

Design of Software Systems (Ontwerp van SoftwareSystemen)

4 Metrics and Software Visualization

Roel Wuyts
2015-2016

Acknowledgements

Slides adopted, with permission, from Prof. Dr. Michele Lanza

<http://www.inf.unisi.ch/faculty/lanza/>

Lecture 04

Metrics & Problem Detection

You cannot control what you cannot measure

Tom de Marco

Metrics are functions that assign numbers to products, processes and resources

Software metrics are measurements which relate to software systems, processes or related documents

Metrics compress system properties and traits into numbers

Let's see some examples..

Examples of size metrics

- ▶ NOM - Number of Methods
- ▶ NOA - Number of Attributes
- ▶ LOC - Number of Lines of Code
- ▶ NOS - Number of Statements
- ▶ NOC - Number of Children

Chidamber & Kemerer, 1994
Lorenz & Kidd, 1994

Cyclomatic Complexity (CYCLO)

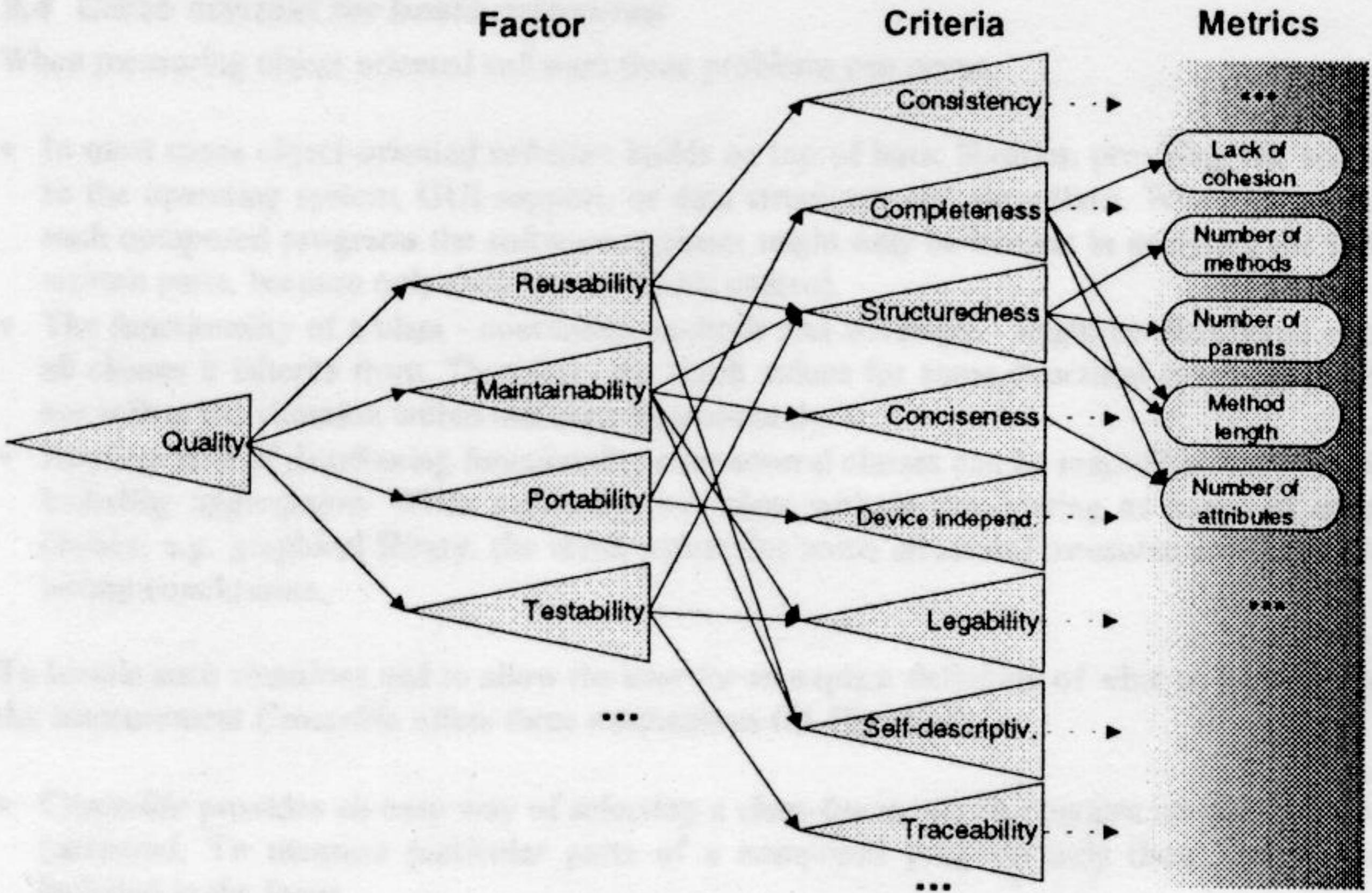
- ▶ The McCabe cyclomatic complexity (CYCLO) counts the number of independent paths through the code of a function
 - ▶ Good: it reveals the minimum number of tests to write
 - ▶ Bad: its interpretation does not directly lead to improvement actions

Weighted Method Count (WMC)

- ▶ WMC sums up the complexity of a class' methods (measured by the metric of your choice, usually CYCLO)
 - ▶ Good: It is configurable, thus adaptable to our precise needs
 - ▶ Bad: Its interpretation does not directly lead to improvement actions

Coupling Between Objects (CBO)

- ▶ CBO shows the number of classes from which methods or attributes are used.
 - ▶ Good: CBO takes into account real dependencies, not just declared ones
 - ▶ Bad: No differentiation of types and/or intensity of coupling



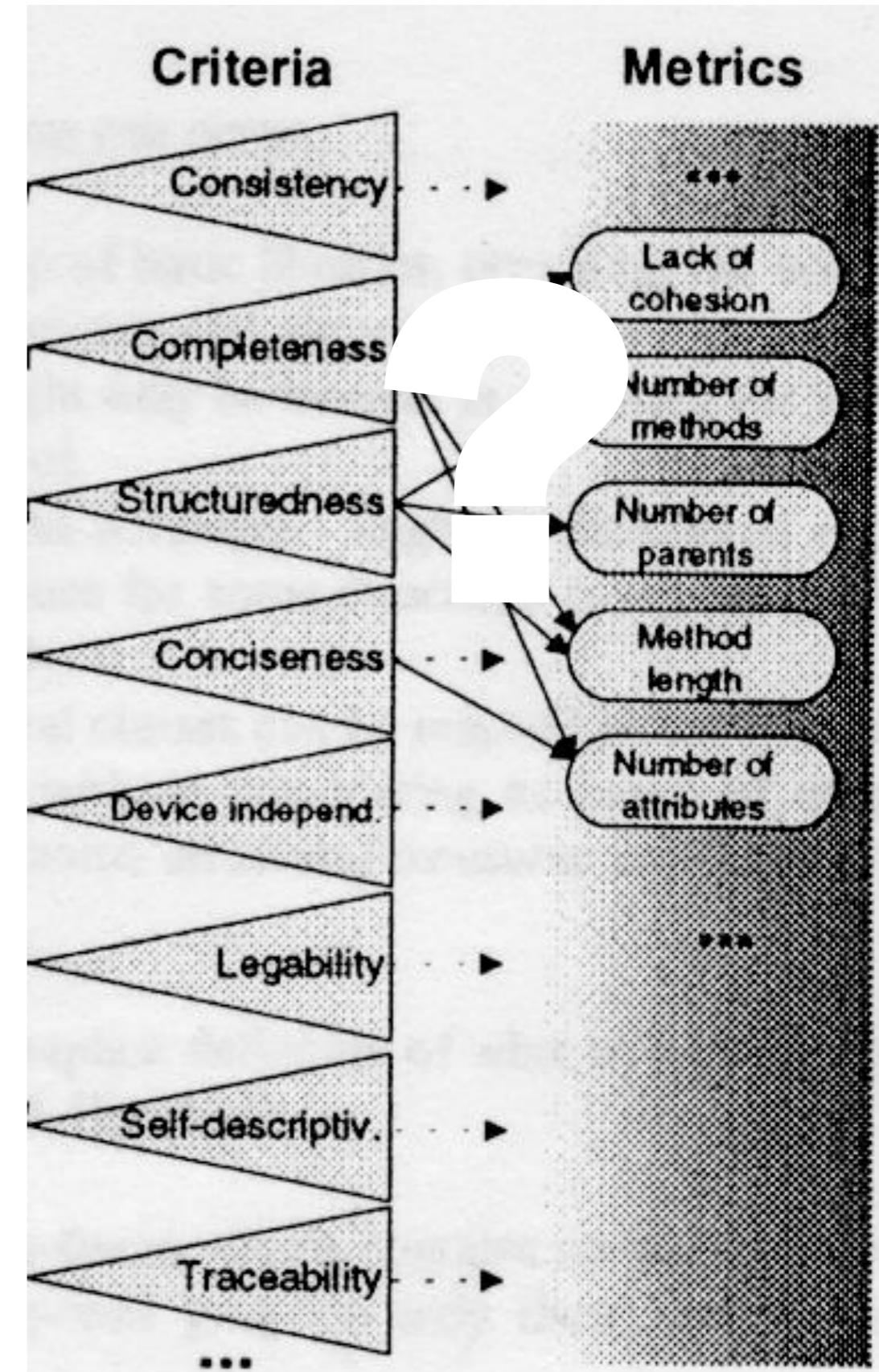
McCall, 1977
Boehm, 1978

Metrics help to assess and improve quality!

Do they?

Problems..

- ▶ Metrics granularity
 - ▶ metrics capture symptoms, not causes of problems
 - ▶ in isolation, metrics do not lead to improvement actions
- ▶ Implicit Mapping
 - ▶ we do not reason in terms of metrics, but in terms of design (principles)



McCall, 1977
Boehm, 1978

2 big obstacles in using metrics:

Thresholds make metrics hard to interpret

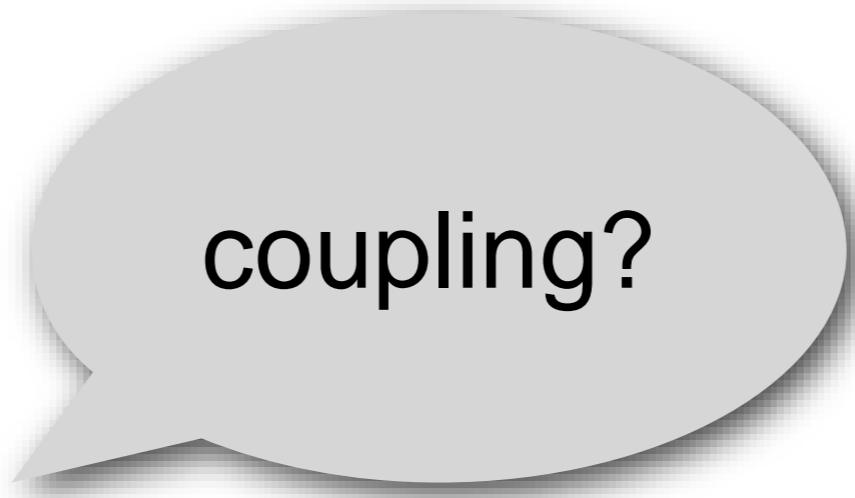
Granularity makes metrics hard to use in isolation

How do I get an
initial understanding of a system?

Metric	Value
LOC	35175
NOM	3618
NOC	384
CYCLO	5579
NOP	19
CALLS	15128
FANOUT	8590
AHH	0,12
ANDC	0,31

Metric	Value
LOC	35175
NOM	3618
NOC	384
CYCLO	5579
NOP	8590
CALLS	0,12
FANOUT	0,31
AvgLOC	

And now what?



coupling?

We need means to compare

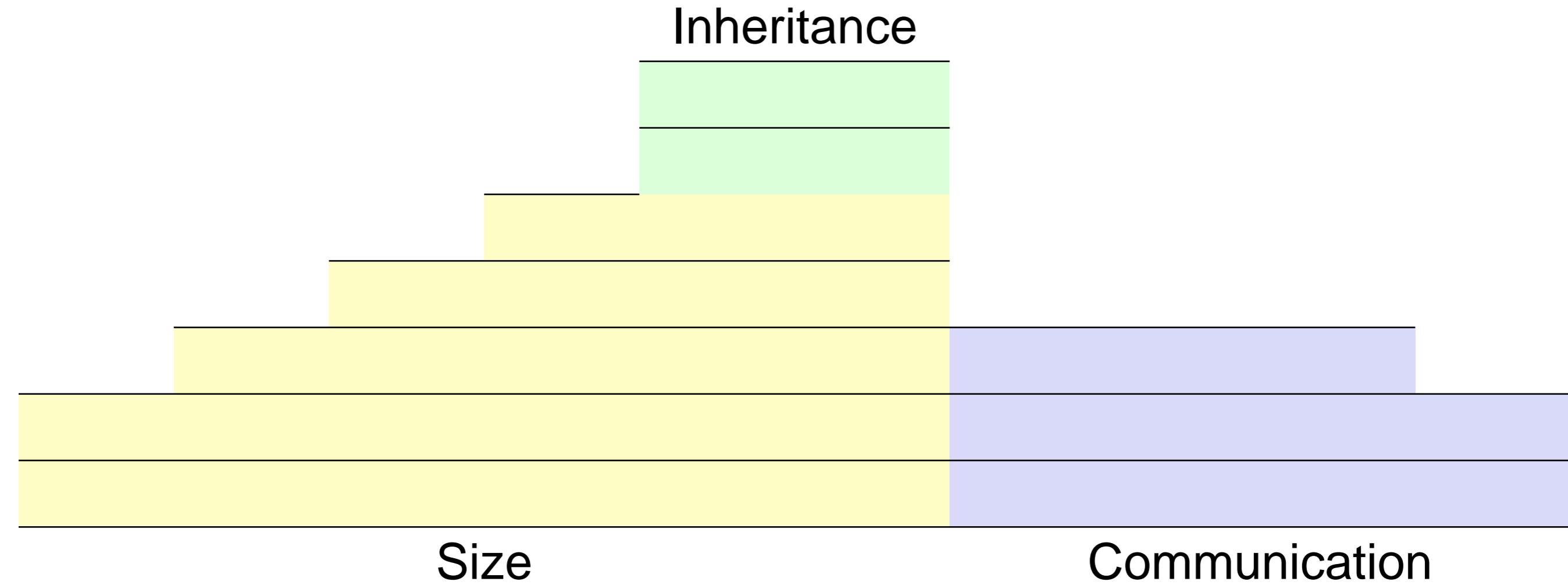


hierarchies?

A photograph of a large pyramid, likely the Great Pyramid of Giza, set against a sky filled with scattered clouds. The pyramid's stone blocks are visible at the base, leading up to its apex. The lighting suggests either sunrise or sunset.

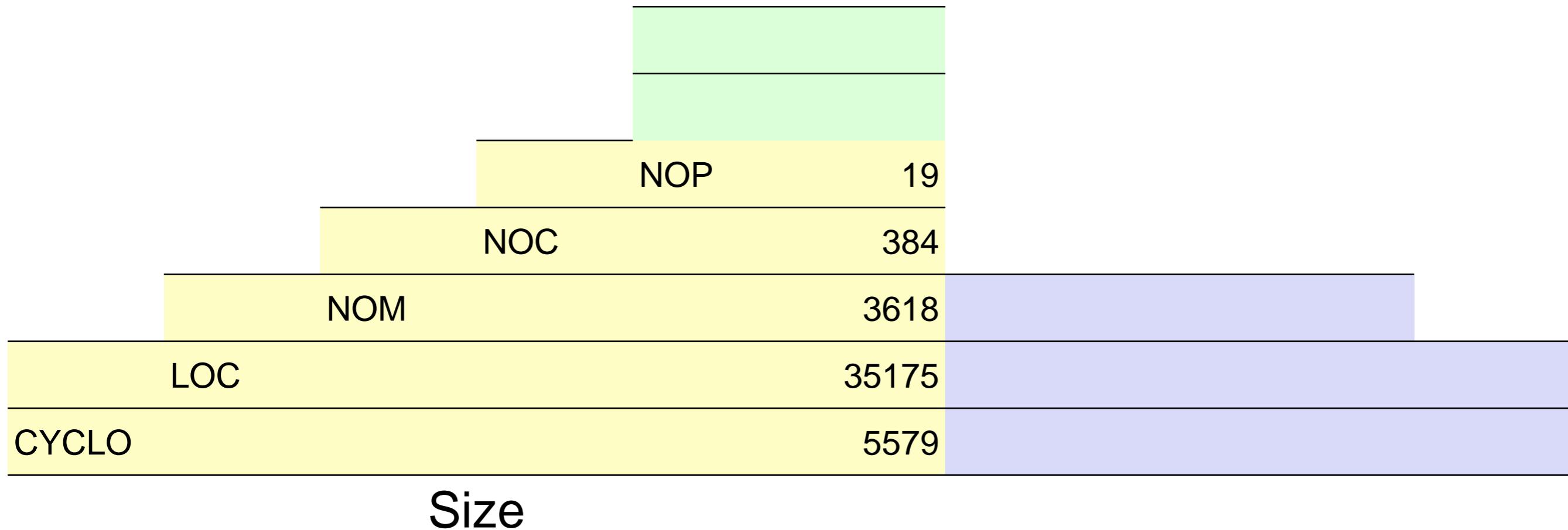
Characterizing Systems with Metrics

The Overview Pyramid provides a metrics overview

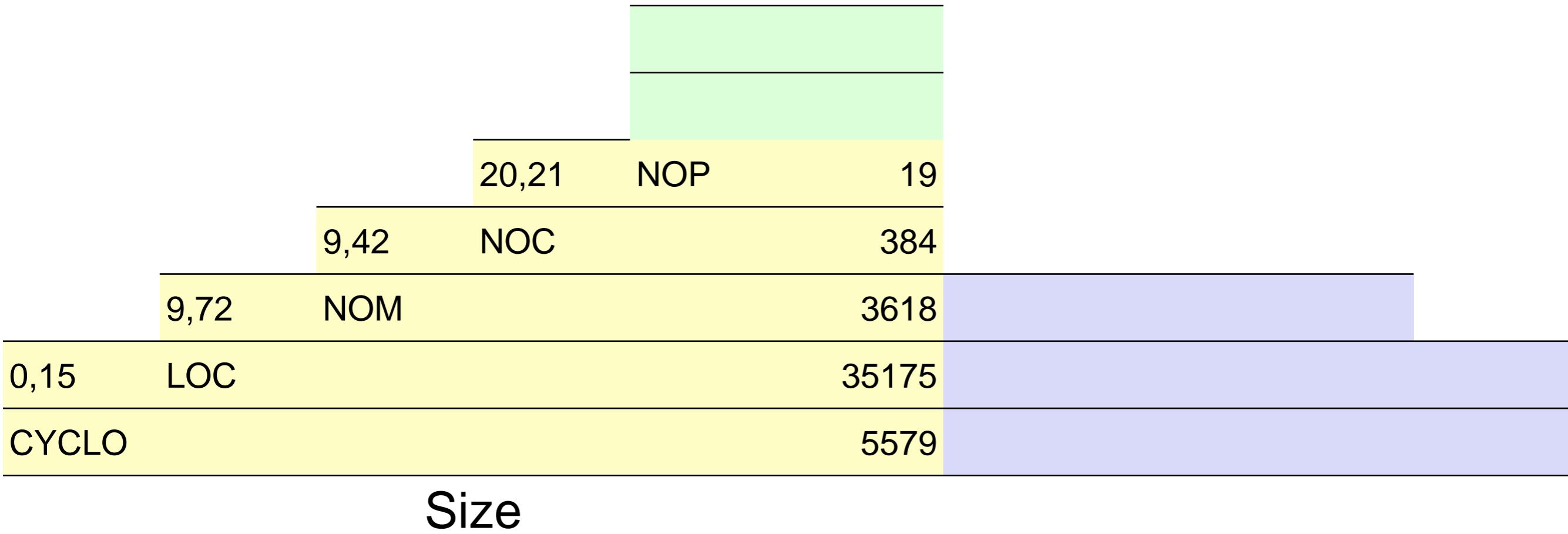


Lanza & Marinescu, 2006

The Overview Pyramid provides a metrics overview

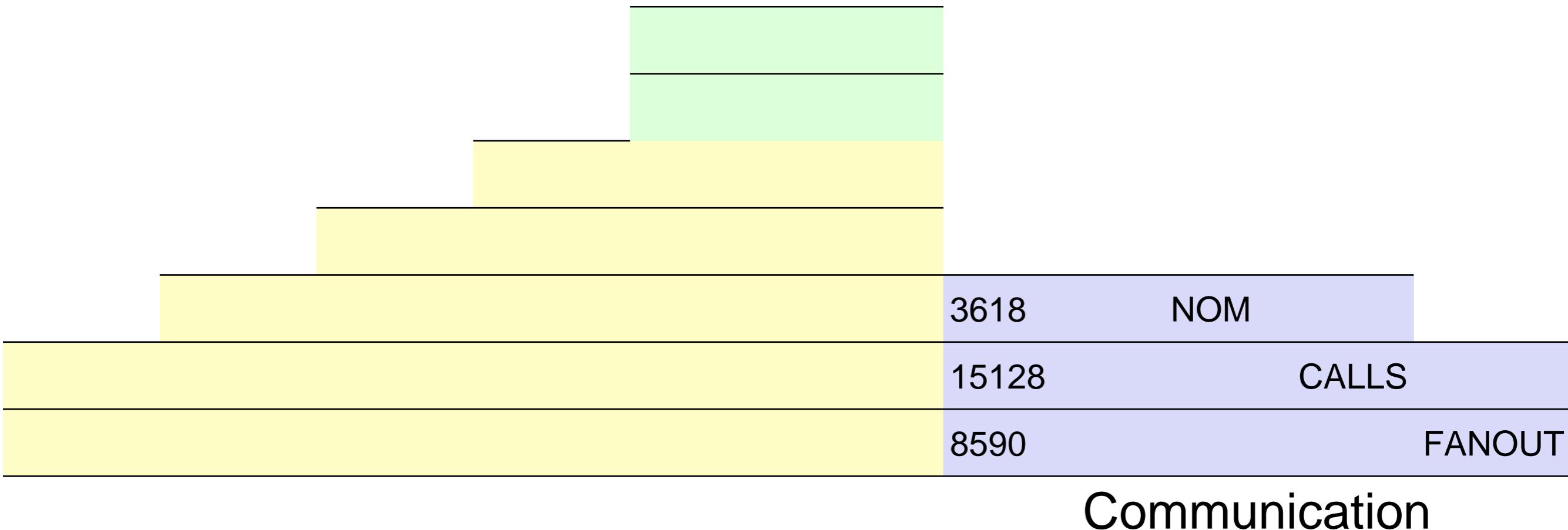


The Overview Pyramid provides a metrics overview

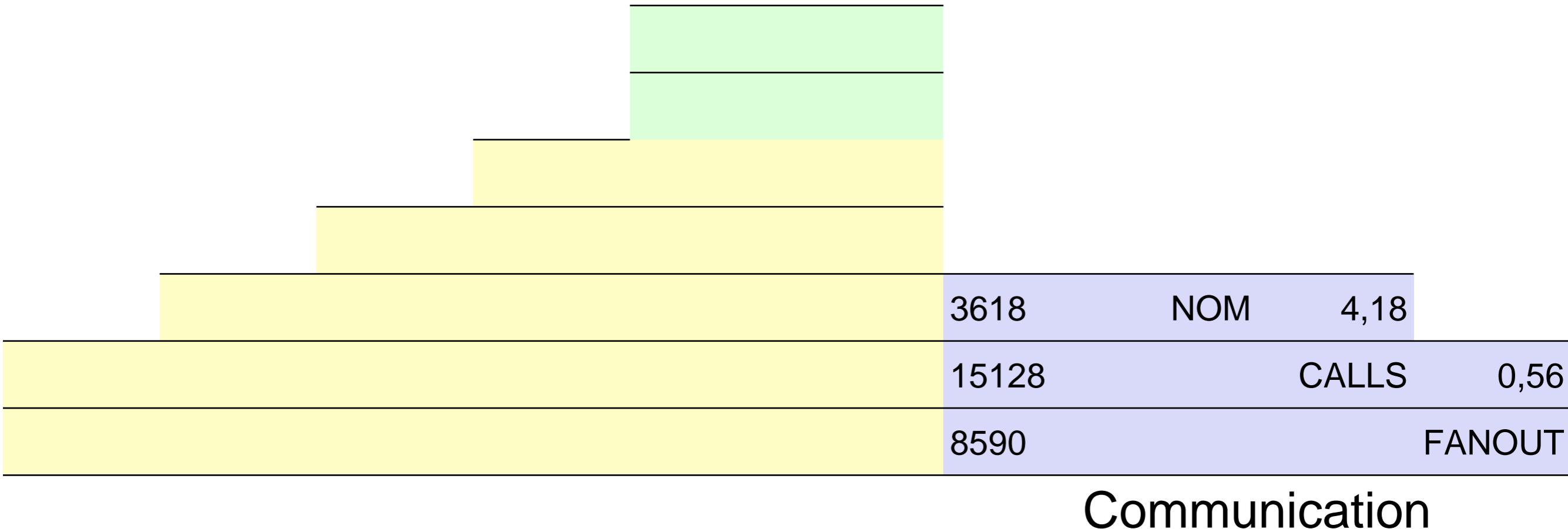


Make relative: CYCLO / LOC = 0,15, etc.

The Overview Pyramid provides a metrics overview

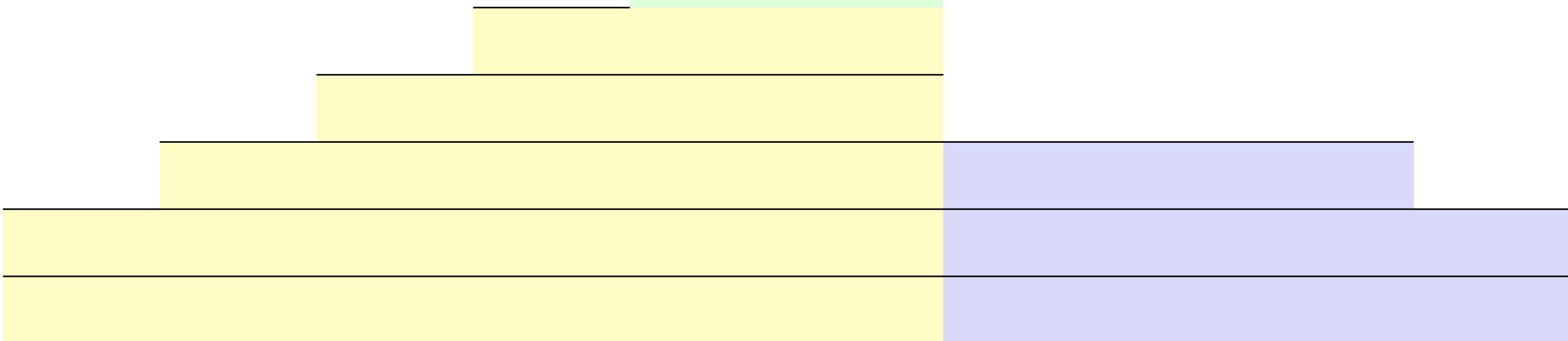


The Overview Pyramid provides a metrics overview

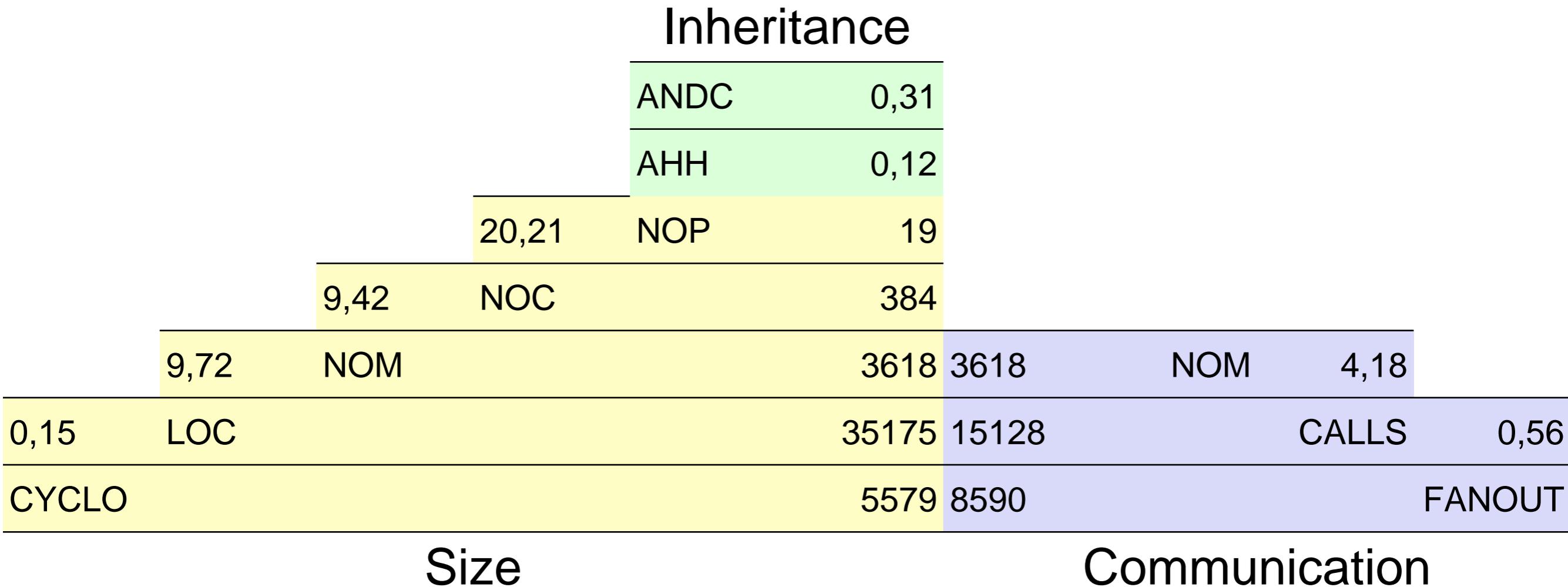


The Overview Pyramid provides a metrics overview

Inheritance	
ANDC	0,31
AHH	0,12

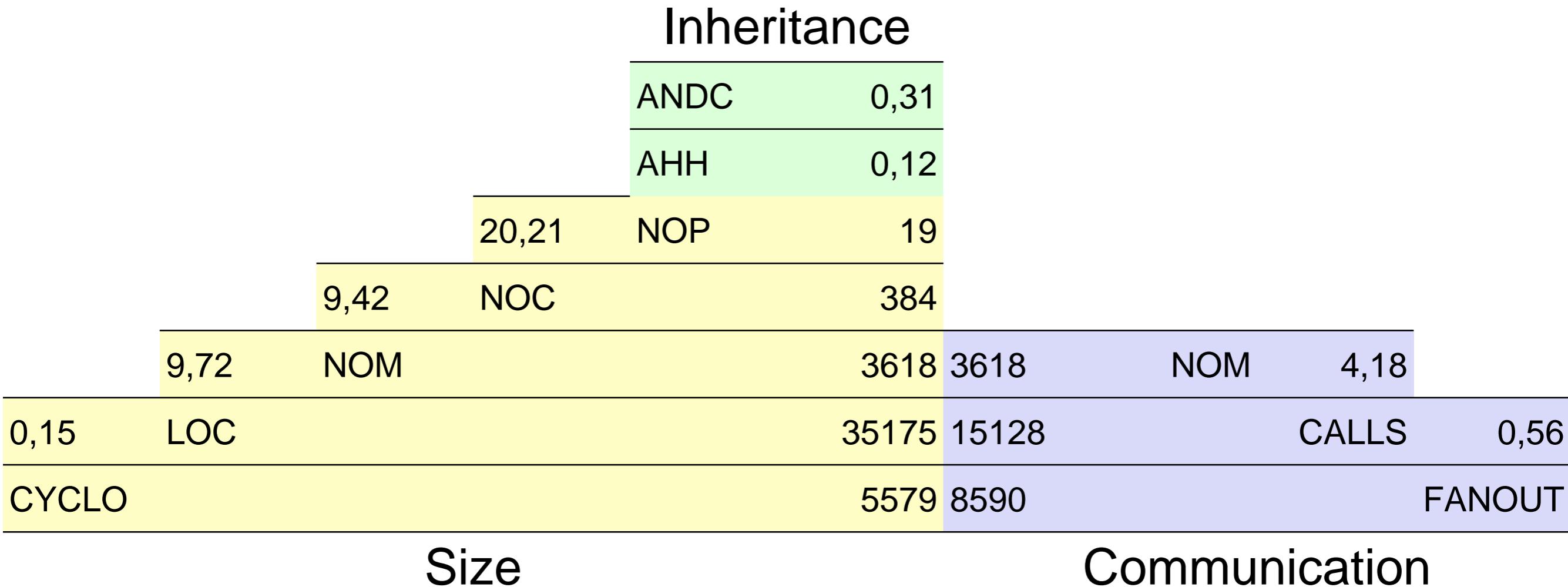


The Overview Pyramid provides a metrics overview

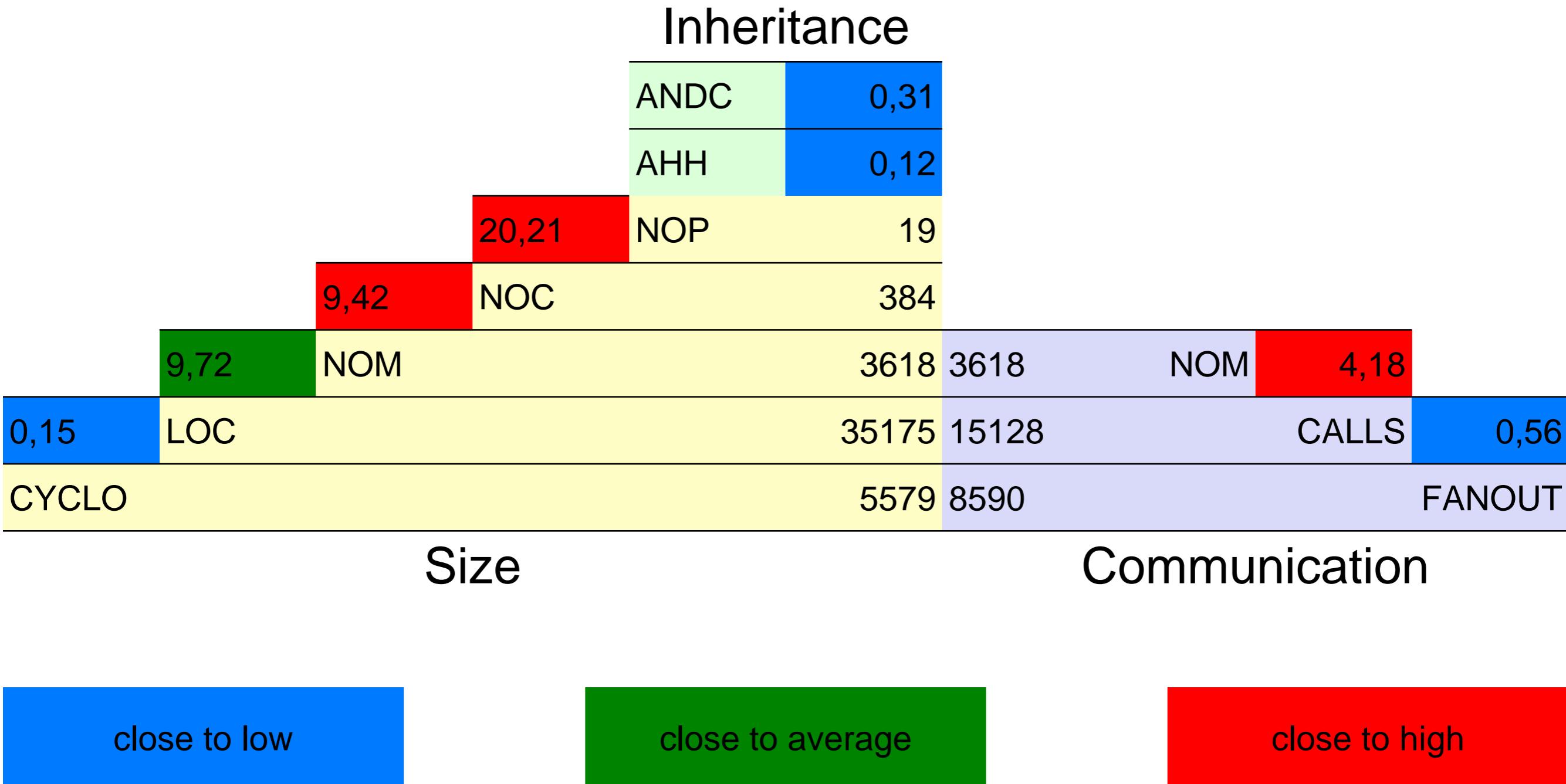


Obtaining Thresholds

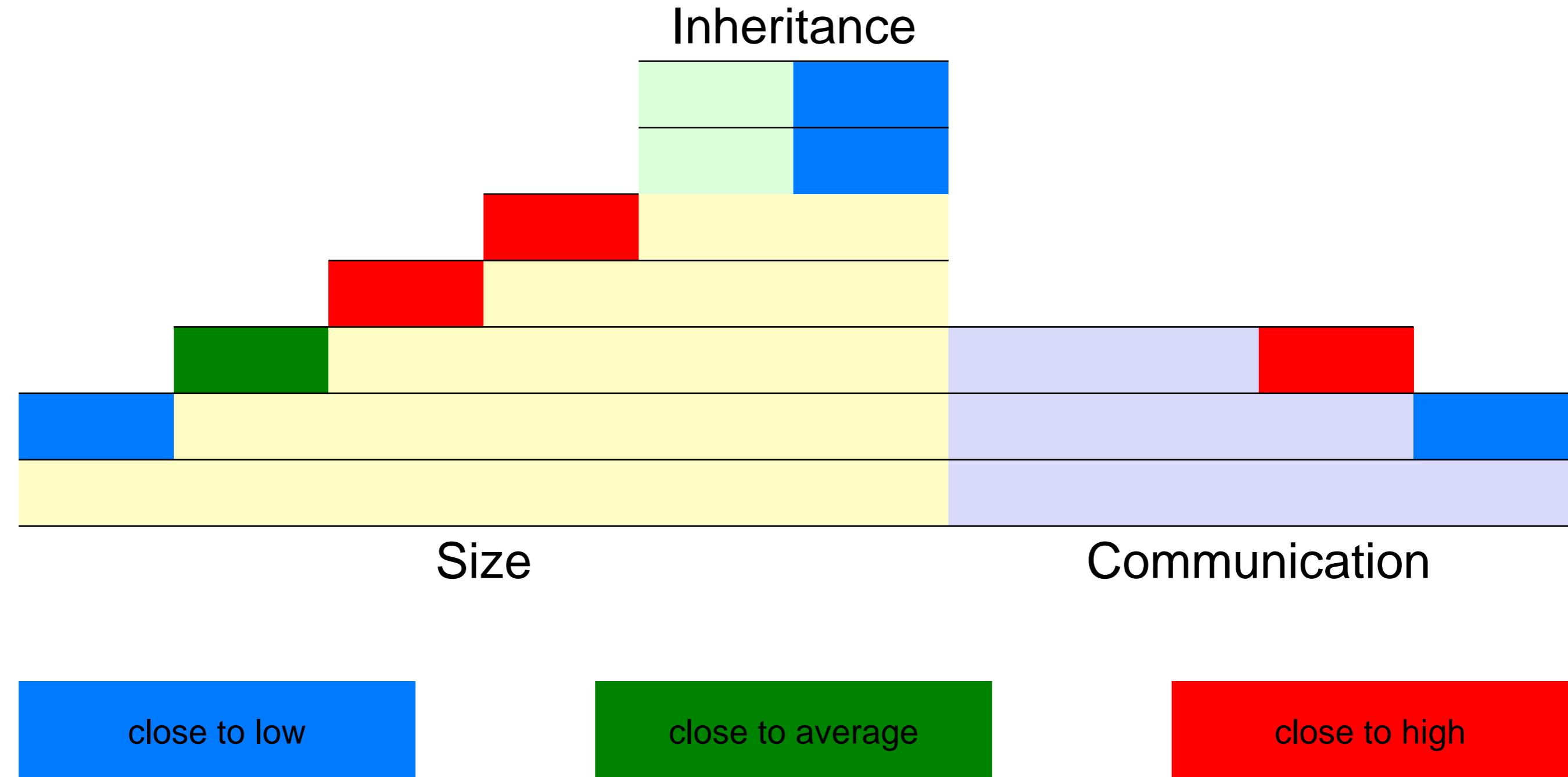
The Overview Pyramid provides a metrics overview



The Overview Pyramid provides a metrics overview



The Overview Pyramid provides a metrics overview



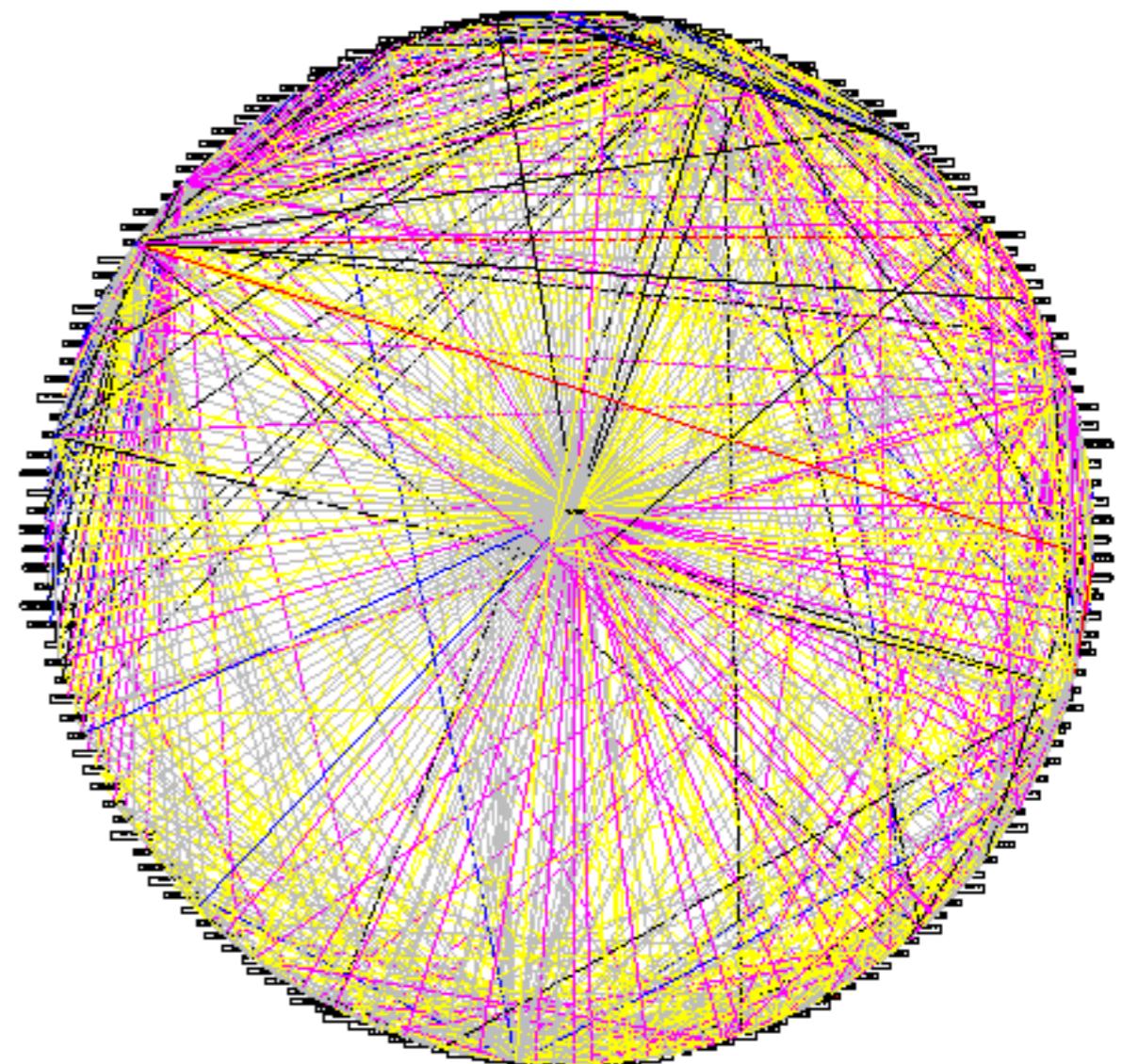
How do I improve my code?

- ▶ Quality is more than zero bugs
- ▶ Quality is about design principles, design heuristics, and best practices
- ▶ Breaking them leads to
 - ▶ Code deterioration
 - ▶ Design problems ~ Maintenance problems

Imagine...

You change a small
design fragment...

...and one third of all
classes require changes!



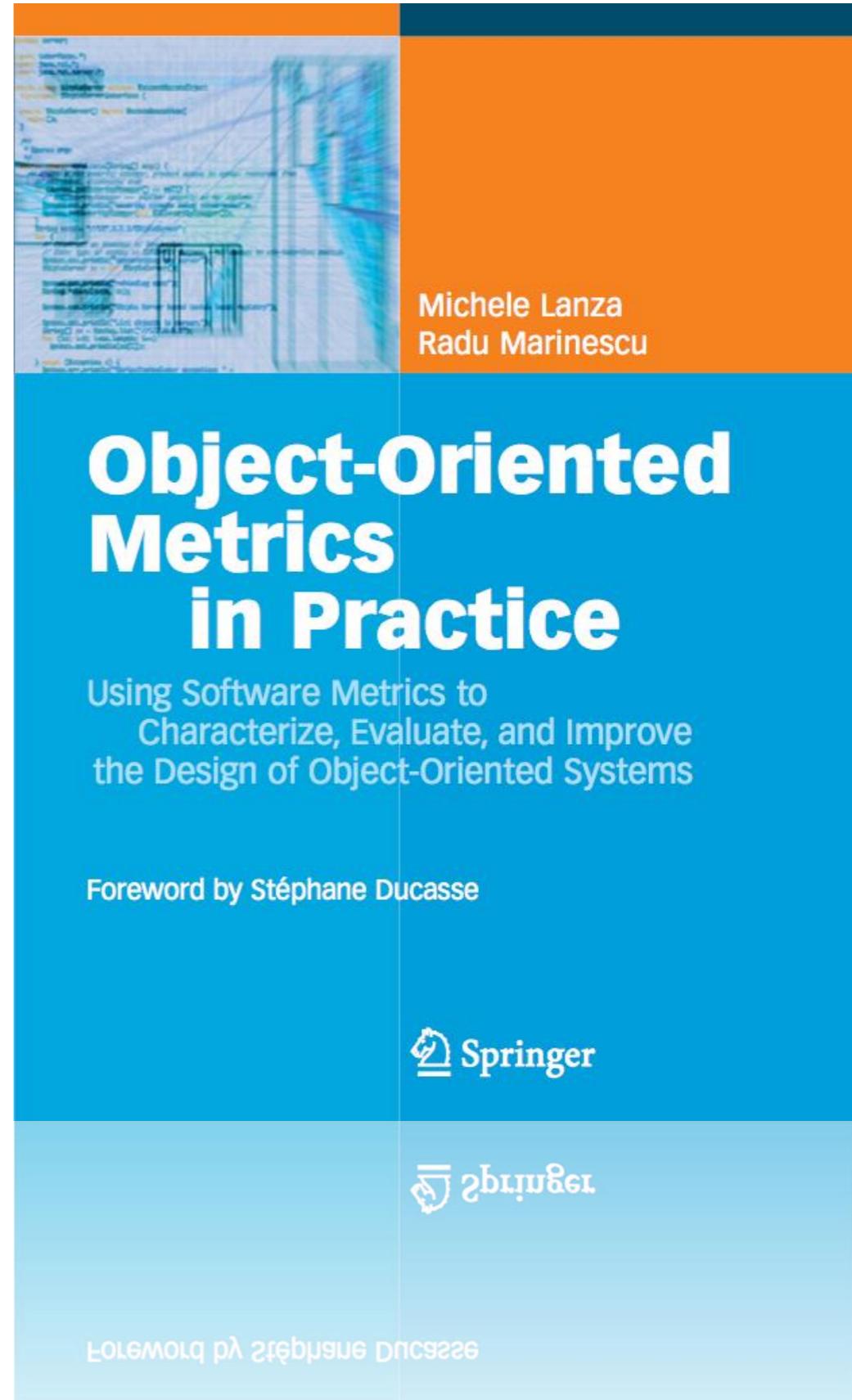
Design Problems

- ▶ Expensive
- ▶ Frequent
- ▶ Unavoidable
- ▶ How can we detect and eliminate them?

Reference

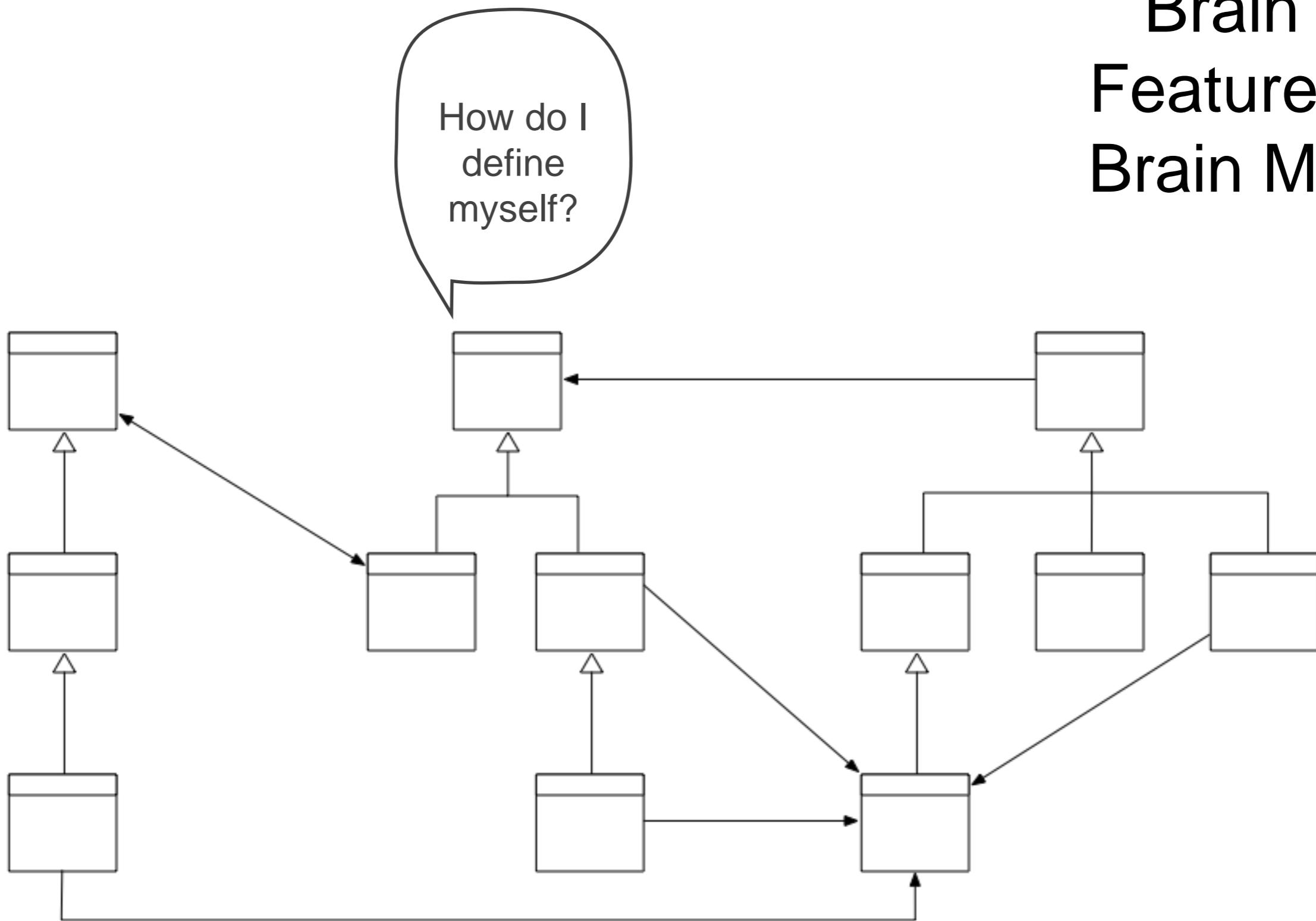
M. Lanza, R. Marinescu
“Object-Oriented Metrics in Practice”

Springer, 2006
ISBN 3-540-24429-8



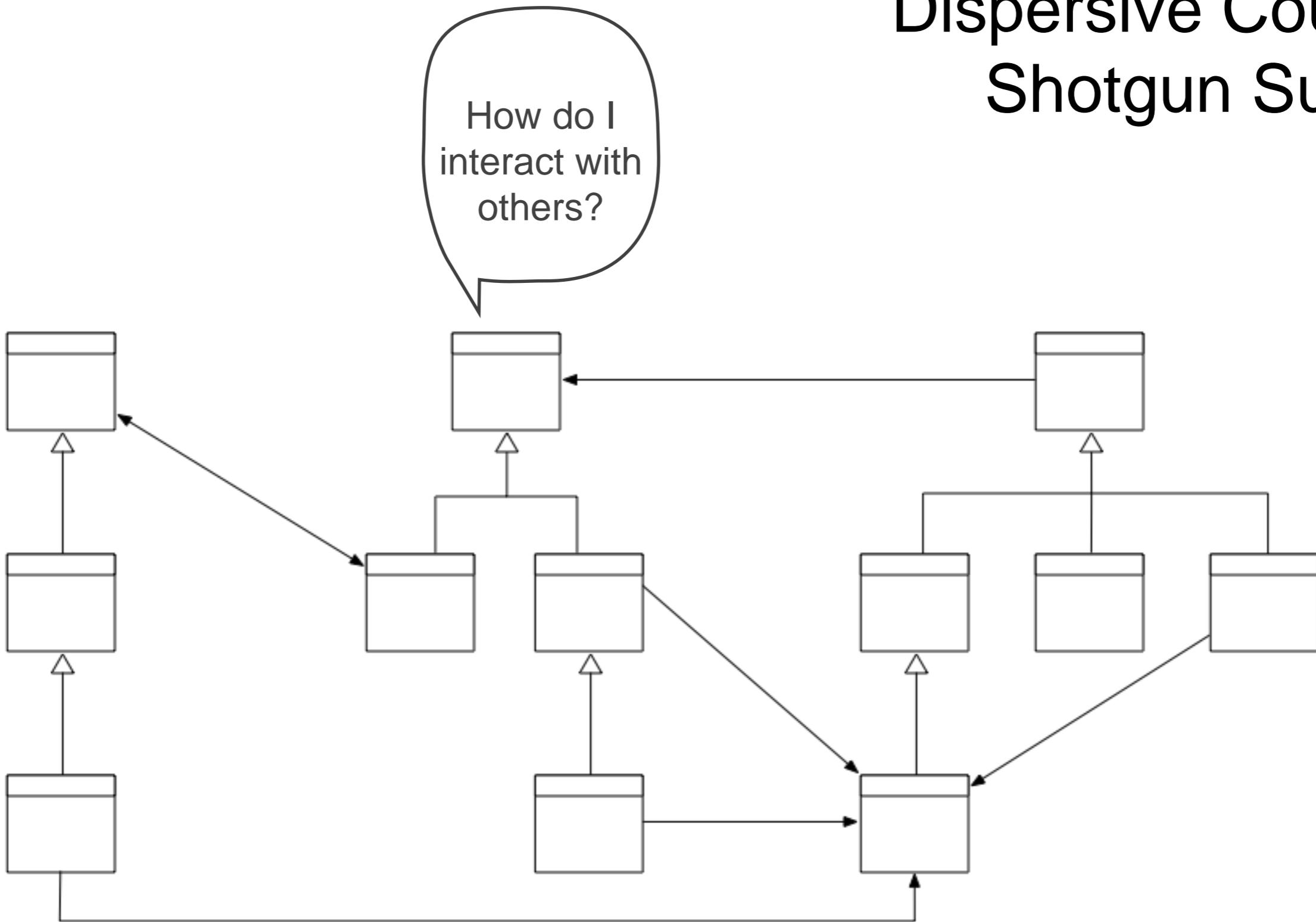
Identity Disharmony

God Class
Data Class
Brain Class
Feature Envy
Brain Method



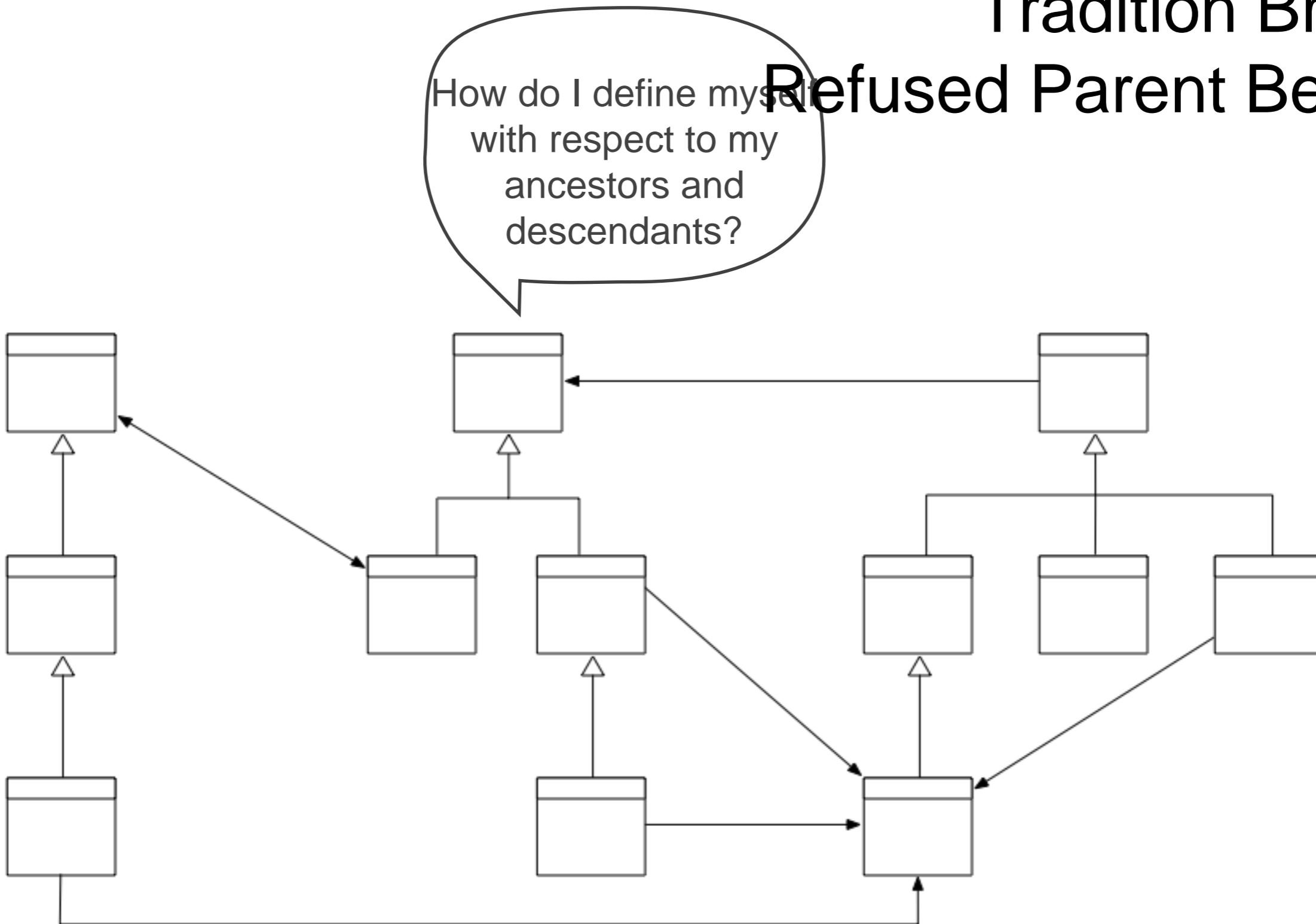
Collaboration Disharmony

Intensive Coupling
Dispersive Coupling
Shotgun Surgery



Classification Disharmony

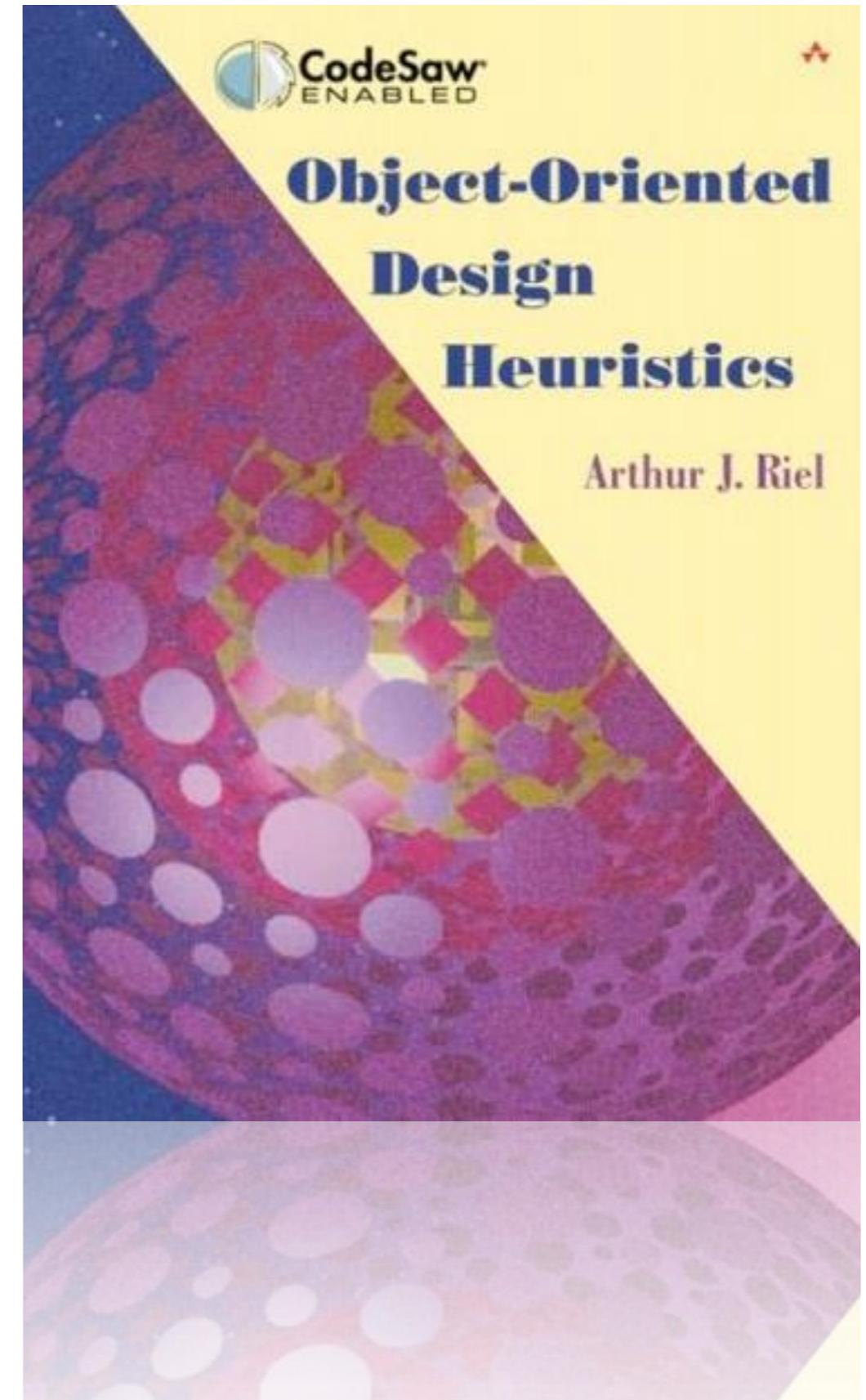
Futile Hierarchy
Tradition Breaker
Refused Parent Bequest



God Class

“In a good object-oriented design
the intelligence of a system is
uniformly distributed among the top-
level classes.”

Arthur Riel, 1996

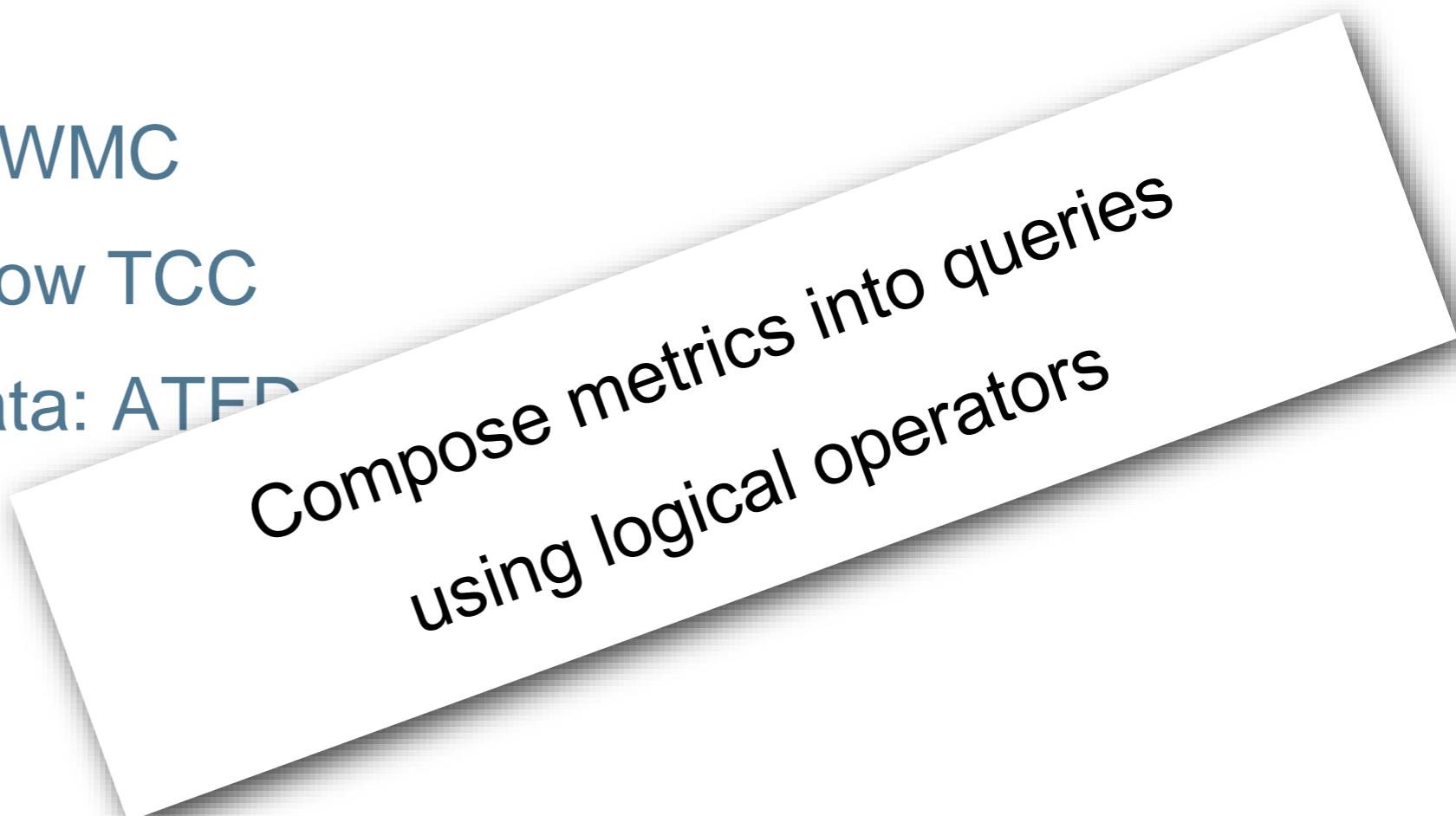


God Classes

- ▶ God Classes tend to centralize the intelligence of the system, to do everything and to use data from small data-classes
- ▶ God Classes tend
 - ▶ to centralize the intelligence of the system
 - ▶ to do everything and
 - ▶ to use data from small data-classes
- ▶ God Classes
 - ▶ centralize the intelligence of the system
 - ▶ do everything
 - ▶ use data from small data-classes

God Classes

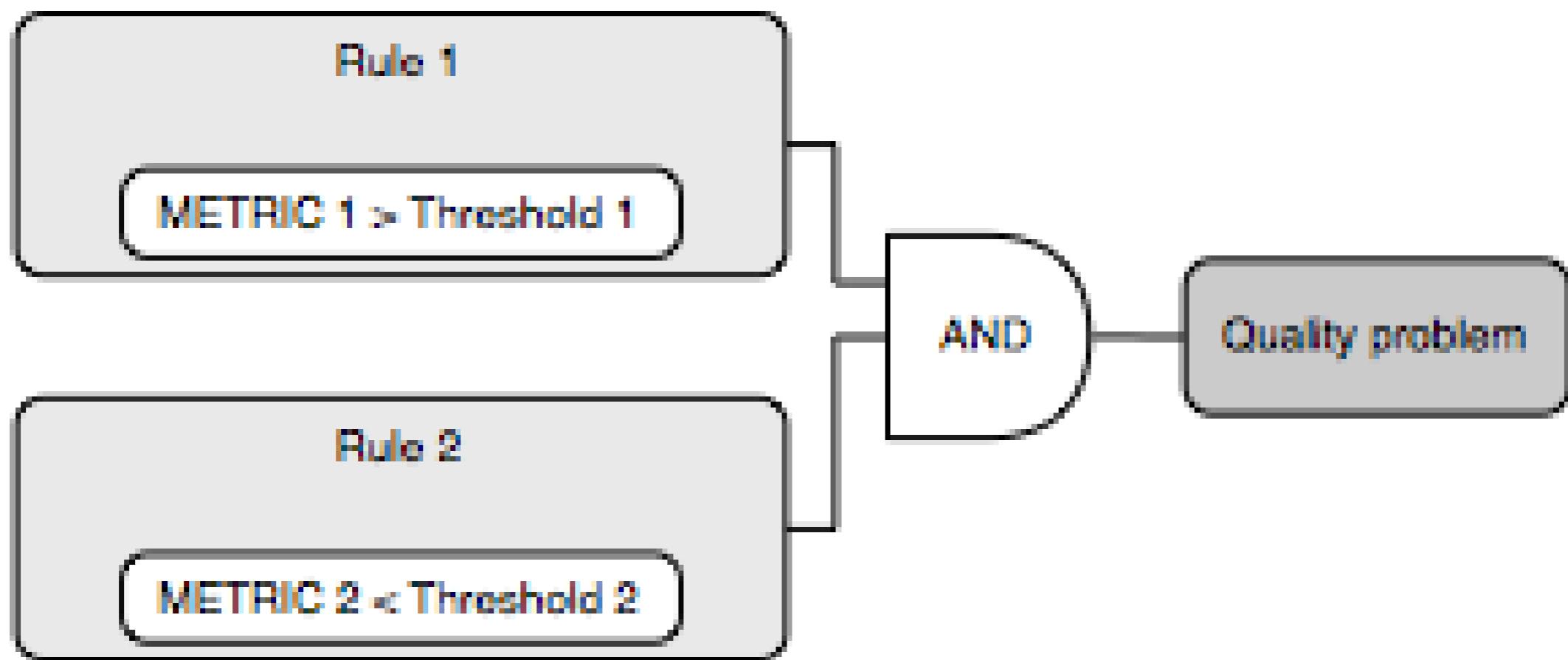
- ▶ God Classes
 - ▶ centralize the intelligence of the system
 - ▶ do everything
 - ▶ use data from small data-classes
- ▶ God Classes
 - ▶ are complex: high WMC
 - ▶ are not cohesive: low TCC
 - ▶ access external data: ATFD



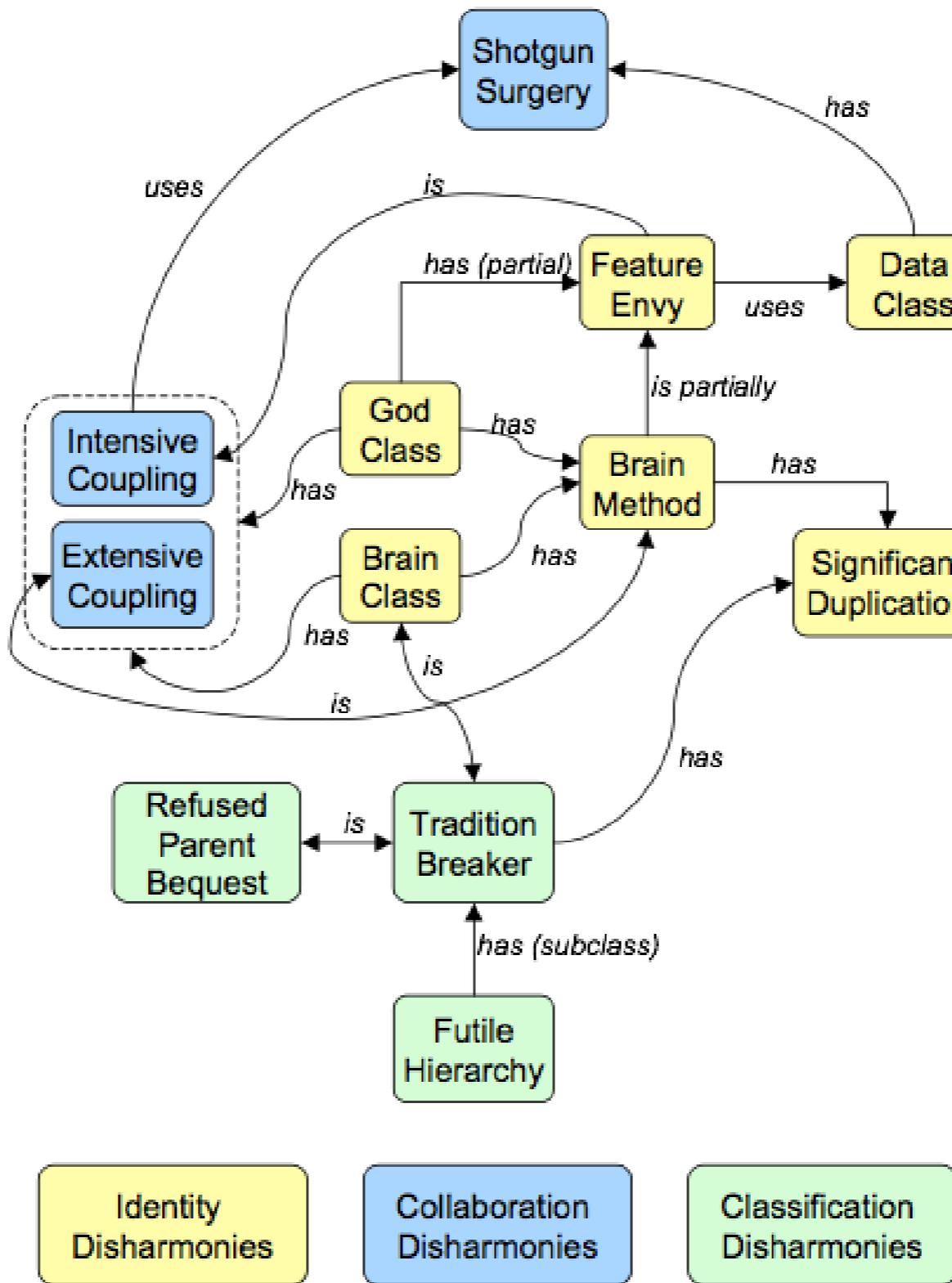
Compose metrics into queries
using logical operators

Detection Strategies

- ▶ Detection strategies are metric-based queries to detect design flaws



Design Flaws do not come alone



Characteristics of a God Class

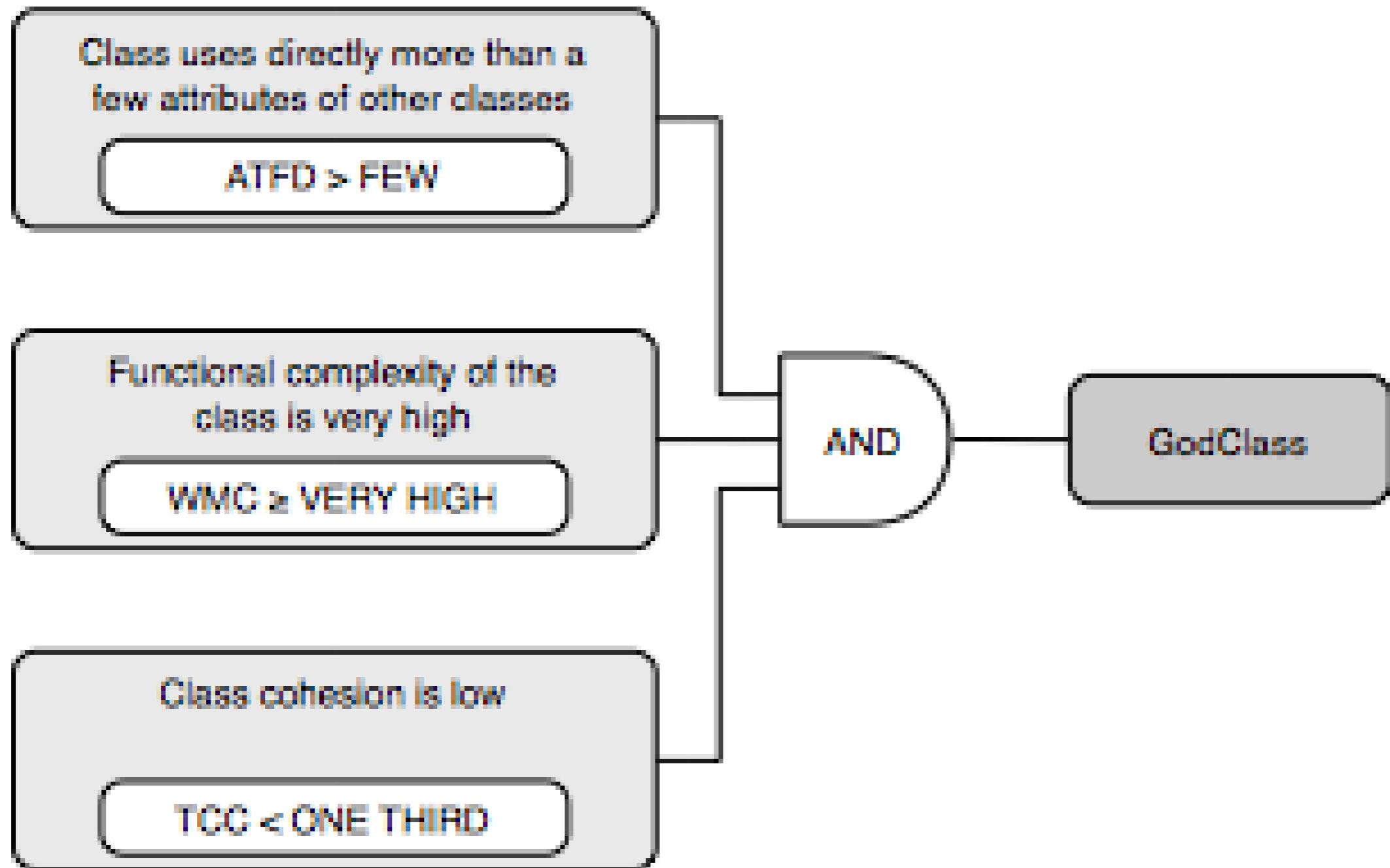
Heavily accesses data of other “lightweight” classes, either directly or using accessor methods.

Is large

God
Class

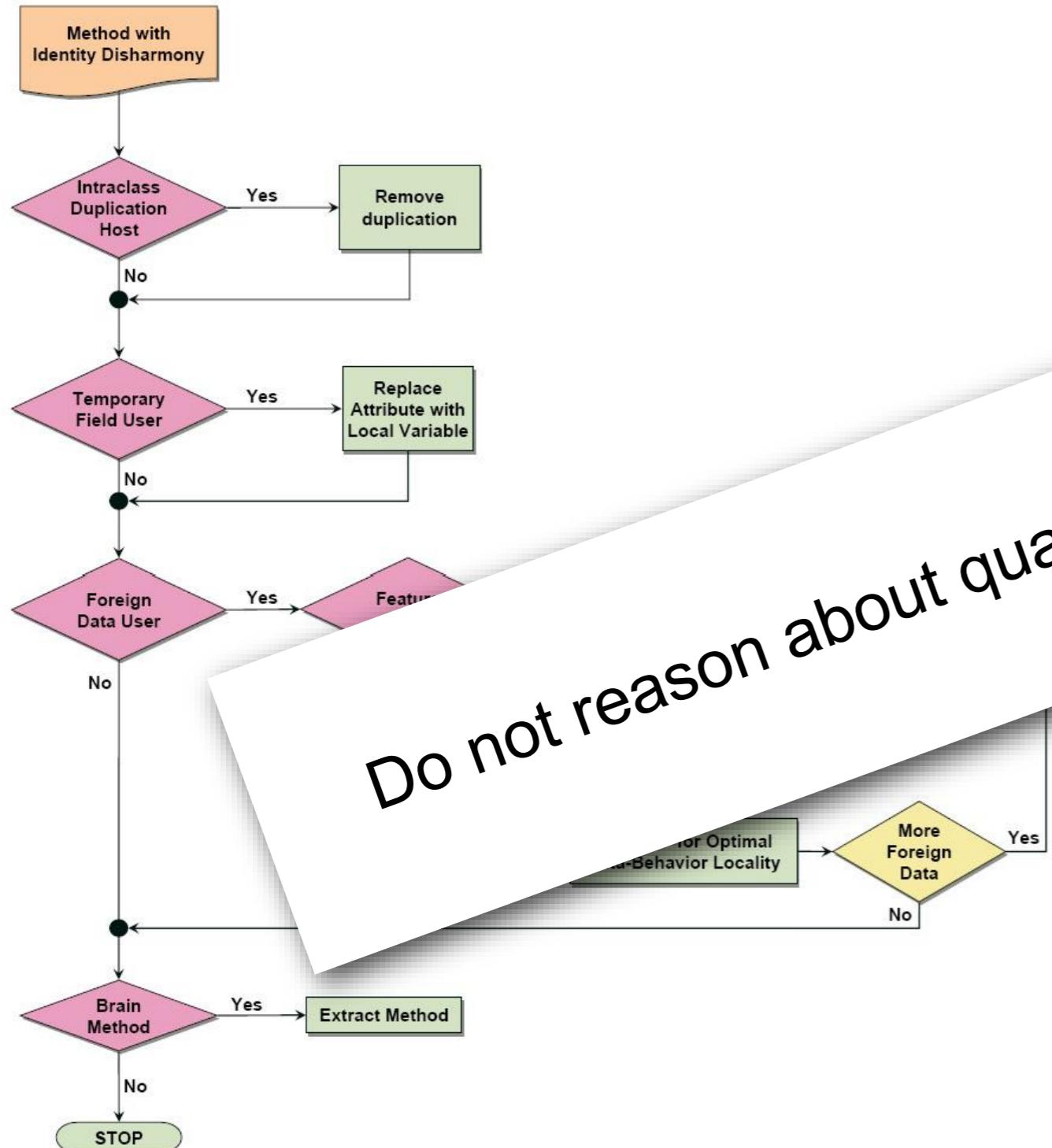
Has a lot of non-communicative behavior

God Class Detection Strategy

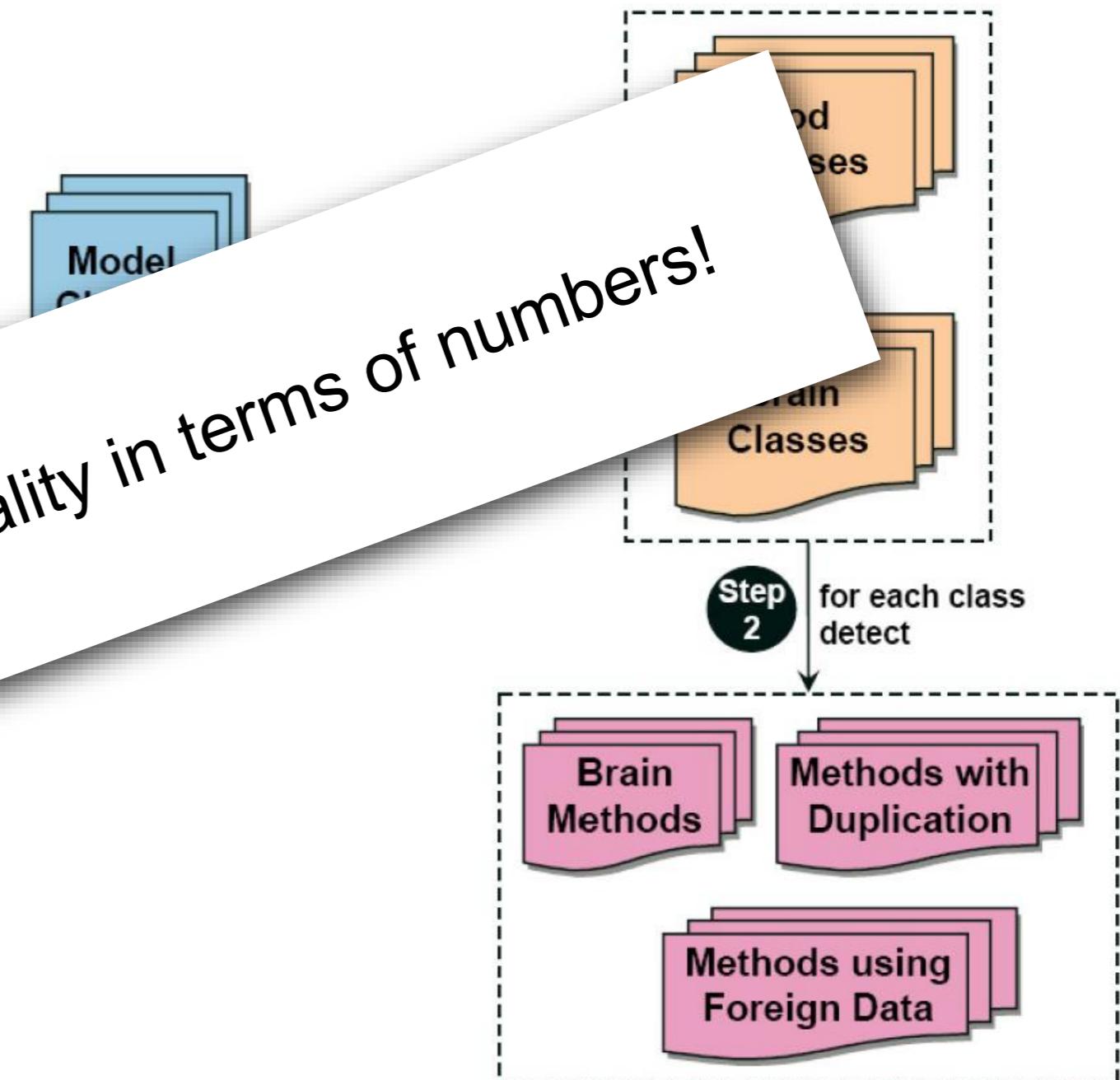


And Now?

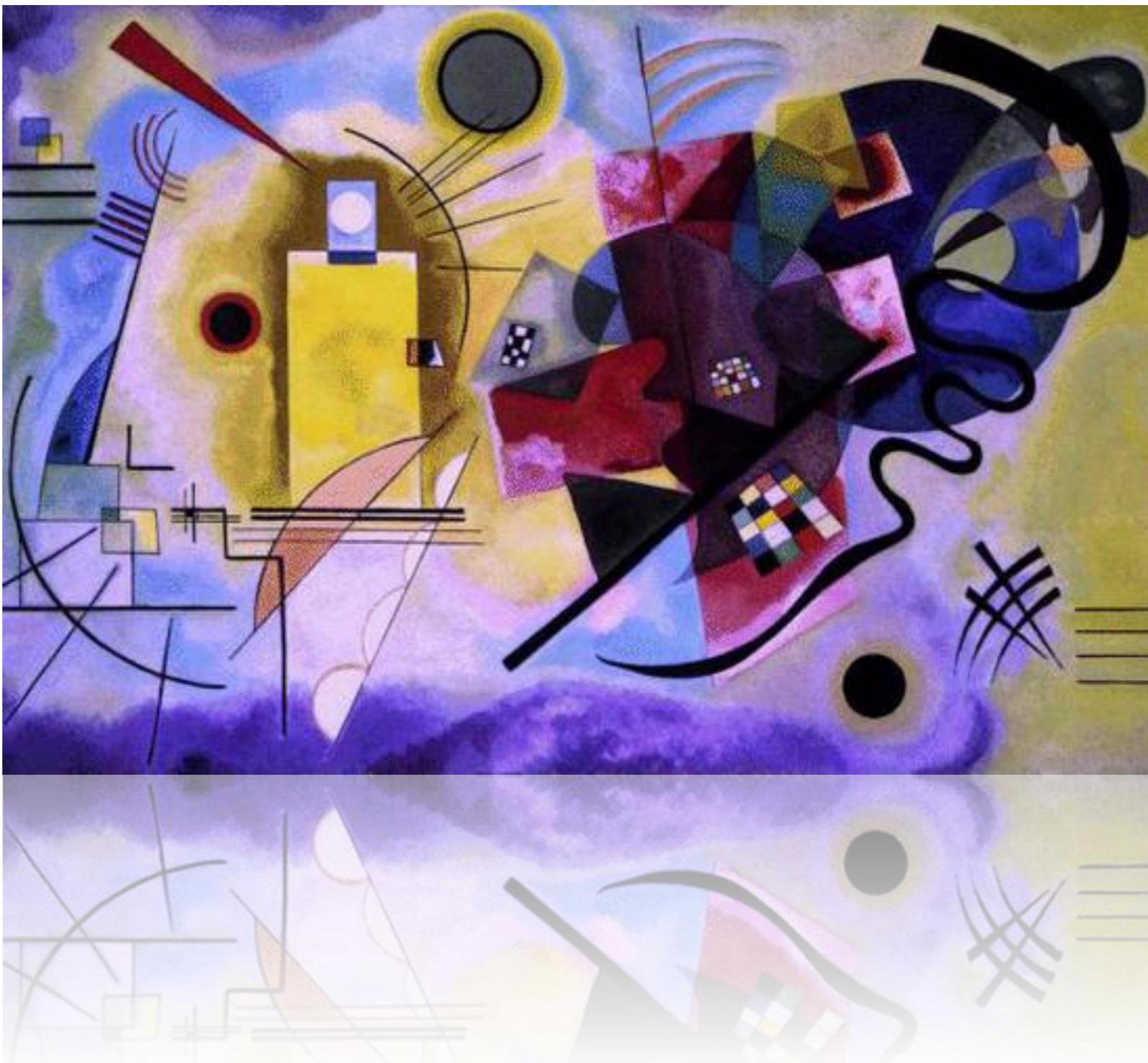
Follow A Clear and Repeatable Process



Do not reason about quality in terms of numbers!



Metrics are only half the truth



Can we understand the beauty of a painting
by measuring its frame and counting its
colors?

Lecture 05

Software Visualization

Source Code = Text

```
<?php wp_head(); ?>

```

Container>

```
<div id="header">
<div class="style_content">
<form action="php bloginfo('home'); ?&gt;/" name="search_box" id="search_box" method="get"&gt;
&lt;label for="input_search" id="label_search"<?php _e('Find this', 'Gluedideas_subtle'); ?></label> <input type="text" id="input_search" class="input" type="text" />
<input type="submit" value="Search" id="button_search" style="background-image: url('image/x-icon'); border: none; width: 100px; height: 30px; margin-left: 10px;"/>
</form>

```

Age" src="php bloginfo('stylesheet_directory'); ?>/assets/images/icon_search.gif" align="top" id="button_search" style="background-image: url('image/x-icon'); border: none; width: 100px; height: 30px; margin-left: 10px;"/></p

```
<h1 id="title"><a href="php echo get_settings('home'); ?&gt;"/&gt;&lt;span&gt;&lt;?php bloginfo('name'); ?&gt;&lt;/span&gt;&lt;/a&gt;&lt;/h1&gt;
&lt;p id="tagline"&gt;&lt;span&gt;&lt;?php bloginfo('description'); ?&gt;&lt;/span&gt;&lt;/p&gt;
</pre

<ul id="menu">



```
<li class="menu-item"><?php echo($sMenuHome); ?>>"/><?php _e('Home', 'Gluedideas_subtle'); ?>
 <?php wp_list_pages('depth=1&title_li=&sort_column=menu_order'); ?>
</pre

</div>


```
if ($iLeadIndex == $aOptions['lead_count'] + 1) {
    echo ('<h2>' . __('Previous Articles') . '</h2>');
}
```


$bShowContent = false;


```
if ($iLeadIndex == $aOptions['lead_count'] + 1) {
    echo ('<h2>' . __('Previous Articles') . '</h2>');
}
```


<div id="post_<?php the_ID(); ?>" class="post<?php echo($sPostClass); ?>">
 <h3 class="title">"<?php the_title(); ?></h3>
 <ul class="metalinks">
 <li class="icon author"><?php _e("Posted by", 'Gluedideas_subtle'); ?> <?php the_author_posts_link(); ?>
 <li class="icon date"><?php the_time(get_option('date_format')) ?>
 <?php if ($aOptions['show_metalinks']) : ?>
 <ul class="metalinks">
 <li class="icon comment">#comments"><?php comments_number(__('No Responses', 'Gluedideas_subtle'), __('One Response', 'Gluedideas_subtle'), __('Responses', 'Gluedideas_subtle')) ?>
 <li class="icon delicious"><a href="http://del.icio.us/post?url=<?php the_permalink() ?>&title=<?php echo urlencode(get_the_title()); ?>">Delicious
 <li class="icon digg">Digg
 <li class="icon technorati"><a href="http://technorati.com/cosmos/search.html?url=<?php the_permalink() ?>">Technorati

 <?php endif; ?>
 <br class="clear" />
 <?php if ($bShowContent) : ?>
 <div class="content">
 <?php the_content(); ?>
 <ul class="links">
 <li class="icon jump"><?php if (strpos(get_the_content('^\n\n'), '^\n\n') > 0) : ?><a href="<?php the_permalink() ?>more...">More...
 <?php if ($gluedideas_subtle) : ?><?php if (strpos(get_the_content('^\n\n'), '^\n\n') > 0) (echo(' or ')); ?><?php _e('Jump to the permalinks'); ?>
 <?php if (strpos(get_the_content('^\n\n'), '^\n\n') > 0) (_e(' or ')); ?><?php _e('Jump to the permalinks'); ?>

 <?php endif; ?>
</pre


```


```

Programming = Writing



```

/*****************************************/
/* micro-Max, */
/* A chess program smaller than 2KB (of non-blank source), by H.G. Muller */
/*****************************************/

/* version 3.2 (2000 characters) features:
   - recursive negamax search
   - quiescence search with recaptures
   - recapture extensions
   - (internal) iterative deepening
   - best-move-first 'sorting'
   - a hash table storing score and best move
   - full FIDE rules (expt minor promotion) and move-legality checking */

#define F(I,S,N) for(I=S;I<N;I++)
#define W(A) while(A)
#define K(A,B) *(int*)(T+A+(B&8)+S*(B&7))
#define J(A) K(y+A,b[y])-K(x+A,u)-K(H+A,t)

#define U 16777224
struct _ {int K,V;char X,Y,D;} A[U]; /* hash table, 16M+8 entries*/

int V=112,M=136,S=128,I=8e4,C=799,Q,N,j; /* V=0x70=rank mask, M=0x88 */

char O,K,L,
w[]={0,1,1,3,-1,3,5,9}, /* relative piece values */
o[]={-16,-15,-17,0,1,16,0,1,16,15,17,0,14,18,31,33,0, /* step-vector lists */
    7,-1,11,6,8,3,6, /* 1st dir. in o[] per piece*/
    6,3,5,7,4,5,3,6}, /* initial piece setup */
b[129], /* board: half of 16x8+dummy*/
T[1035], /* hash translation table */

n[]= ".?+nkbrq??"NKBRQ"; /* piece symbols on printout*/

D(k,q,l,e,J,Z,E,z,n) /* recursive minimax search, k=moving side, n=depth*/
int k,q,l,e,J,Z,E,z,n; /* (q,l)=window, e=current eval. score, E=e.p. sqr.*/
{
    /* e=score, z=prev.dest; J,Z=hashkeys; return score*/
    int j,r,m,v,d,h,i=9,F,G;
    char t,p,u,x,y,X,Y,H,B;
    struct _ *a=A;
    /* lookup pos. in hash table*/
    j=(k*E^J)&U-9; /* try 8 consec. locations */
    W((h=A[++j].K)&&h-Z&&-i); /* first empty or match */
    a+=i;j:0; /* dummy A[0] if miss & full*/
    if(a->K) /* hit: pos. is in hash tab */
    {d=a->D;v=a->V;X=a->X;
     if(d>=n) /* if depth sufficient: */
     {if(v>=l|X&S&&v<=q|X&8) return v; /* use if window compatible */
      d=n-1; /* or use as iter. start */
      }X&=~M;Y=a->Y; /* with best-move hint */
      Y=d?Y:0; /* don't try best at d=0 */
    }else d=X=Y=0;
    N++; /* node count (for timing) */
    W(d++<n|z==8&N<1e7&d<98) /* iterative deepening loop */
    {x=B=X;
     Y|=8&Y>>4; /* request try noncastl. 1st*/
     m=d>1?-l:e;
     do{u=b[x];
     if(u&k)
     {r=p=u&7;
      j=o[p+16];
      W(r=p>2&r<0?-r:o[++j]) /* loop over directions o[] */
      {A: /* resume normal after best */
       y=x;F=G=S; /* (x,y)=move, (F,G)=castl.R*/
       do{H=y+=r; /* y traverses ray */
       }
     }
     }
     }
     }
     }

if(i<0||E-S&&b[E]&&y-E<2&E-y<2)m=l; /* K capt. or bad castling */
if(m>=l)goto C; /* abort on fail high */

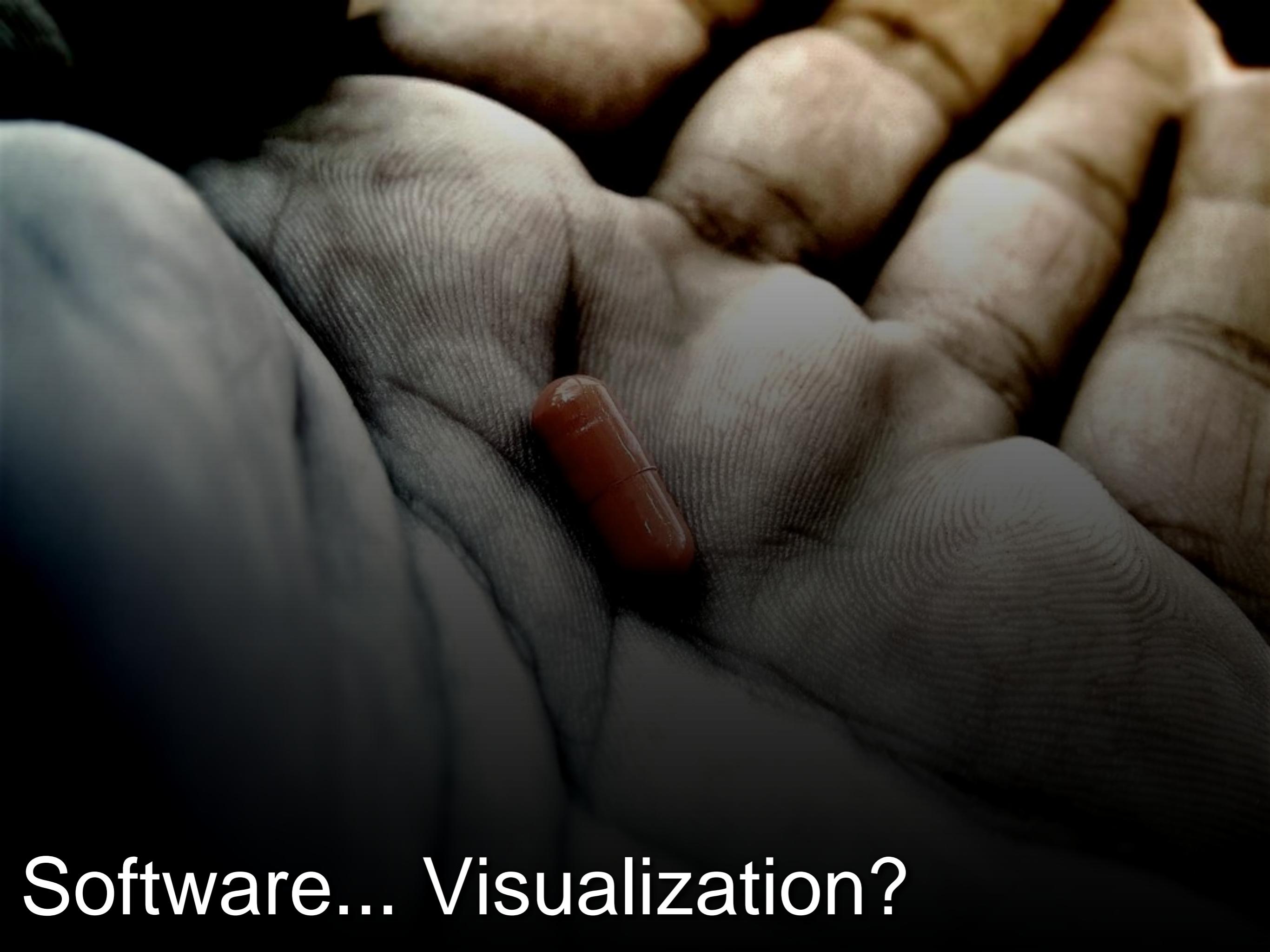
if(h=d-(y!=z)) /* remaining depth(-recapt.)*/
{v=p<6?b[x+8]-b[y+8]:0; /* center positional pts. */
 b[G]=b[H]=b[x]=0;b[y]=u&31; /* do move, strip virgin-bit*/
 if(!(G&M)){b[F]=k+6;v+=30;} /* castling: put R & score */
 if(p<3) /* pawns: */
 {v=9*((x-2)&M||b[x-2]!=u)+ /* structure, undefended */
   ((x+2)&M||b[x+2]!=u)-1; /* squares plus bias */
  if(y+r+1&S){b[y]|=7;i+=C;} /* promote p to Q, add score*/
 }
 v=-D(24-k,-l-(l>e),m>q?-q,-e-v-i, /* recursive eval. of reply */
   J+J(0),Z+J(8)+G-S,F,y,h); /* J,Z: hash keys */
 v-=v>e; /* delayed-gain penalty */
 if(z==9) /* called as move-legality */
 {if(v!=l&x==K&y==L) /* checker: if move found */
  {Q=-e-i;O=F;return l;} /* & not in check, signal */
  v=m; /* (prevent fail-lows on */
 }
 /* K-capt. replies */
 b[G]=k+38;b[F]=b[y]=0;b[x]=u;b[H]=t; /* undo move, G can be dummy */
 if(Y&8){m=v;Y&=~8;goto A;} /* best=1st done, redo normal*/
 if(v>m){m=v;X=x;Y=y|S&G;} /* update max, mark with S */
 }
 /* if non castling */
 t+=p<5; /* fake capt. for nonsliding*/
 if(p<3&6*k+(y&V)==S /* pawn on 3rd/6th, or */
  ||(u&~24)==36&j==7&& /* virgin K moving sideways,*/
  G&M&&b[G=(x|7)-(r>>1&7)]&32 /* 1st, virgin R in corner G*/
  &&!{b[G^1]|b[G^2]}) /* 2 empty sqrs. next to R */
 {F=y;t--; /* unfake capt., enable e.p.*/
 }W(!t); /* if not capt. continue ray*/
 }}W((x=x+9&~M)-B); /* next sqr. of board, wrap */
 C:if(m>l/4|m<-l/4)d=99; /* mate is indep. of depth */
 m=m+l?m:-D(24-k,-l,l,0,J,Z,S,S,1)/2; /* best loses K: (stale)mate*/
 if(!a->K|(a->X&M)!=M|a->D<=d) /* if new/better type/depth:*/
 {a->K=Z;a->V=m;a->D=d;A->K=0; /* store in hash,dummy stays*/
  a->X=X|8*(m>q)|S*(m<l);a->Y=Y; /* empty, type (limit/exact)*/
 }
 /* encoded in X S,8 bits */
 /*if(z==8)printf("%2d ply, %9d searched, %6d by (%2x,%2x)\n",d-1,N,m,X,Y&0x77);*/
 }
 if(z&8){K=X;L=Y&~M;}
 return m;
}

main()
{
int j,k=8,*p,c[9];

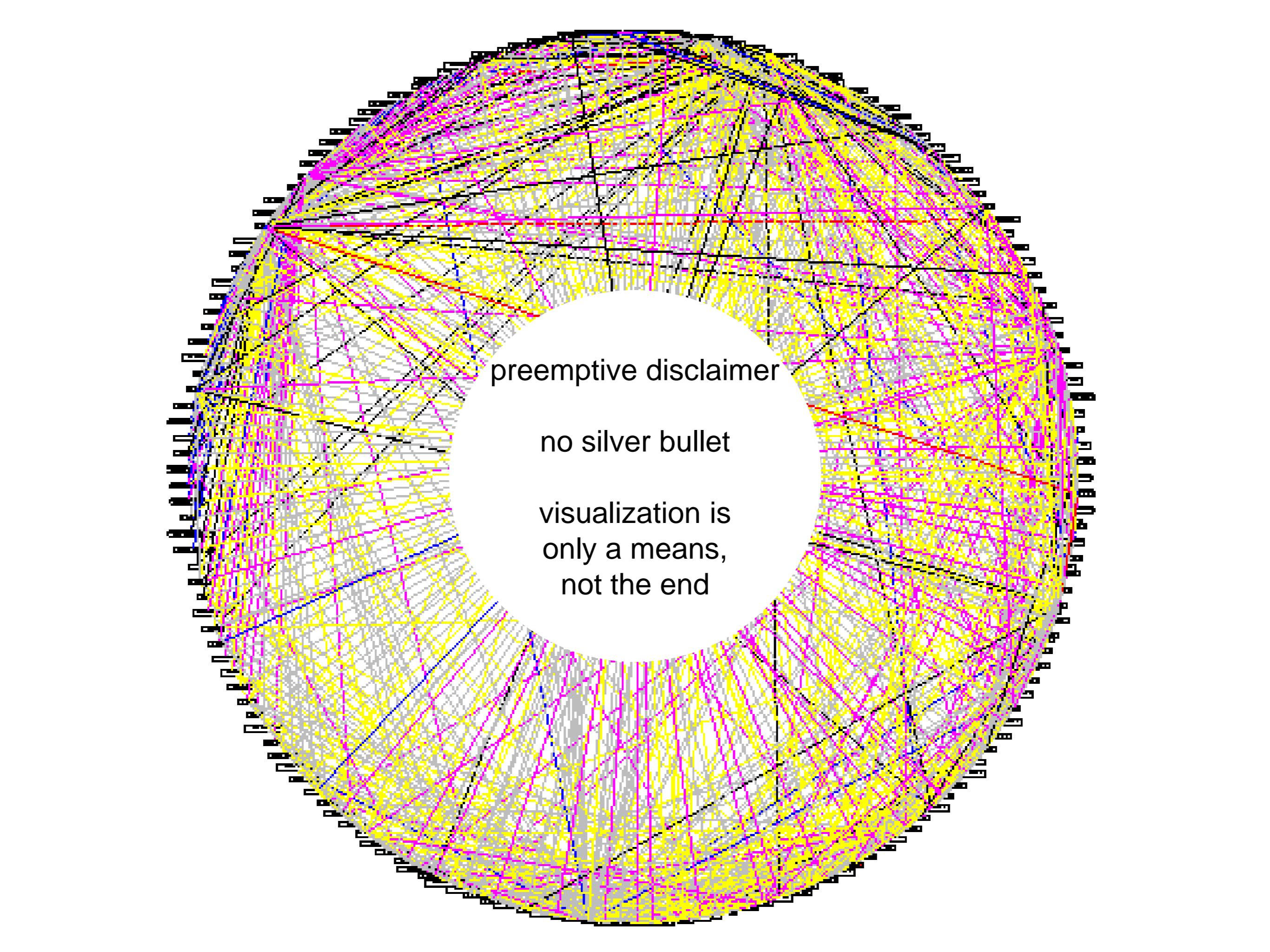
F(i,0,8)
{b[i]=(b[i+V]=o[i+24]+40)+8;b[i+16]=18;b[i+96]=9; /* initial board setup*/
 F(j,0,8)b[16*j+i+8]=(i-4)*(i-4)+(j-3.5)*(j-3.5); /* center-pts table */
}
F(i,M,1035)T[i]=random()>>9;

W(1) /* play loop */
{F(i,0,121)printf(" %c",i&8&&(i+=7)?10:n[b[i]&15]); /* print board */
 p=c;W((*p++=getchar())>10); /* read input line */
 N=0;
 if(*c-10){K=c[0]-16*c[1]+C;L=c[2]-16*c[3]+C;}else /* parse entered move */
 D(k,-l,l,Q,1,1,O,8,0); /* or think up one */
}

```



Software... Visualization?



preemptive disclaimer

no silver bullet

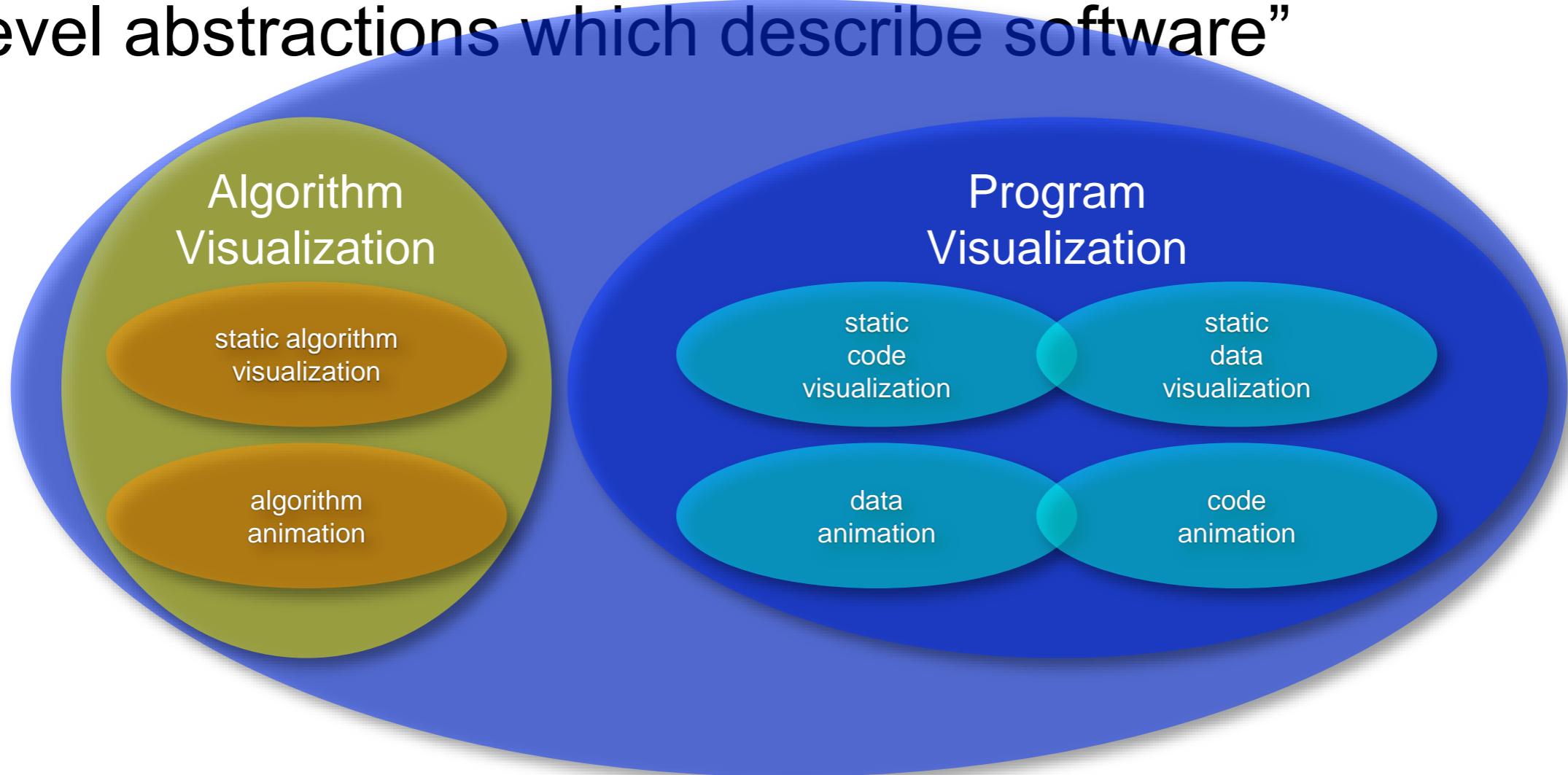
visualization is
only a means,
not the end

not software visualization

```
#include <math.h>
#include <sys/time.h>
#include <X11/Xlib.h>
#include <X11/keysym.h>
double L,o,P
,_=dt,T,Z,D=1,d,
s[999],E,h=.8,I,
J,K,w[999],M,m,O
,n[999],j=33e-3,i=
1E3,r,t,u,v,W,S=
74.5,l=221,X=7.26,
a,B,A=32.2,c,F,H;
int N,q,C,y,p,U;
Window z; char f[52]
; GC k; main(){ Display*e=
XOpenDisplay(0); z=RootWindow(e,0); for(XSetForeground(e,k=XCreateGC
; scanf("%lf%lf%lf",y+n,w+y, y+s)+1; y++); XSelectInput(e,z= XCreate
0,0,WhitePixel(e,0) ),KeyPressMask); for(XMapWindow(e,z); ; T=sin(j)
; K= cos(j); N=1e4; M+= H*_; Z=D*K; F+=_*P; r=E*K; W=cos( O
sin(j); a=B*T*D-E*W; XCLEARWINDOW(e,z); t=T*E+ D*B*W; i=1e-3
*T*B,E*d/K *B+v+B/K*F*D)*_; p<y; ){ T=p[s]+i; E=c-r; W=sin(j)
]== 0|K <fabs(W=T*r-I*E +D*P) |fabs(D=t *D+Z *T
*D; N-1E4&& XDrawLine(e ,z,k,N ,U,q,C); N+=1; U+=1;
XDrawString(e,z,k ,20,380,f,17); D=v/1000000.0; s+=1;
} } }
```

Software Visualization

- ▶ Program Visualization: “The visualization of the actual program code or data structures in static or dynamic form”
- ▶ Algorithm Visualization: “The visualization of the higher-level abstractions which describe software”



Software Visualization in Context

- ▶ There are many good-looking visualizations, but...
- ▶ When it comes to maintenance & evolution, there are several issues:
 - ▶ Scalability
 - ▶ Information Retrieval
 - ▶ What to visualize
 - ▶ How to visualize
 - ▶ Limited time
 - ▶ Limited resources

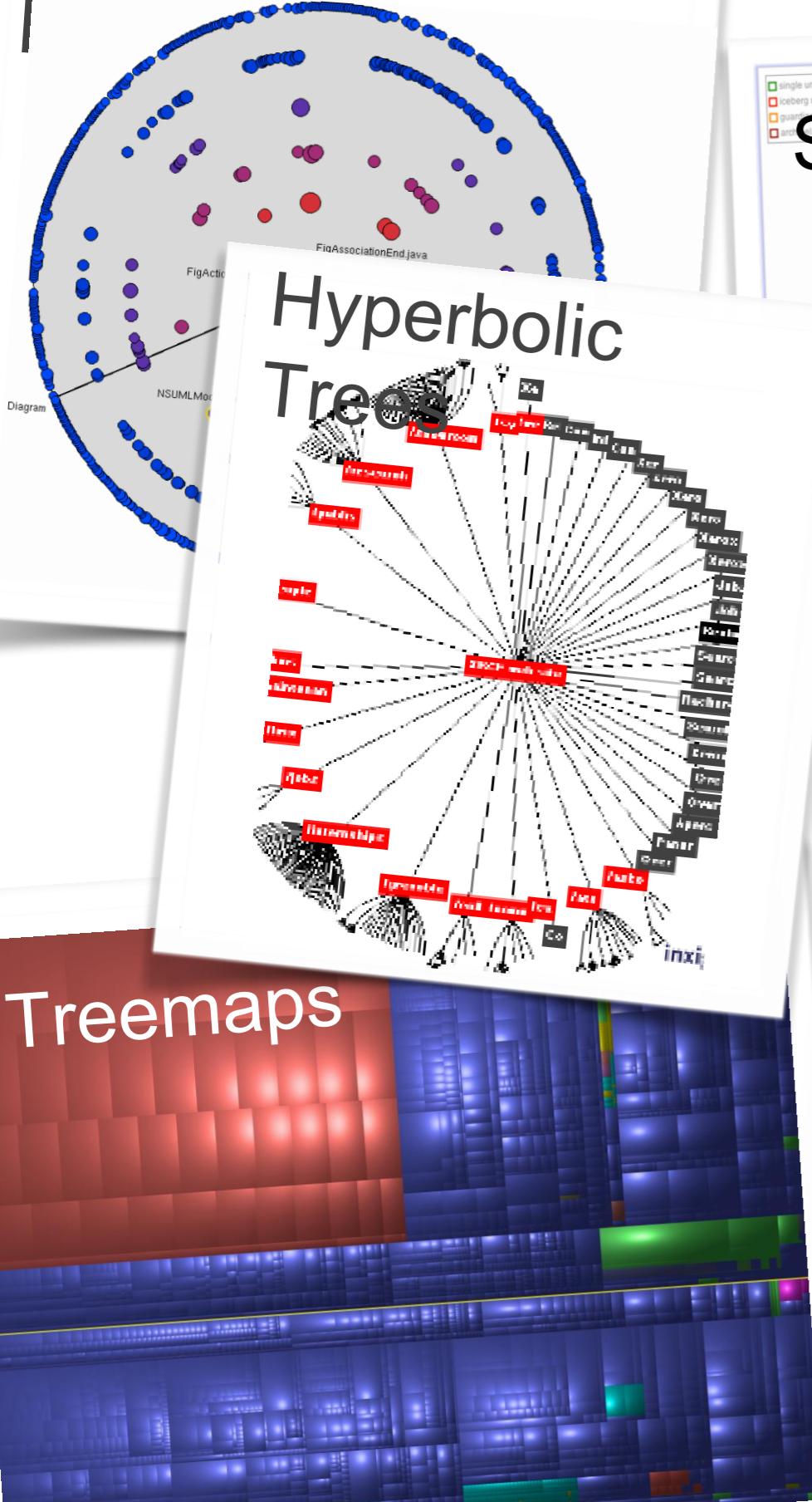
Program Visualization

- ▶ “The visualization of the actual program code or data structures in either static or dynamic form”
- ▶ Overall goal: generate views of a system to understand it
- ▶ Surprisingly complex problem domain/research area
 - ▶ Visual Aspects: Efficient use of space, overplotting problems, layout issues, HCI issues, GUI issues, lack of conventions (colors, shapes, etc.)
 - ▶ Software Aspects
 - ▶ Granularity (complete systems, subsystems, modules, classes, etc.)
 - ▶ When to apply (first contact, known/unknown parts, forward engineering?)
 - ▶ Methodology

Static Code Visualization

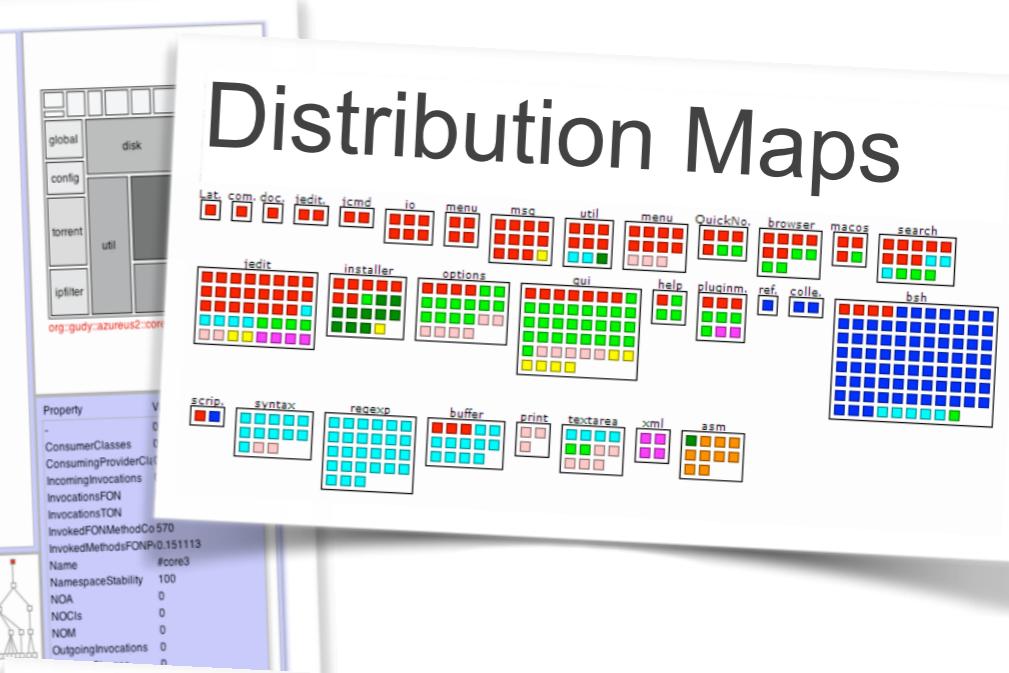
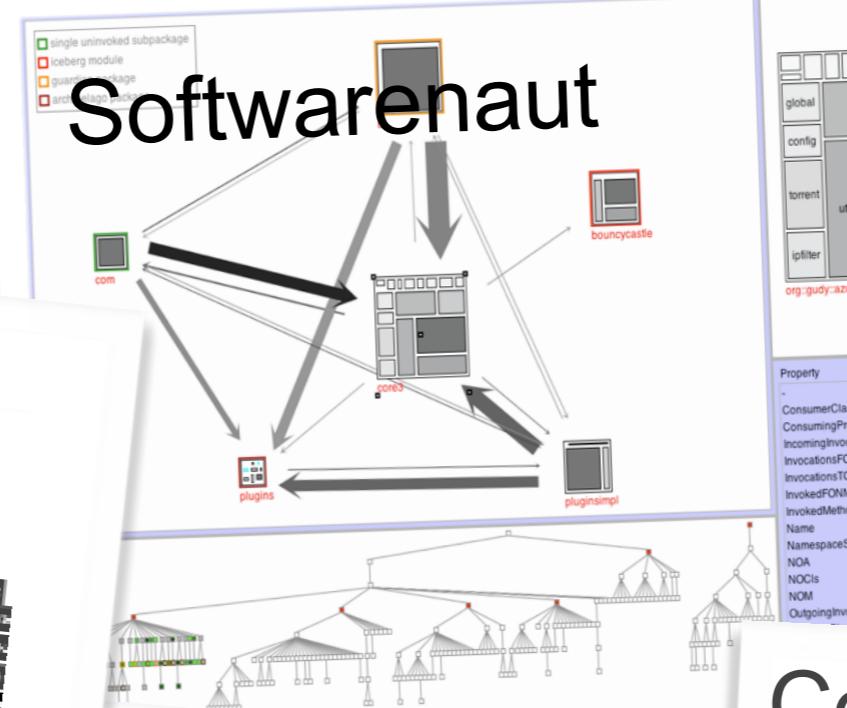
- ▶ The visualization of information that can be extracted from a system at “compile-time”
- ▶ Directly influenced by programming languages and their paradigms
 - ▶ Object-Oriented: classes, methods, attributes, inheritance, ...
 - ▶ Procedural: procedures, invocations, imports, ...
 - ▶ Functional: functions, function calls, ...

The Evolution

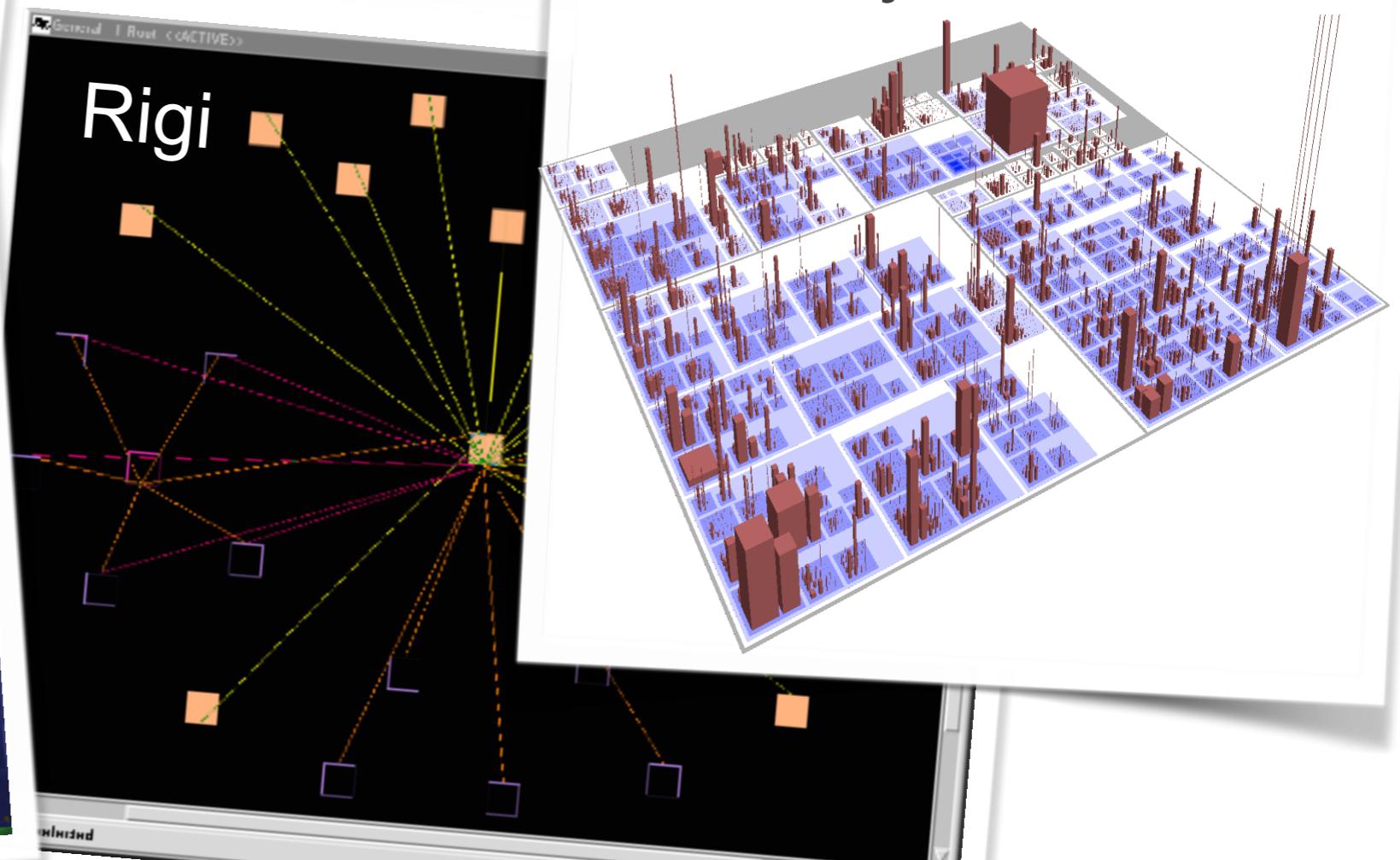


Treemaps

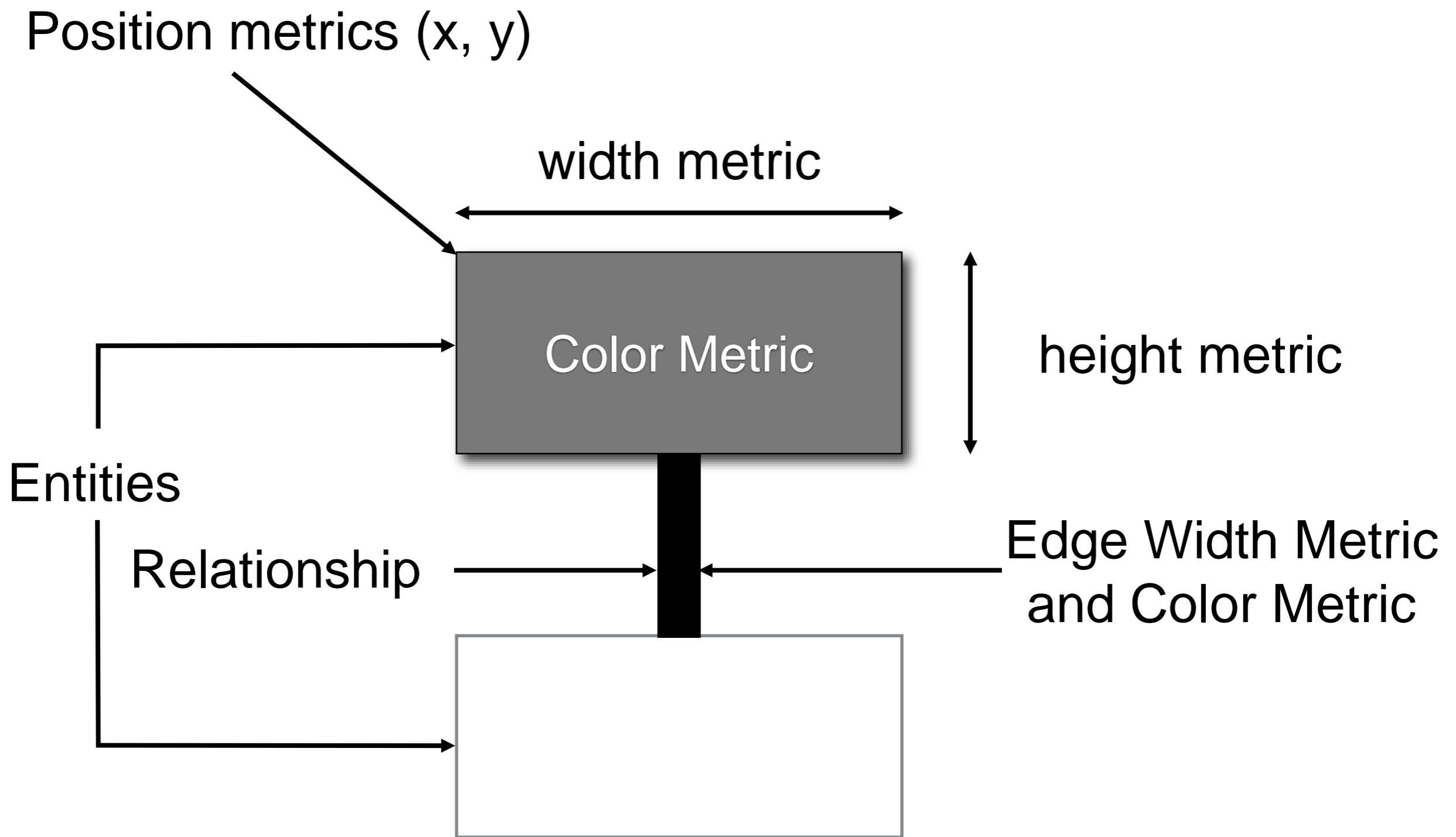
Examples



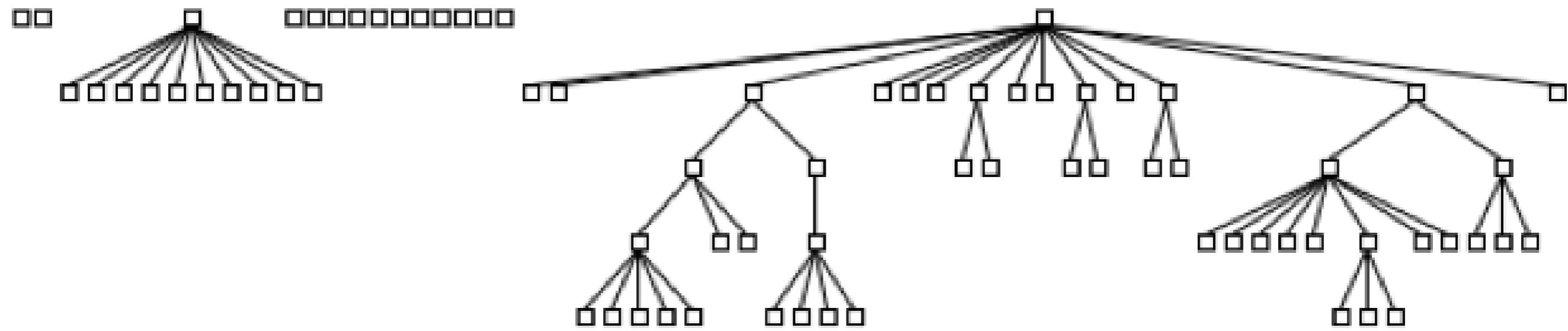
Code City



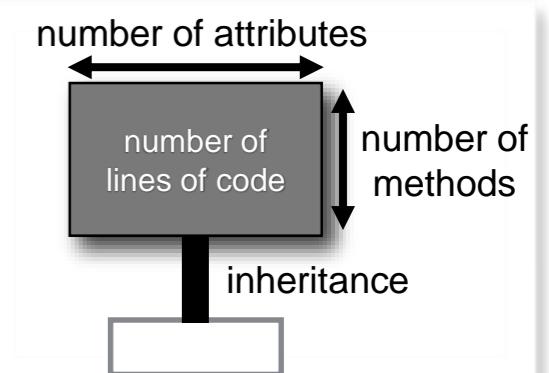
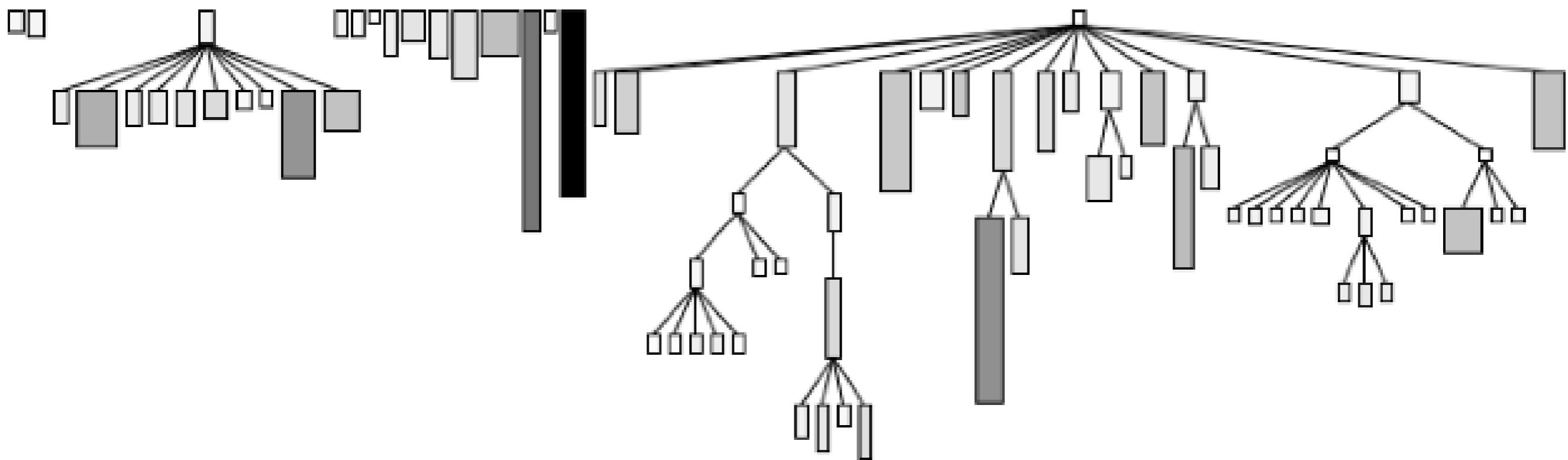
The Polymetric View Principle

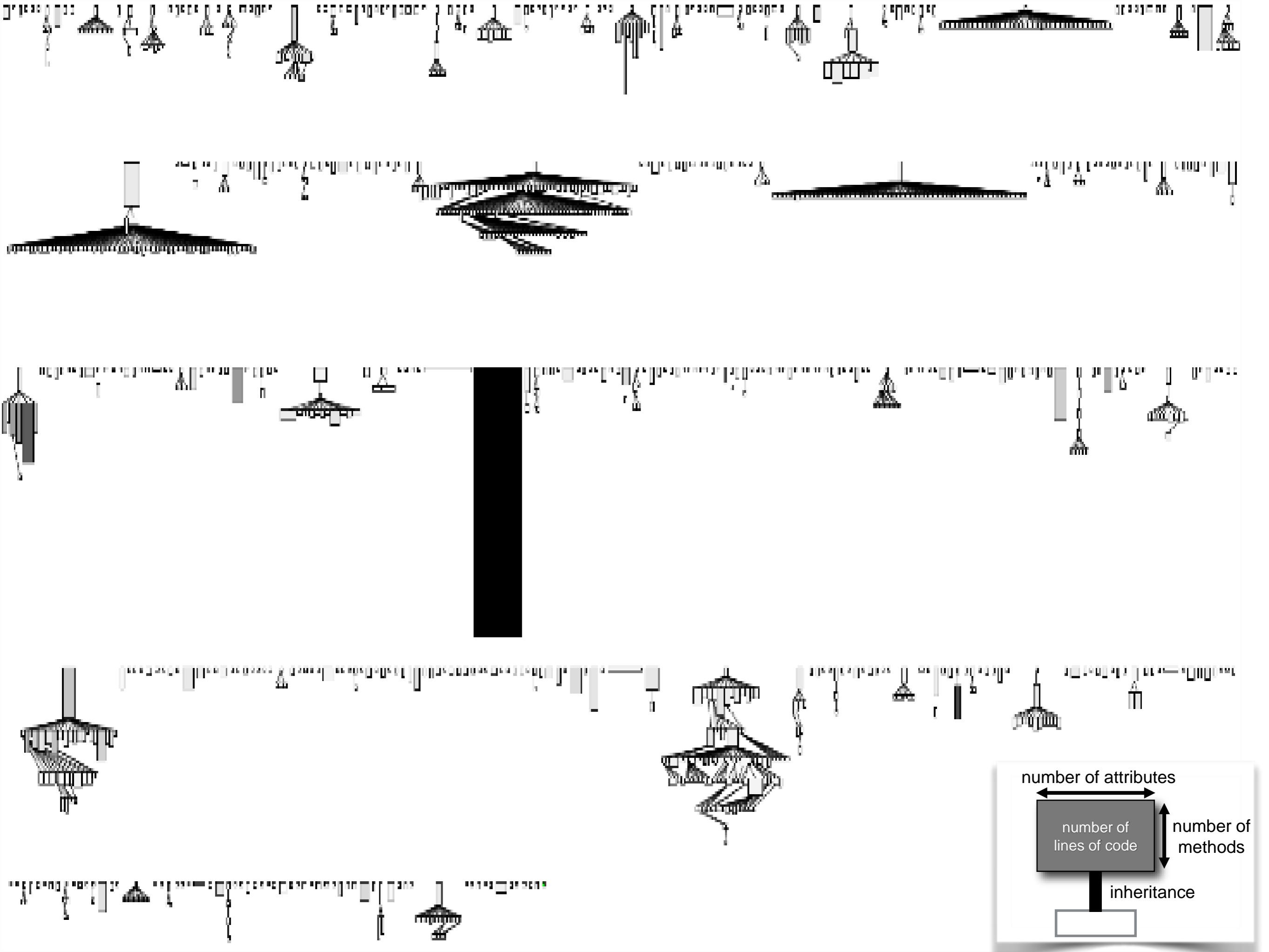


Class Hierarchy

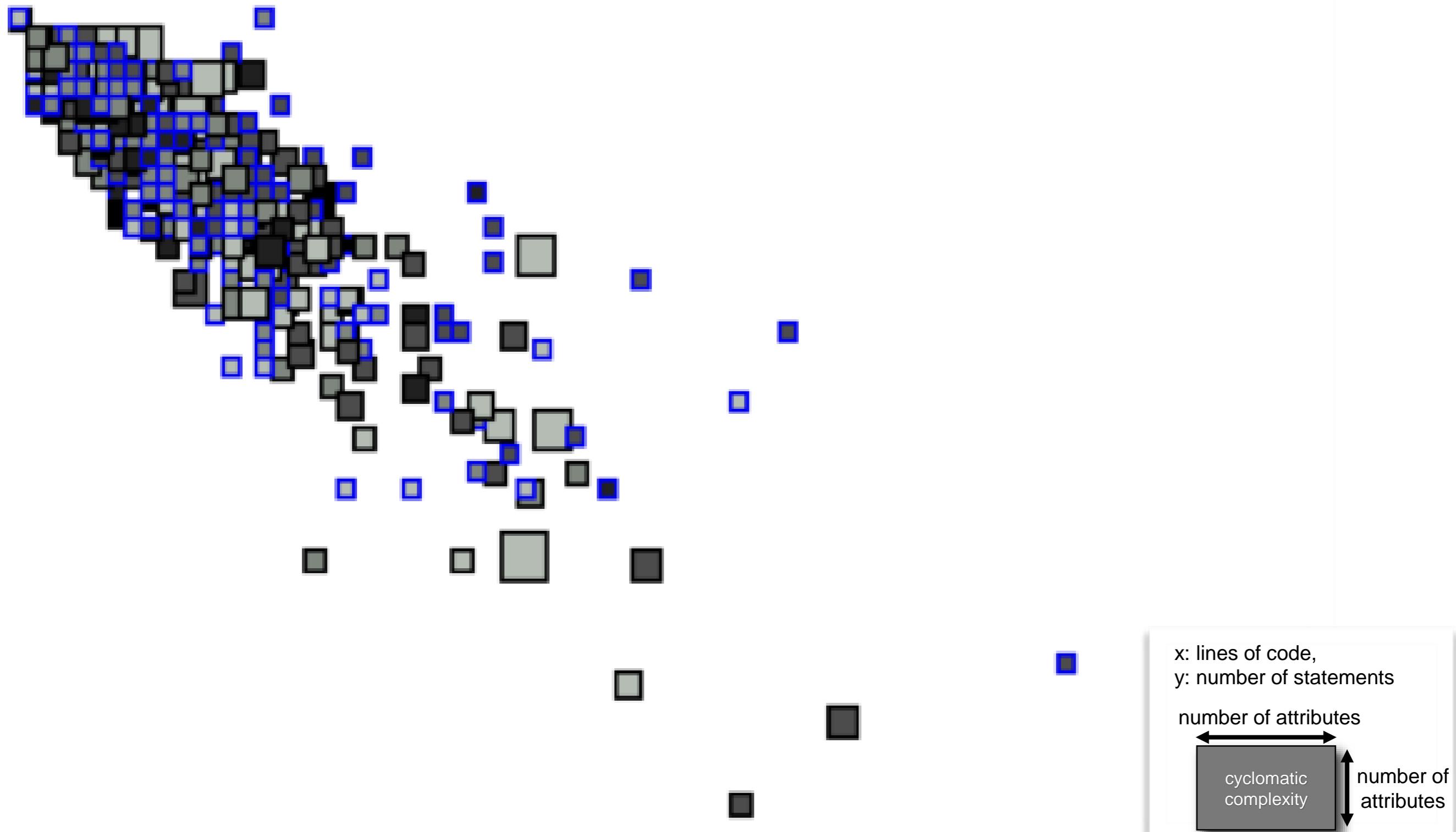


System Complexity View

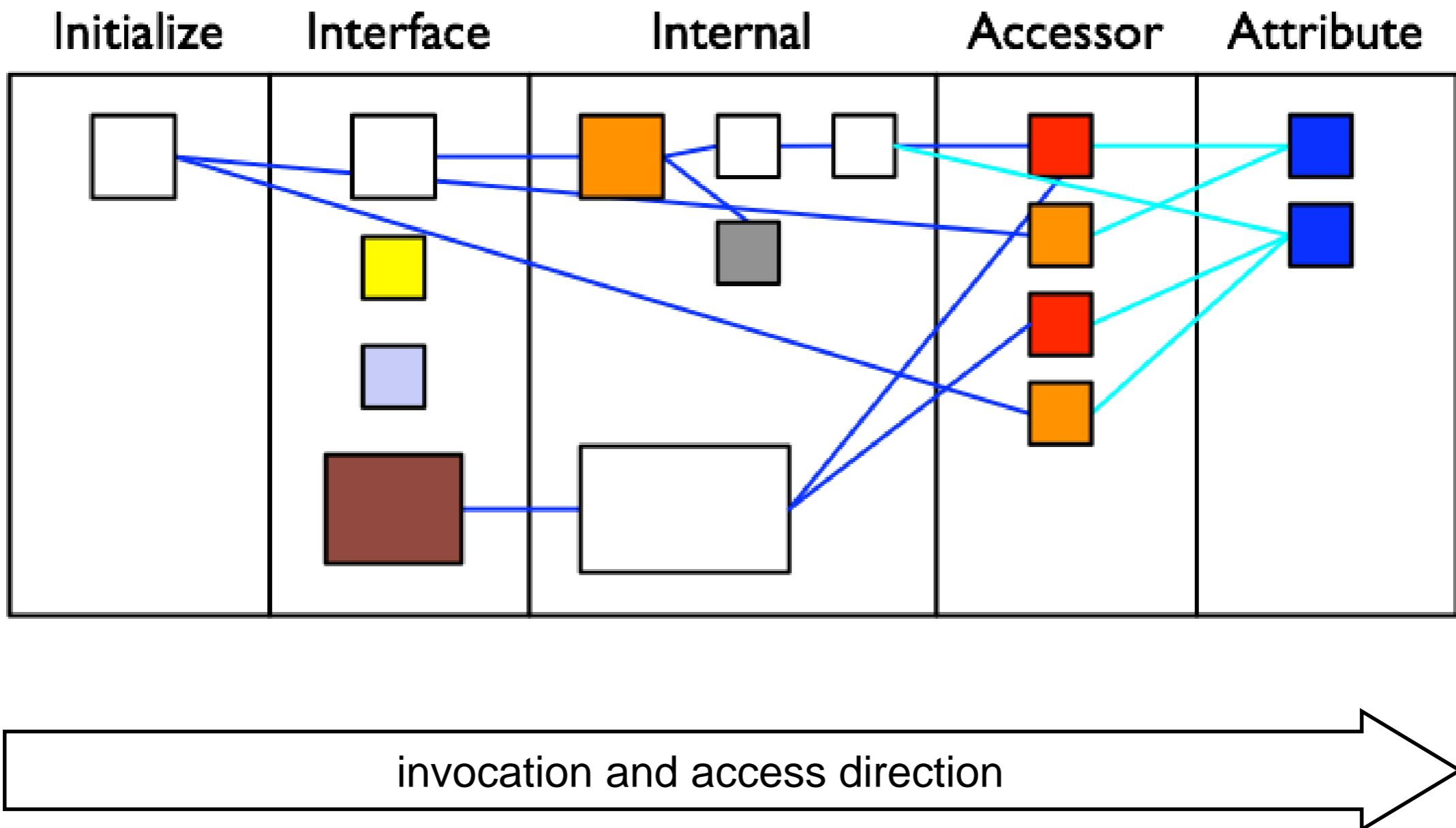




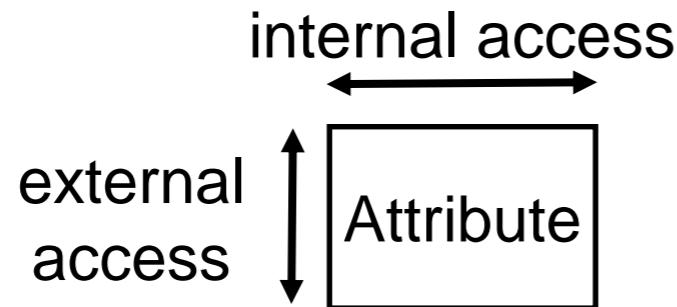
Method Structure Correlation View



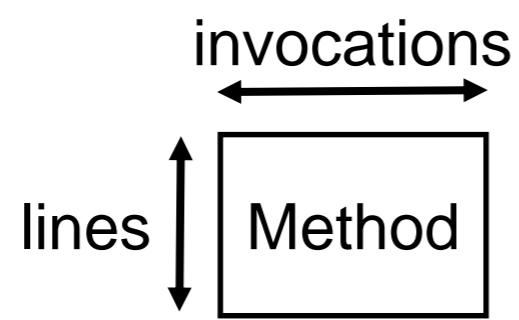
Increasing Information Granularity: The Class Blueprint



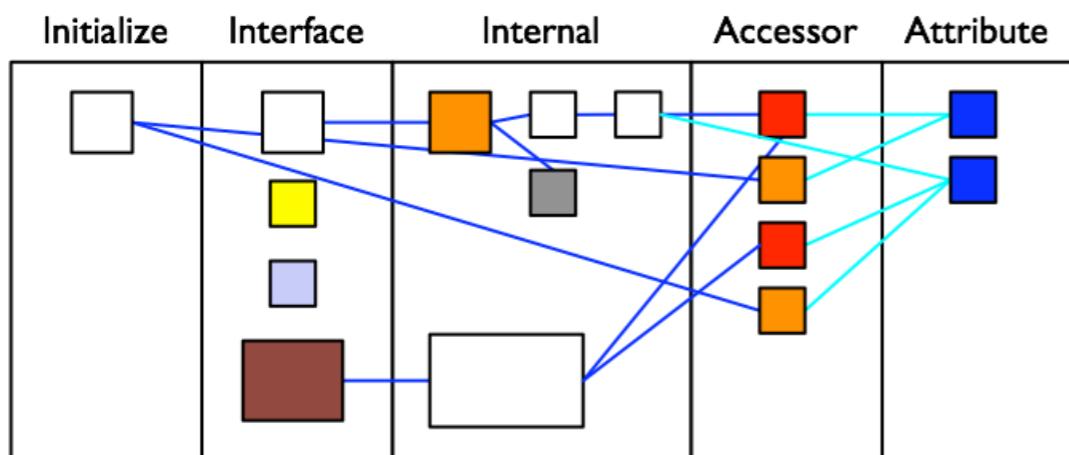
Detailing Class Blueprints



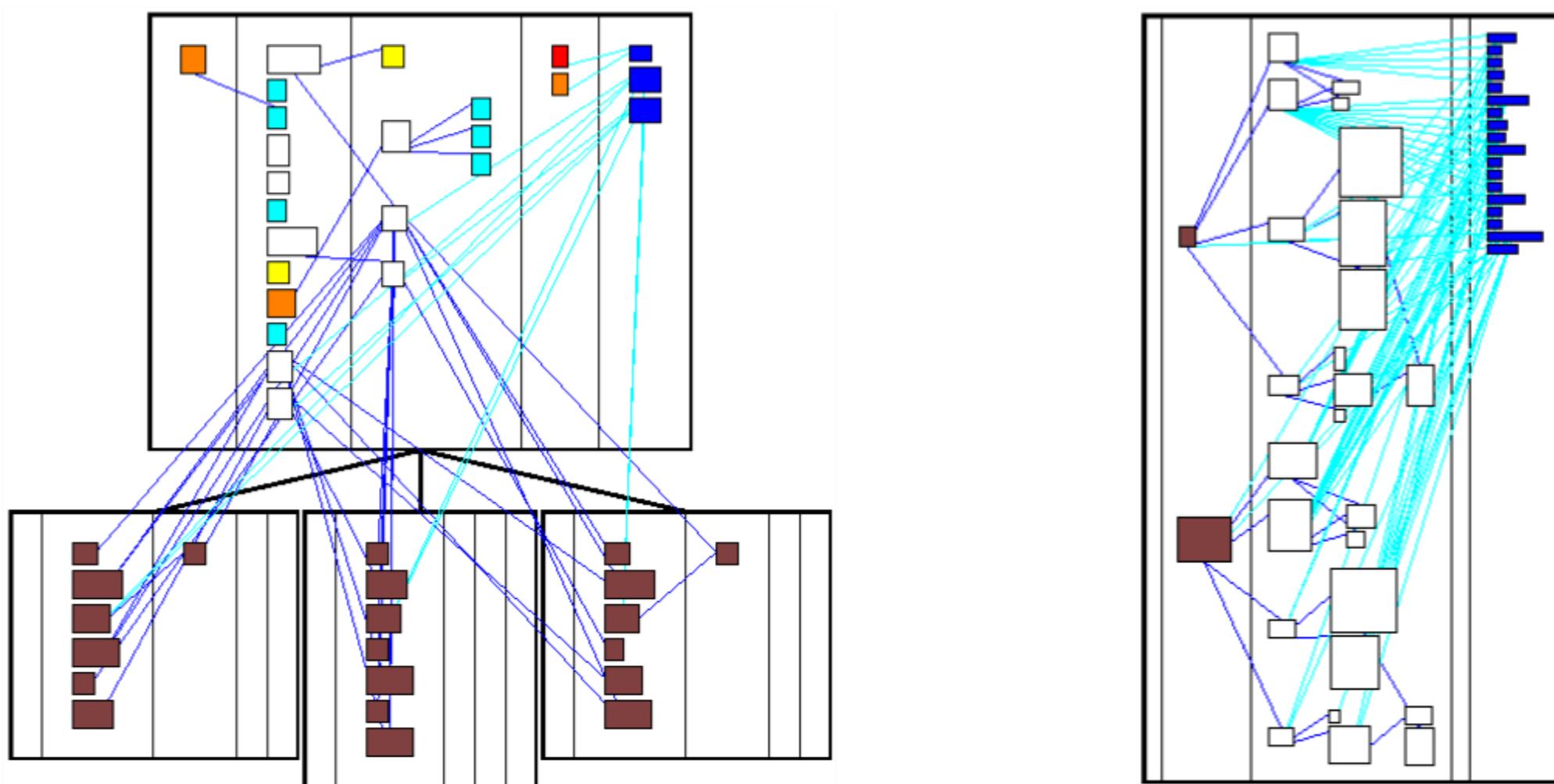
Access
Invocation



Regular			Constant
Overriding			Delegating
Extending			Setter
Abstract			Getter



A Pattern Language based on Class Blueprints



internal access

external access

Attribute

invocations

lines

Method

Regular

Constant

Overriding

Delegating

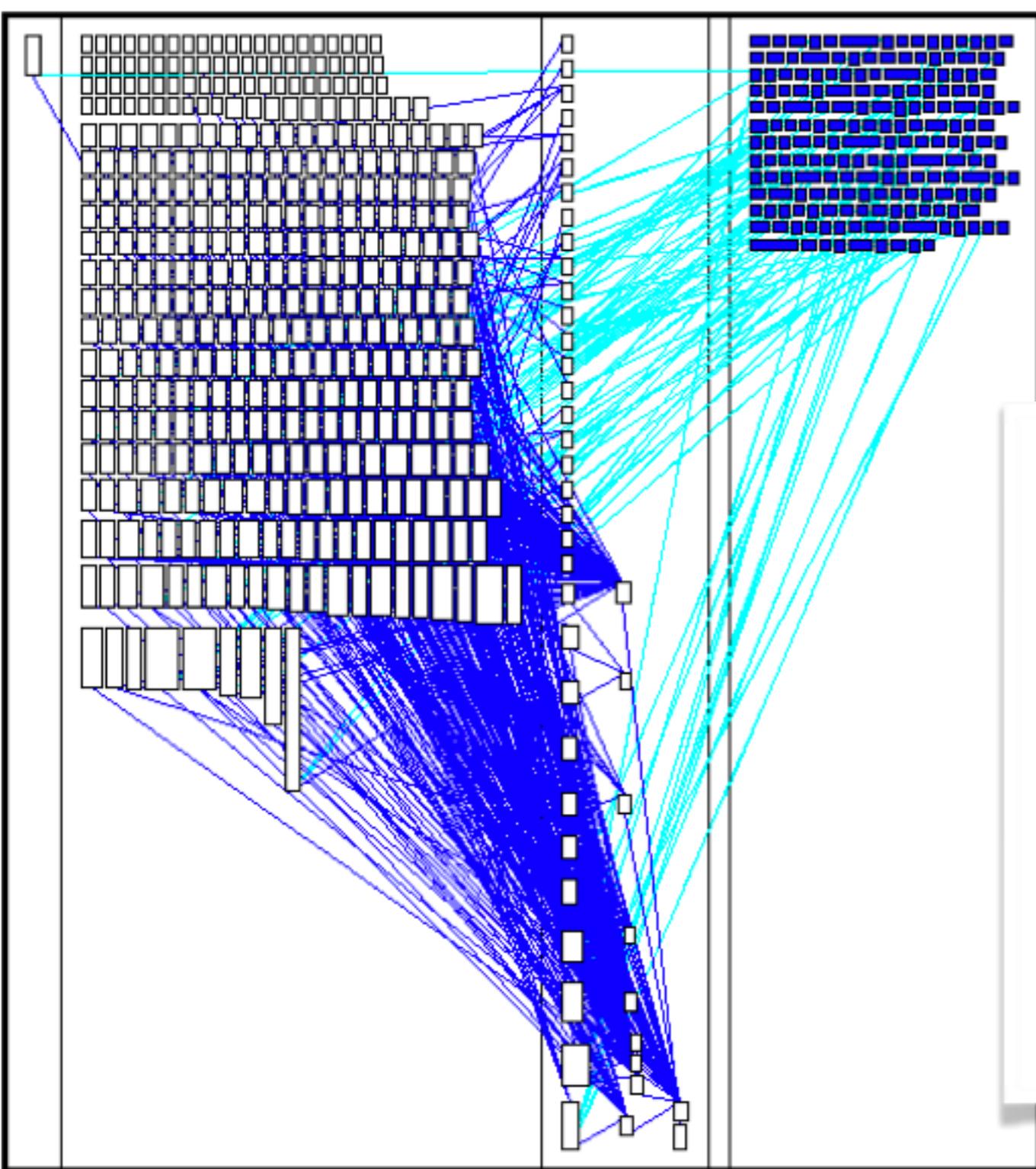
Extending

Setter

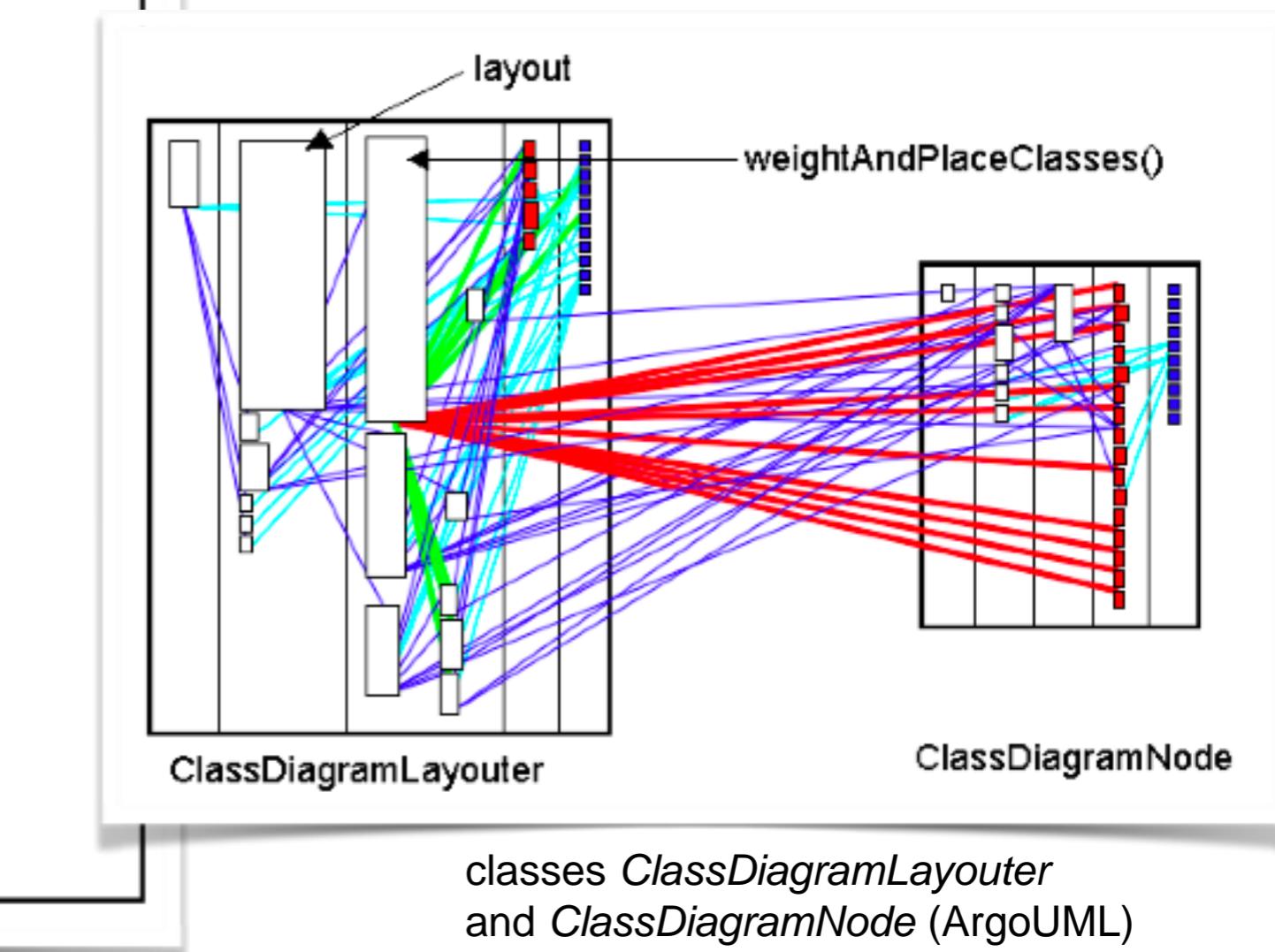
Abstract

Getter

What do you see here ?



class *ModelFacade* (ArgoUML)



classes *ClassDiagramLayouter*
and *ClassDiagramNode* (ArgoUML)

Reflections on Static Visualization

- ▶ **Pros**
 - ▶ Intuitive
 - ▶ Aesthetically pleasing
- ▶ **Cons**
 - ▶ Several approaches are orthogonal to each other
 - ▶ No conventions
 - ▶ Too easy to produce meaningless results
 - ▶ Scaling up is possible at the expense of semantics
- ▶ **Orthogonally**
 - ▶ Without programming knowledge it's only colored boxes and arrows..

References

M. Lanza, R. Marinescu, *Object-Oriented Metrics in Practice*, Springer, 2006.

M. Lanza, *Object-Oriented Reverse Engineering - Coarse-grained, Fine-grained, and Evolutionary Software Visualization*, Ph.D. Thesis, University of Berne, Switzerland, 2003.

<http://www.inf.usi.ch/faculty/lanza/Downloads/Lanz03b.pdf>

Overview of the Overview Pyramid:

<http://pdepend.org/documentation/handbook/reports/overview-pyramid.html>

You are free to:

Share — copy and redistribute the material in any medium or format

Adapt — remix, transform, and build upon the material
for any purpose, even commercially.

The licensor cannot revoke these freedoms as long as you follow the license terms.

Under the following terms:



Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.



ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.

No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.