Software Visualization

The Humane Solution

by Eberhard Gräther

ACCU 2019

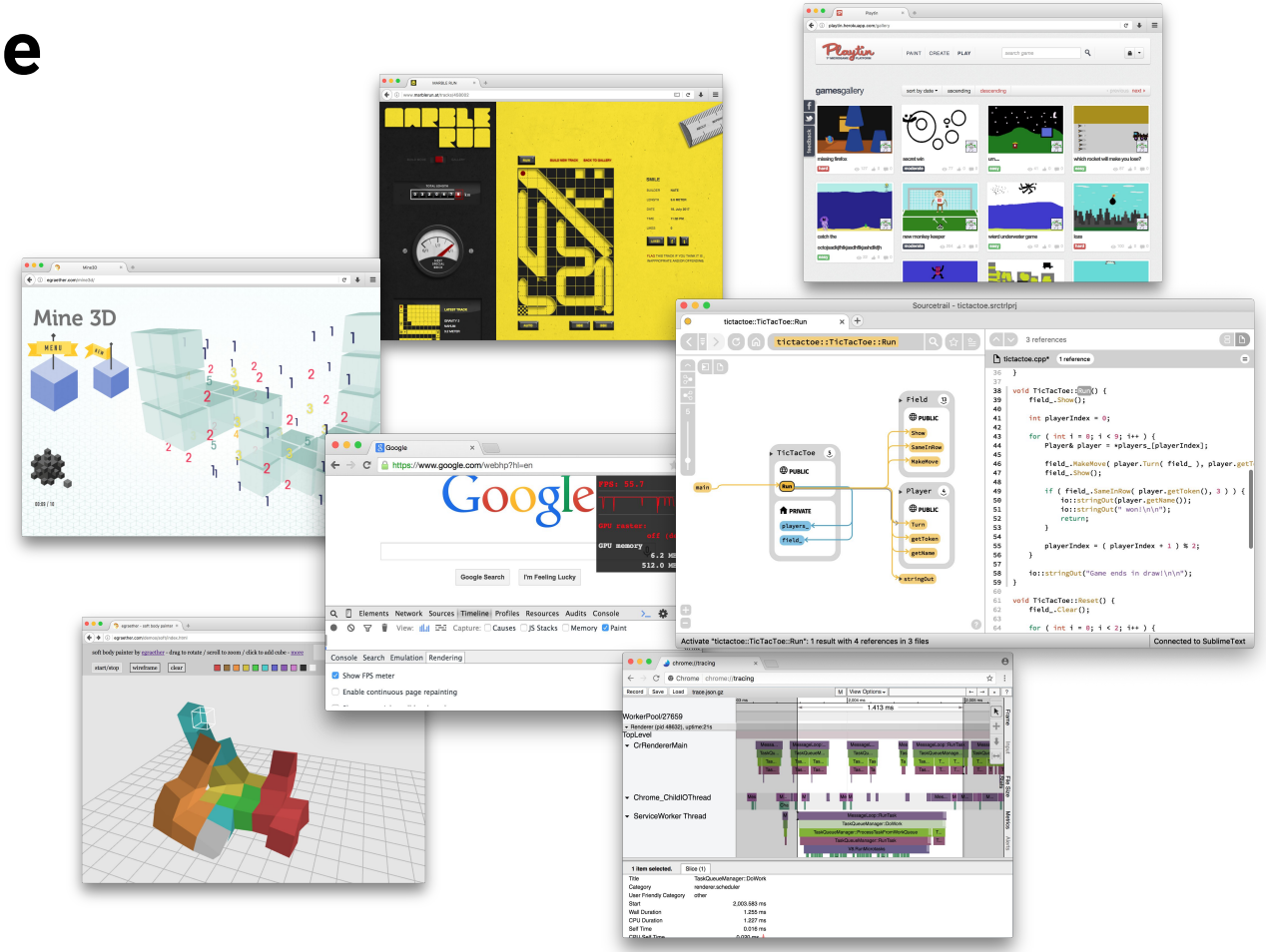
Agenda

- Reading Code
- Information Visualization
- Software Visualization
  - Structure
  - Behaviour
  - Evolution
- Interactive Dependency Graph

About me

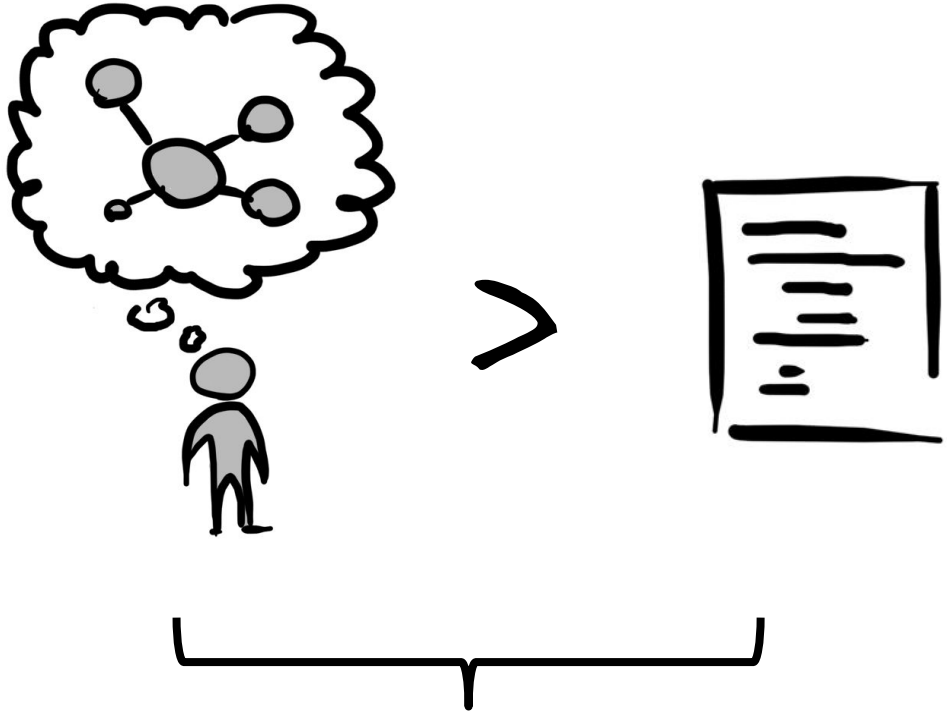
Eberhard Gräther
egraether@coati.io
@egraether

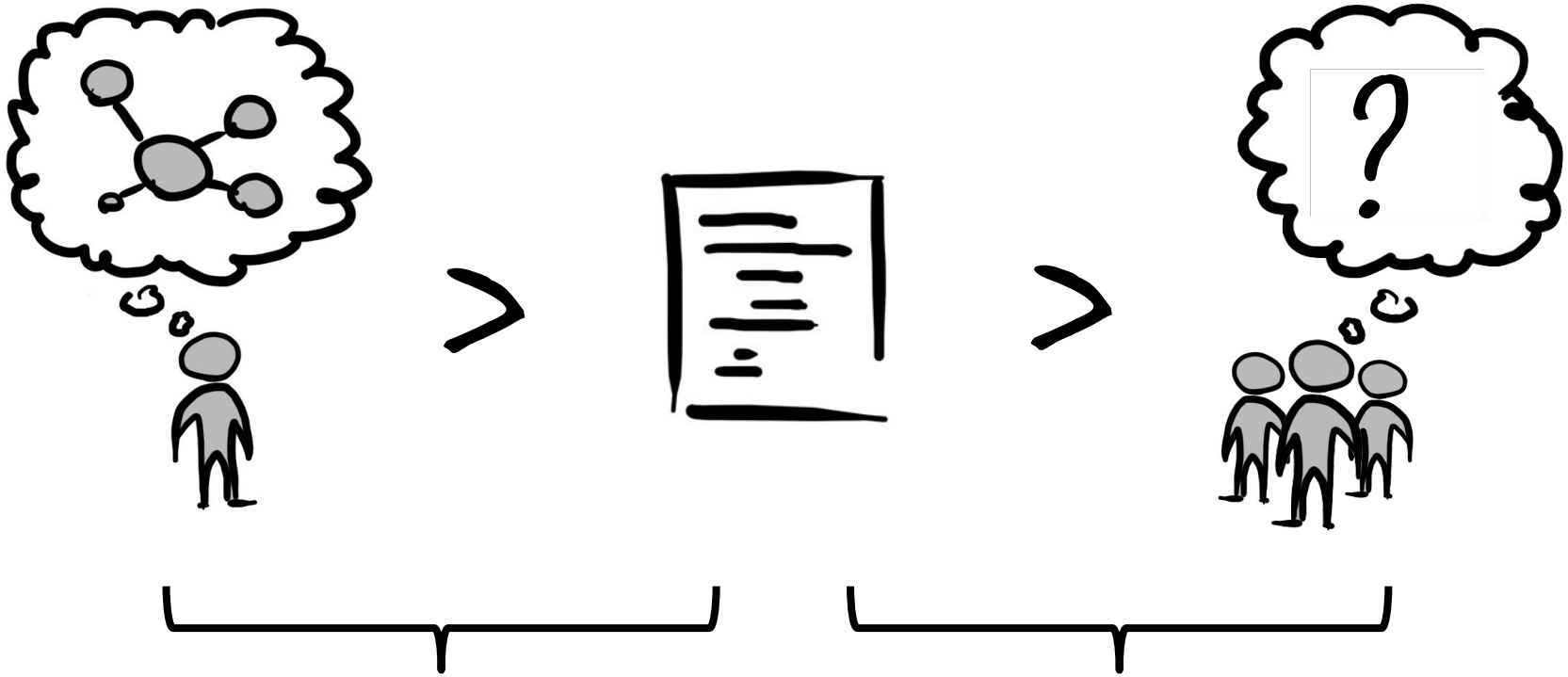
Founder
Software Engineer
Interaction Designer
at **Coati Software**



Reading Code

1/5

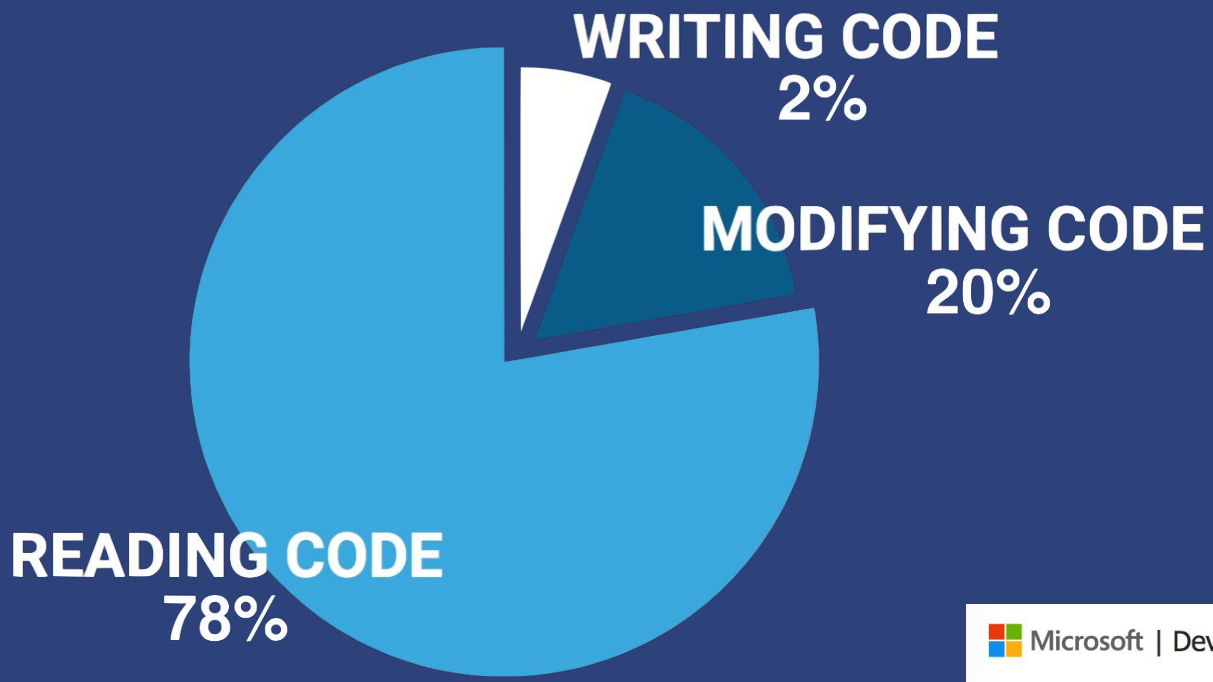

WRITING


?
WRITING
READING

*“Indeed, the ratio of **time spent reading vs. writing is well over 10:1**. We are constantly reading old code as part of the effort to write new code.”*

- Robert C. Martin

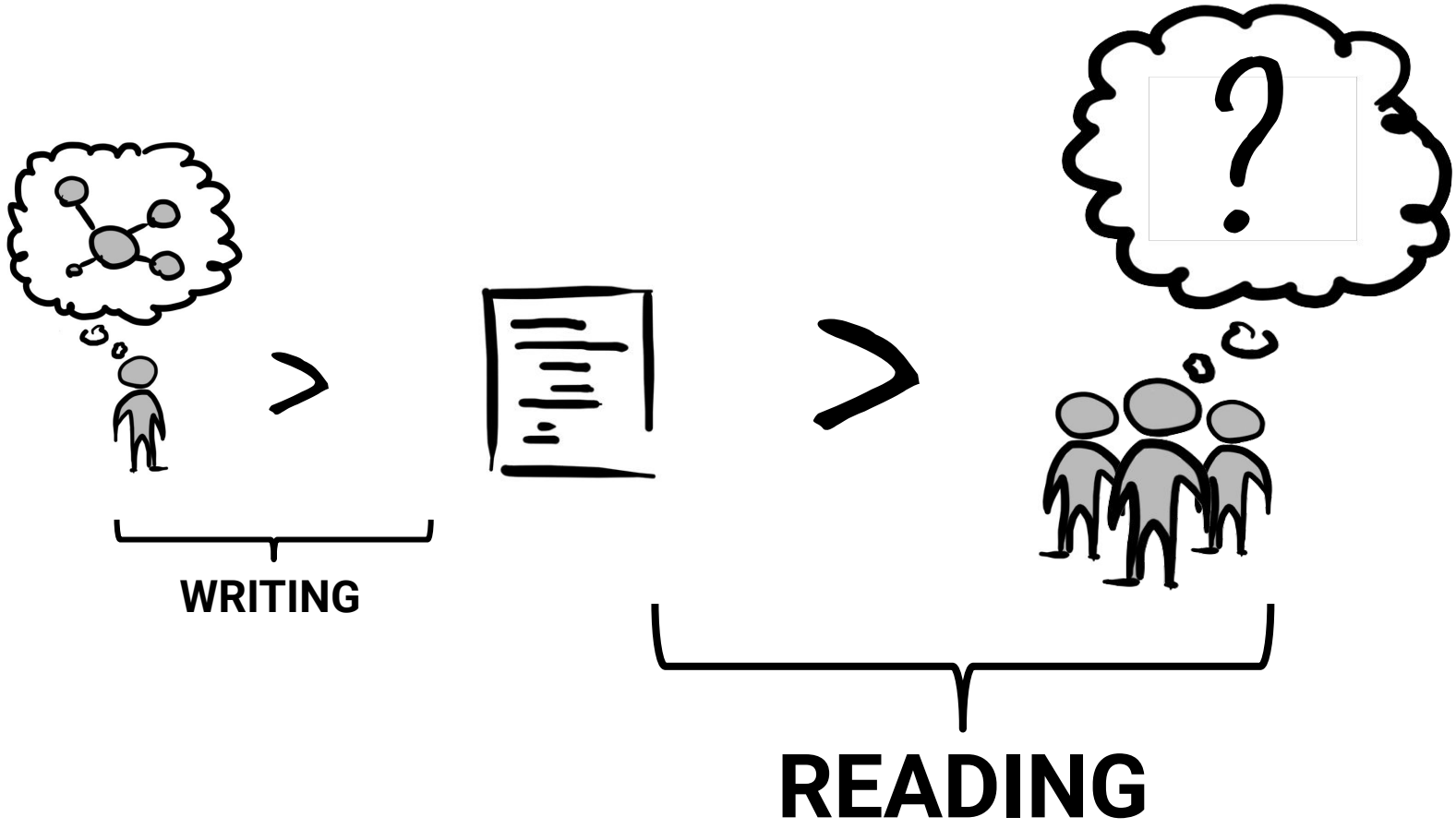
Clean Code: A Handbook of Agile Software Craftsmanship

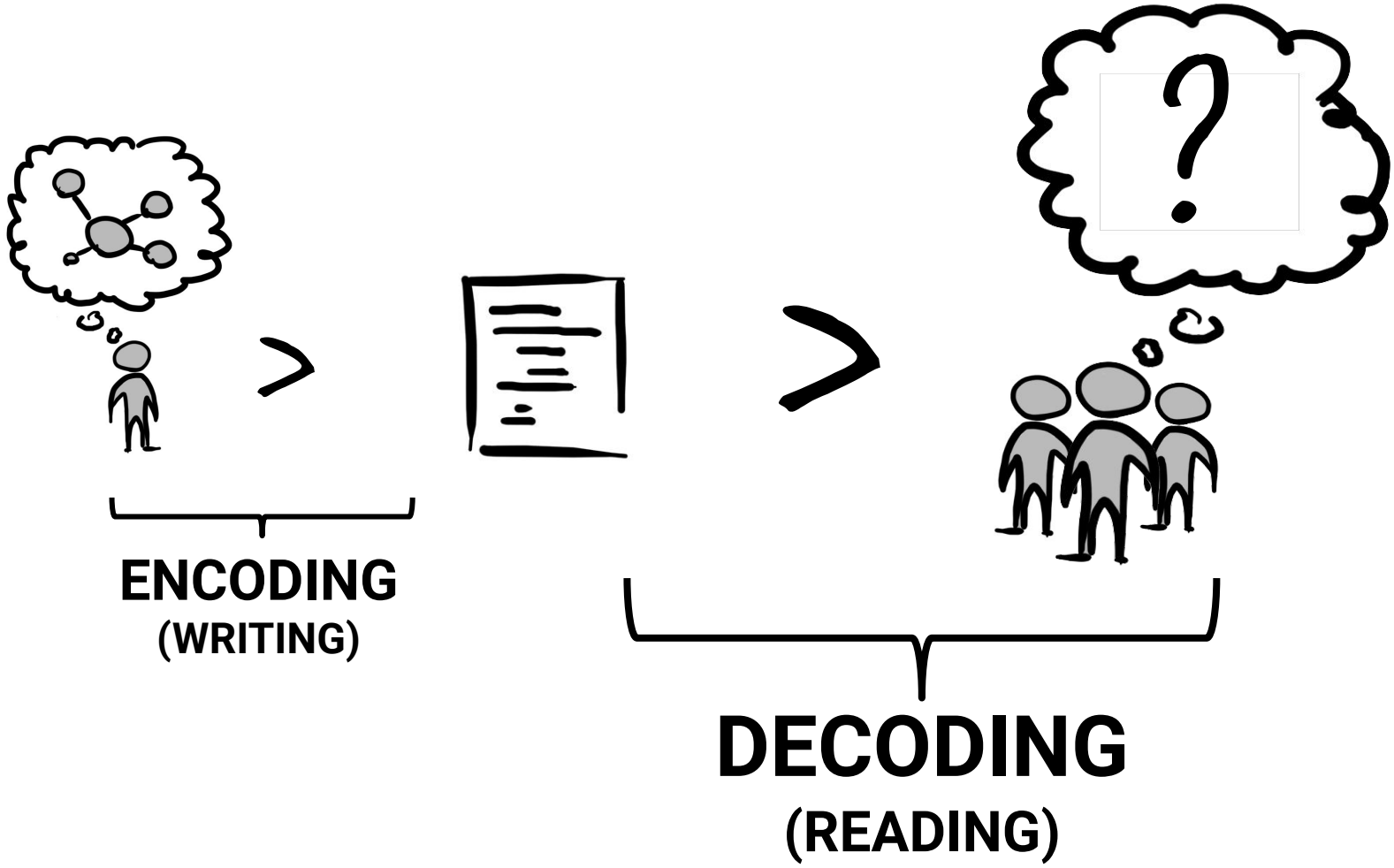


Microsoft | Developer

Peter Hallam's WebLog

* https://blogs.msdn.microsoft.com/peterhal/2006/01/04/what-do-programmers-really-do-anyway-aka-part-2-of-the-yardstick-saga/


WRITING
READING


?
ENCODING
(WRITING)
DECODING
(READING)

Source code is textual data with high information density.

```
1
2  template <typename T>
3  class Countable
4  {
5  public:
6      Countable() {
7          ++objectCount;
8      }
9
10     int getCount() const {
11         return objectCount;
12     }
13
14 private:
15     static int objectCount;
16 };
17
18 template <typename T> int Countable<T>::objectCount(0);
19
20 class Object : public Countable<Object>
21 {};
22
23 int main()
24 {
25     Object o;
26     int c = o.getCount();
27     return 0;
28 }
29
```

Source code is textual data with high information density.

For readability:

- **syntax highlighting**
- **naming convention**
- **whitespace**

```
1
2   template <typename T>
3   class Countable
4   {
5   public:
6       Countable() {
7           ++objectCount;
8       }
9
10      int getCount() const {
11          return objectCount;
12      }
13
14  private:
15      static int objectCount;
16  };
17
18  template <typename T> int Countable<T>::objectCount(0);
19
20  class Object : public Countable<Object>
21  {};
22
23  int main()
24  {
25      Object o;
26      int c = o.getCount();
27      return 0;
28  }
29
```

Source code is textual data with high information density.

For readability:

- **syntax highlighting**
- **naming convention**
- **whitespace**

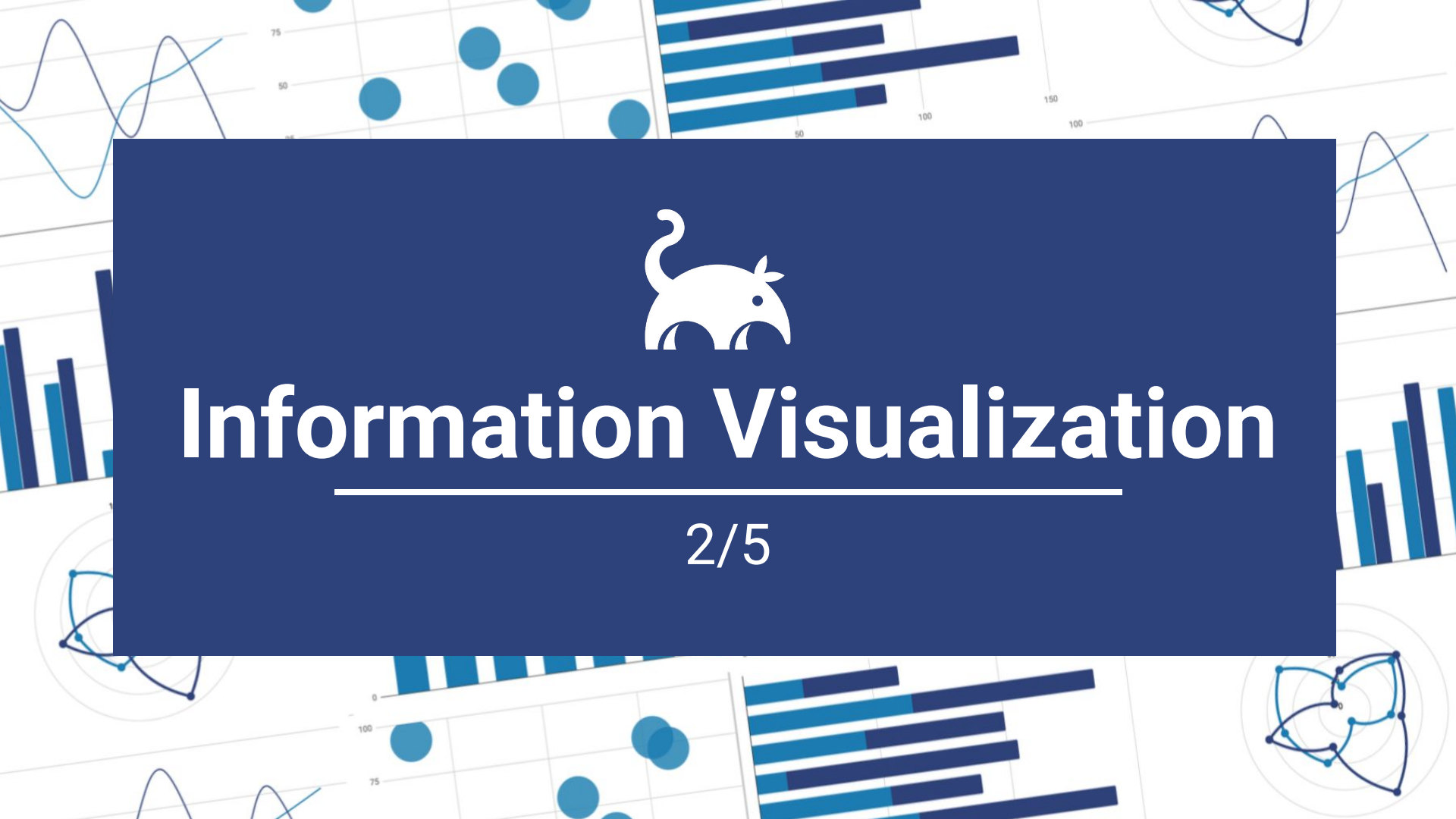
Hard to process for our mind!

```
1
2   template <typename T>
3   class Countable
4   {
5   public:
6       Countable() {
7           ++objectCount;
8       }
9
10      int getCount() const {
11          return objectCount;
12      }
13
14  private:
15      static int objectCount;
16  };
17
18  template <typename T> int Countable<T>::objectCount(0);
19
20  class Object : public Countable<Object>
21  {};
22
23  int main()
24  {
25      Object o;
26      int c = o.getCount();
27      return 0;
28  }
29
```

*“Programmers tend to adapt to the level of representation **provided by the computer**, instead of adapting the computers representations to their **perceptive abilities**.”*

- Stephan Dhiel

Software Visualization - Visualizing the Structure, Behaviour, and Evolution of Software



Information Visualization

2/5

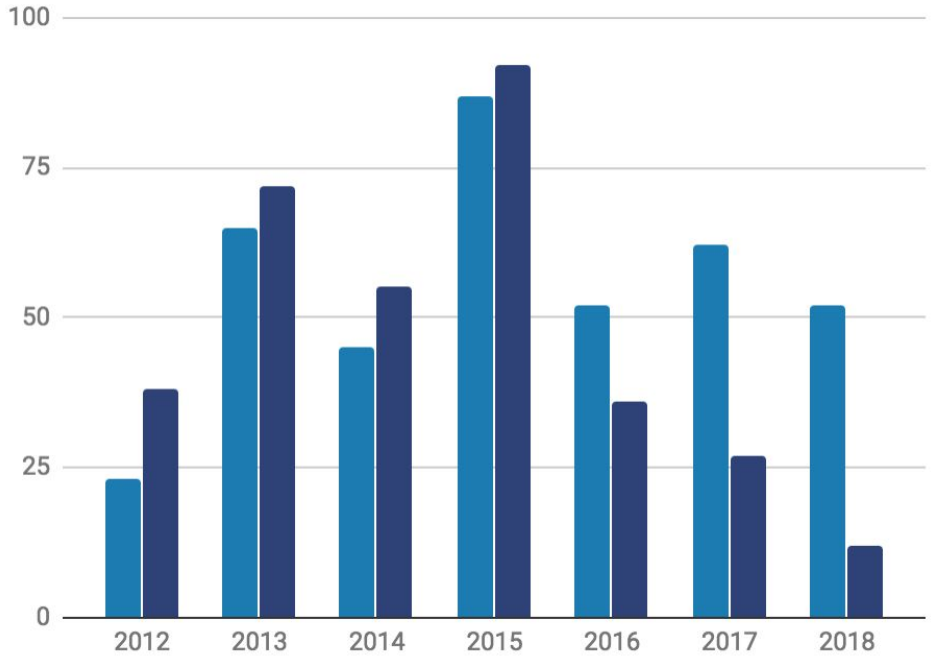
*“The **principal task of information visualization** is to **allow information to be derived** from data.”*

- Robert Spence

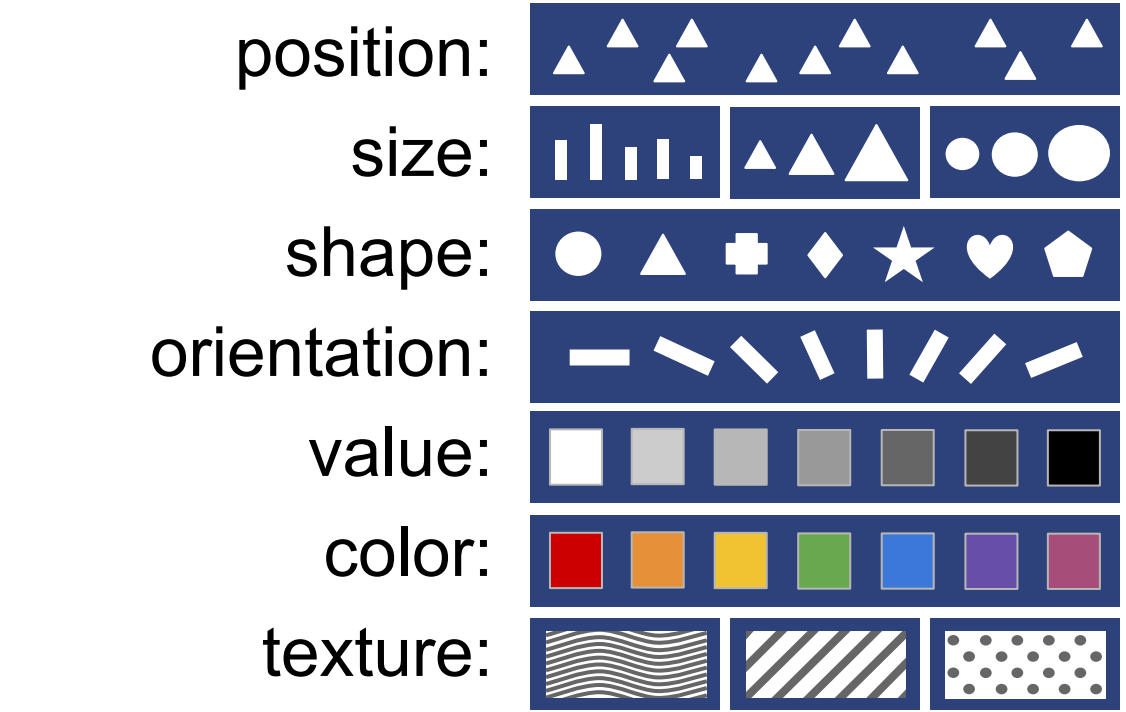
Information Visualization: Design for Interaction

Sample

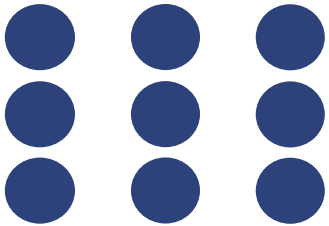
2012	23	38
2013	65	72
2014	45	55
2015	87	92
2016	52	36
2017	62	27
2018	52	12



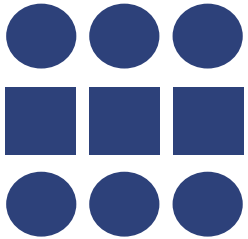
Visual Variables



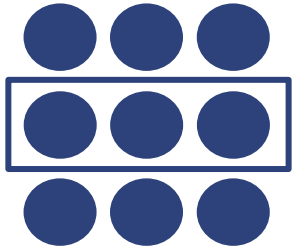
Gestalt Laws



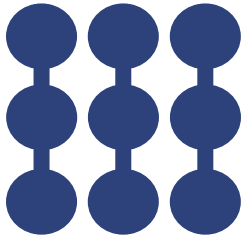
Proximity



Similarity



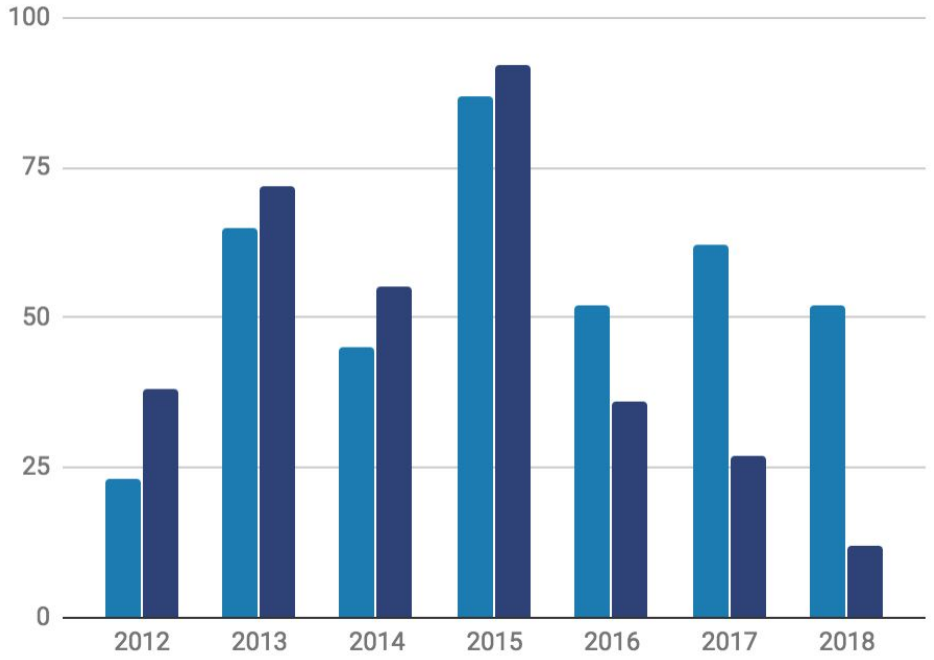
Enclosure



Connection

Sample

2012	23	38
2013	65	72
2014	45	55
2015	87	92
2016	52	36
2017	62	27
2018	52	12

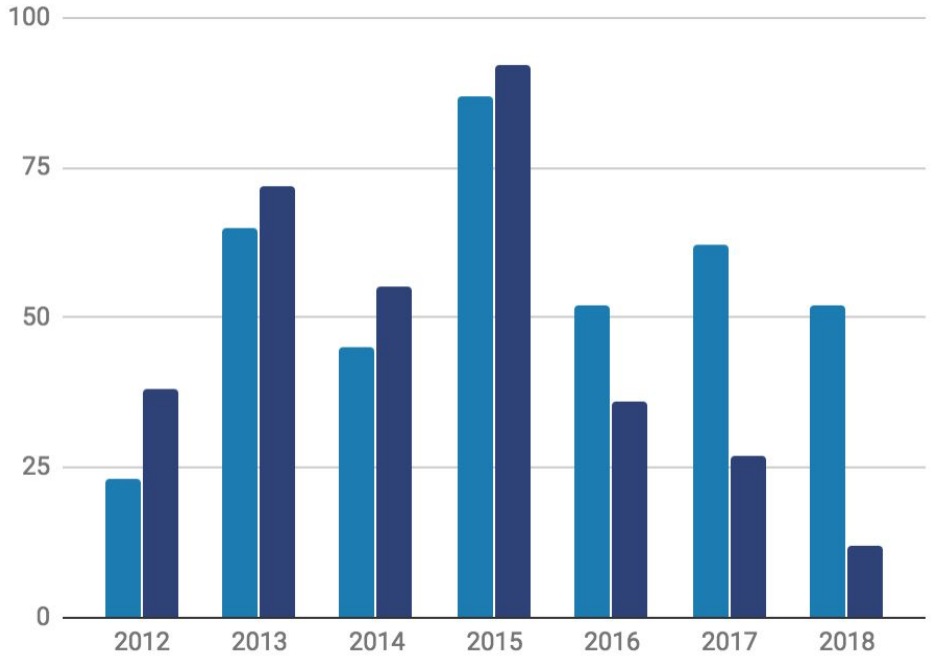


Sample

2012	23	38
2013	65	72
2014	45	55
2015	87	92
2016	52	36
2017	62	27
2018	52	12

Sample

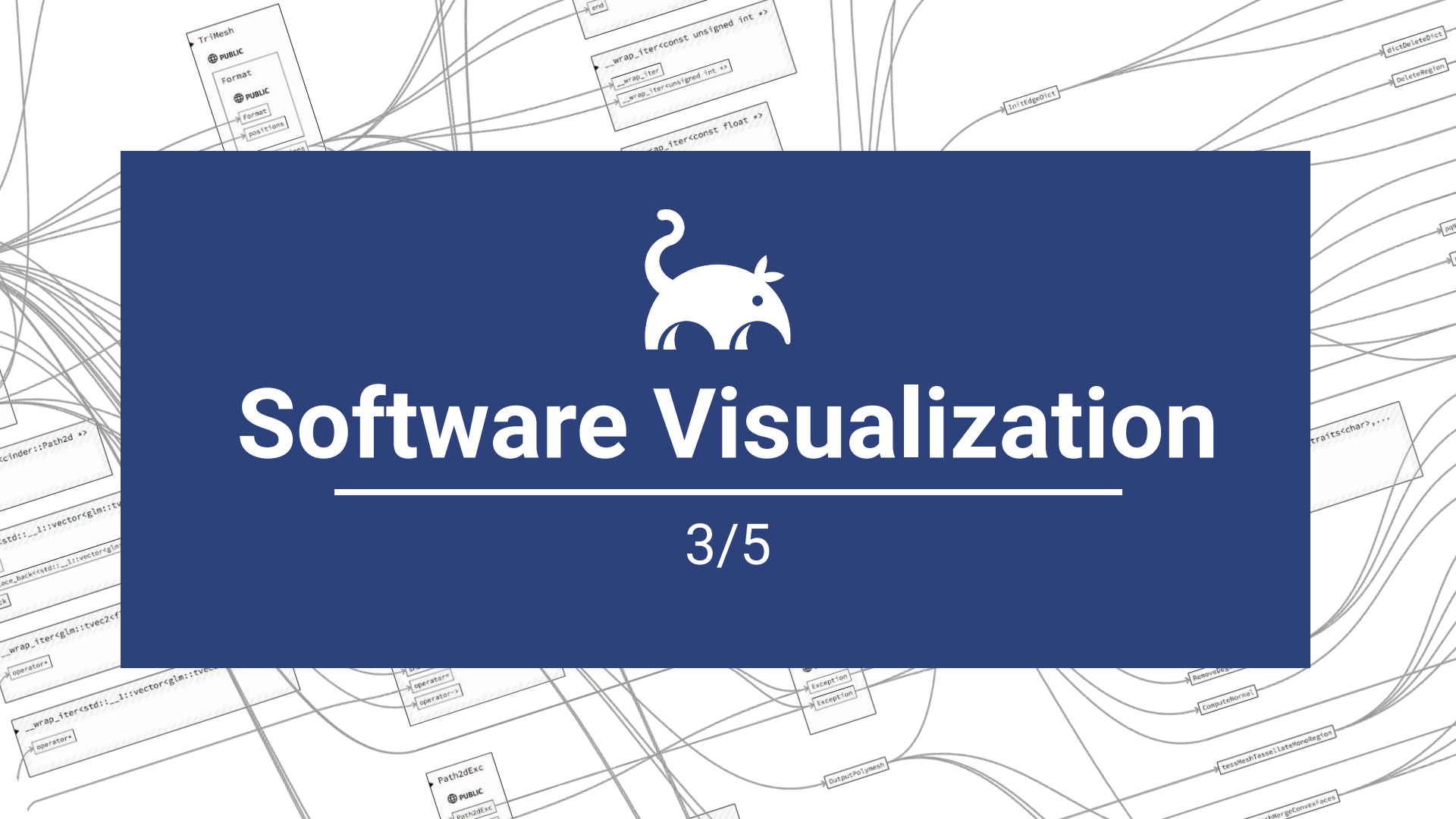
2012	23	38
2013	65	72
2014	45	55
2015	87	92
2016	52	36
2017	62	27
2018	52	12



Sample 2

```
template<typename T>class Countable{
public:Countable(){++objectCount;}int
getCount()const{return objectCount;}
private:static int objectCount;};
template<typename T>int Countable<T>::
objectCount(0);class Object:public
Countable<Object>{};int main(){Object
o;int c=o.getCount();return 0;}
```

```
template <typename T>
class Countable
{
public:
    Countable() {
        ++objectCount;
    }

    int getCount() const {
        return objectCount;
    }

private:
    static int objectCount;
};

template <typename T> int Countable<T>::objectCount(0);

class Object : public Countable<Object>
{};

int main()
{
    Object o;
    int c = o.getCount();
    return 0;
}
```



Software Visualization

3/5

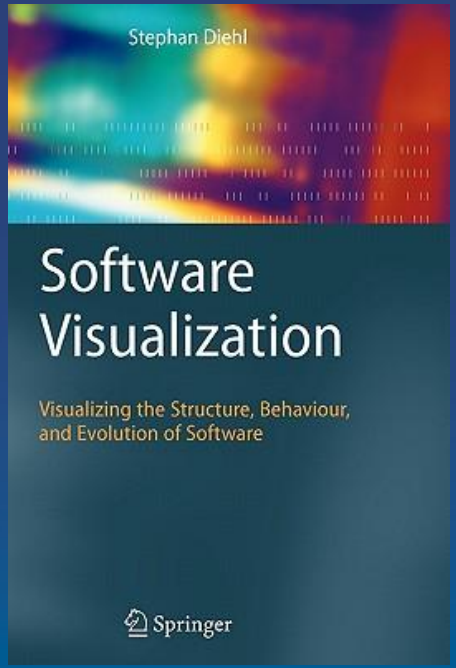
*“The **goal of software visualization** is to help to **comprehend software systems** and to **improve the productivity** of the software development process.”*

- Stephan Dhiel

Software Visualization - Visualizing the Structure, Behaviour, and Evolution of Software

Software Visualization

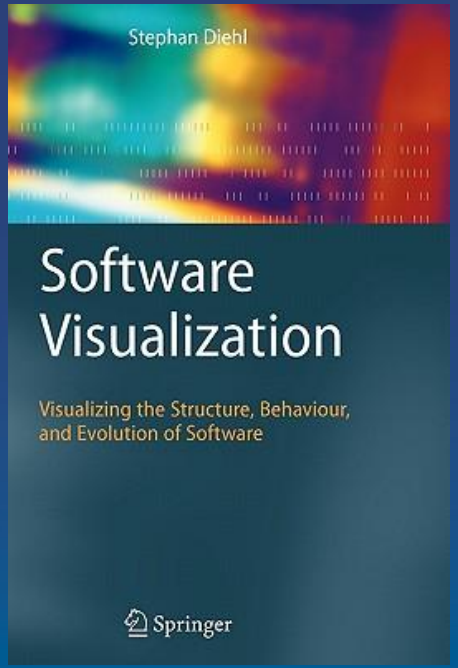
- **Structure**:
  code is data (static analysis)
- **Behaviour**:
  data from execution (dynamic analysis)
- **Evolution**:
  history is data (version control)



Software Visualization, Stephan Dhiel, 2007
https://www.springer.com/us/book/9783540465041

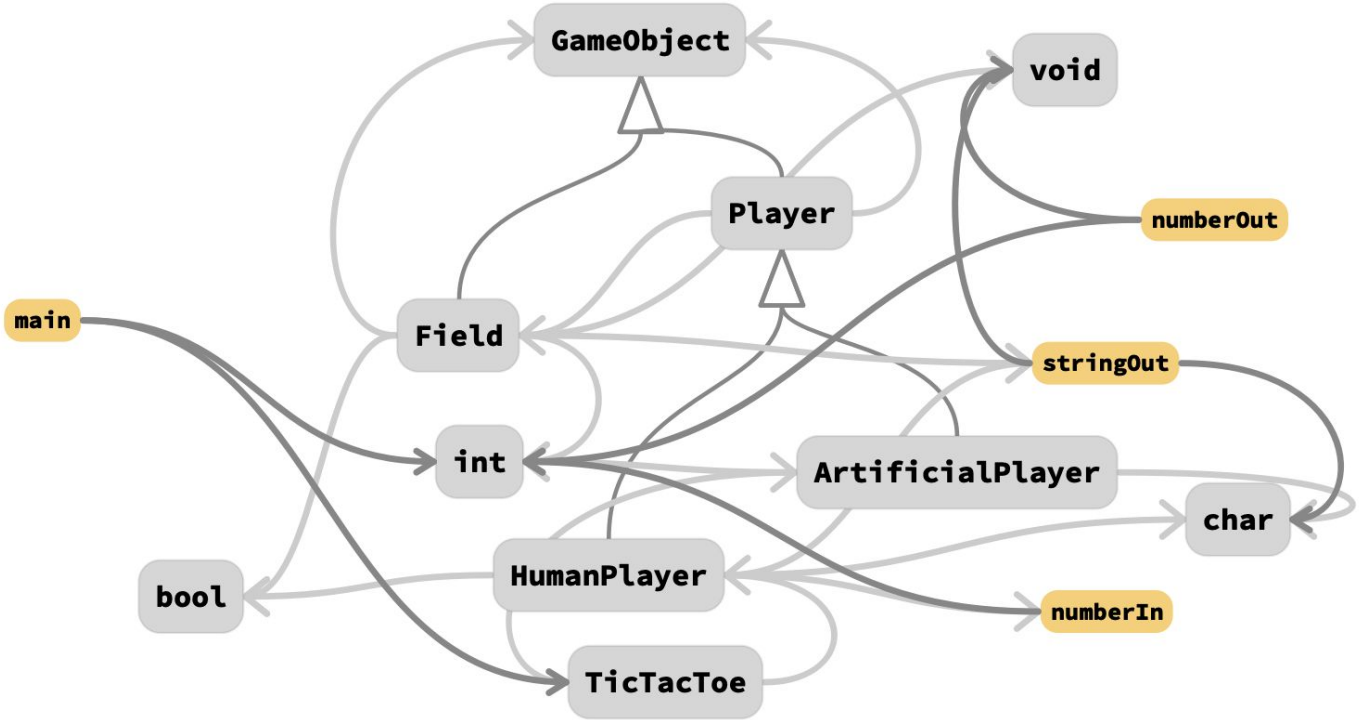
Software Visualization

- **Structure**:
  code is data (static analysis)
- **Behaviour**:
  data from execution (dynamic analysis)
- **Evolution**:
  history is data (version control)

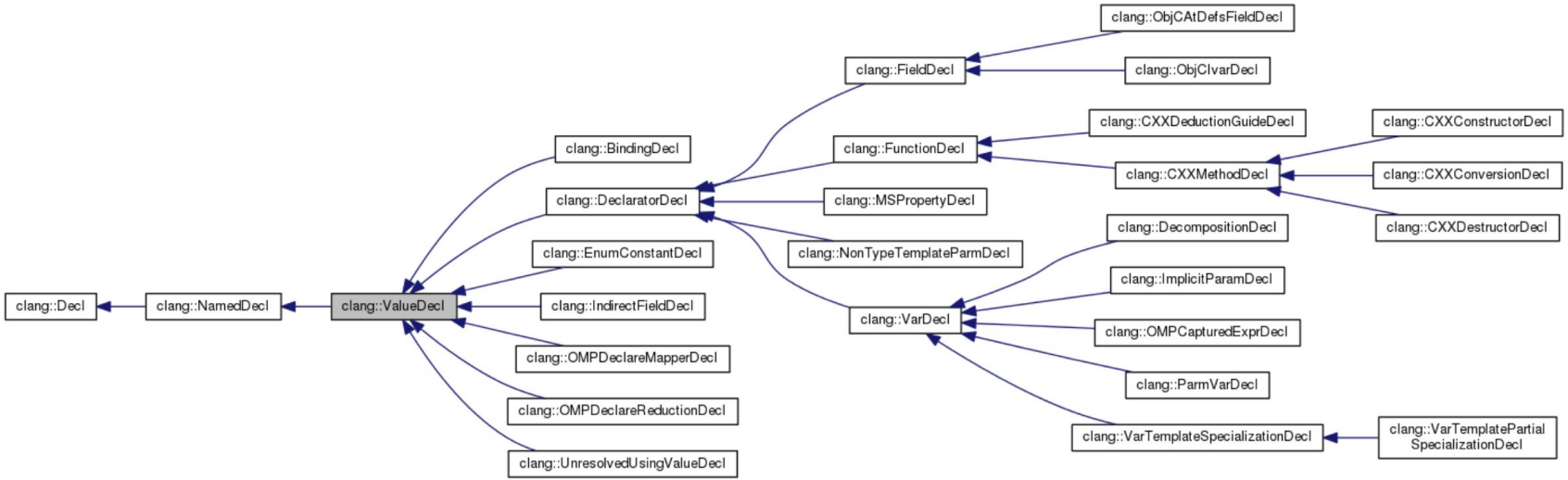


Software Visualization, Stephan Dhiel, 2007
https://www.springer.com/us/book/9783540465041

Structure: Node-Link-Diagram

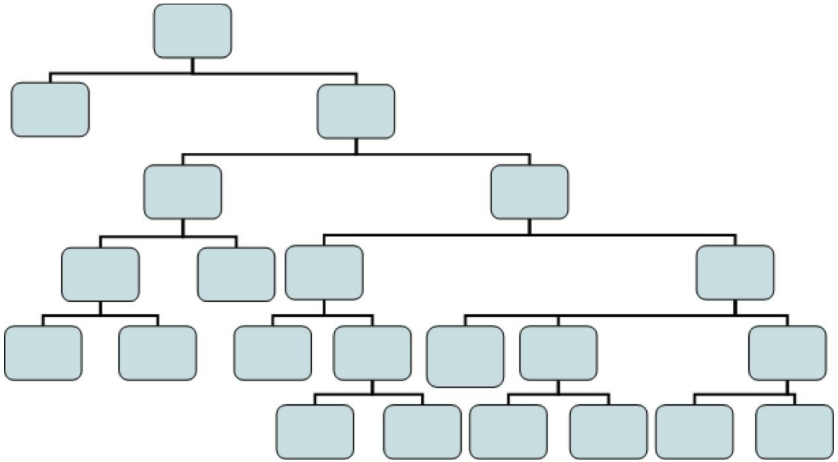


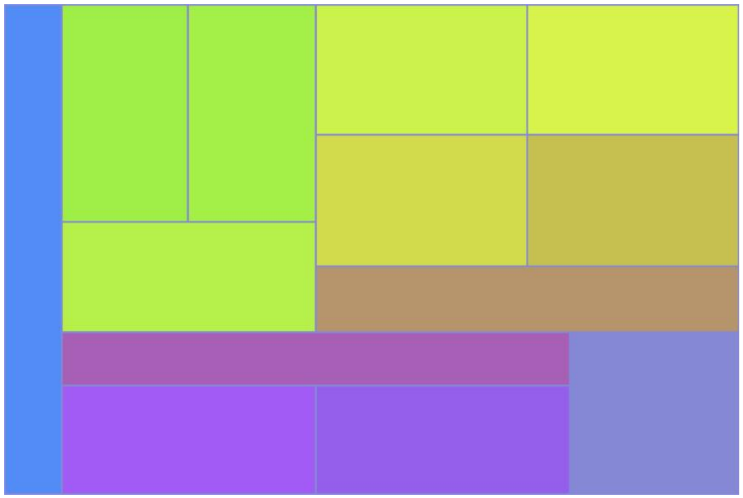
Structure: Doxygen Inheritance Tree



https://clang.llvm.org/doxygen/classclang_1_1ValueDecl.html

Structure: Treemap





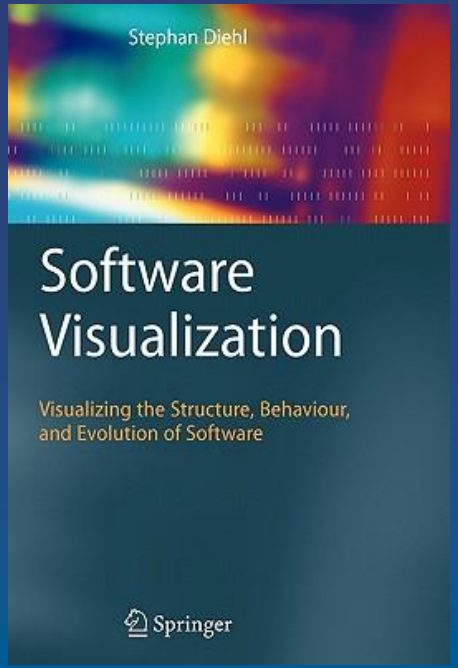
Ward, Matthew O., Georges Grinstein, and Daniel Keim. 2015. Interactive Data Visualization: Foundations, Techniques, and Applications, Second Edition - 360 Degree Business. 2nd. Natick, MA, USA: A. K. Peters, Ltd. ISBN: 1482257378.

Structure: NDepend Code Metrics

https://www.ndepend.com/Doc/Treemap/TreemapColorCursor.png

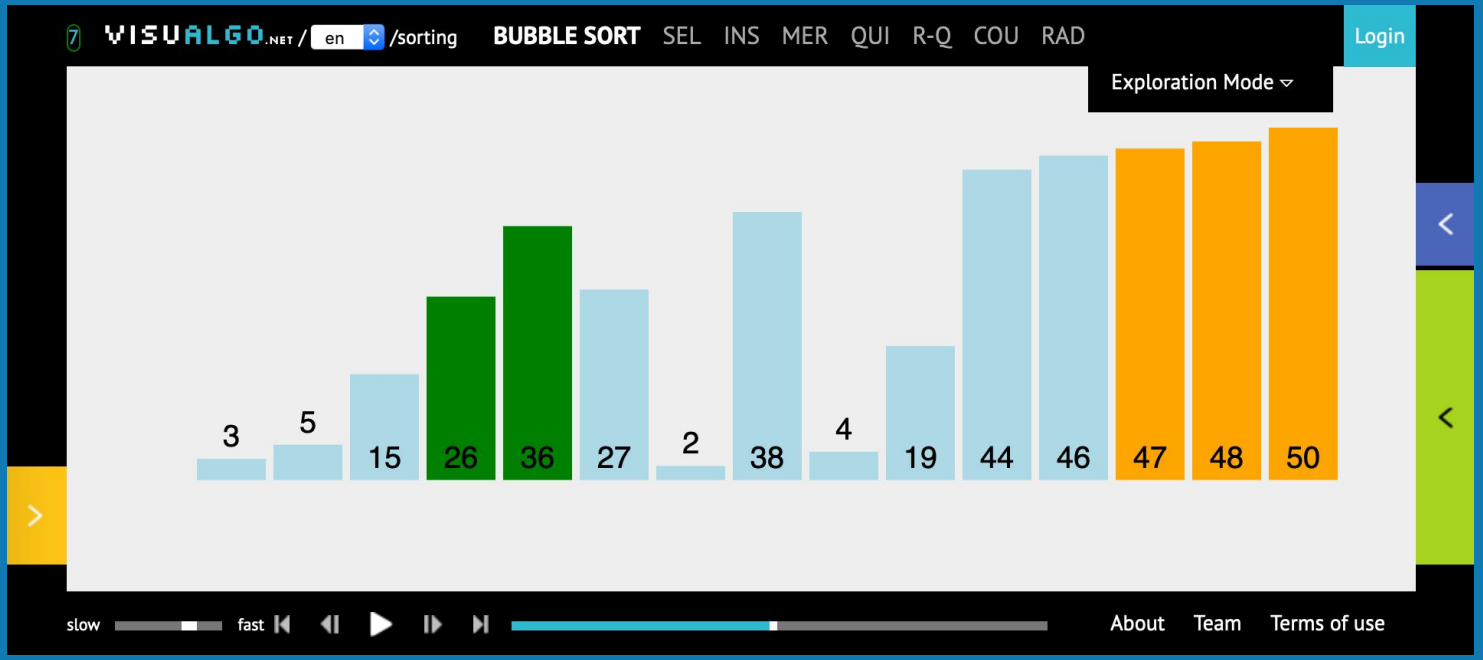
Software Visualization

- **Structure**:
  code is data (static analysis)
- **Behaviour**:
  data from execution (dynamic analysis)
- **Evolution**:
  history is data (version control)



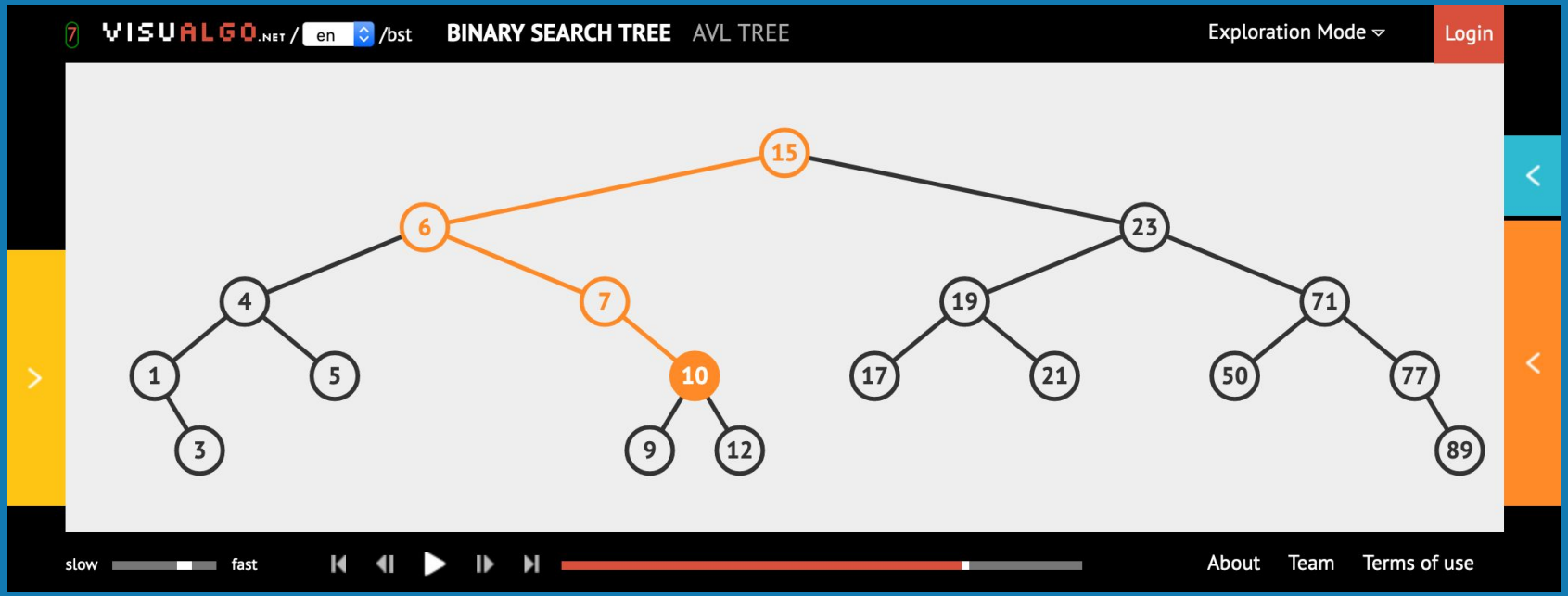
Software Visualization, Stephan Dhiel, 2007
https://www.springer.com/us/book/9783540465041

Behaviour: Visualgo Sorting



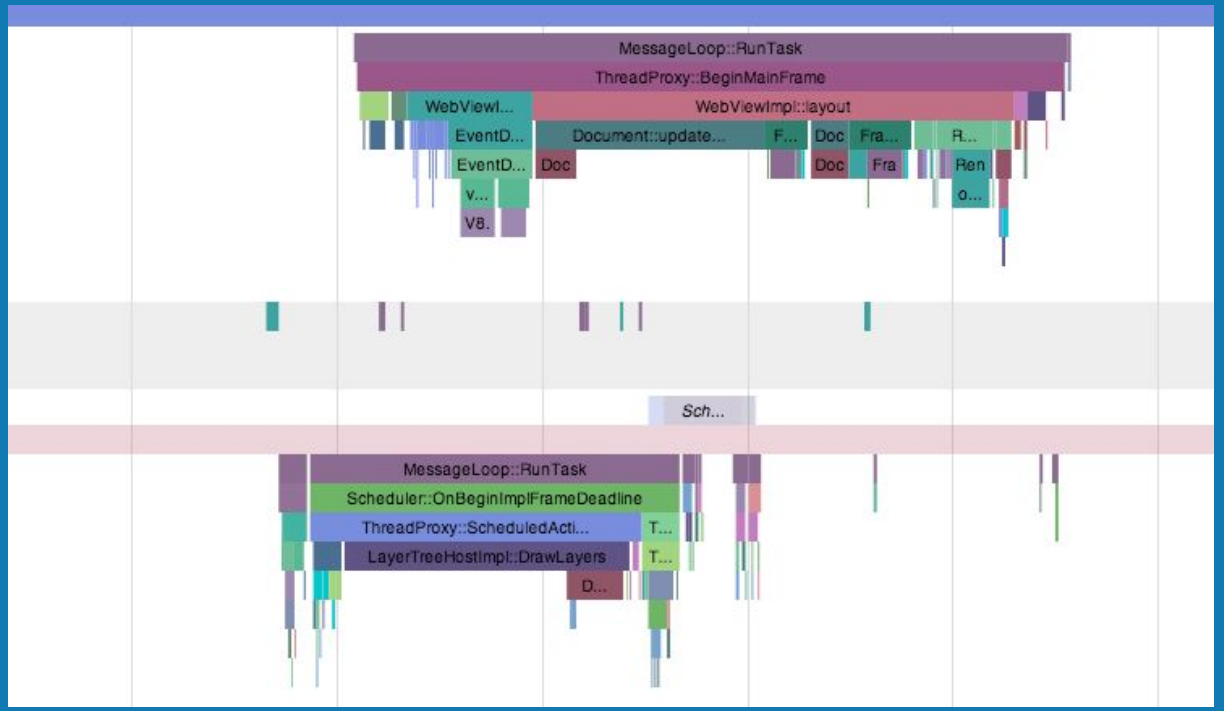
https://visualgo.net/en/sorting

Behaviour: Visualgo Binary Search



https://visualgo.net/en/bst

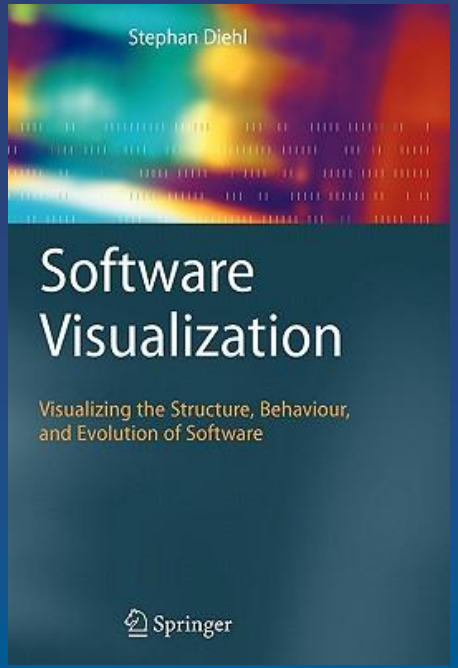
Behaviour: TraceViewer Flame Chart



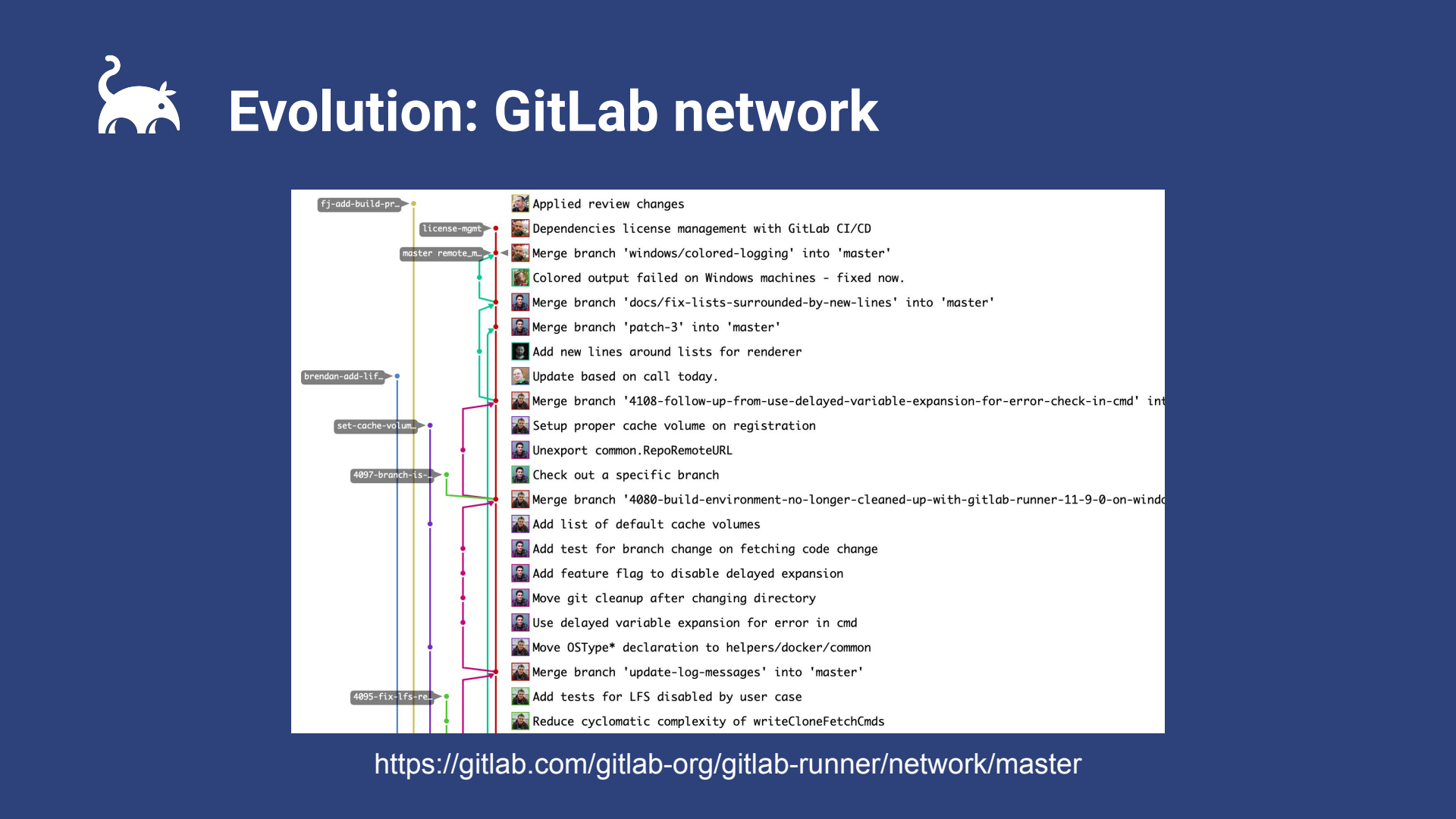
https://www.chromium.org/developers/how-tos/trace-event-profiling-tool/using-frameviewer/03-individual%20trace.png

Software Visualization

- **Structure**:
  code is data (static analysis)
- **Behaviour**:
  data from execution (dynamic analysis)
- **Evolution**:
  history is data (version control)



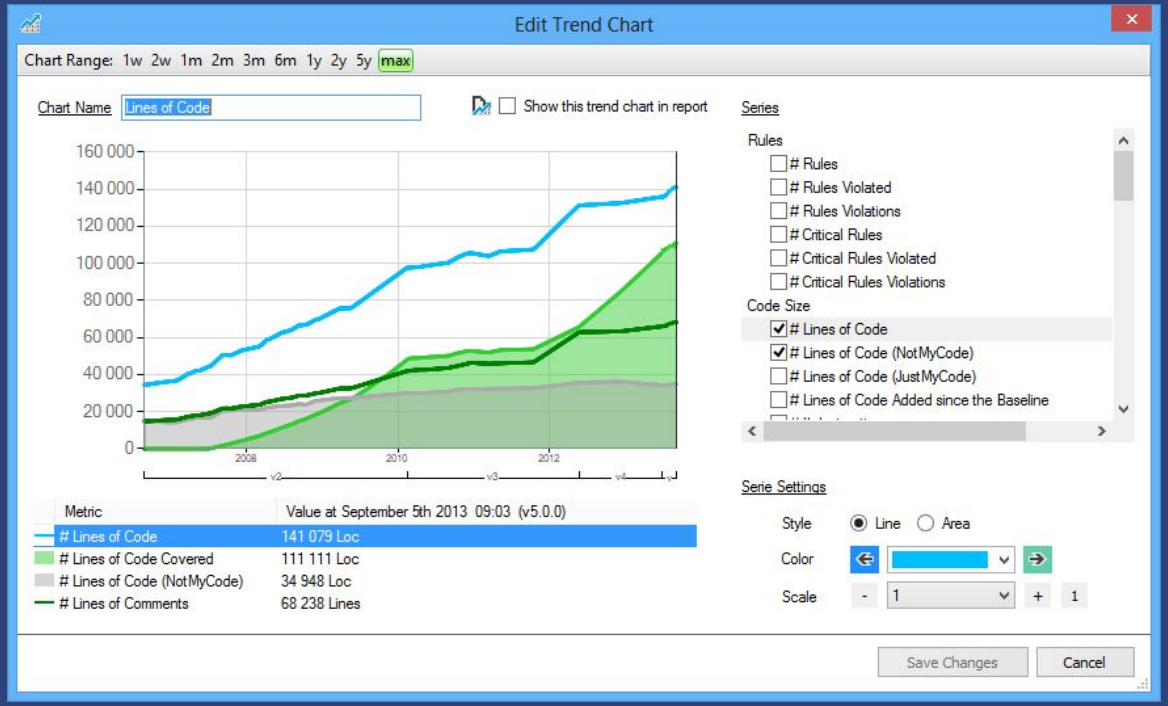
Software Visualization, Stephan Dhiel, 2007
https://www.springer.com/us/book/9783540465041



Evolution: GitLab network

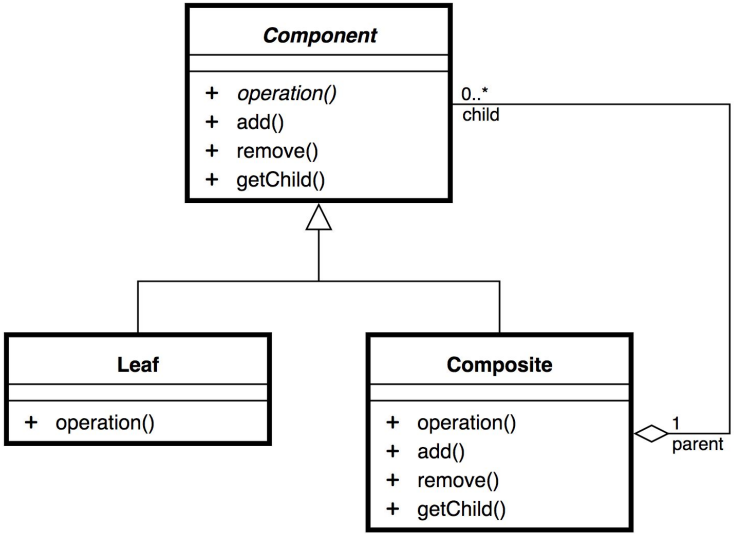
https://gitlab.com/gitlab-org/gitlab-runner/network/master

Evolution: NDepend Trend Monitoring



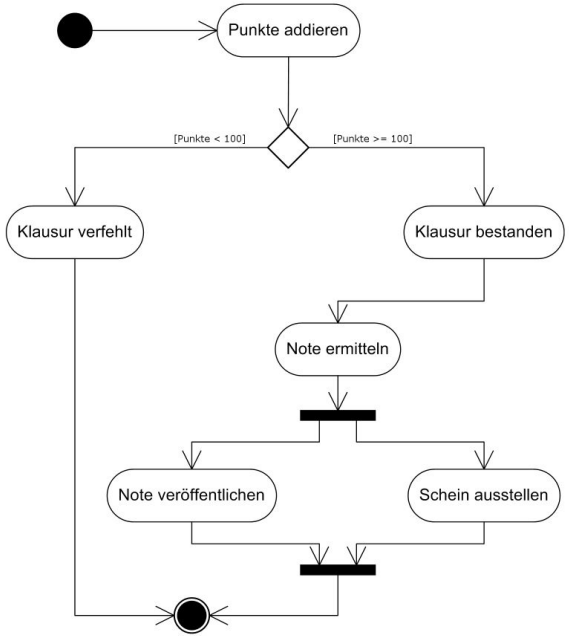
https://www.ndepend.com/Doc/Trend/TrendChartEdit.PNG

UML



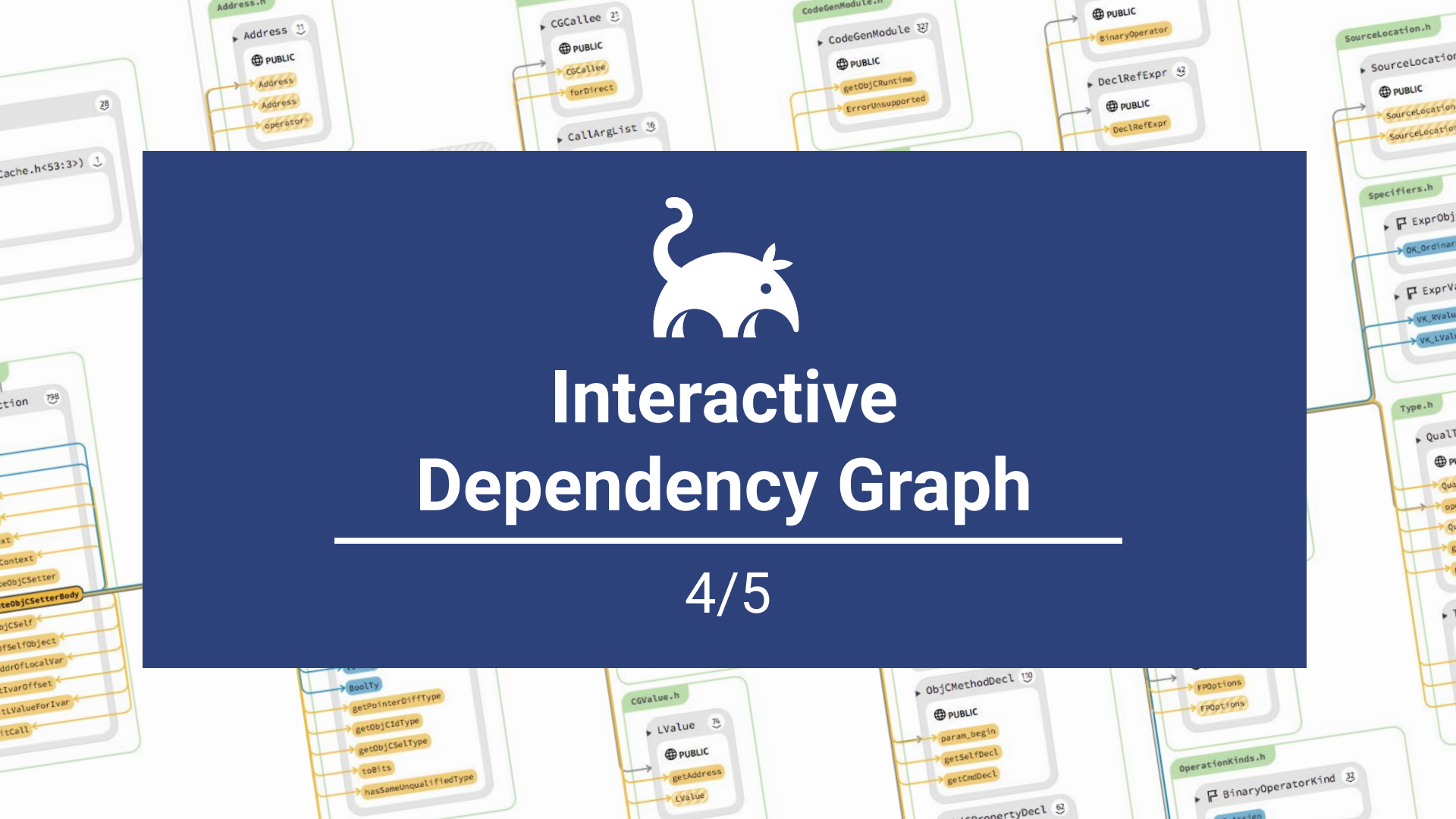
Class Diagram

By Composite_UML_class_diagram.svg: Trashtoyderivative work: « Aaron Rotenberg « Talk « (Composite_UML_class_diagram.svg) [Public domain], via Wikimedia Commons (https://commons.wikimedia.org/wiki/File:Composite_UML_class_diagram_(fixed).svg)



Activity Diagram

By Original PNG Aktivity diagram 2.png by Stern (Redrawn in SVG) [CC BY-SA 3.0 (https://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons (https://commons.wikimedia.org/wiki/File:Uml-Activity-Beispiel1.svg)


Interactive
Dependency Graph
4/5

*“**Dependency** is one of the **most critical problems in software development.** Much legacy code work involves breaking dependencies so that change can be easier.”*

- Michael Feathers

Working Effectively with Legacy Code

Goals

- Overview of existing symbols
- Dependencies between symbols

```
1
2  template <typename T>
3  class Countable
4  {
5  public:
6      Countable() {
7          ++objectCount;
8      }
9
10     int getCount() const {
11         return objectCount;
12     }
13
14 private:
15     static int objectCount;
16 };
17
18 template <typename T> int Countable<T>::objectCount(0);
19
20 class Object : public Countable<Object>
21 {};
22
23 int main()
24 {
25     Object o;
26     int c = o.getCount();
27     return 0;
28 }
29
```

Graph

Symbols (nodes):

- Type
- Function (Method)
- Variable (Field)

Relationships (edges):

- inherits
- specializes
- calls
- accesses

```
1
2  template <typename T>
3  class Countable
4  {
5  public:
6      Countable() {
7          ++objectCount;
8      }
9
10     int getCount() const {
11         return objectCount;
12     }
13
14 private:
15     static int objectCount;
16 };
17
18 template <typename T> int Countable<T>::objectCount(0);
19
20 class Object : public Countable<Object>
21 {};
22
23 int main()
24 {
25     Object o;
26     int c = o.getCount();
27     return 0;
28 }
29
```

Nodes

Type

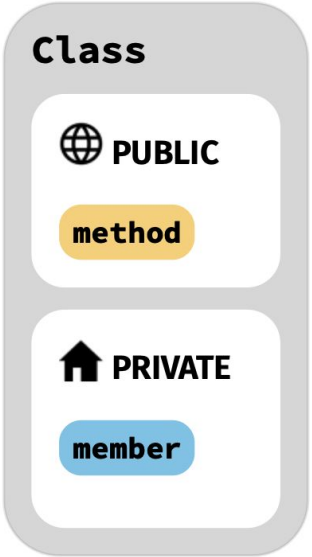
```
class Type { };
```

function

```
void function() { }
```

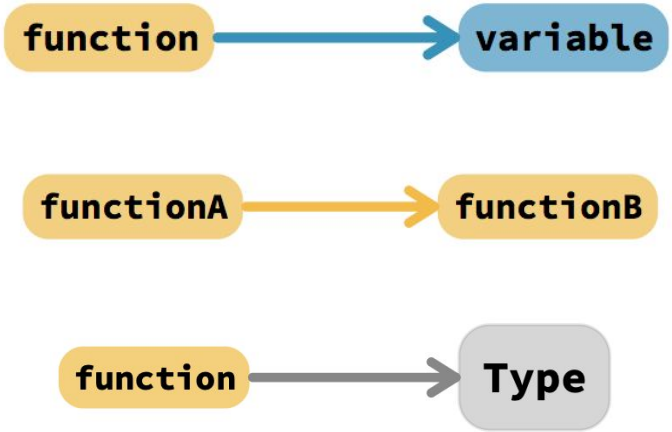
variable

```
int variable;
```




```
class Class
{
public:
    void method();
private:
    int field;
};
```

Edges - “uses”

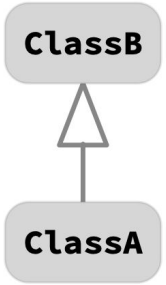



```
void function() { variable = 1; }
```

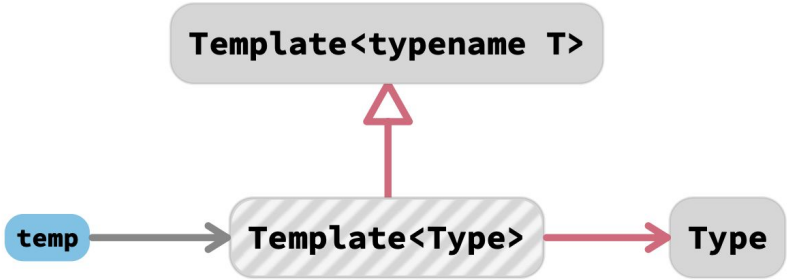
```
void functionA() { functionB(); }
```

```
void function() { Type t; }
```

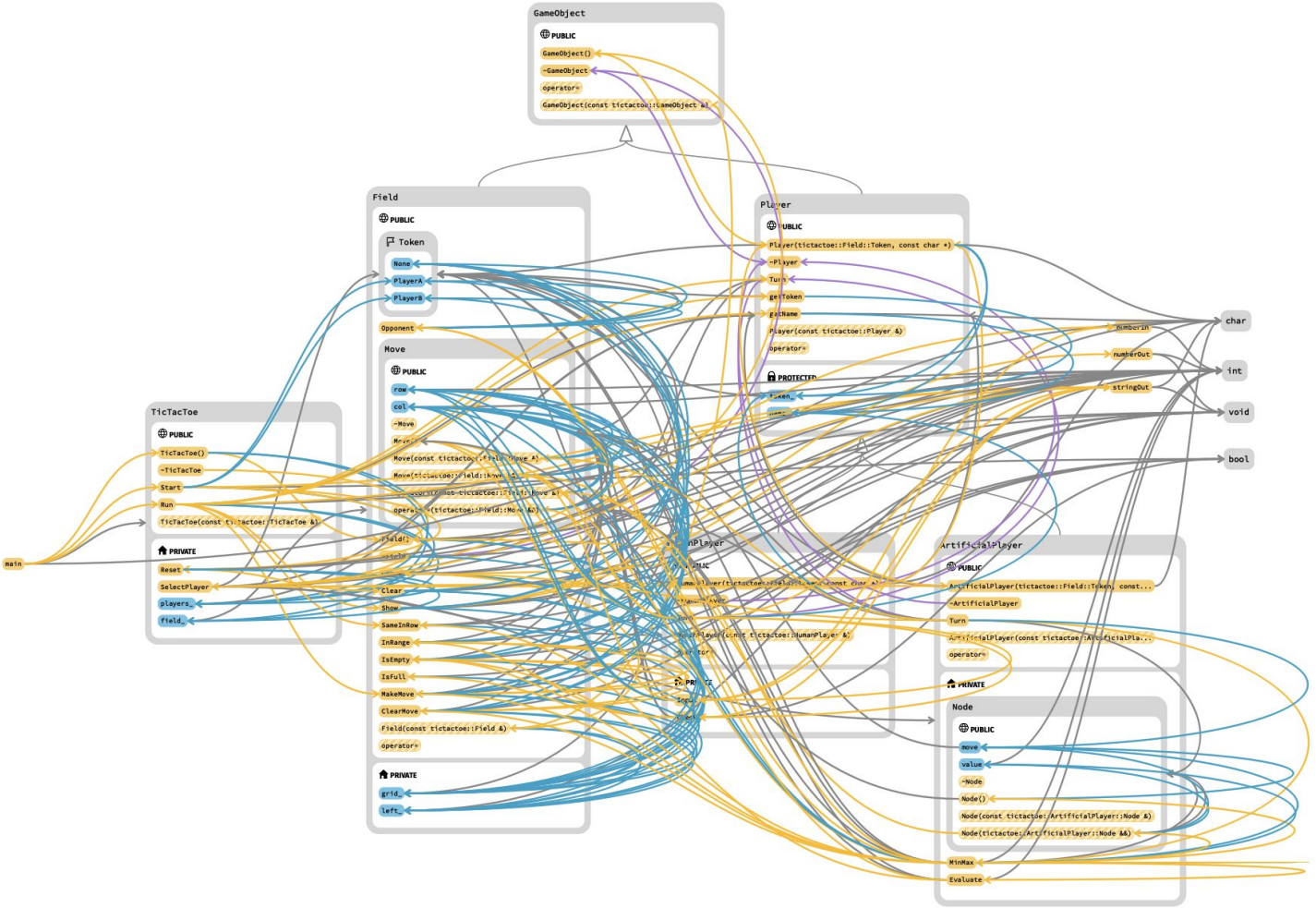
Edges - “is a”



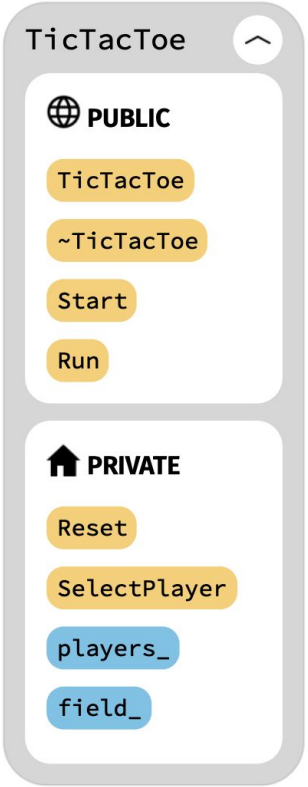

```
class ClassA
    : public ClassB
{};
```



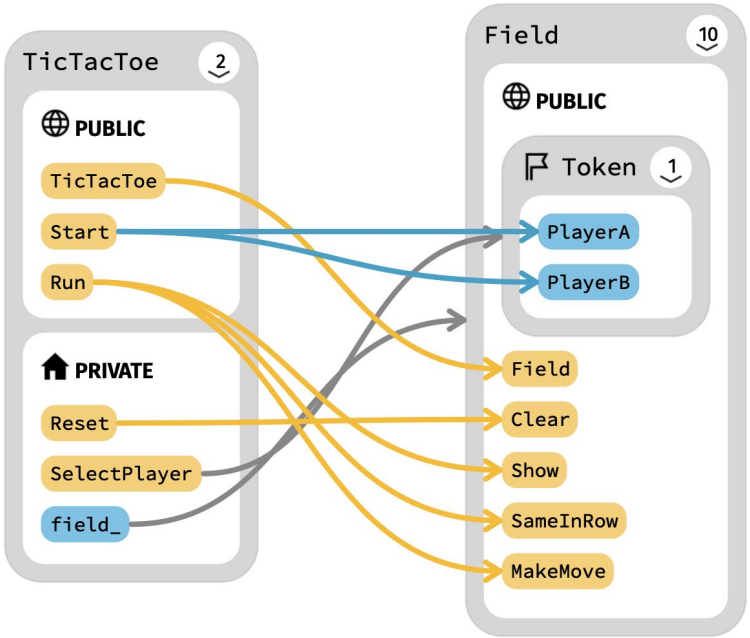

```
template <typename T>
    class Template
{};
Template<Type> temp;
```

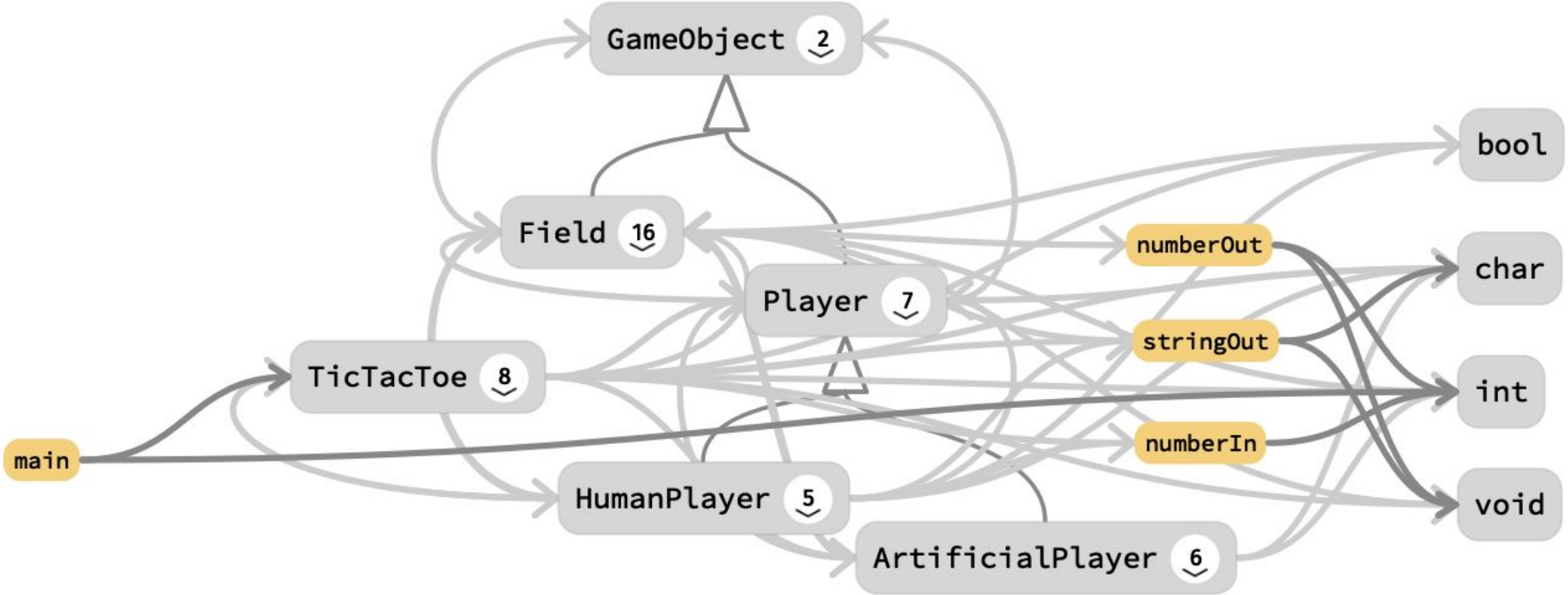

GameObject
PUBLIC
GameObject()
~GameObject
operator=
GameObject(const tictactoe::GameObject &)
Field
PUBLIC
Token
None
PlayerA
PlayerB
Opponent
Move
PUBLIC
row
col
~Move
Move(const tictactoe::Field::Move &)
Clear
Show
SameInRow
InRange
IsEmpty
IsFull
MakeMove
ClearMove
field(const tictactoe::Field &)
operator=
PRIVATE
grid_
left_
Player
PUBLIC
Player(tictactoe::Field::Token, const char *)
~Player
Turn
getToken
getName
Player(const tictactoe::Player &)
operator=
PROTECTED
token_
name_
TicTacToe
PUBLIC
TicTacToe()
~TicTacToe
Start
Run
TicTacToe(const tictactoe::TicTacToe &)
PRIVATE
Reset
SelectPlayer
players_
field_
main
ArtificialPlayer
PUBLIC
ArtificialPlayer(tictactoe::Field::Token, const...
~ArtificialPlayer
Turn
ArtificialPlayer(const tictactoe::ArtificialPla...
operator=
PRIVATE
Node
PUBLIC
move
value
~Node
Node()
Node(const tictactoe::ArtificialPlayer::Node &)
Node(tictactoe::ArtificialPlayer::Node &&)
MinMax
Evaluate
numberIn
numberOut
stringOut
char
int
void
bool

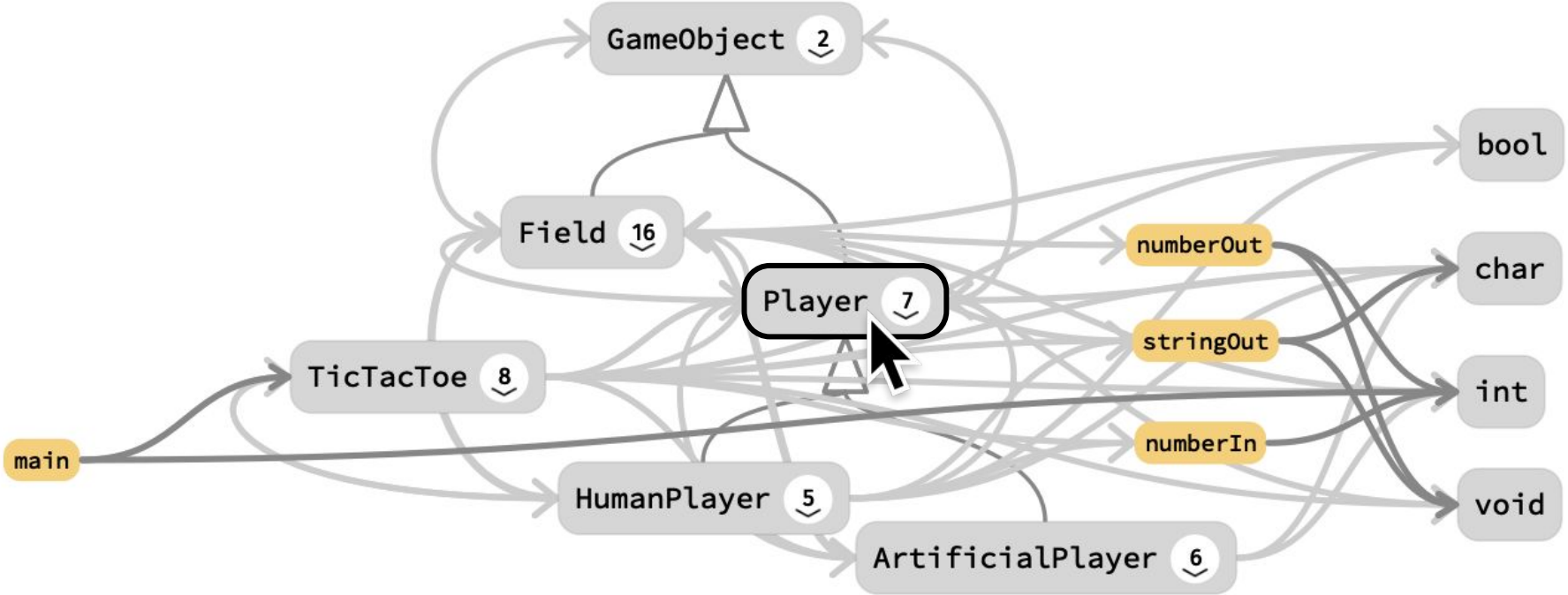
Nodes - expand/collapse



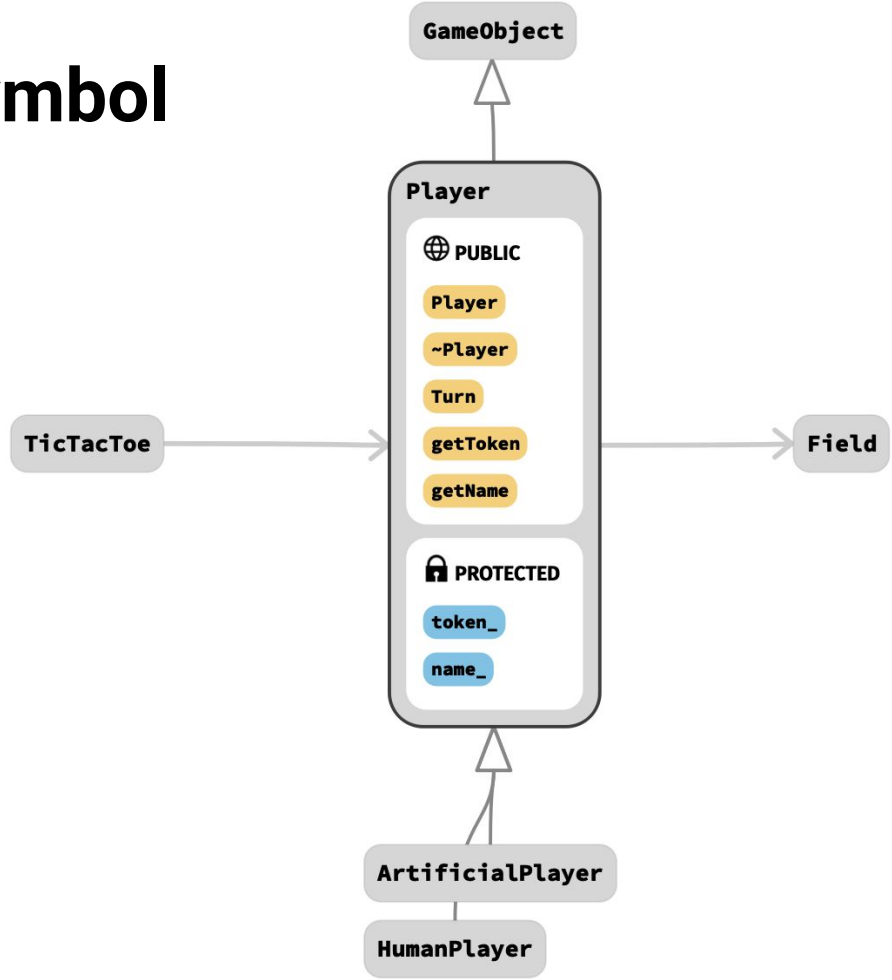
Edges - aggregation



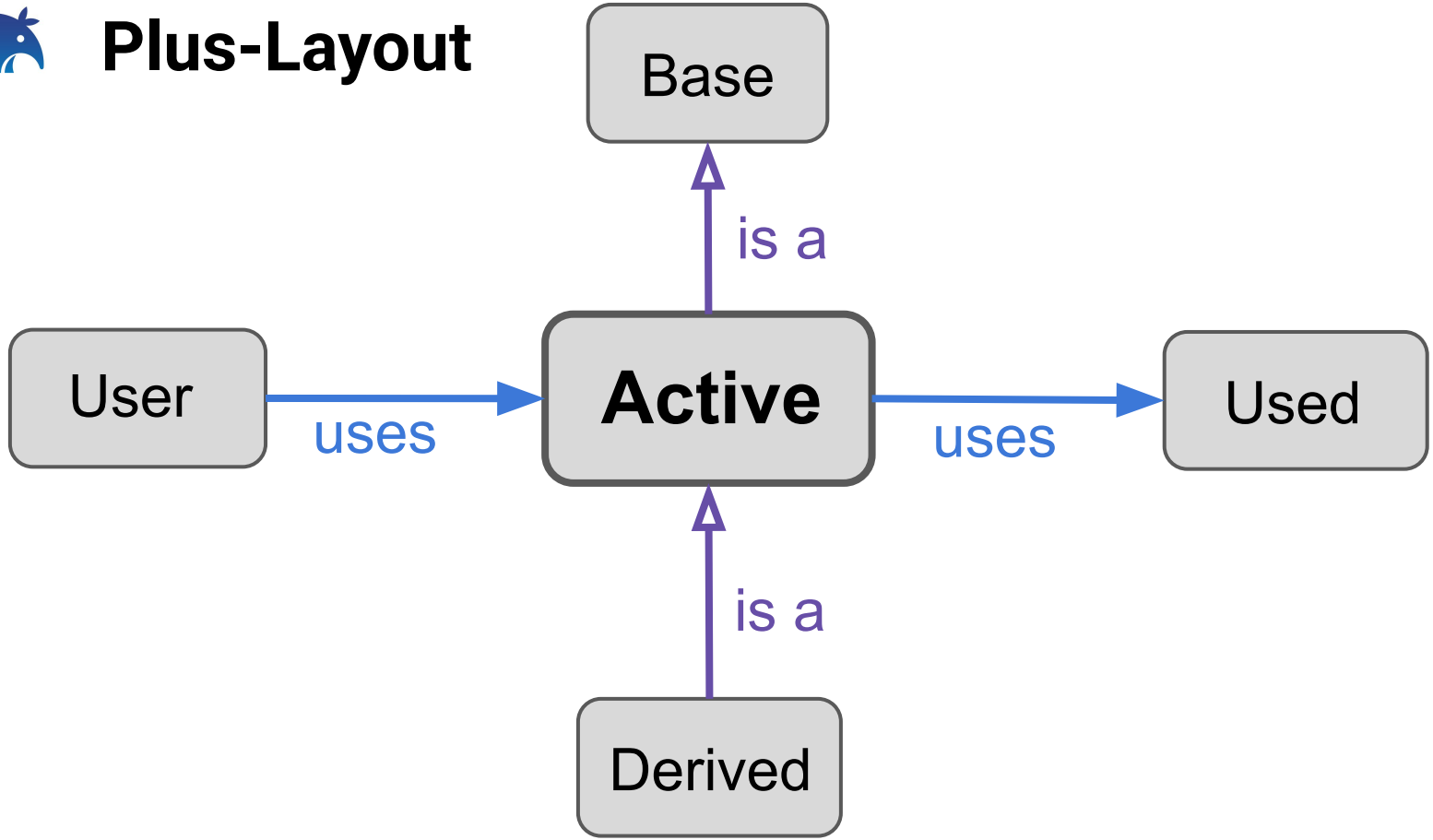

GameObject 2
Field 16
Player 7
TicTacToe 8
main
HumanPlayer 5
ArtificialPlayer 6
numberOut
stringOut
numberIn
bool
char
int
void

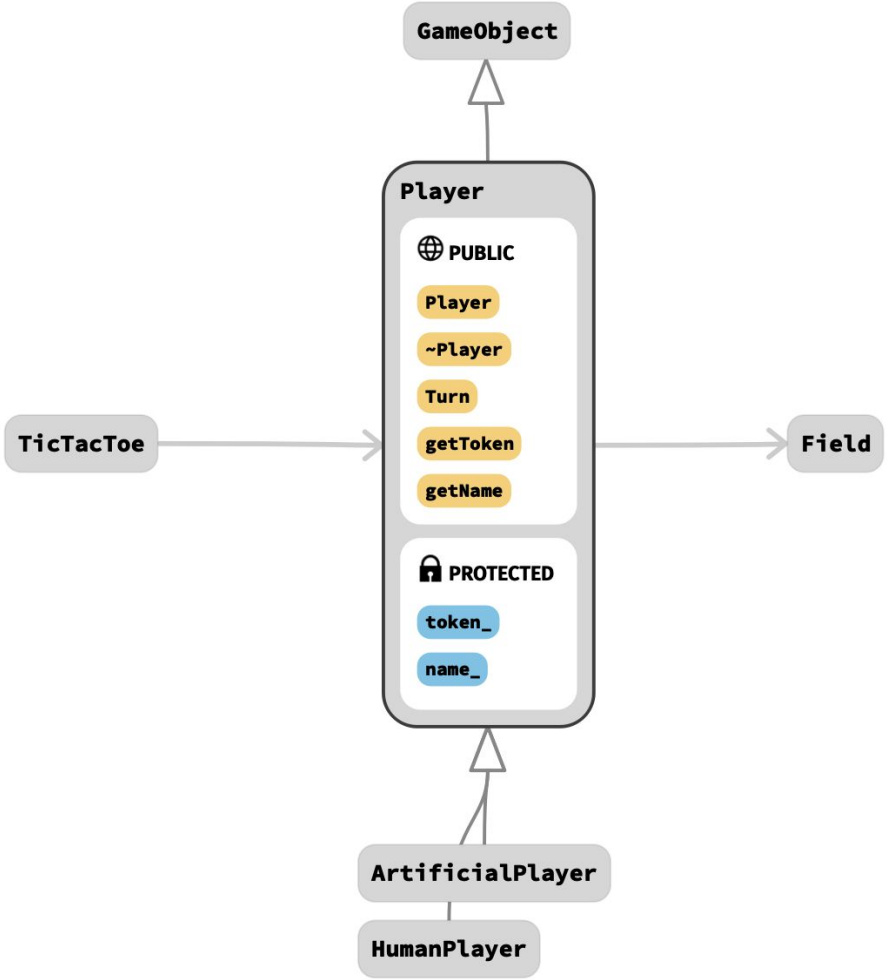

GameObject 2
Field 16
Player 7
TicTacToe 8
HumanPlayer 5
ArtificialPlayer 6
main
numberOut
stringOut
numberIn
bool
char
int
void

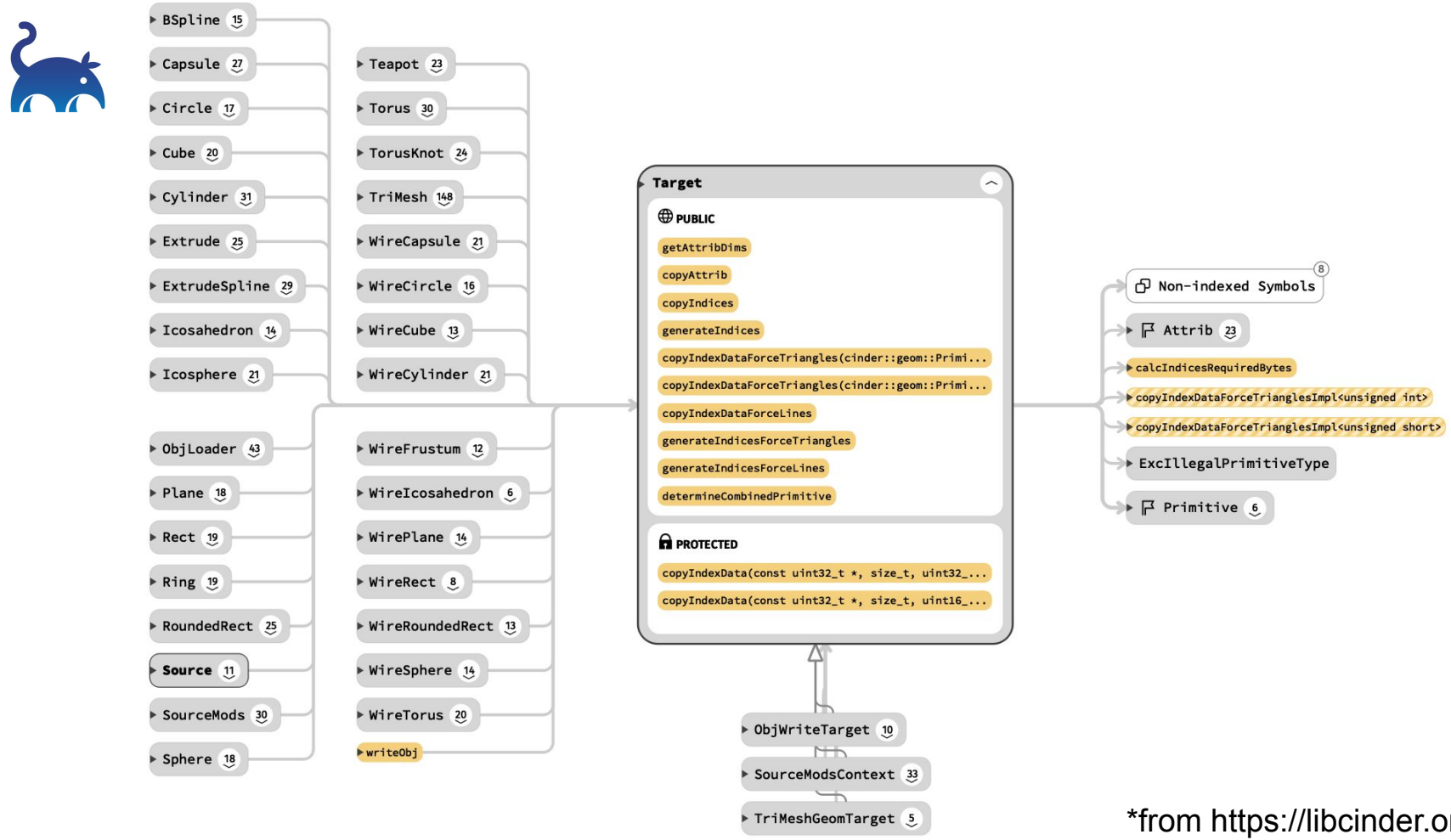
Active Symbol



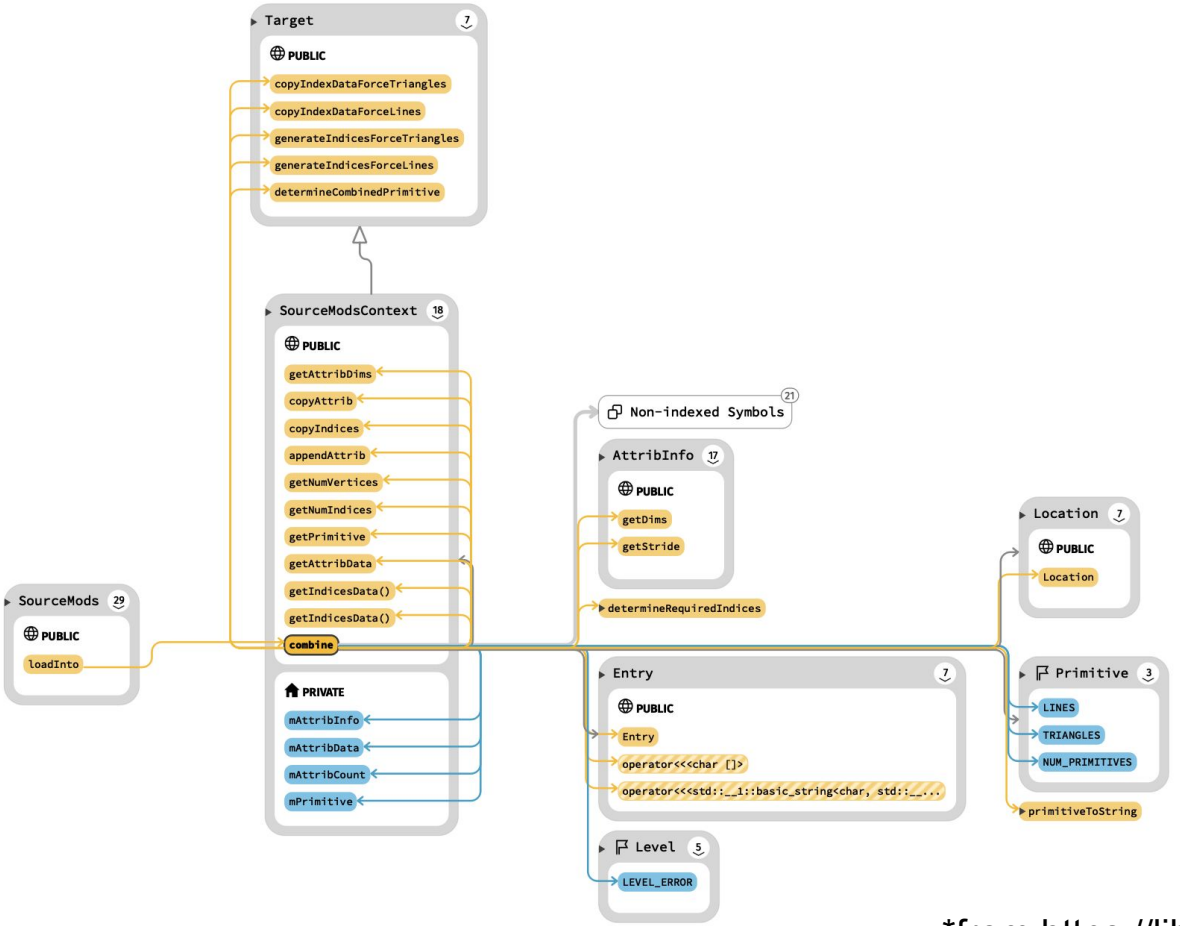
Plus-Layout



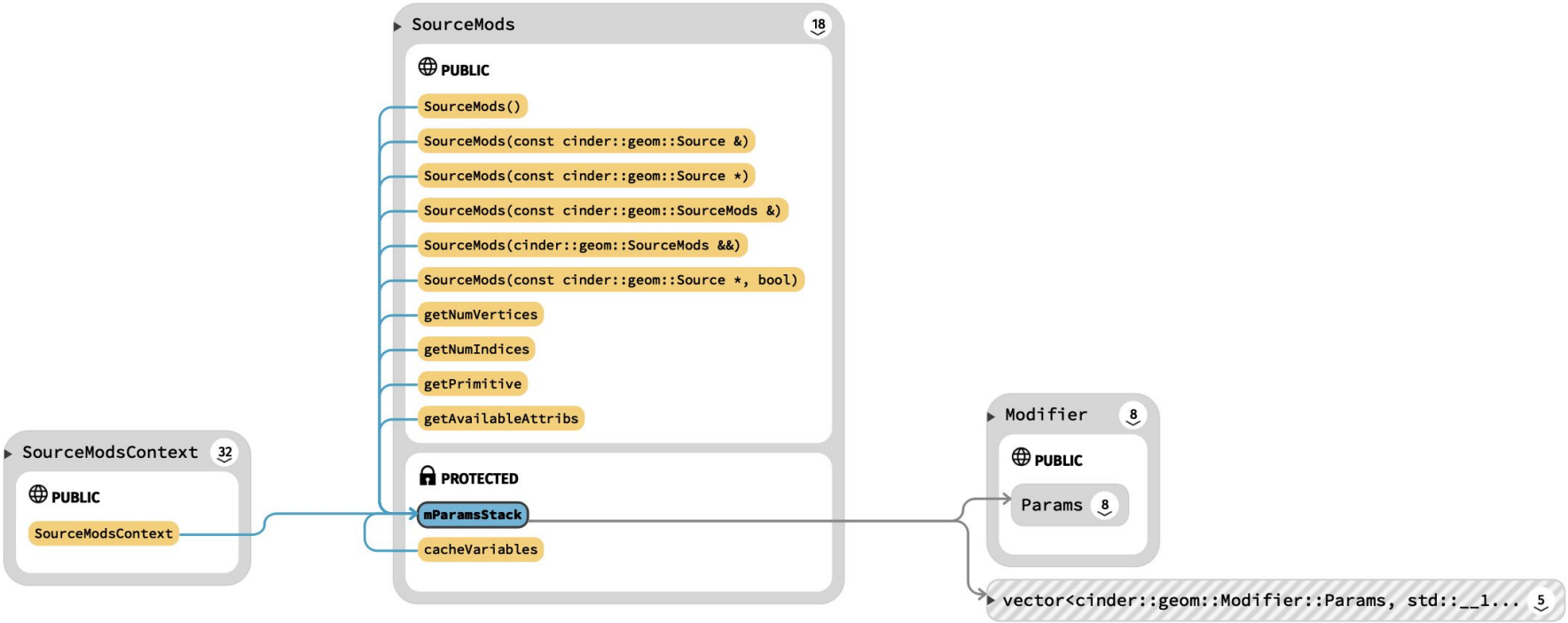

GameObject
Player
PUBLIC
Player
~Player
Turn
getToken
getName
PROTECTED
token_
name_
TicTacToe
Field
ArtificialPlayer
HumanPlayer



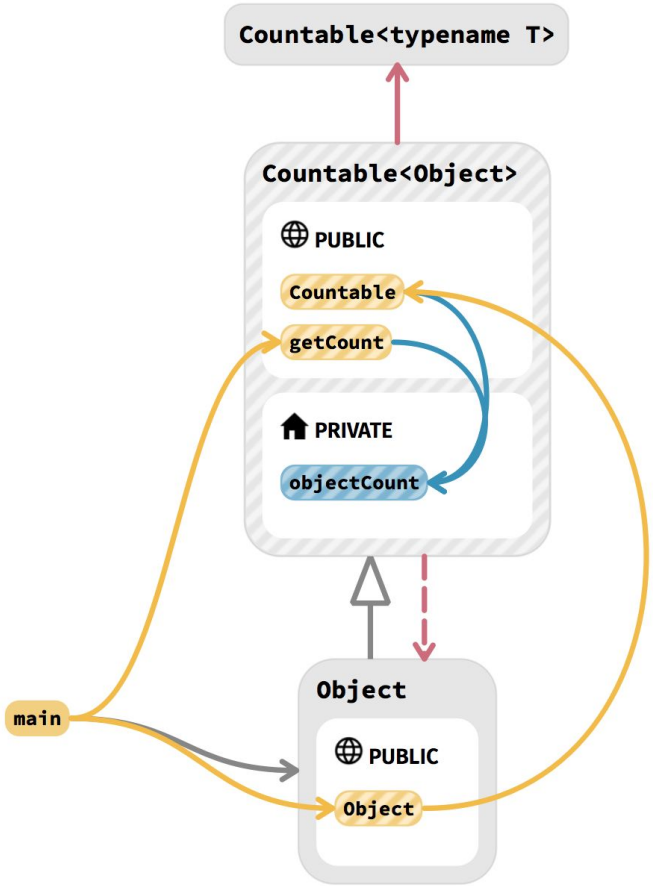
*from https://libcinder.org/



*from https://libcinder.org/

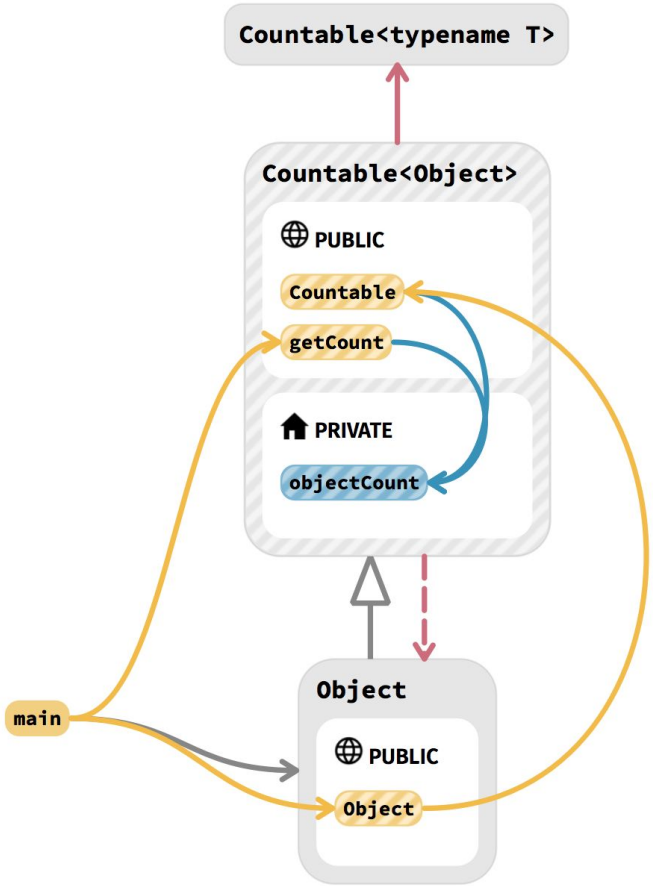


*from https://libcinder.org/



Pros

- Provides fast overview
- Brings information together
- Shows dependencies
- Fast to process



Pros

- \+ Provides fast overview
- \+ Brings information together
- \+ Shows dependencies
- \+ Fast to process

Cons

- — Lacks a lot of information
- — Can be cluttered
- — Notation needs to be learned

Pros

- Ground truth
- Shows all information
- Familiar syntax from writing

```
1
2   template <typename T>
3   class Countable
4   {
5   public:
6       Countable() {
7           ++objectCount;
8       }
9
10      int getCount() const {
11          return objectCount;
12      }
13
14  private:
15      static int objectCount;
16  };
17
18  template <typename T> int Countable<T>::objectCount(0);
19
20  class Object : public Countable<Object>
21  {};
22
23  int main()
24  {
25      Object o;
26      int c = o.getCount();
27      return 0;
28  }
29
```

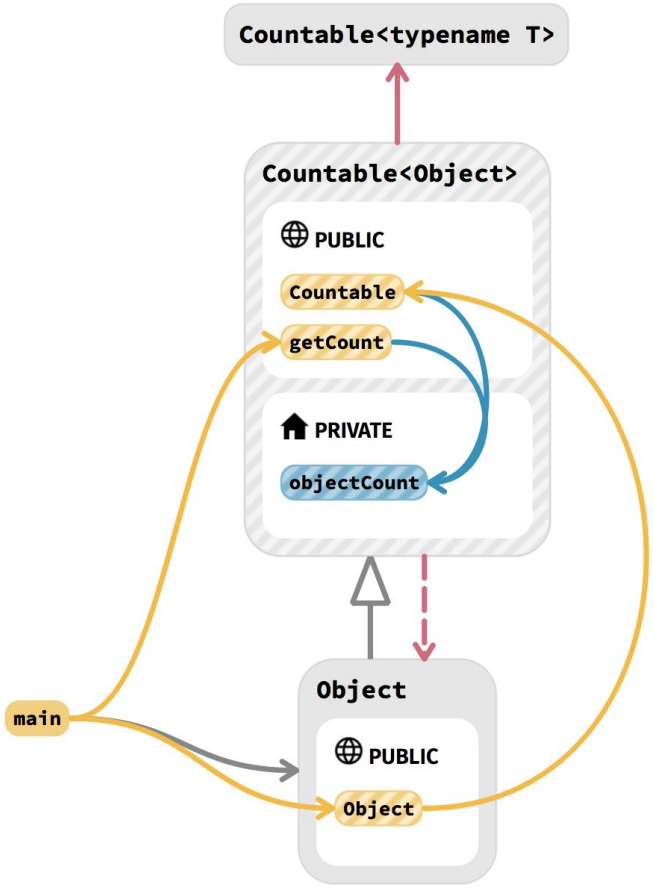
Pros

- Ground truth
- Shows all information
- Familiar syntax from writing

Cons

- Large amount of data
- Information spread across multiple locations/files
- Slow to process

```
1
2   template <typename T>
3   class Countable
4   {
5   public:
6       Countable() {
7           ++objectCount;
8       }
9
10      int getCount() const {
11          return objectCount;
12      }
13
14  private:
15      static int objectCount;
16  };
17
18  template <typename T> int Countable<T>::objectCount(0);
19
20  class Object : public Countable<Object>
21  {};
22
23  int main()
24  {
25      Object o;
26      int c = o.getCount();
27      return 0;
28  }
29
```



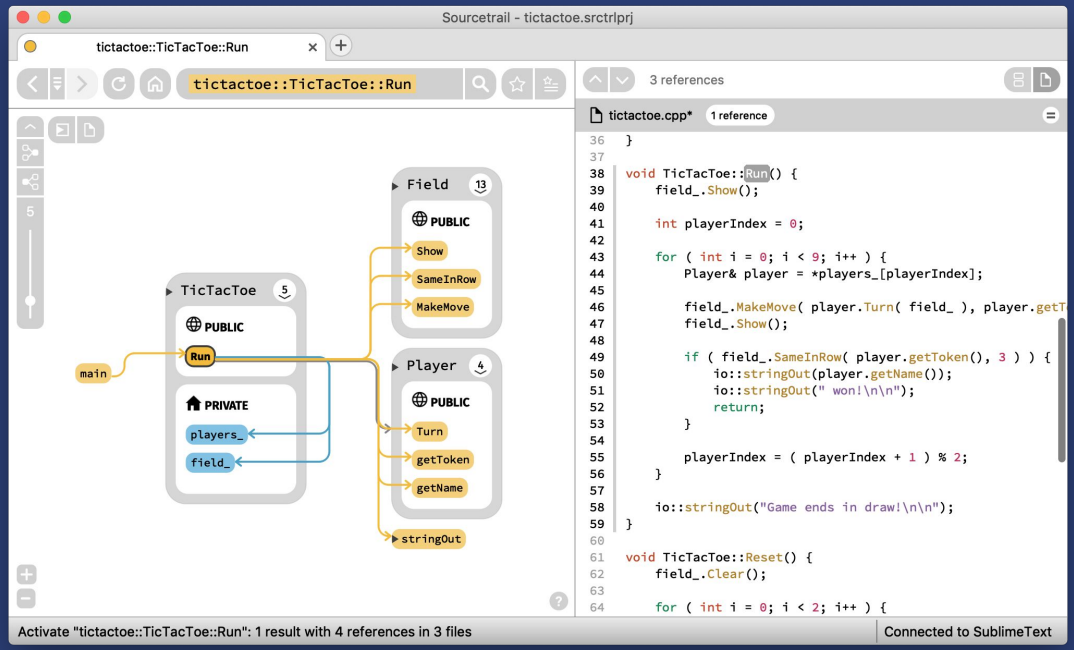

```
1
2   template <typename T>
3   class Countable
4   {
5   public:
6       Countable() {
7           ++objectCount;
8       }
9
10      int getCount() const {
11          return objectCount;
12      }
13
14  private:
15      static int objectCount;
16  };
17
18  template <typename T> int Countable<T>::objectCount(0);
19
20  class Object : public Countable<Object>
21  {};
22
23  int main()
24  {
25      Object o;
26      int c = o.getCount();
27      return 0;
28  }
29
```

SOURCETRAIL

search

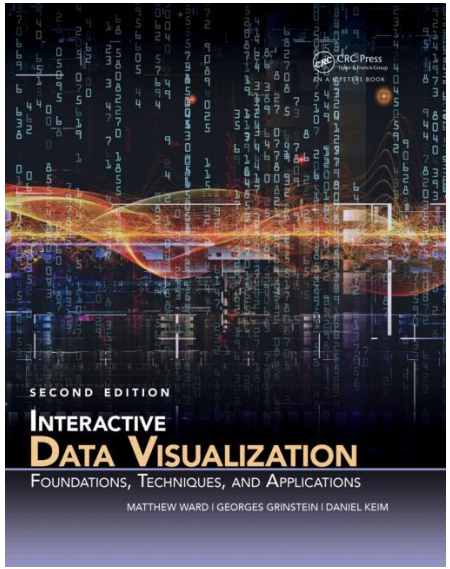
code

graph



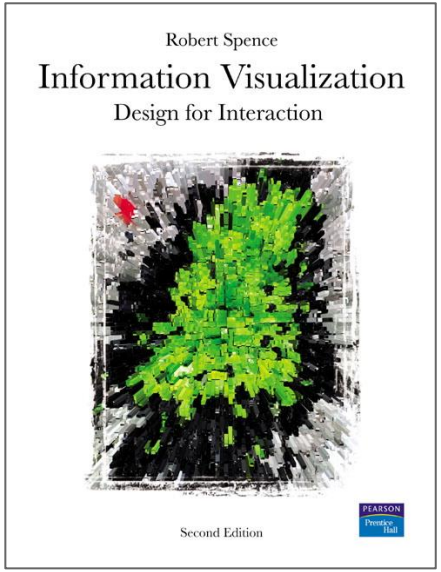
https://sourcetrail.com

Literature



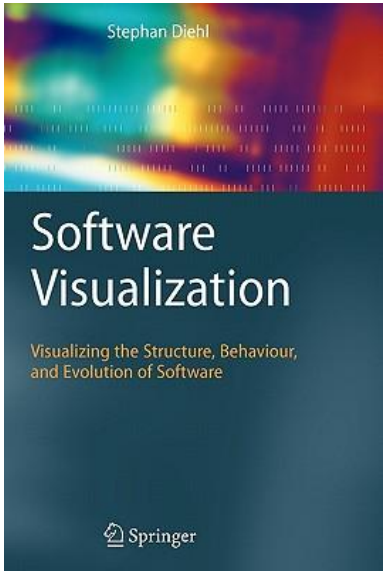
Interactive Data Visualization
M. Ward, G. Grinstein, D. Keim

https://www.crcpress.com/Interactive-Data-Visualization-Foundations-Techniques-and-Applications/Ward-Grinstein-Keim/p/book/9781482257373



Information Visualization: Design for Interaction
Robert Spence

http://catalogue.pearsoned.co.uk/educator/product/Information-Visualization-Design-for-Interaction/0132065509.page



Software Visualization
Stephan Dhiel

https://www.springer.com/gp/book/9783540465041#

Thank you!
Questions?

https://sourcetrail.com

mail@sourcetrail.com
Twitter: @Sourcetrail

Eberhard Gräther
@egraether
egraether@coati.io

Coati Software KG
Jakob-Haringer-Straße 1/127
5020 Salzburg
Austria