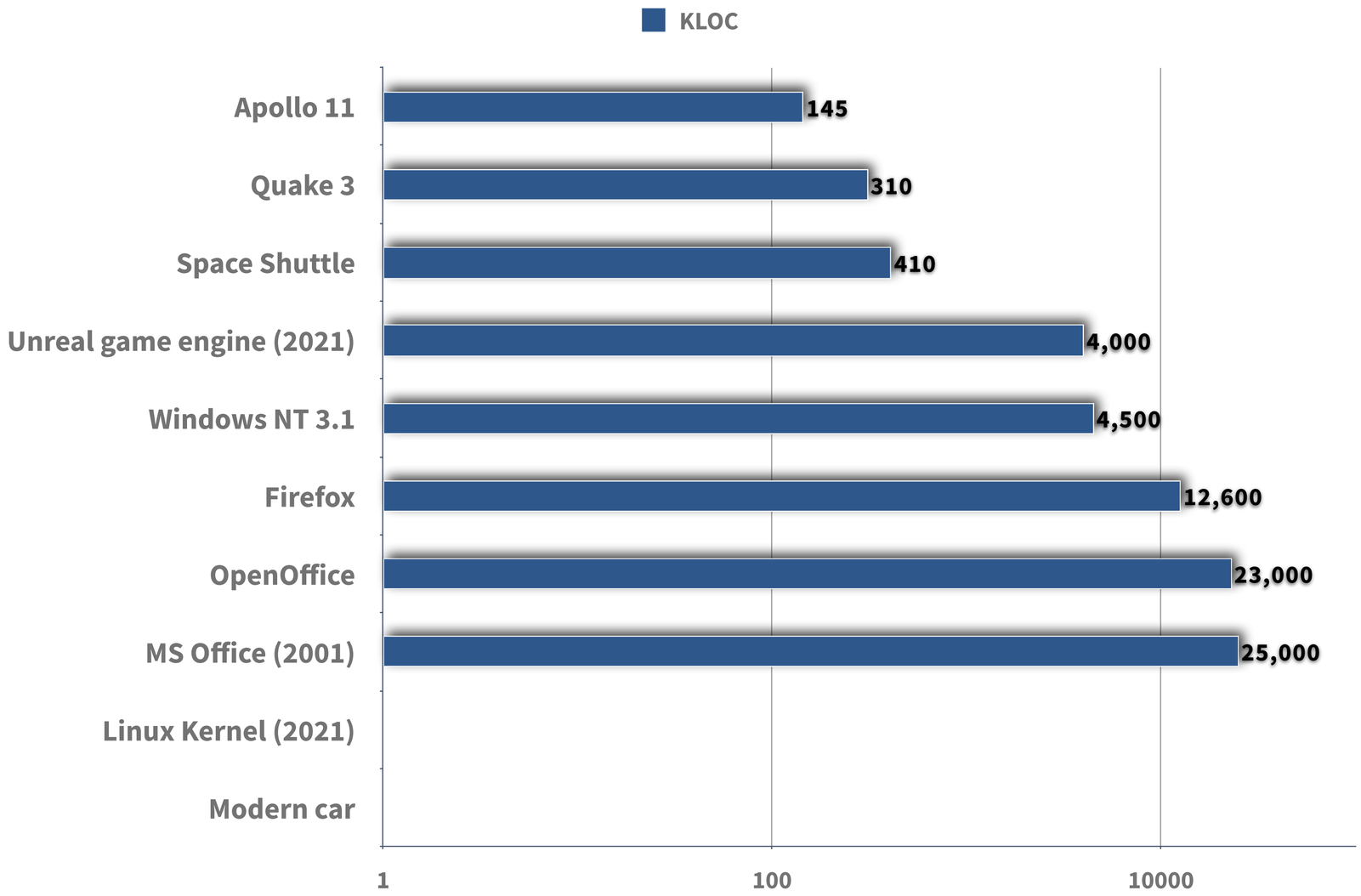
Scale



https://github.com/chrislgarry/Apollo-11

Margaret Hamilton next to the 1969 Apollo 11 moon lander code.

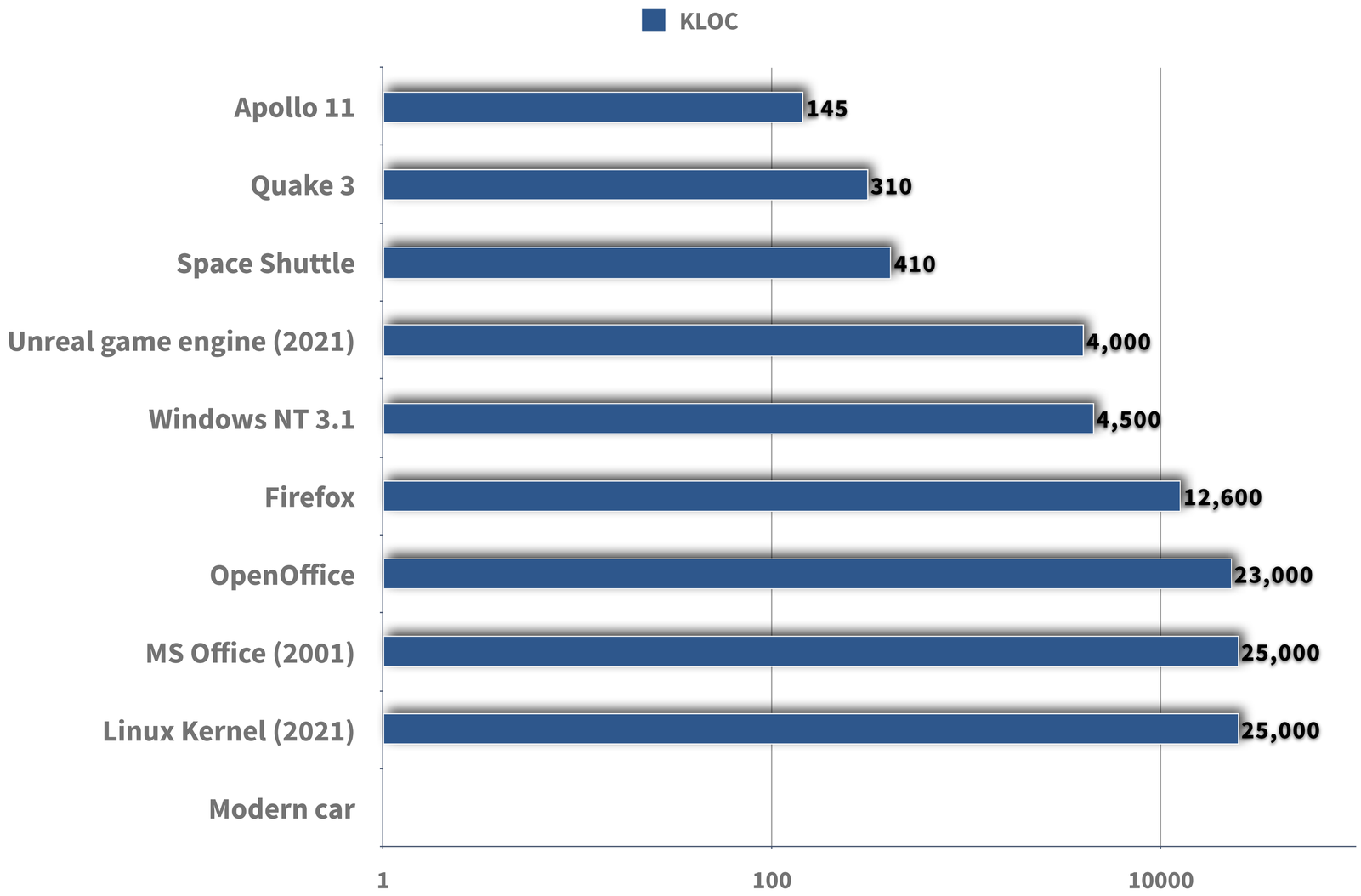
Scale



https://github.com/chrislgarry/Apollo-11

Margaret Hamilton next to the 1969 Apollo 11 moon lander code.

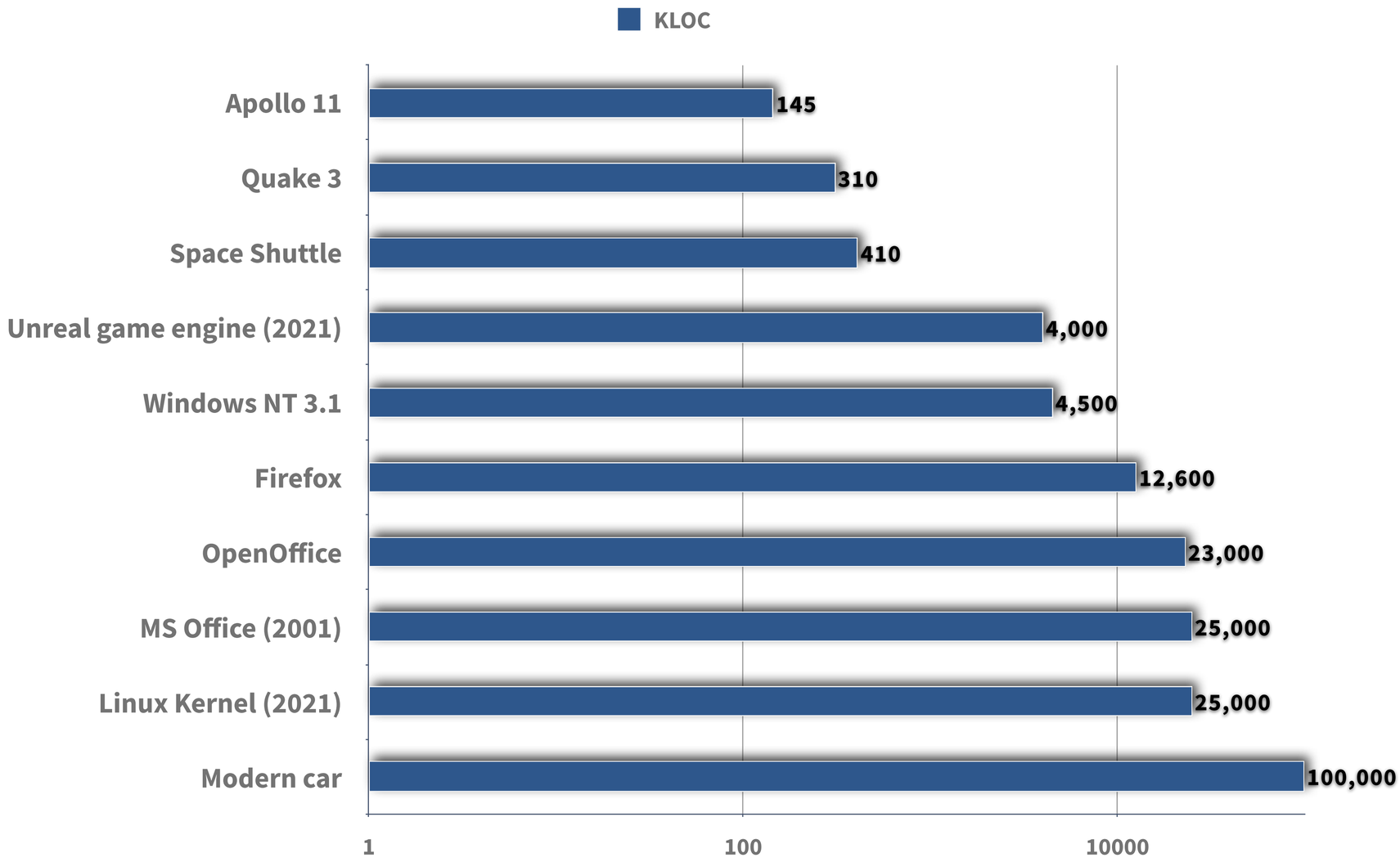
Scale



https://github.com/chrislgarry/Apollo-11

Margaret Hamilton next to the 1969 Apollo 11 moon lander code.

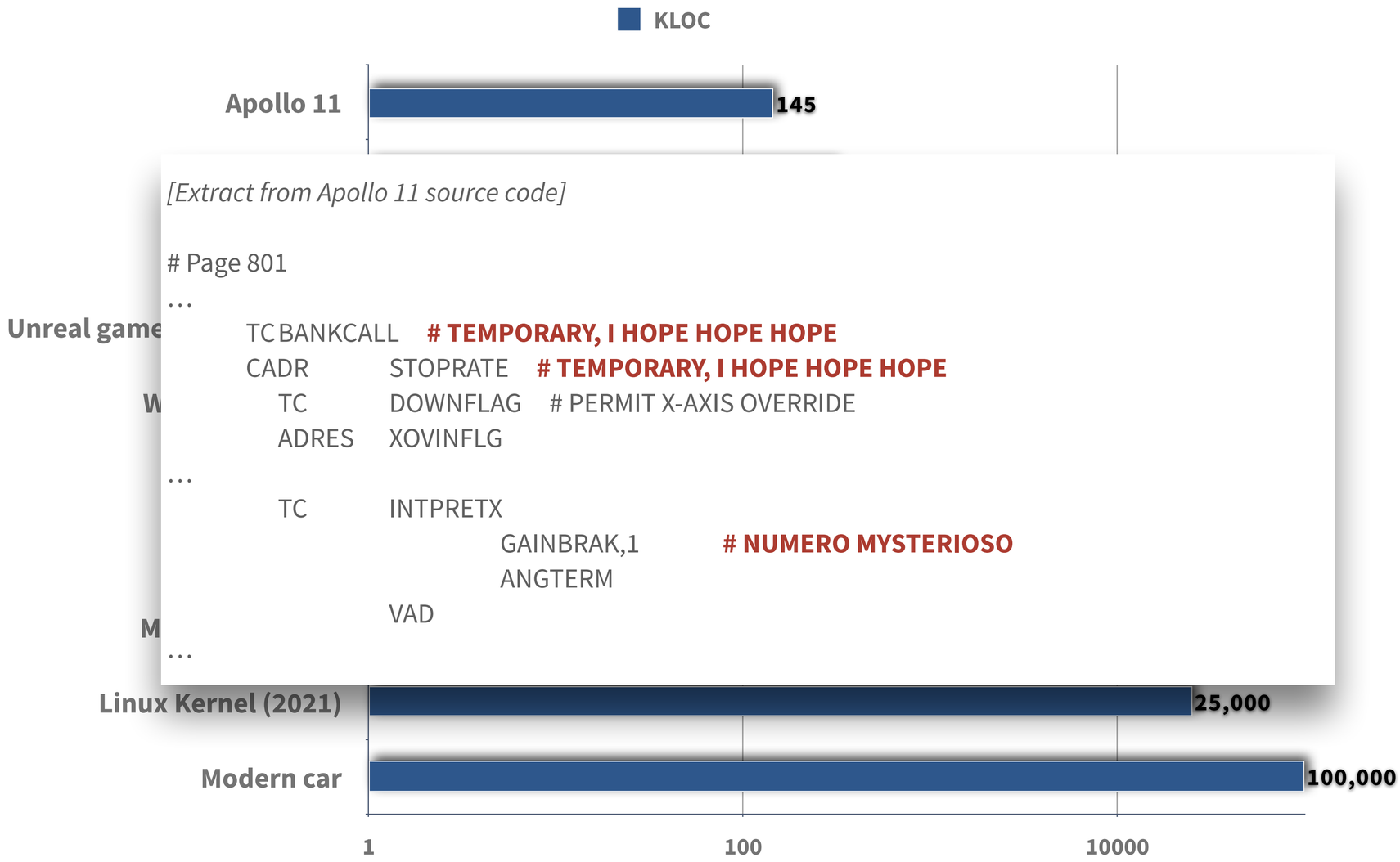
Scale



https://github.com/chrislgarry/Apollo-11

Margaret Hamilton next to the 1969 Apollo 11 moon lander code.

Scale



[Extract from Apollo 11 source code]

```
# Page 801
...
        TC BANKCALL    # TEMPORARY, I HOPE HOPE HOPE
        CADR        STOPRATE    # TEMPORARY, I HOPE HOPE HOPE
           TC       DOWNFLAG    # PERMIT X-AXIS OVERRIDE
           ADRES    XOVINFLG
...
           TC       INTPRETX
                       GAINBRAK,1          # NUMERO MYSTERIOSO
                       ANGTERM
                    VAD
...
```

https://github.com/chrislgarry/Apollo-11

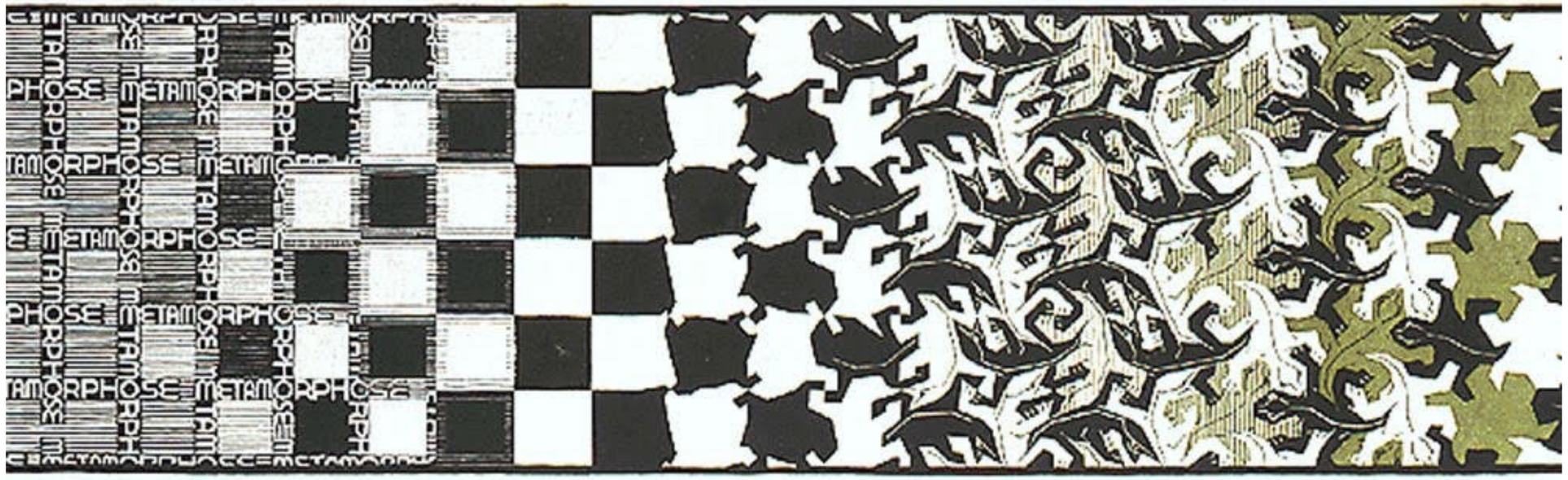
Margaret Hamilton next to the 1969 Apollo 11 moon lander code.

Continuous change

Code structure is continuously changing.

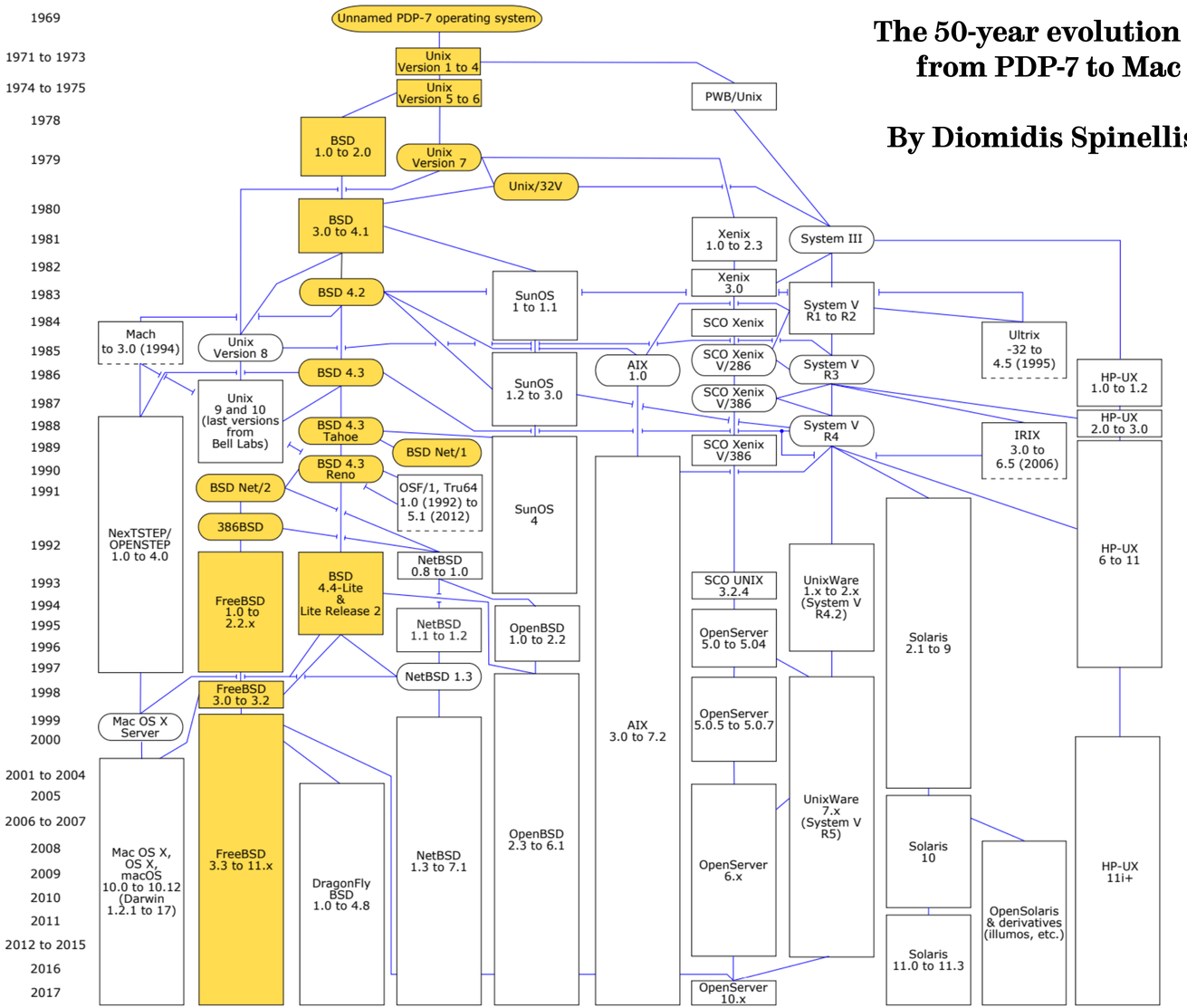
Impossible to maintain a fixed, reliable document of architecture or design.

Developer knowledge of the system rapidly becomes outdated.

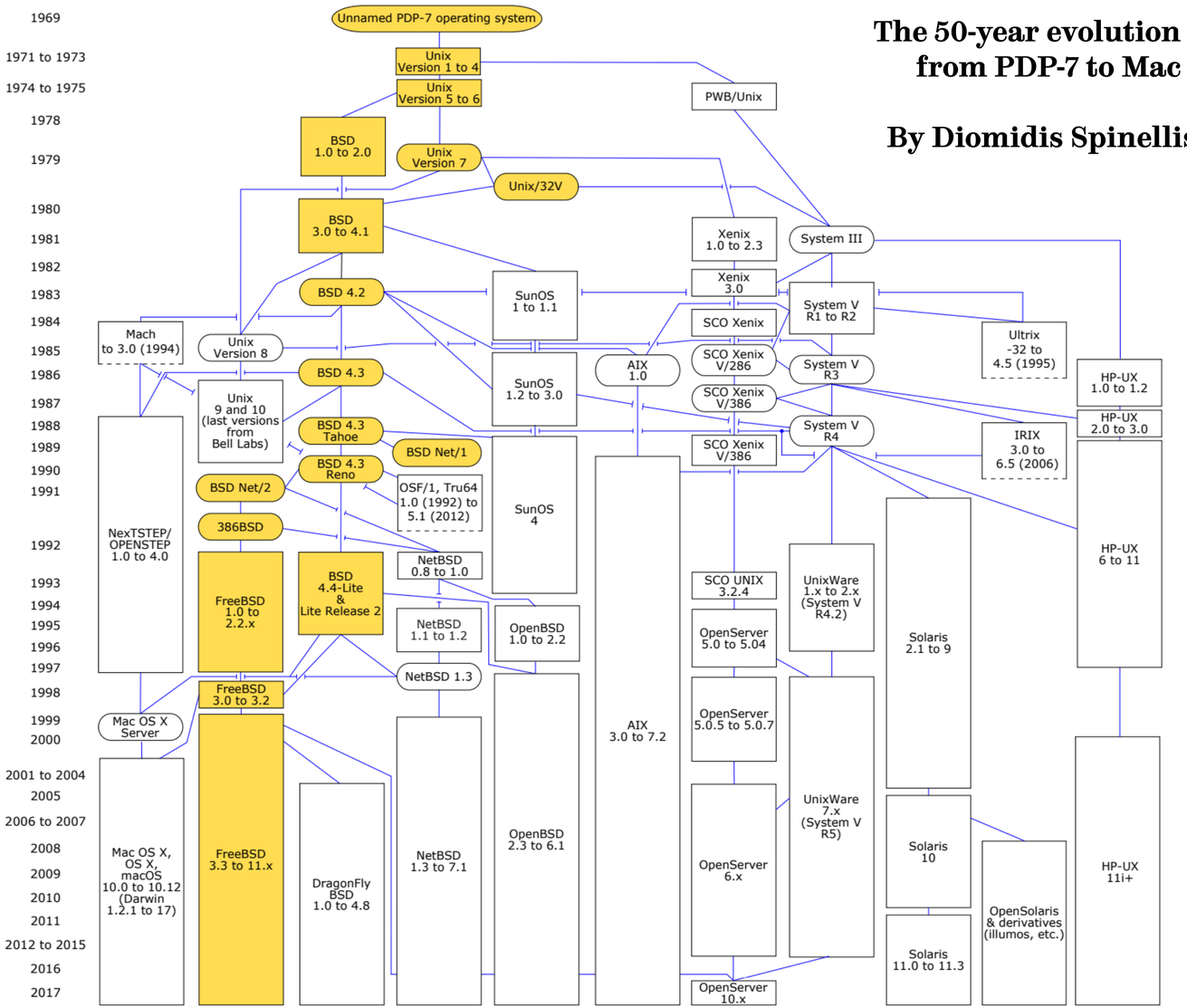


Metamorphosis,
M.C. Escher

An example: Unix



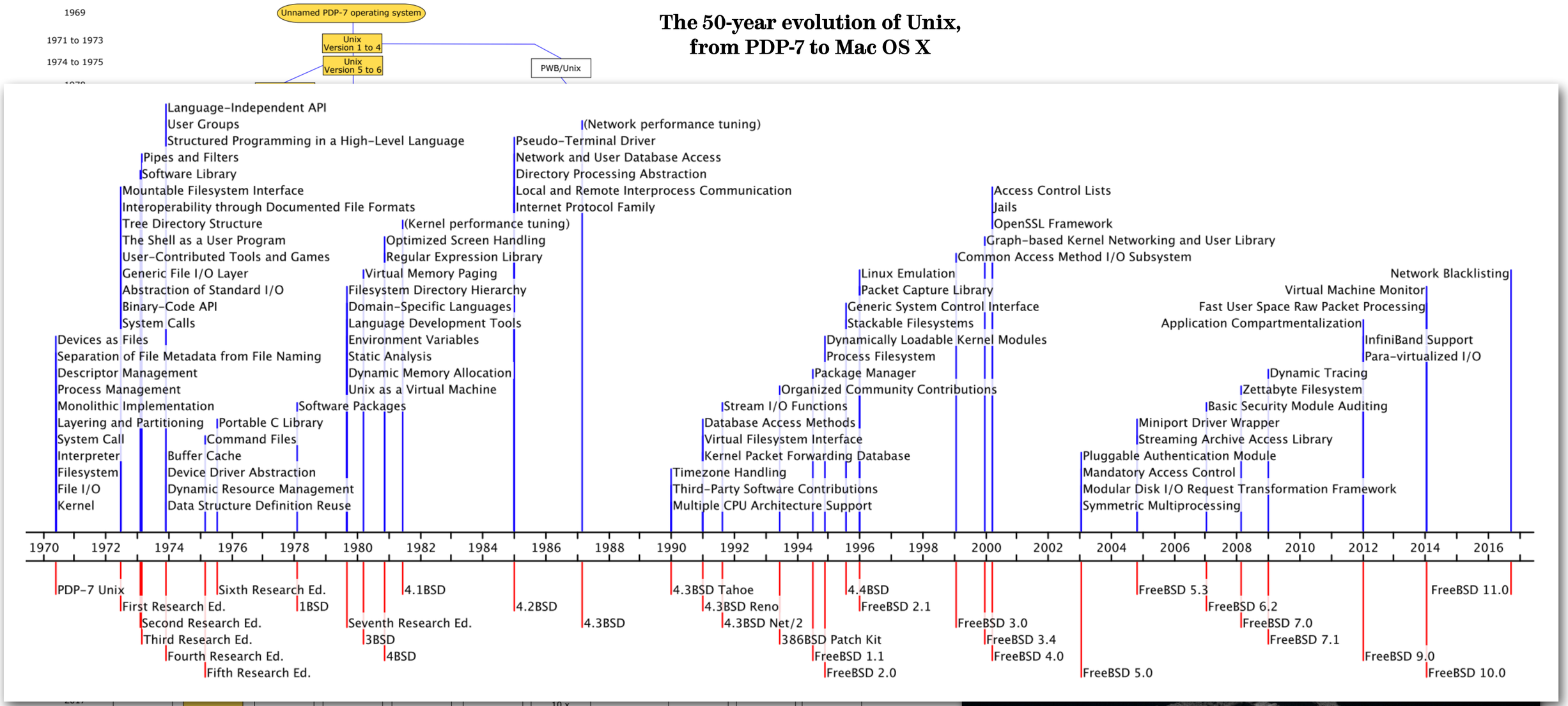
An example: Unix



The 50-year evolution of Unix, from PDP-7 to Mac OS X

By Diomidis Spinellis (2019)

An example: Unix



Complexity

Functionality emerges from interactions.

An intricate network of dependencies and data-flows.

Difficult to anticipate from source code alone.

Behaviour is “invisible’’ - relies on the ability to execute code mentally.

Calls between files in the Flare Actionscript visualisation library

Dynamism

Legacy code is:

- Dynamic
- Built on abstractions that may no longer be appropriate.
- Have been gradually re-purposed over time.

Difficult to conceptualise.

Difficult to plan a re-engineering strategy…

- … figure out where to start, …
- … what to change,
- and how.

Dynamism

Legacy code is:

- Dynamic
- Built on abstractions that may no longer be appropriate.
- Have been gradually re-purposed over time.

Difficult to conceptualise.

Difficult to plan a re-engineering strategy…

- … figure out where to start, …
- … what to change,
- and how.

Why not start from scratch?

It’s extremely risky.

A lack of requirements and specification.

- They are implicit, “baked in” to the source code of the legacy system.
- Extremely difficult to extract.
- Highly complex; they have evolved and grown, potentially over decades.

Difficult to anticipate cost and ensure reliability.

Difficult to ensure that the replacement is able to replace the old system.

High stakes for business critical systems.

- Failure at “switchover” could lead to calamity.

TSB bank split from Lloyds Bank in 2013.

- Had used legacy system to manage bank accounts

TSB bank split from Lloyds Bank in 2013.

- Had used legacy system to manage bank accounts

TSB decided in 2018 to migrate accounts from Lloyds system to new platform to facilitate digital banking.

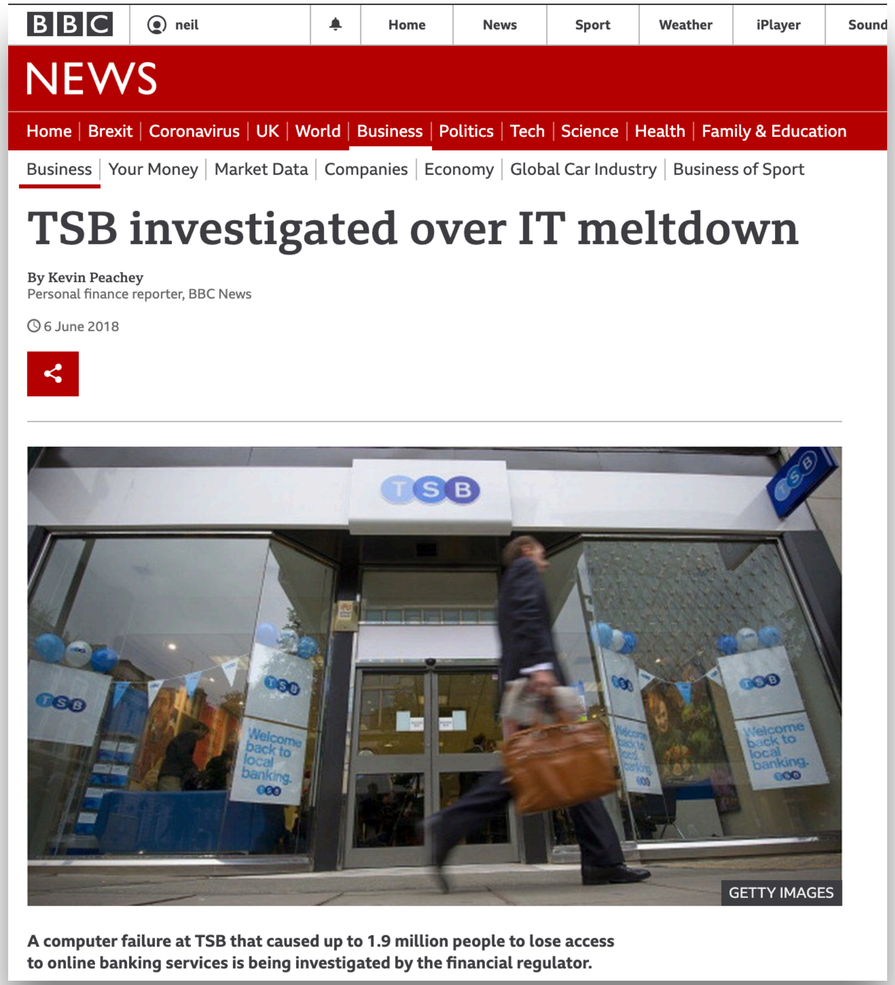
TSB bank split from Lloyds Bank in 2013.

- Had used legacy system to manage bank accounts

TSB decided in 2018 to migrate accounts from Lloyds system to new platform to facilitate digital banking.

Led to disruption to millions of TSB customers, including:

- Loss of internet and banking services for >1 week.
- Customers being able to view each others' banking details.
- Money disappearing from accounts.



BBC | neil | Home | News | Sport | Weather | iPlayer | Sound

NEWS

Home | Brexit | Coronavirus | UK | World | Business | Politics | Tech | Science | Health | Family & Education

Business | Your Money | Market Data | Companies | Economy | Global Car Industry | Business of Sport

TSB investigated over IT meltdown

By Kevin Peachey
Personal finance reporter, BBC News

6 June 2018

A computer failure at TSB that caused up to 1.9 million people to lose access to online banking services is being investigated by the financial regulator.

TSB bank split from Lloyds Bank in 2013.

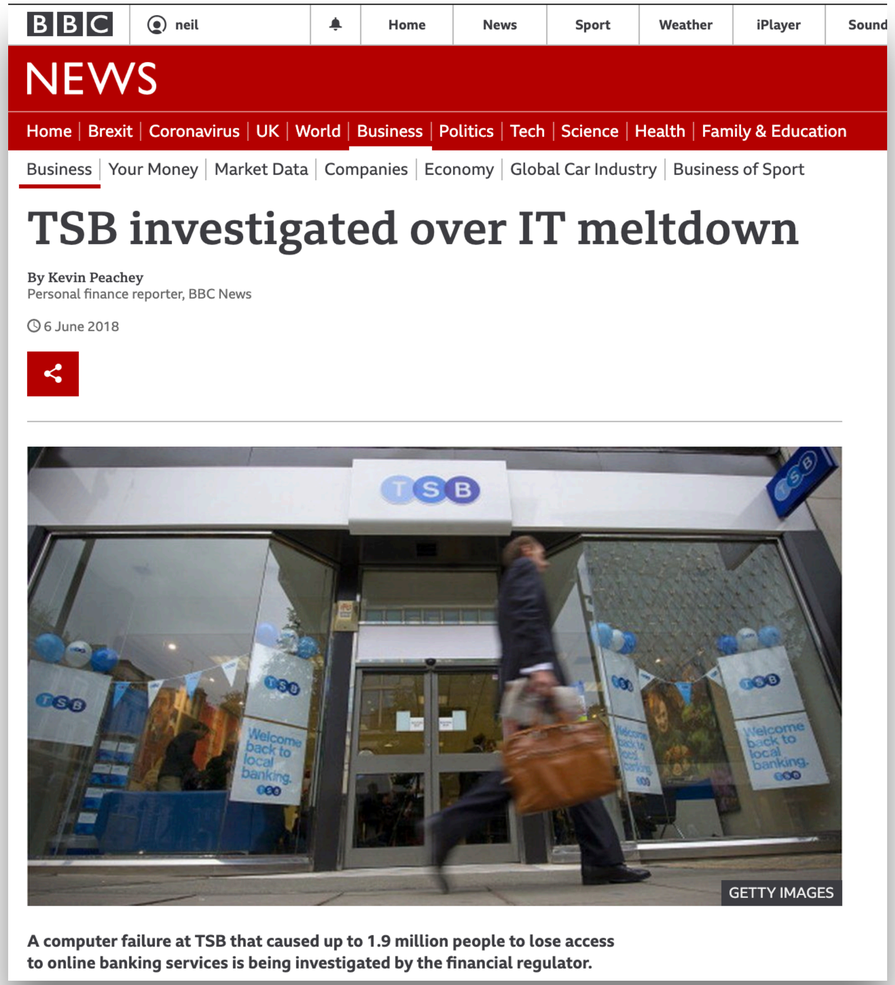
- Had used legacy system to manage bank accounts

TSB decided in 2018 to migrate accounts from Lloyds system to new platform to facilitate digital banking.

Led to disruption to millions of TSB customers, including:

- Loss of internet and banking services for >1 week.
- Customers being able to view each others' banking details.
- Money disappearing from accounts.

Led to estimated £366m loss for TSB.



Take-aways

Many external drivers for change.

- Often happens by necessity, rapid bug fixes, added features.
- Lack of oversight and consideration of all of the various **architectural viewpoints and perspectives.**

Lots of “horizontal” and “vertical” dependencies.

- Changing one component leads to knock-on effects and can have unintended consequences.

Lots of intrinsic challenges

- Scale, continual change, dynamism, …

Reengineering is an important skill-set to have!