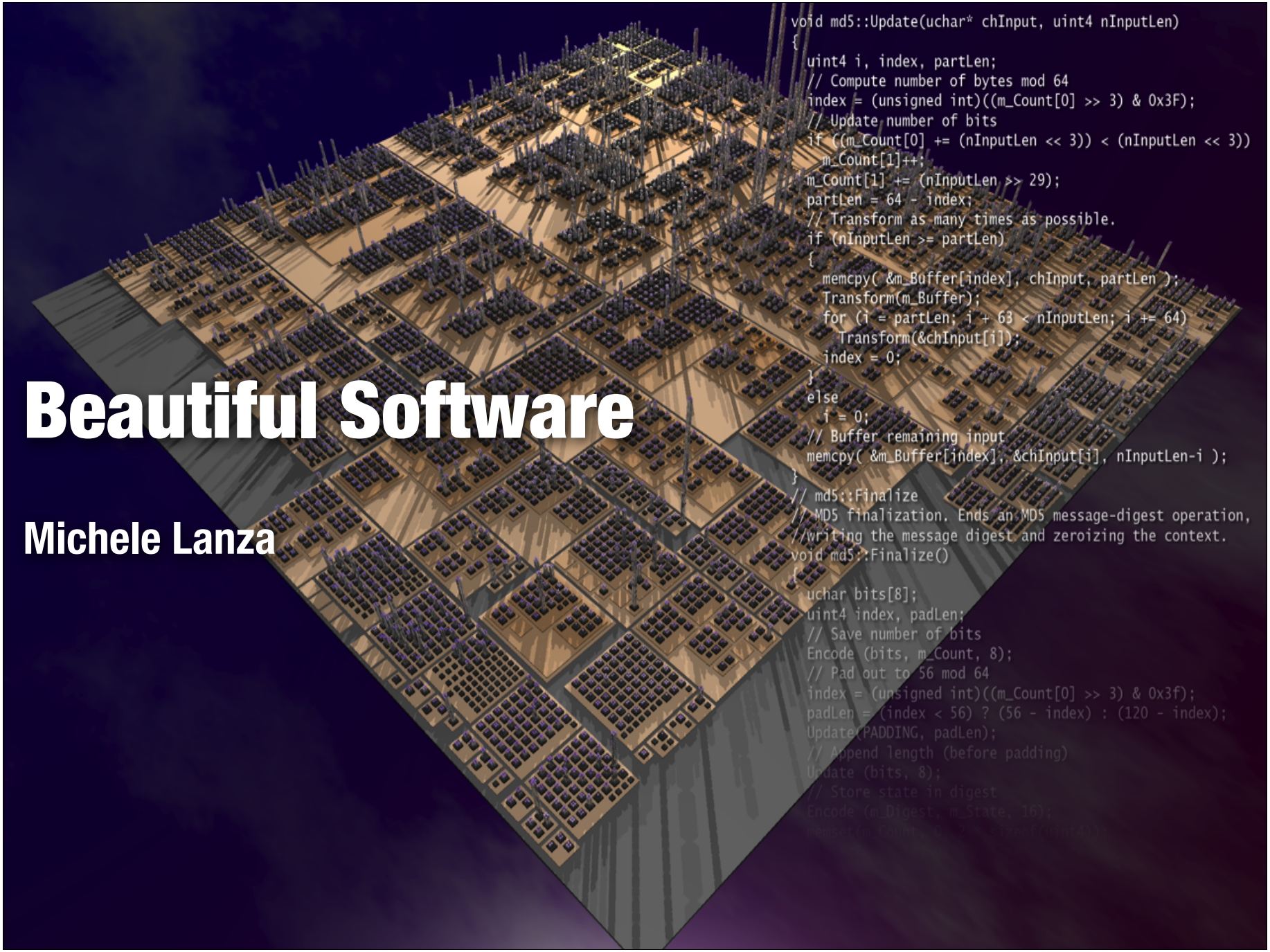
~~Of Code and Change:~~ Beautiful Software

Michele Lanza

REVEAL @ Faculty of Informatics
University of Lugano, Switzerland

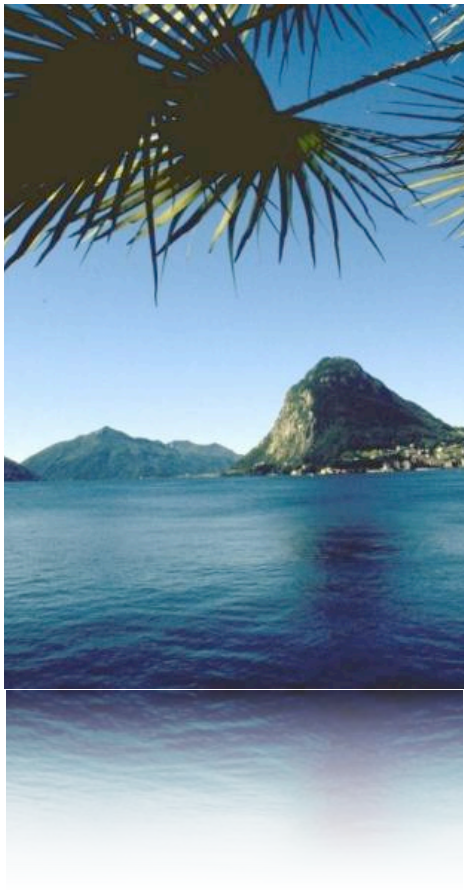


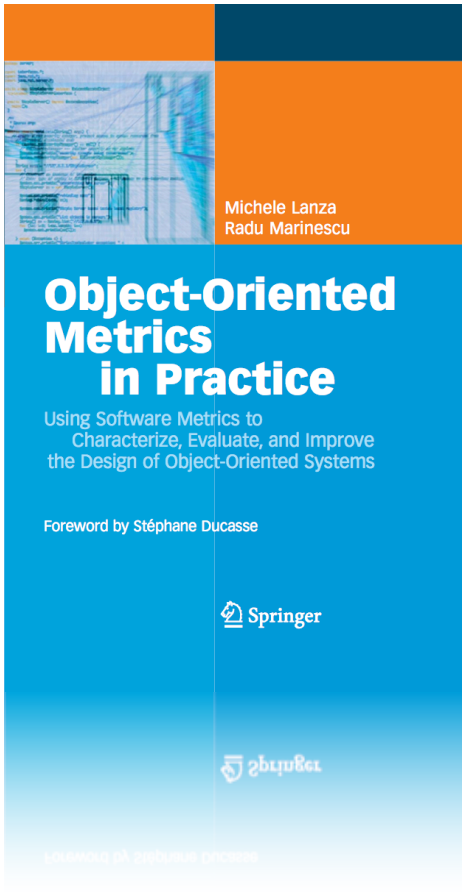
Beautiful Software

Michele Lanza

Prologue

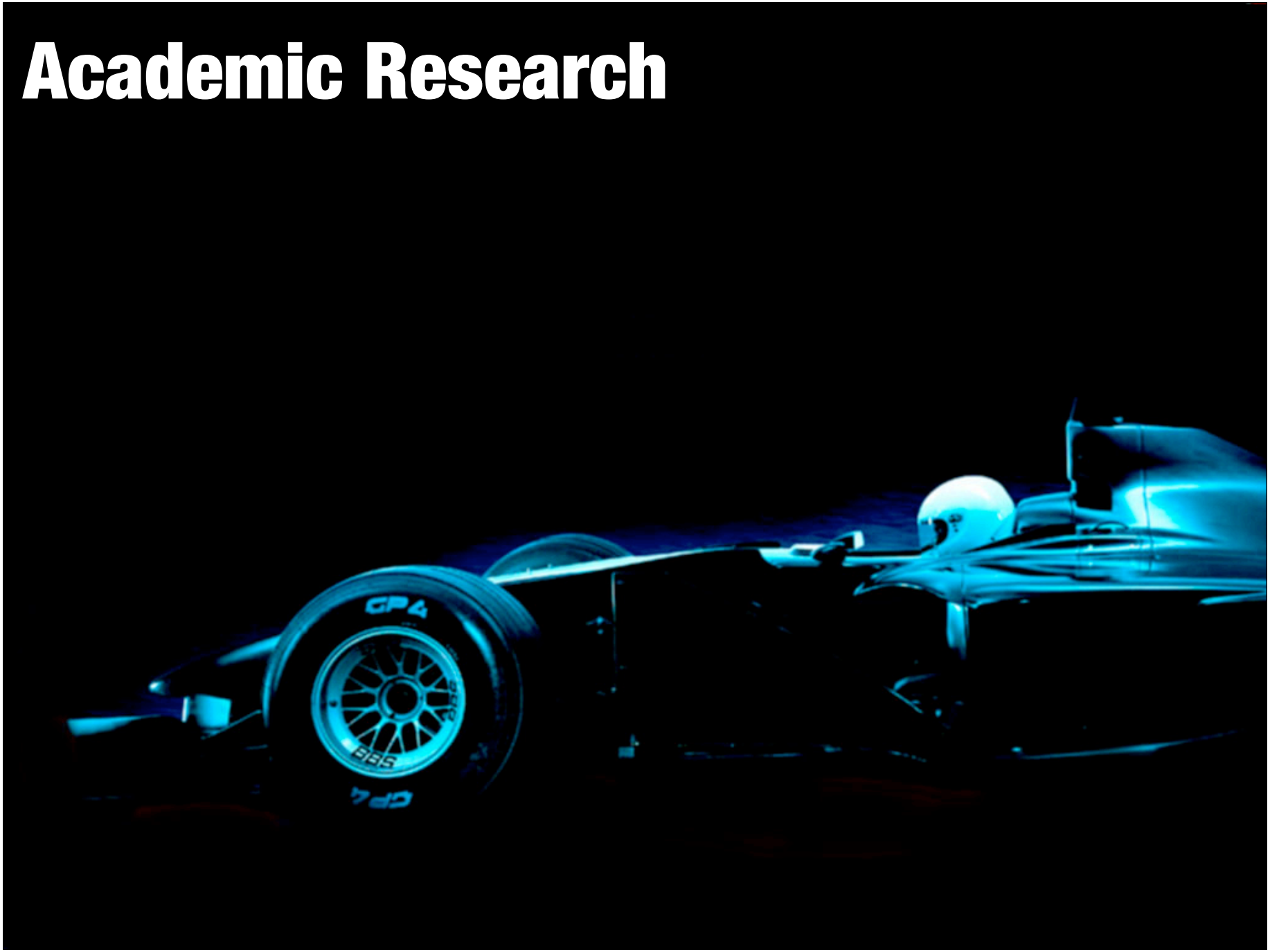
Who the heck is this guy?





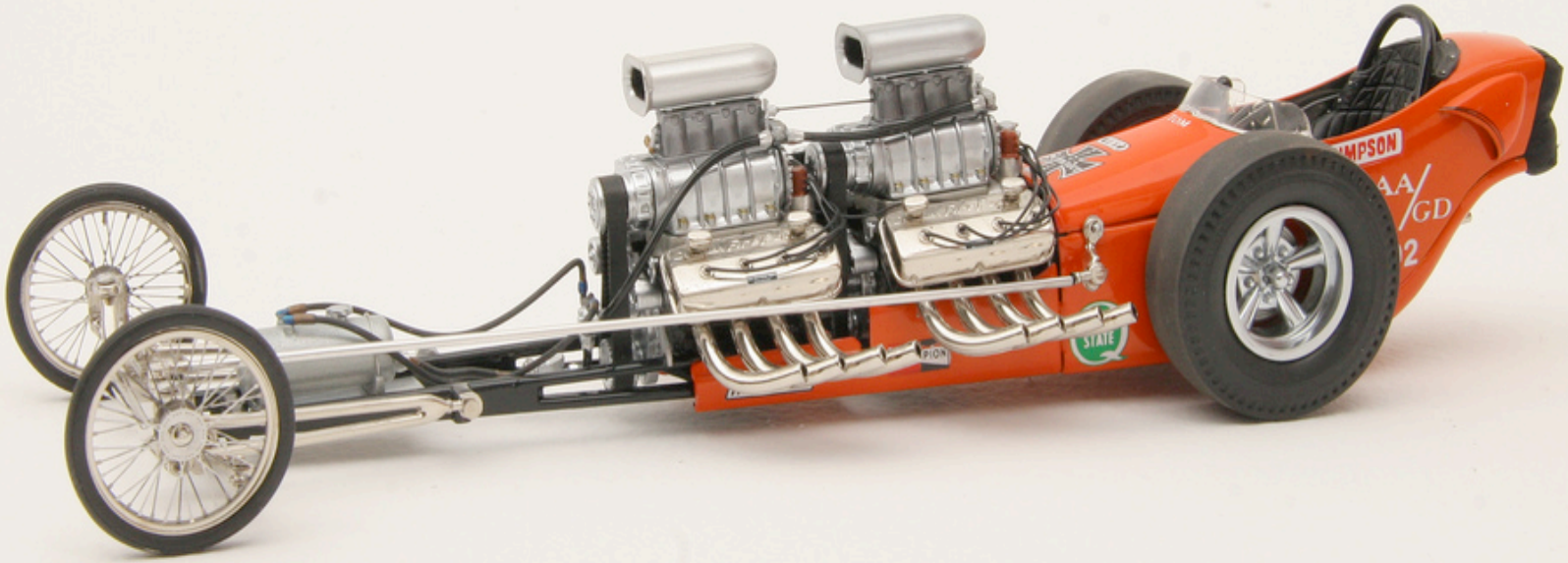
Michele Lanza

Academic Research




Industrial Reality

Military Research

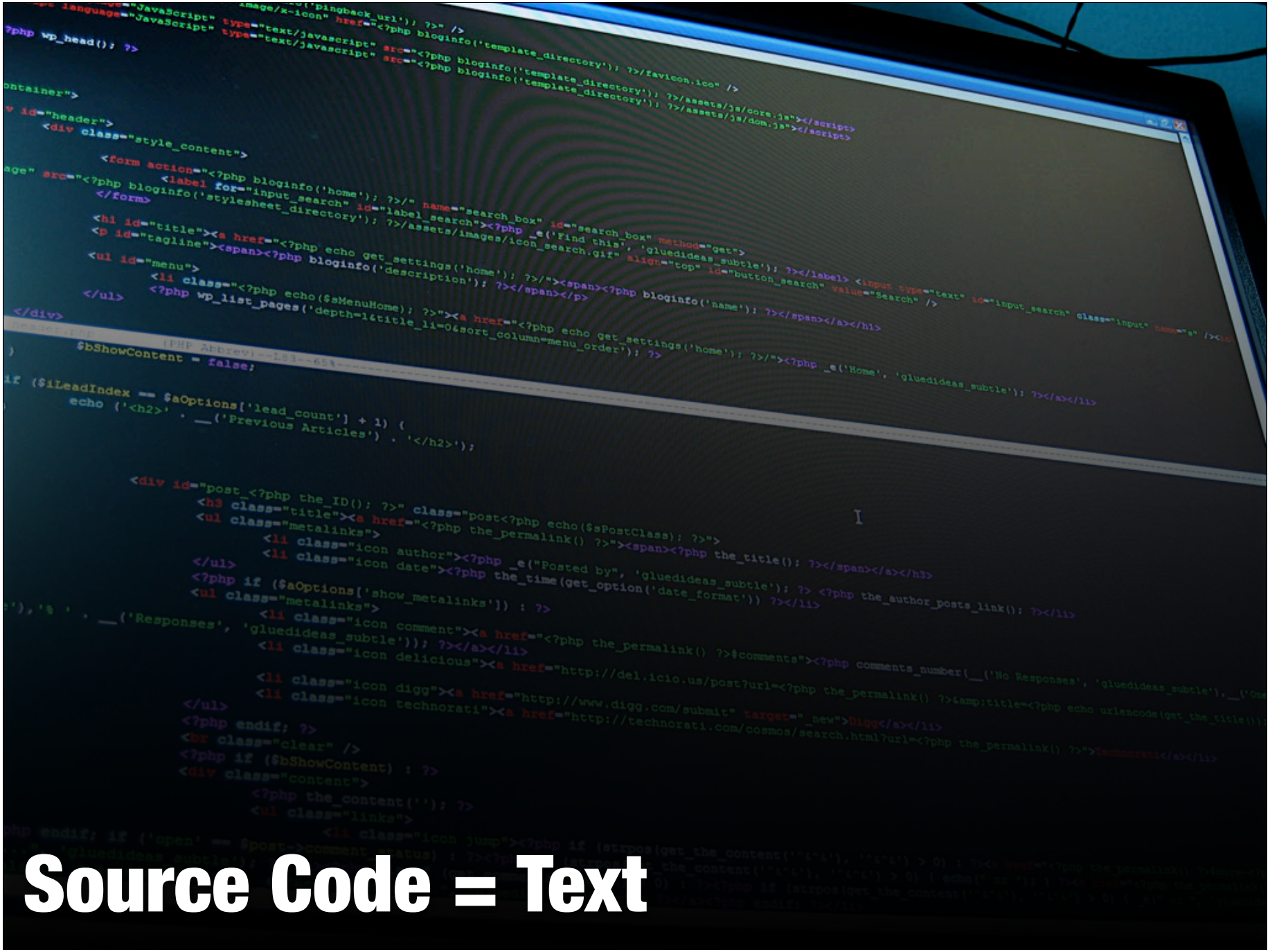


Part 1

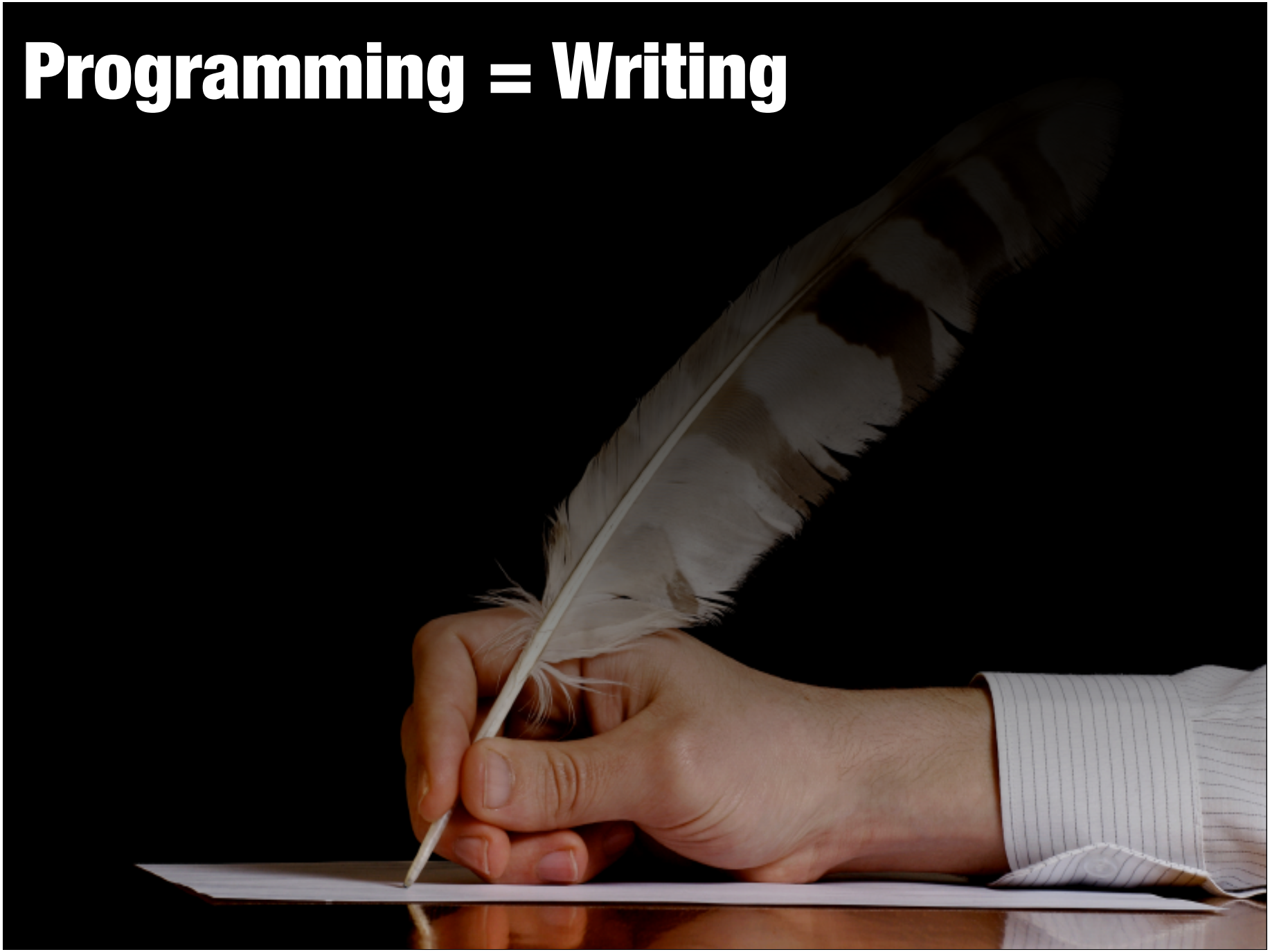
Software Visualization 101

What is Software?

[Software is] anything but hardware, meaning that the "hard" are the parts that are tangible (able to hold) while the "soft" part is **the intangible objects inside the computer.**

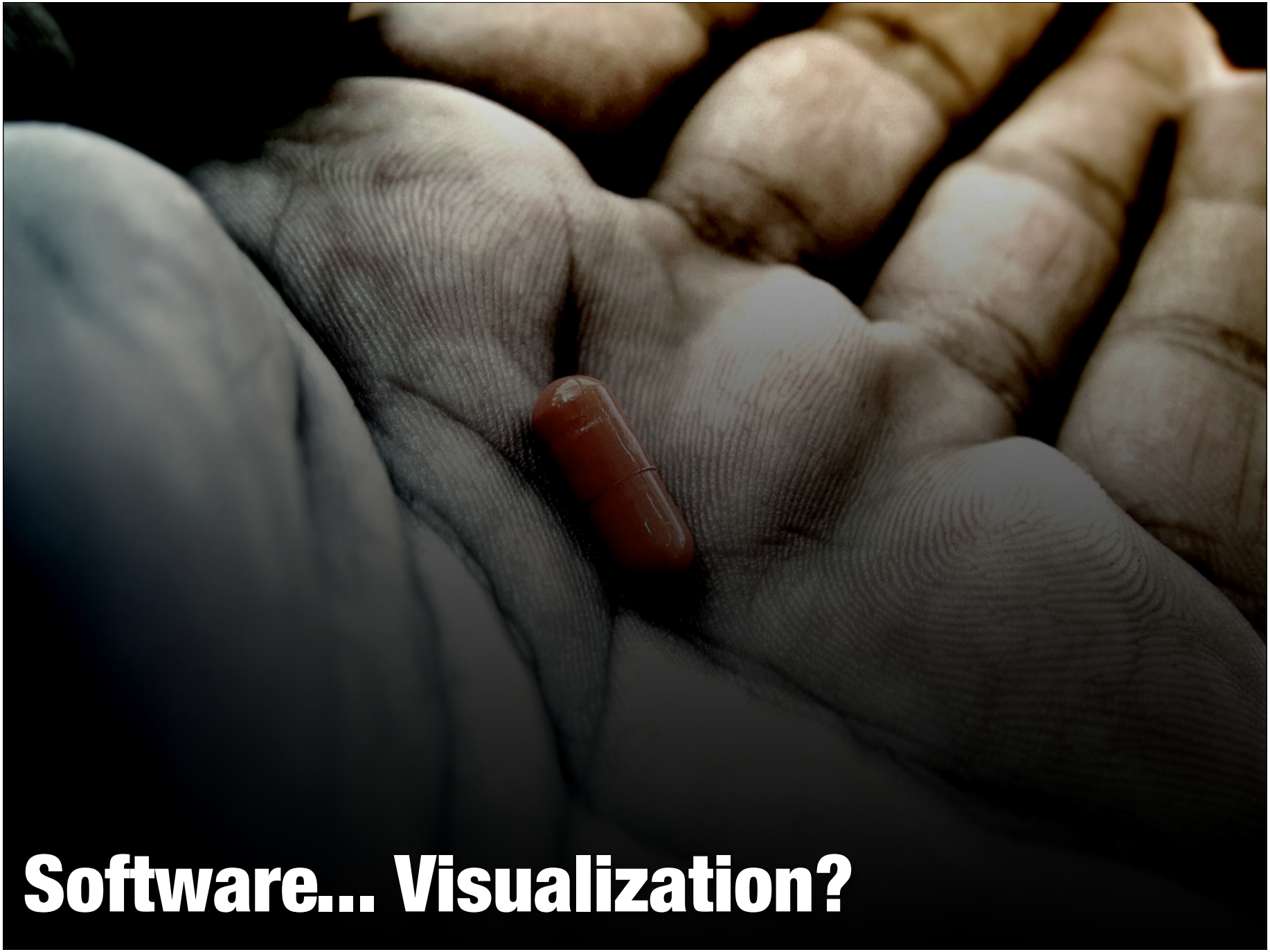

Source Code = Text

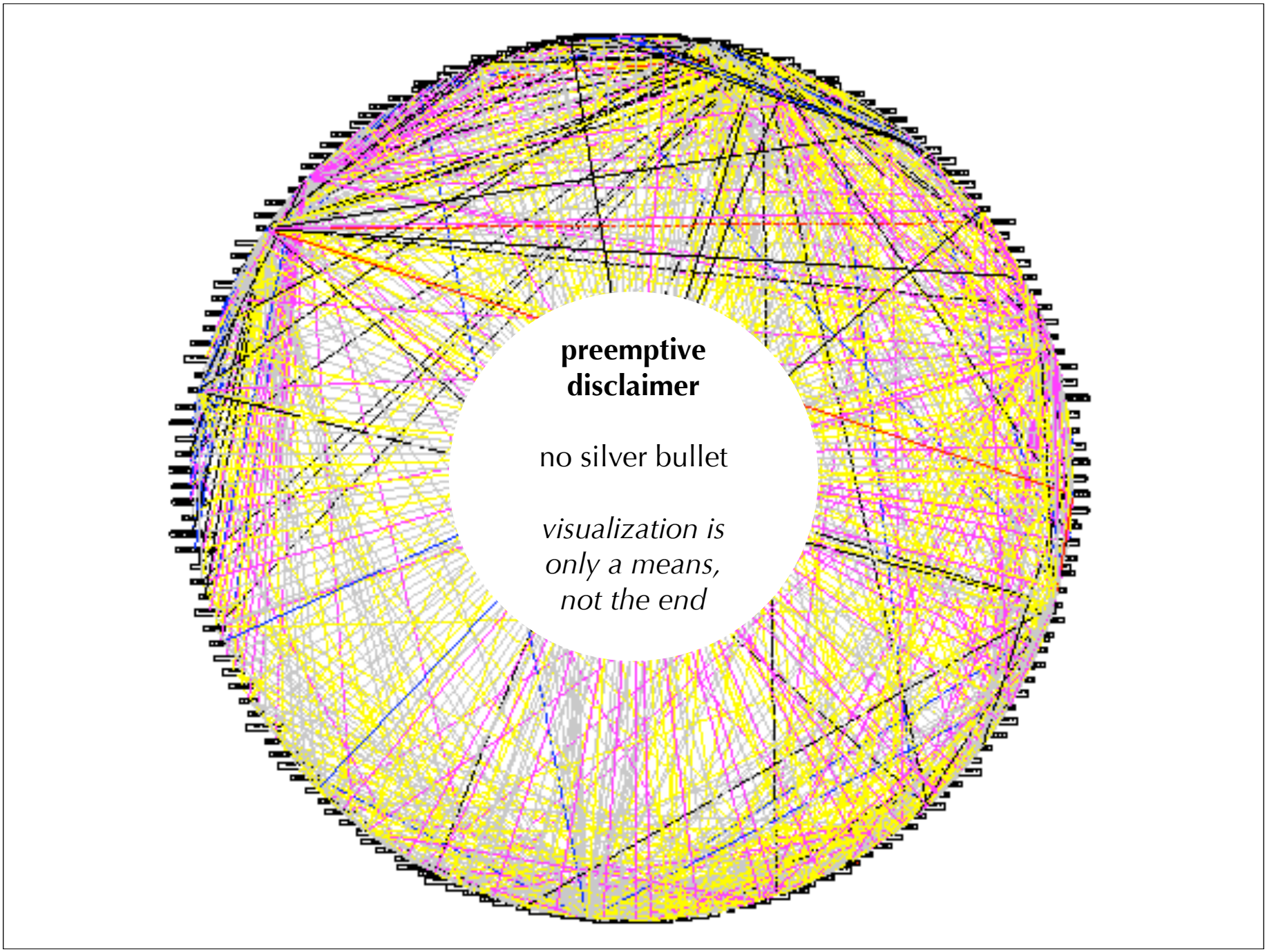
Programming = Writing



```
/************************************************************************/
/*                              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 ptomotion) 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,i;       /* 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;                        /* examine stored data      */
  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;                                /* start iter., no best yet */
 N++;                                          /* node count (for timing)  */
 W(d++<n|z==8&N<1e7&d<98)                      /* iterative deepening loop */
 {x=B=X;                                       /* start scan at prev. best */
  Y|=8&Y>>4;                                   /* request try noncastl. 1st*/
  m=d>1?-I:e;                                  /* unconsidered:static eval */
  do{u=b[x];                                   /* scan board looking for   */
   if(u&k)                                     /*  own piece (inefficient!)*/
   {r=p=u&7;                                   /* p = piece type (set r>0) */
    j=o[p+16];                                 /* first step vector f.piece*/
    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(Y&8)H=y=Y&~M;                         /* sneak in prev. best move */
      if(y&M)break;                            /* board edge hit           */
      if(p<3&y==E)H=y^16;                      /* shift capt.sqr. H if e.p.*/
      t=b[H];if(t&k|p<3&!(r&7)!=!t)break;      /* capt. own, bad pawn mode */
      i=99*w[t&7];                             /* value of capt. piece t   */
      if(i<0||E-S&&b[E]&&y-E<2&E-y<2)m=I;       /* 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?-m:-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!=-I&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>I/4|m<-I/4)d=99;                        /* mate is indep. of depth  */
  m=m+I?m:-D(24-k,-I,I,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   */
 }                                                   /*(in unused half b[])*/
 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,-I,I,Q,1,1,O,8,0);                            /* or think up one    */
  F(i,0,U)A[i].K=0;                                  /* clear hash table   */
  if(D(k,-I,I,Q,1,1,O,9,2)==I)k^=24;                 /* check legality & do*/
 }
}
```

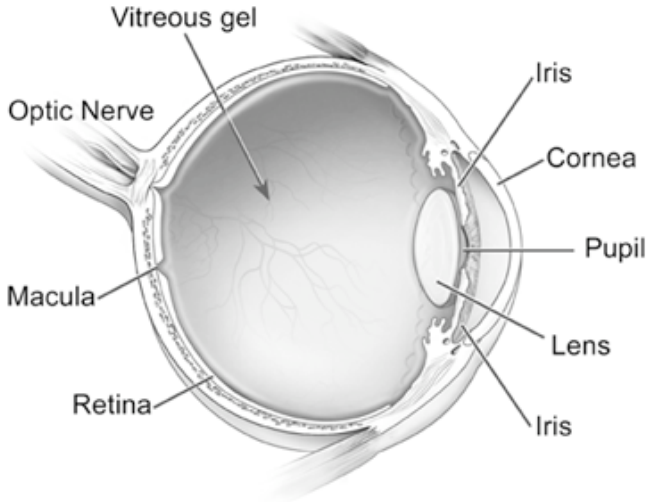
Old Habits Die Hard


Software... Visualization?


preemptive disclaimer
no silver bullet
visualization is only a means, not the end

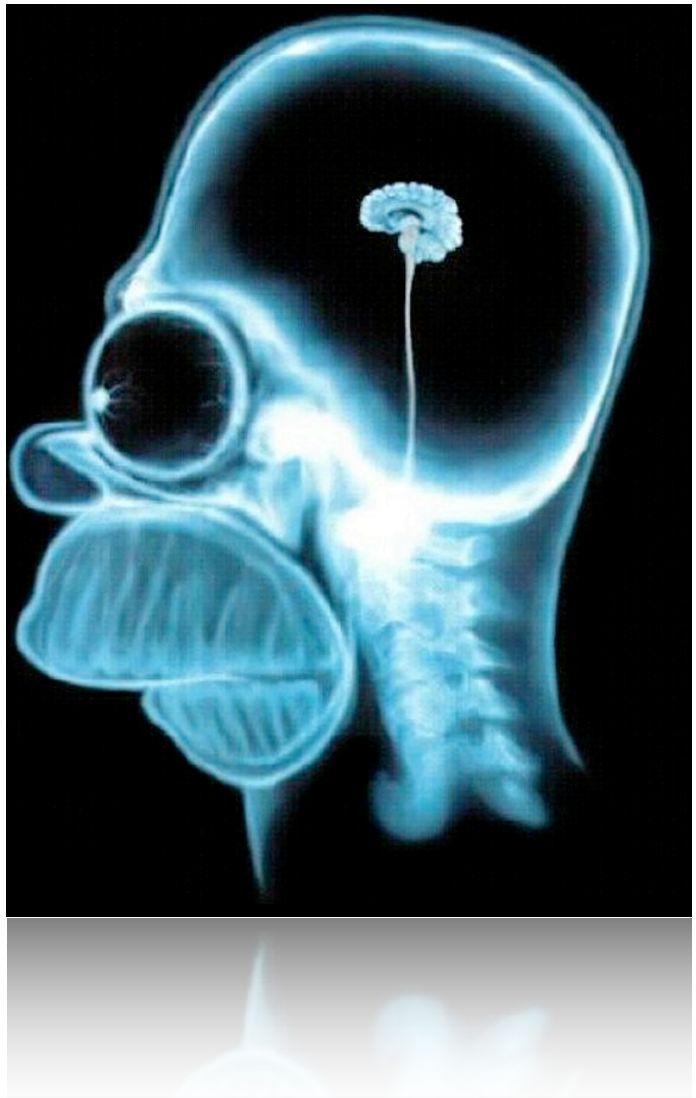
We are Visual Beings

70% of all brain inputs come through the eyes



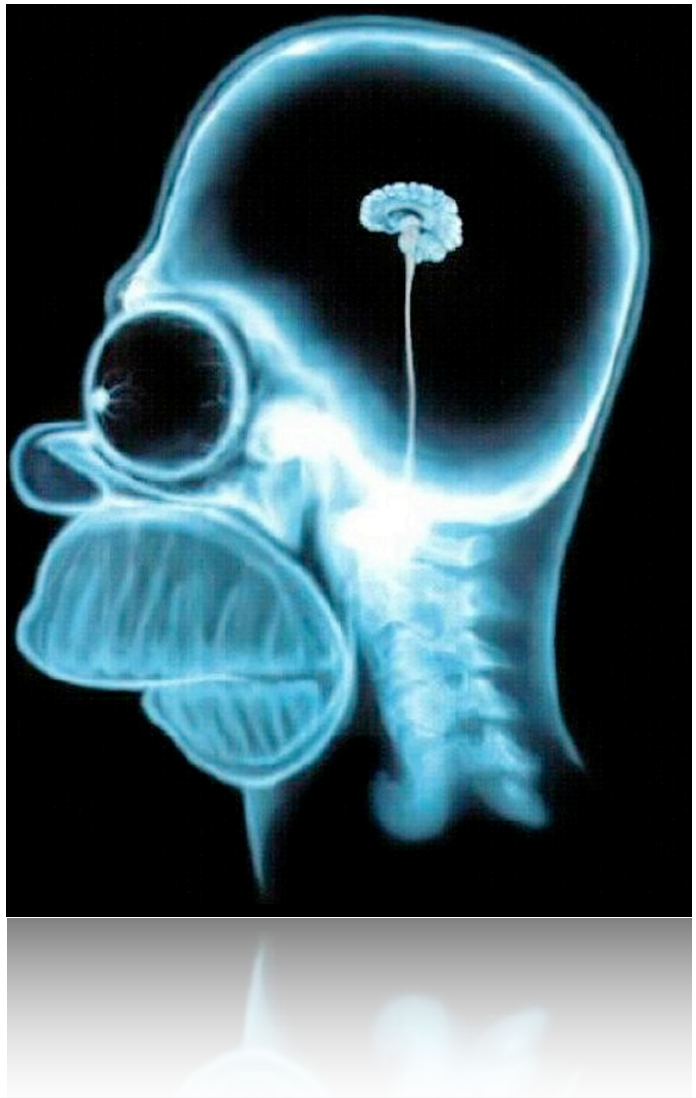
We see with our brain

- We have 3 types of memory that process visual information
  - *Iconic*, the visual sensory register
  - *Short-term*, the brain's working memory
  - *Long-term*



We see with our brain

- We have 3 types of memory that process visual information
  - ***Iconic*, the visual sensory register**
  - ***Short-term*, the brain's working memory**
  - *Long-term*



Iconic and Short-term Memory

- Iconic memory is a “buffer” that retains information for less than 1 second before passing it to short-term memory
  - Perception is *very fast, automatic & subconscious*, therefore called **pre-attentive**
- Short-term memory processes information in the form of “chunks”
  - Temporary, a couple of seconds
  - Limited capacity: 3 - 9 chunks

Exemplifying Pre-attentive Processing

8789364082376403128764532984732984732094873290845
389274-0329874-32874-2319847509834098340983240983 2
049823-0984903281453209481-0839393947896387436598

8789364082376403128764532984732984732094873290845
389274-0329874-32874-2319847509834098340983240983 2
049823-0984903281453209481-0839393947896387436598

Preattentive Attributes Examples

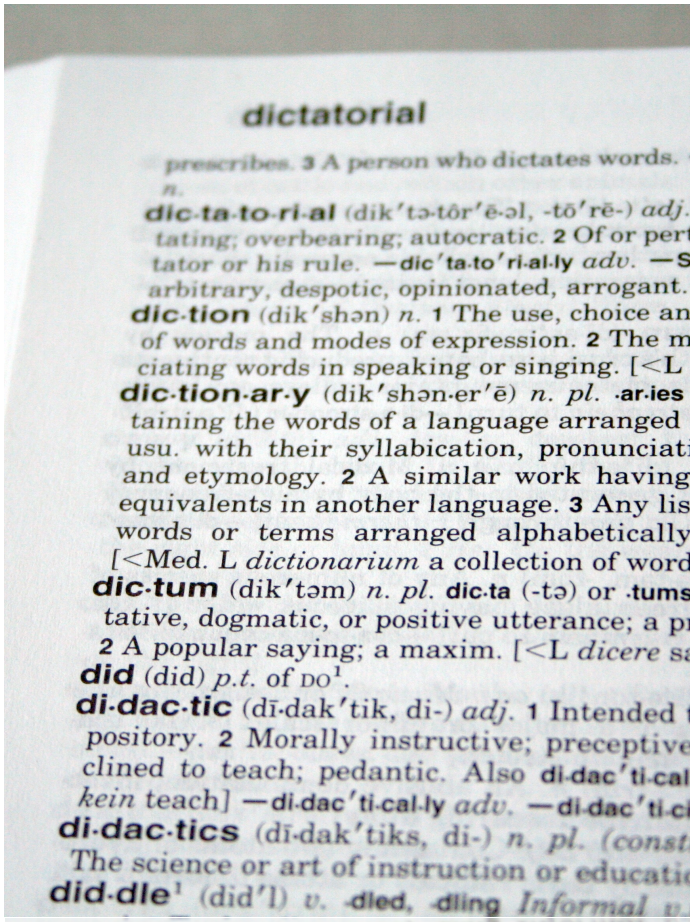
Orientation Line Length Line Width Size

Shape Curvature Added Marks Enclosure

Software Visualization

*"**The use of** the crafts of typography, graphic design, animation, and cinematography with modern human-computer interaction and **computer graphics** technology **to facilitate** both **the** human **understanding** and effective use **of** computer **software**."*

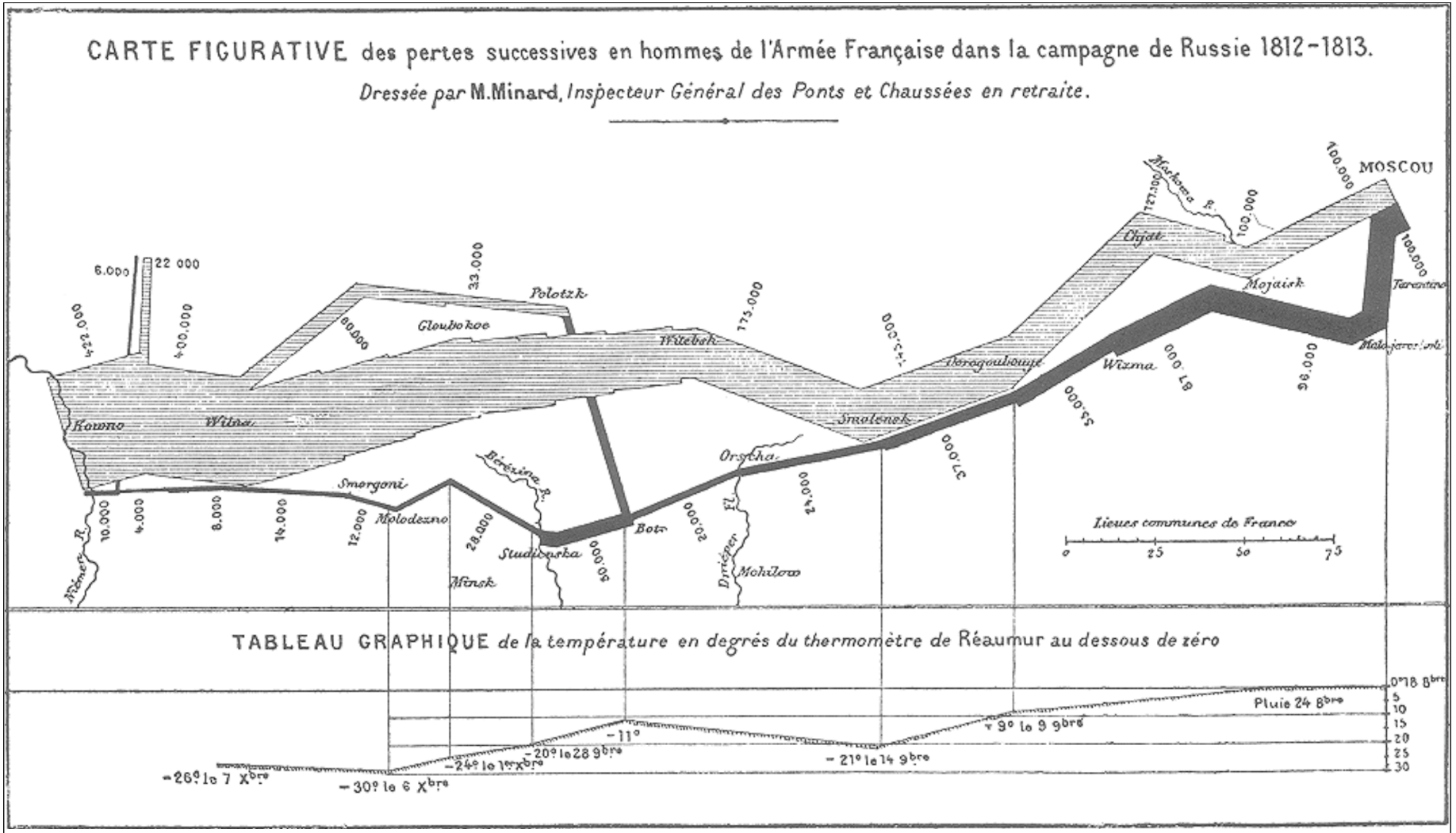
John Stasko, 1998



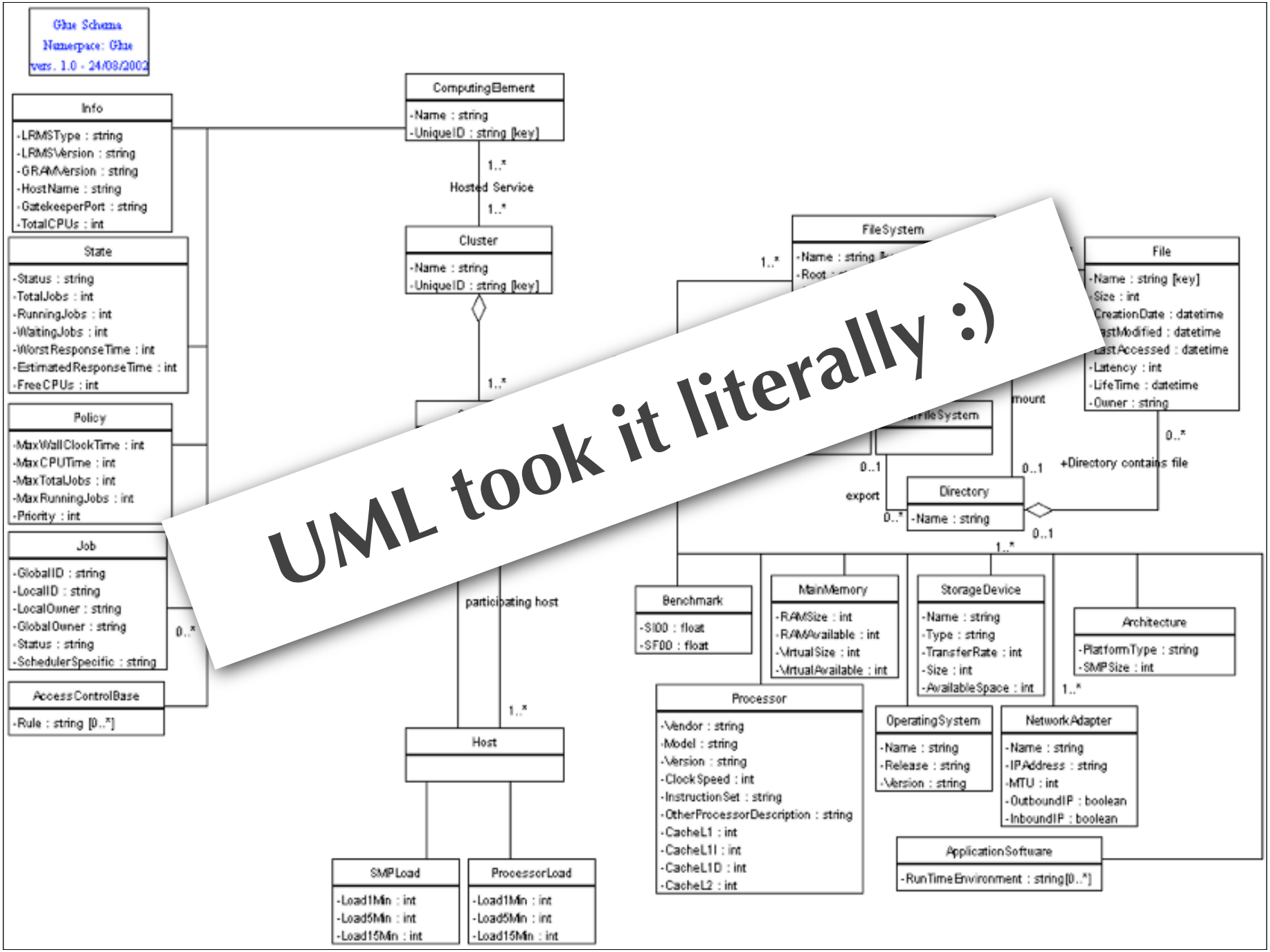

not software visualization


Visualization is about stories

1854, London, cholera epidemic



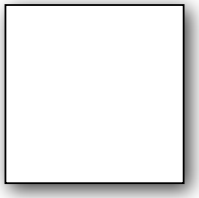
A Picture is Worth a Thousand Words

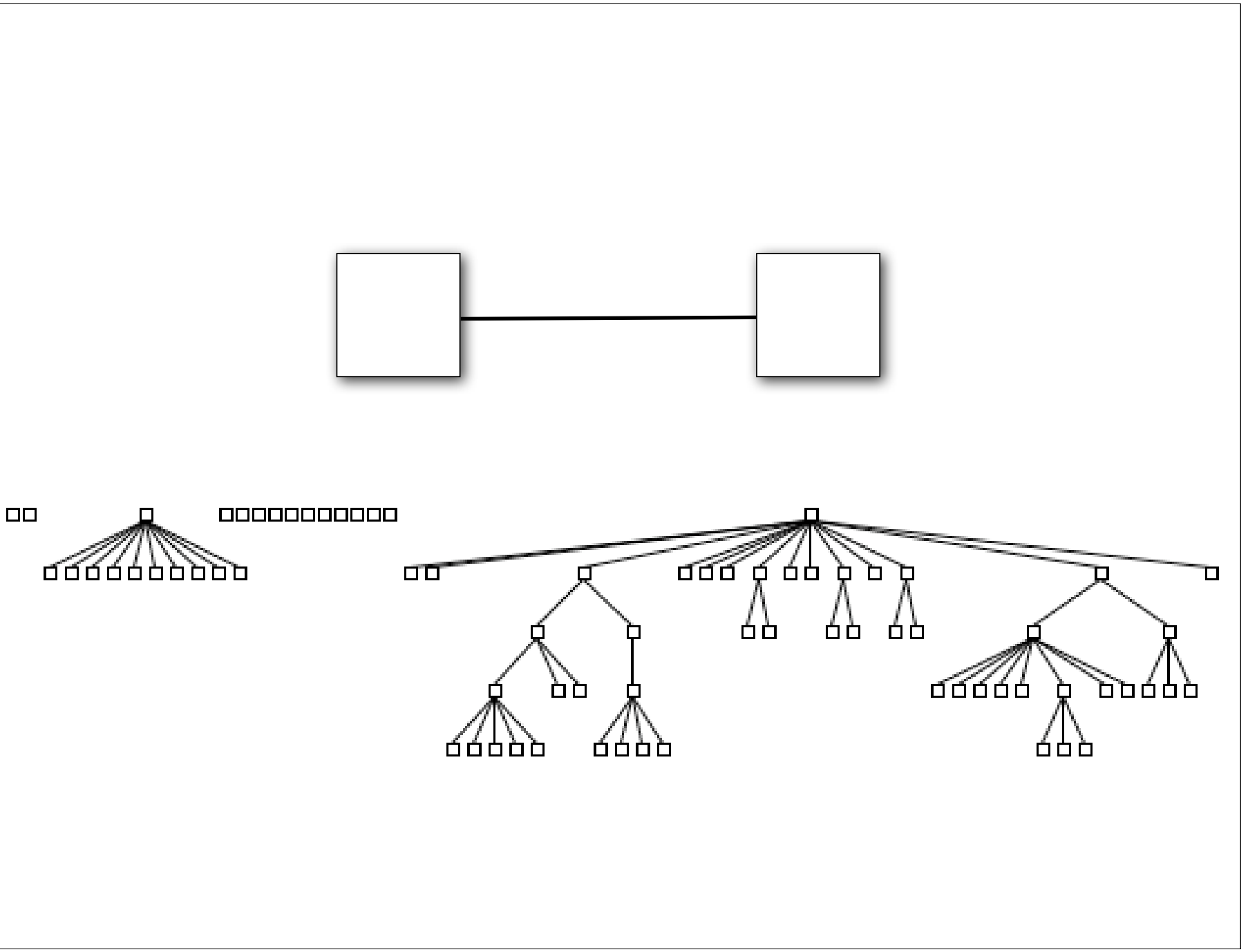

Glue Schema
Namespace: Glue
vers. 1.0 - 24/08/2002
Info
-LRMSType : string
-LRMSVersion : string
-GRAMVersion : string
-HostName : string
-GatekeeperPort : string
-TotalCPUs : int
ComputingElement
-Name : string
-UniqueID : string [key]
1..*
Hosted Service
1..*
Cluster
-Name : string
-UniqueID : string [key]
State
-Status : string
-TotalJobs : int
-RunningJobs : int
-WaitingJobs : int
-WorstResponseTime : int
-EstimatedResponseTime : int
-FreeCPUs : int
Policy
-MaxWallClockTime : int
-MaxCPUTime : int
-MaxTotalJobs : int
-MaxRunningJobs : int
-Priority : int
Job
-GlobalID : string
-LocalID : string
-LocalOwner : string
-GlobalOwner : string
-Status : string
-SchedulerSpecific : string
0..*
AccessControlBase
-Rule : string [0..*]
participating host
1..*
Host
SMPLoad
-Load1Min : int
-Load5Min : int
-Load15Min : int
ProcessorLoad
-Load1Min : int
-Load5Min : int
-Load15Min : int
FileSystem
-Name : string
-Root :
1..*
mount
0..1
0..1
export
0..*
Directory
-Name : string
0..1
+Directory contains file
0..*
File
-Name : string [key]
-Size : int
-CreationDate : datetime
-LastModified : datetime
-LastAccessed : datetime
-Latency : int
-LifeTime : datetime
-Owner : string
1..*
Benchmark
-SI00 : float
-SF00 : float
MainMemory
-RAMSize : int
-RAMAvailable : int
-VirtualSize : int
-VirtualAvailable : int
StorageDevice
-Name : string
-Type : string
-TransferRate : int
-Size : int
-AvailableSpace : int
Architecture
-PlatformType : string
-SMPSize : int
1..*
Processor
-Vendor : string
-Model : string
-Version : string
-ClockSpeed : int
-InstructionSet : string
-OtherProcessorDescription : string
-CacheL1 : int
-CacheL1I : int
-CacheL1D : int
-CacheL2 : int
OperatingSystem
-Name : string
-Release : string
-Version : string
NetworkAdapter
-Name : string
-IPAddress : string
-MTU : int
-OutboundIP : boolean
-InboundIP : boolean
ApplicationSoftware
-RunTimeEnvironment : string[0..*]
UML took it literally :)

Software is intangible,
having no physical shape or size.

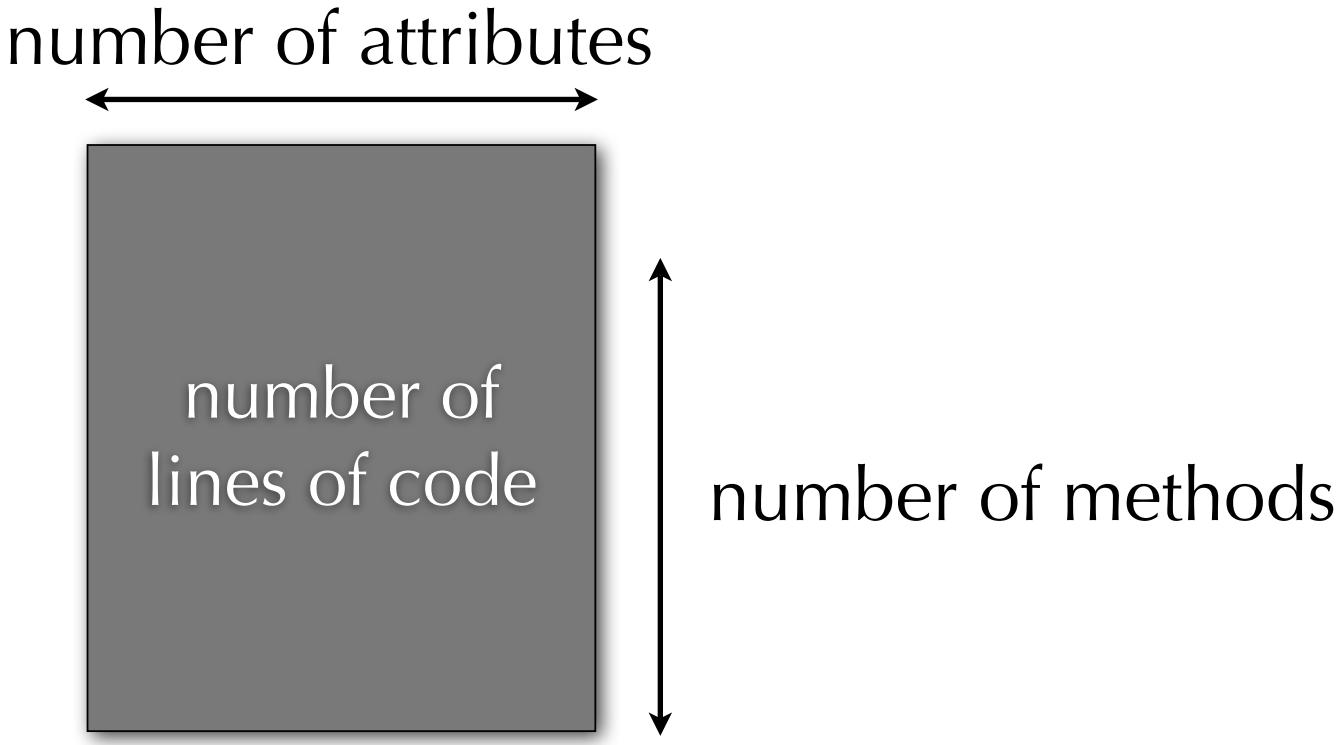
Thomas Ball, 1996

simple is beautiful

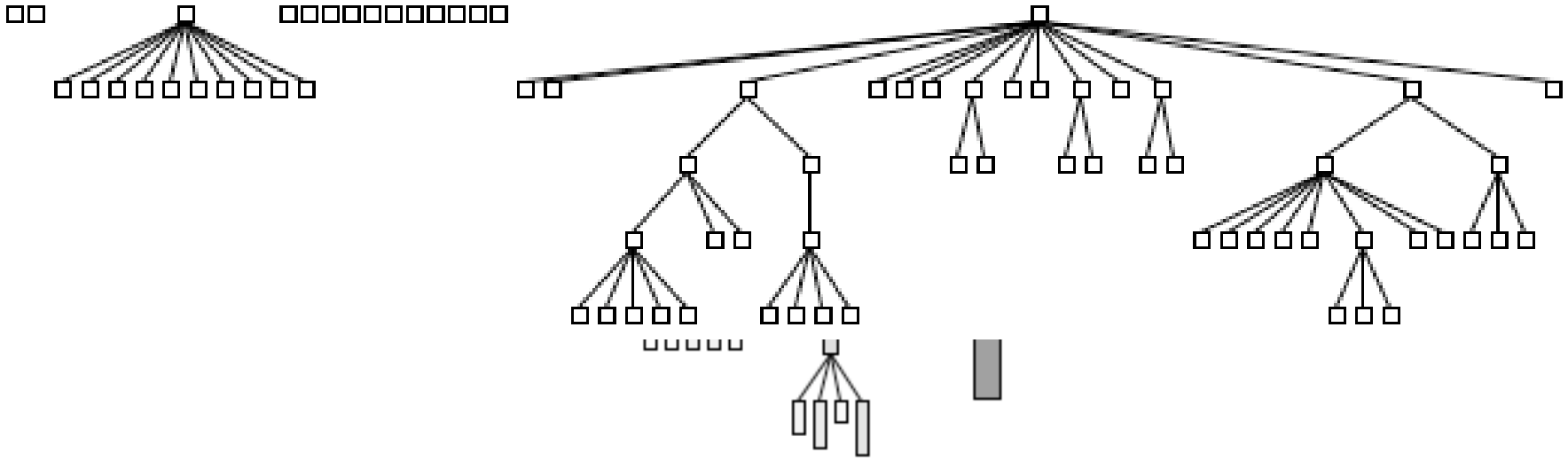




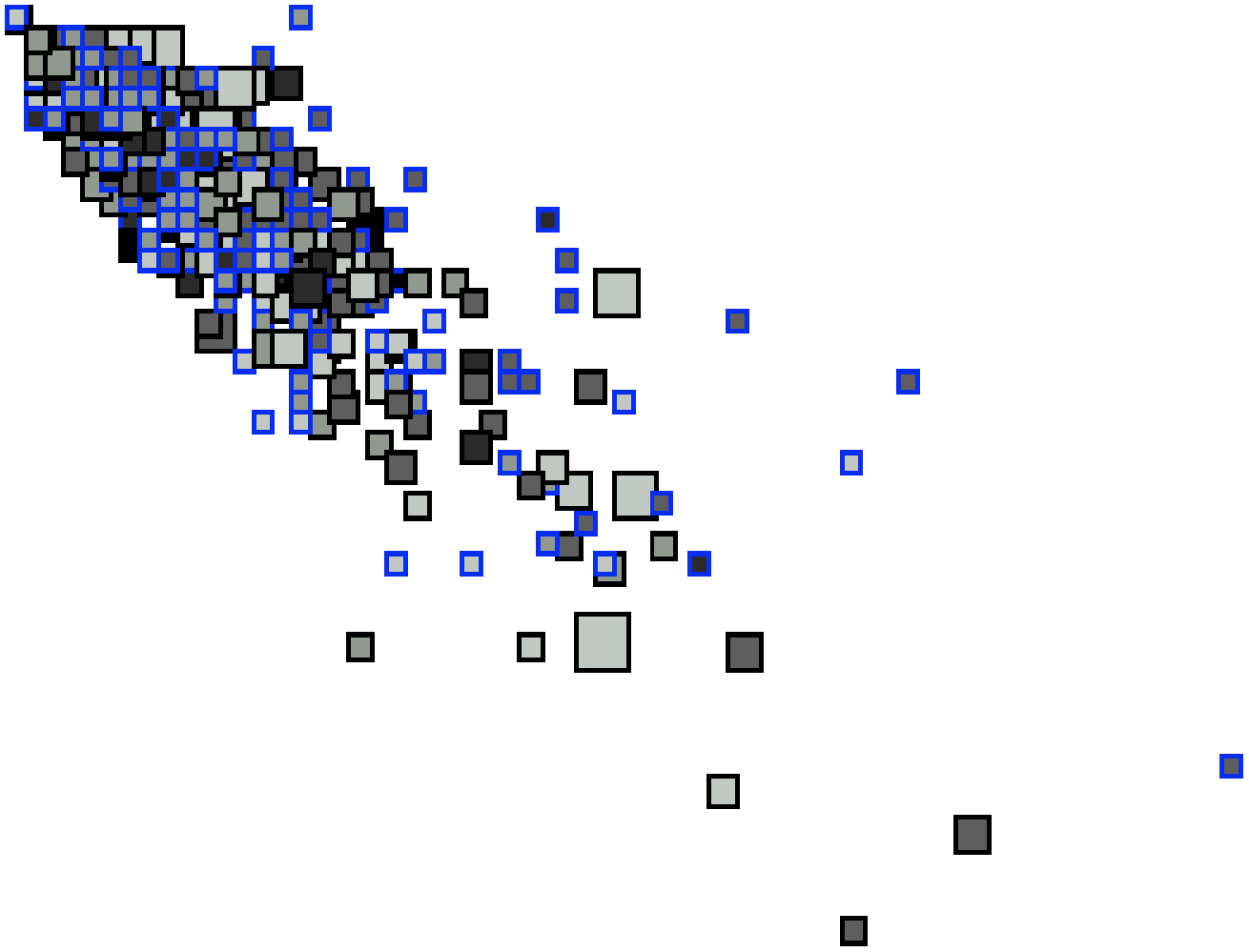
The Polymetric View Principle

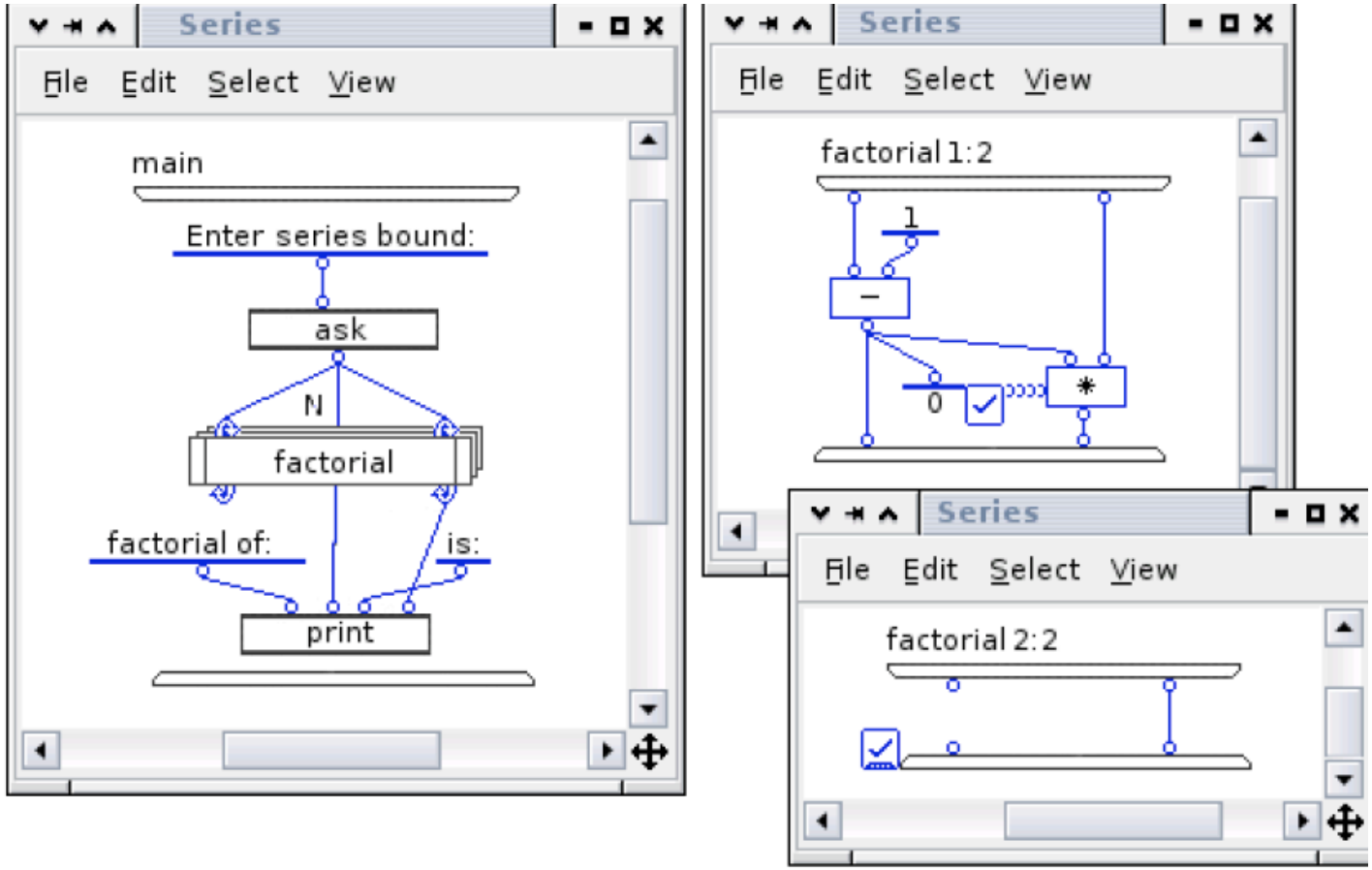


System Complexity View

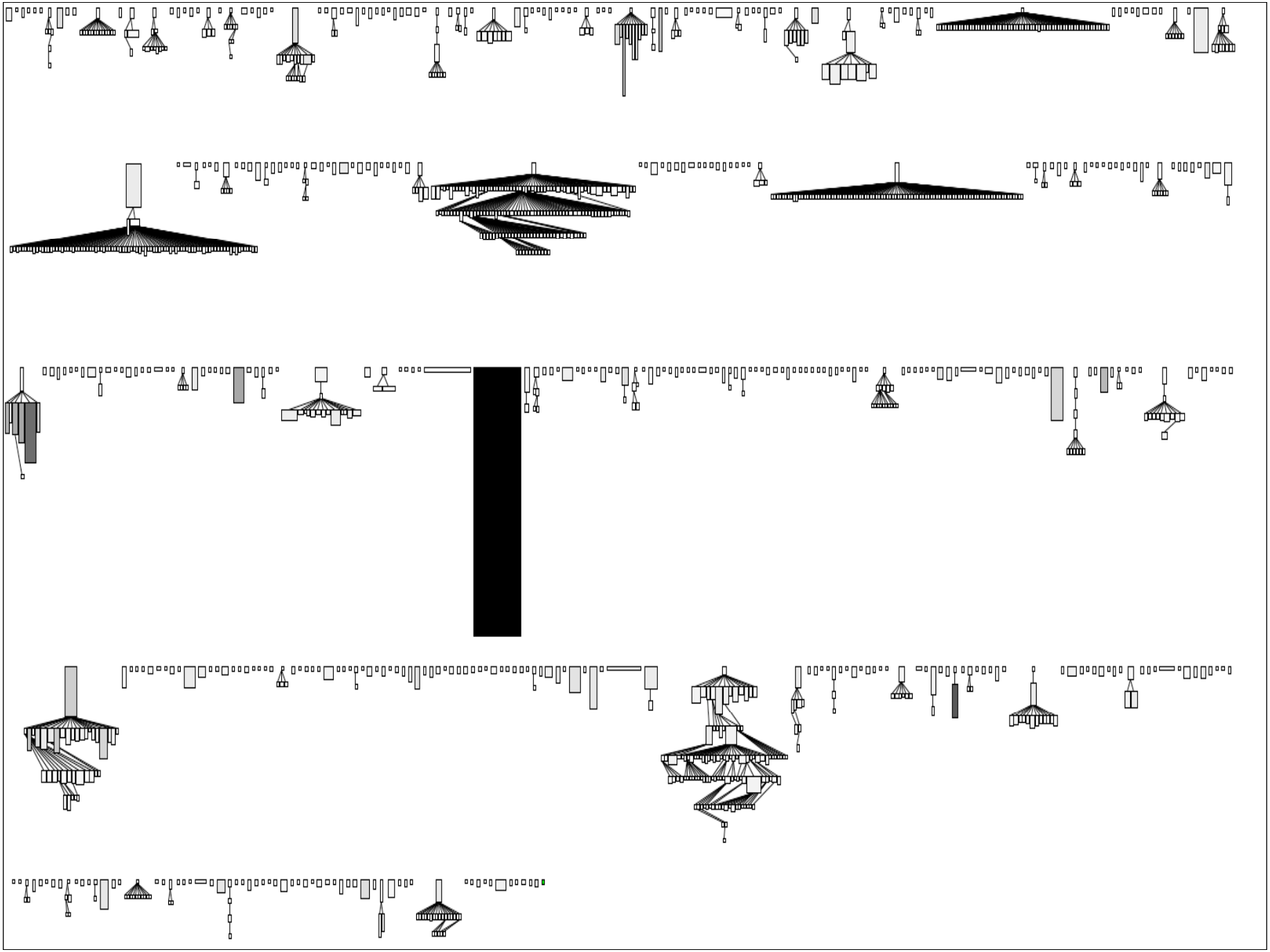


a simple and powerful concept

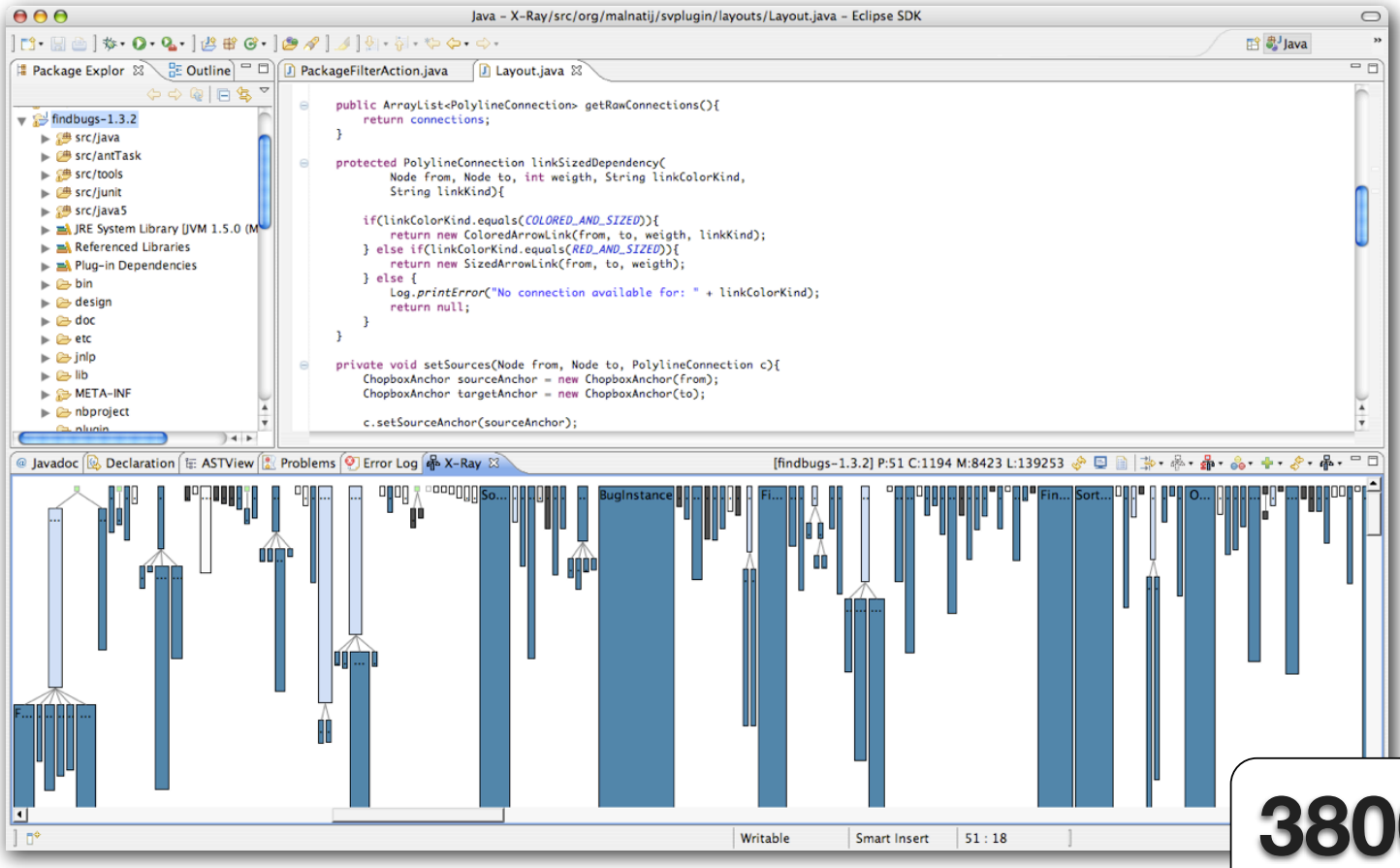




Software Visualization is not Visual Programming



http://atelier.inf.unisi.ch/~malnatij/xray.php



Released:
Nov 2007

3800 +
downloads

free

Where’s the Beauty?

Part 2

Software is beautiful


The best defense is attack

Richard Wettel, PhD student

- R. Wettel, M. Lanza; **Program Comprehension through Software Habitability**. In Proceedings of ICPC 2007 (15th International Conference on Program Comprehension), pp. 231 - 240, IEEE CS Press, 2007

- R. Wettel, M. Lanza; **Visualizing Software Systems as Cities**. In Proceedings of VISSOFT 2007 (4th International Workshop on Visualizing Software for Understanding and Analysis), pp. 92 - 99, IEEE CS Press, 2007

- R. Wettel, M. Lanza; **Visually Localizing Design Problems with Disharmony Maps**. In Proceedings of Softvis 2008 (4th International ACM Symposium on Software Visualization), pp. 155 - 164, ACM Press, 2008.

- R. Wettel, M. Lanza; **Visual Exploration of Large-scale System Evolution**. In Proceedings of WCRE 2008 (15th Working Conference on Reverse Engineering), to be published, IEEE CS Press, 2008

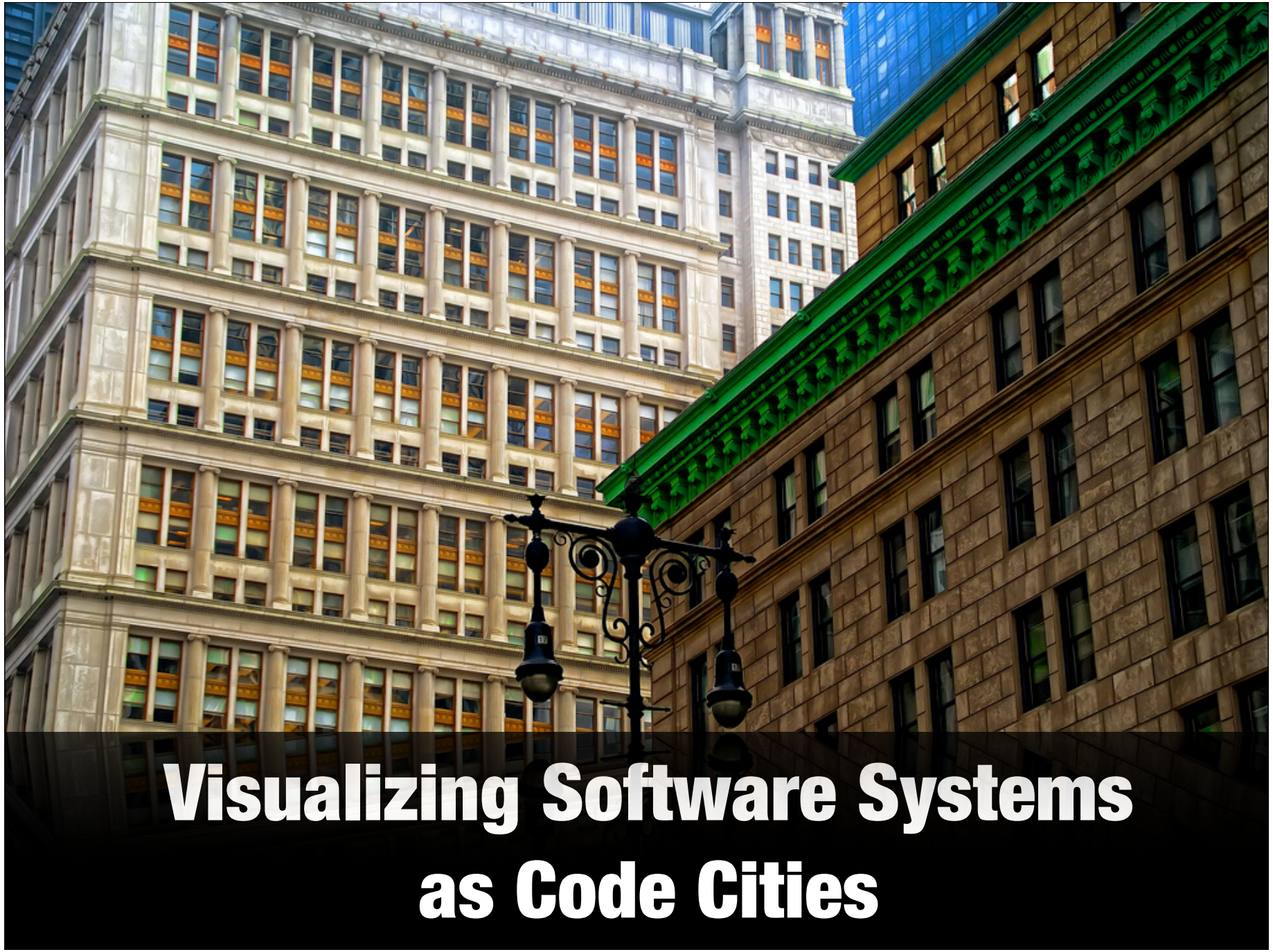
How can we solve Ball's dilemma?

Metaphors..

Habitability is the characteristic of source code that enables programmers, coders, bug-fixers, and people coming to the code later in its life to understand its construction and intentions and to change it comfortably and confidently.

Richard Gabriel

On “Habitability and Piecemeal Growth”; in “Patterns of Software”

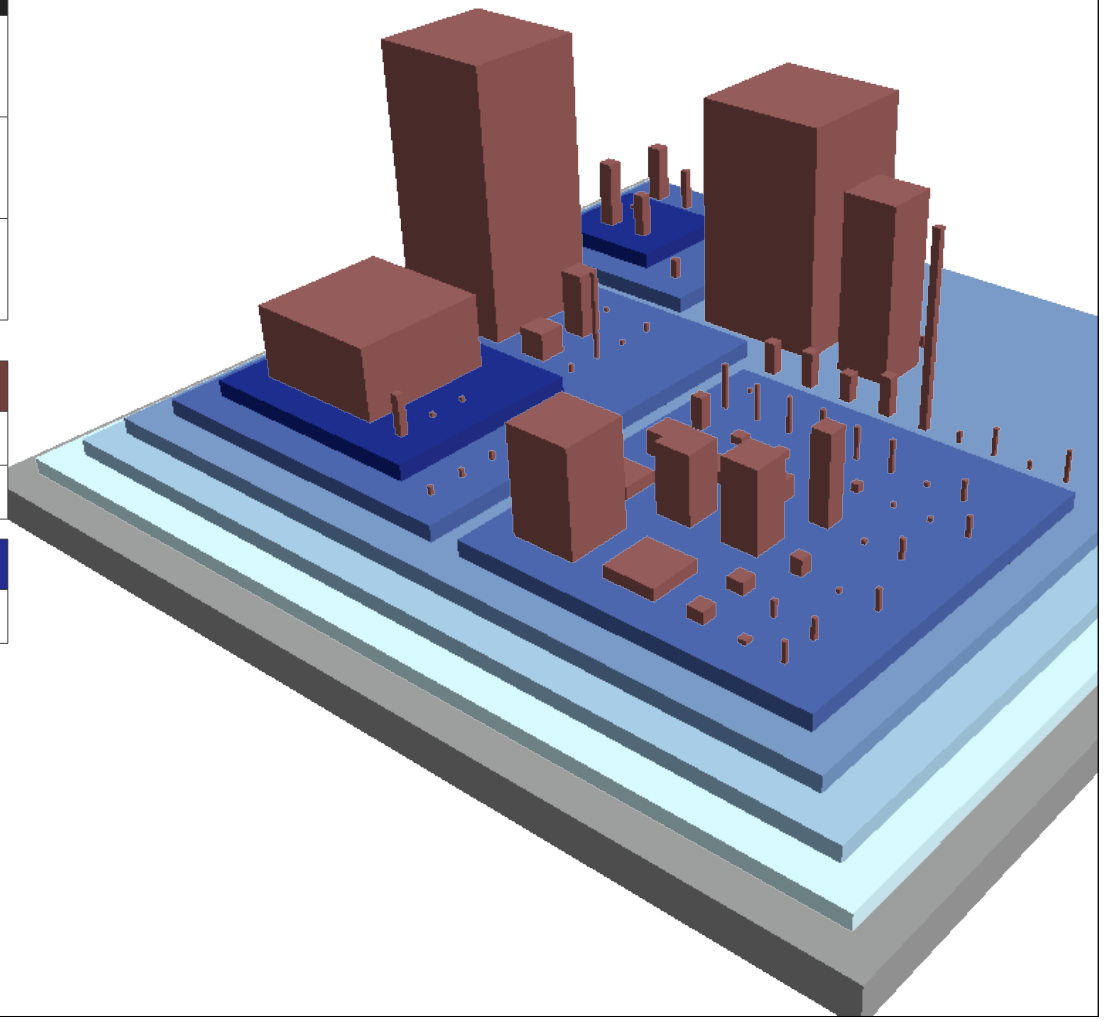

Visualizing Software Systems
as Code Cities

The City Metaphor

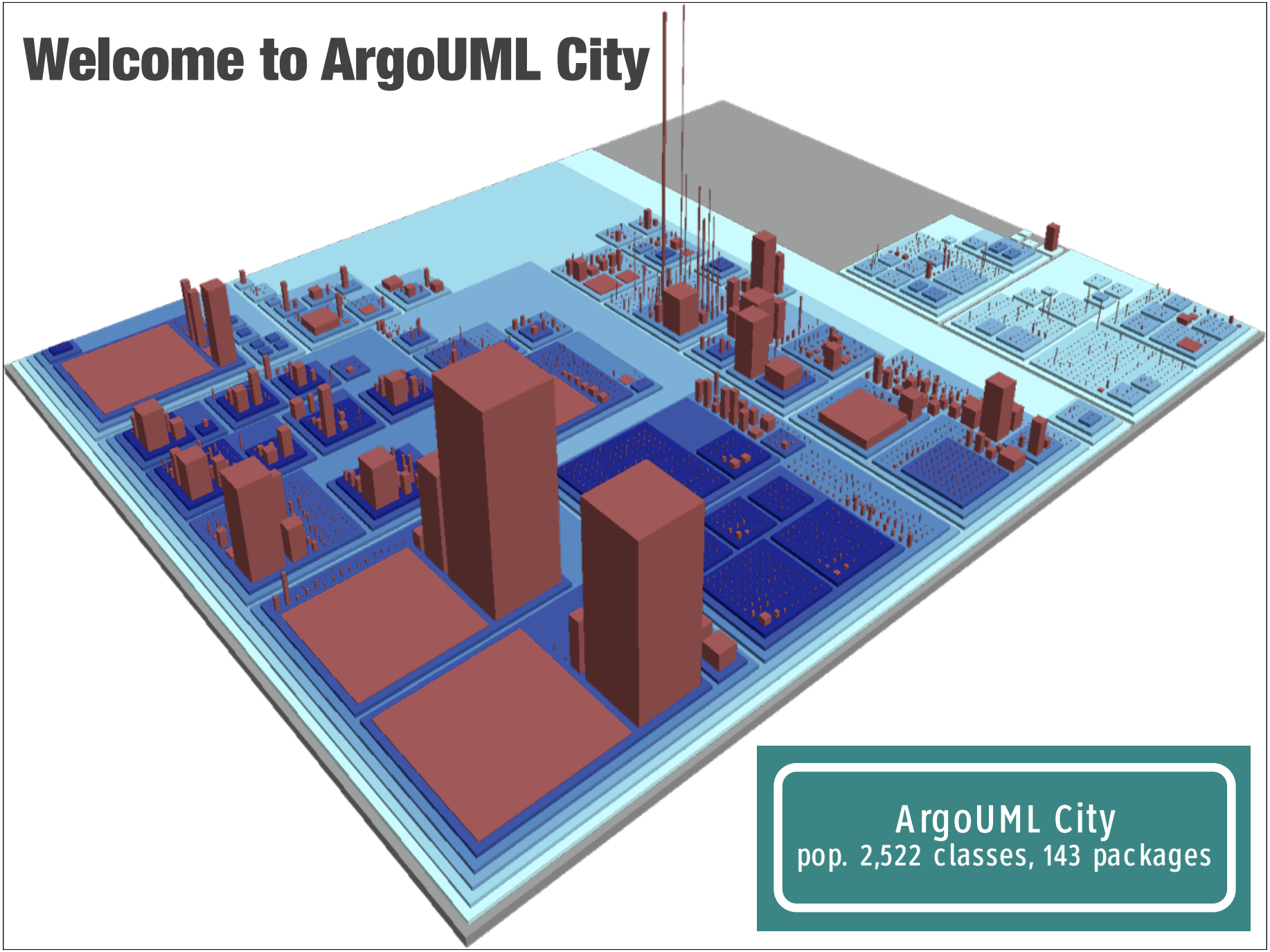
domain mapping	
classes	buildings
packages	districts
system	city

class metric	building property
number of methods (NOM)	height
number of attributes (NOA)	width, length

package metric	district property
nesting level	color

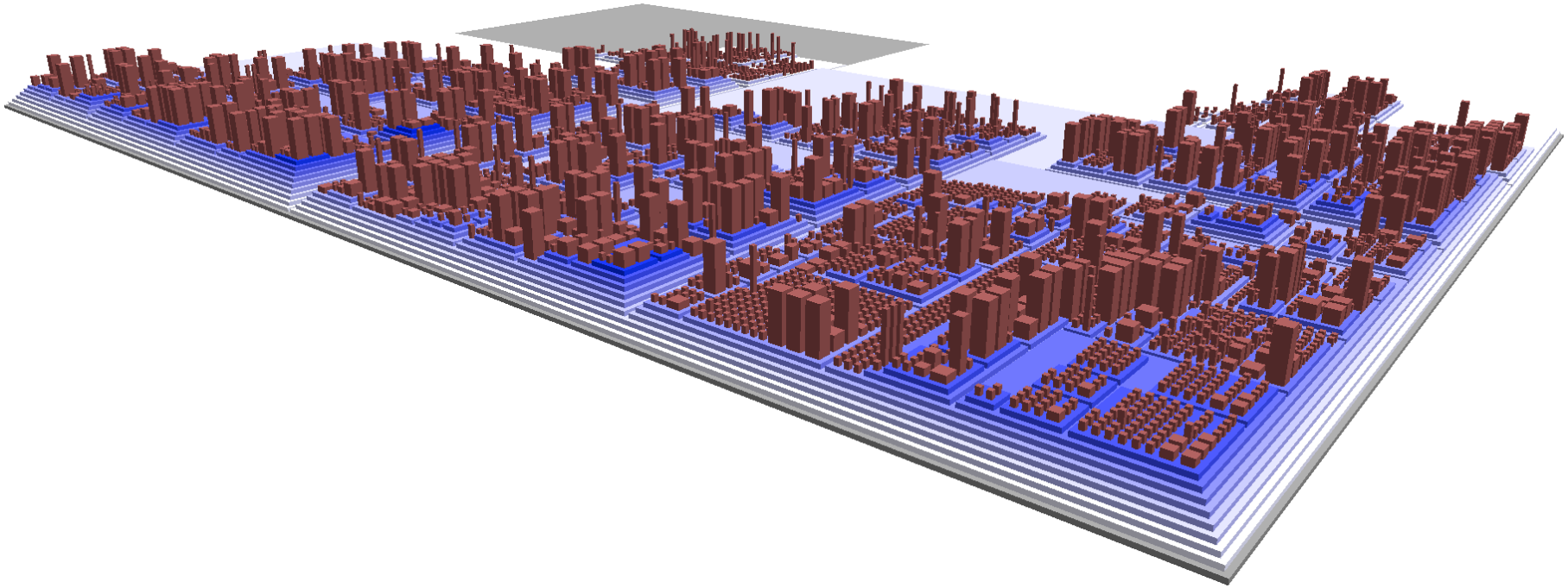


Welcome to ArgoUML City



ArgoUML City
pop. 2,522 classes, 143 packages

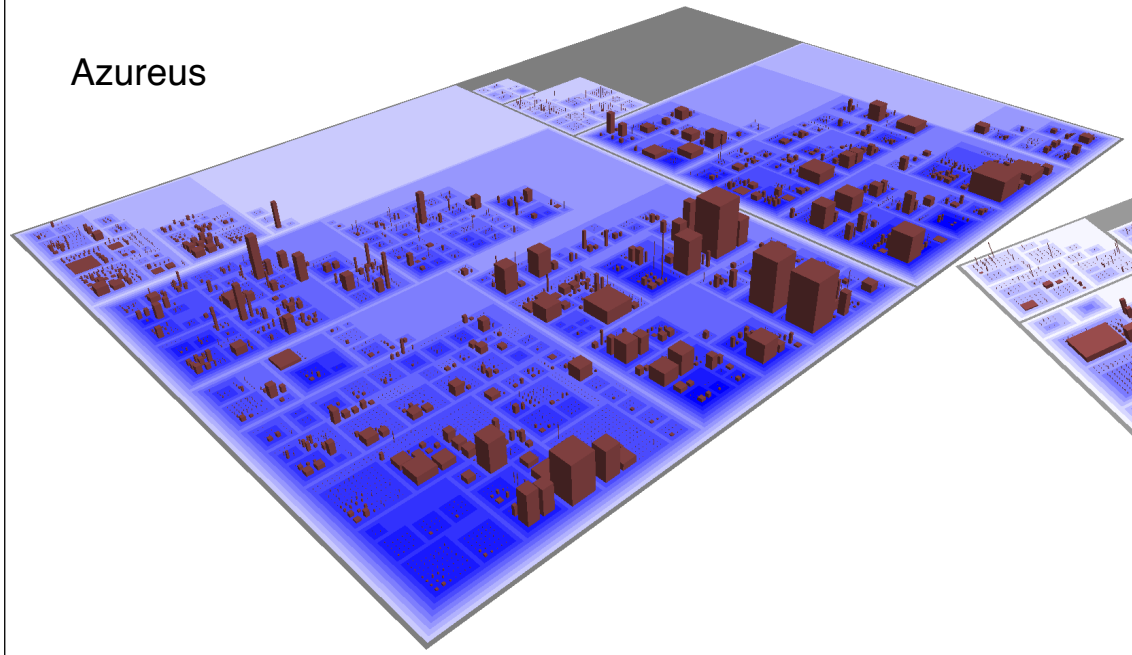
Software Topology



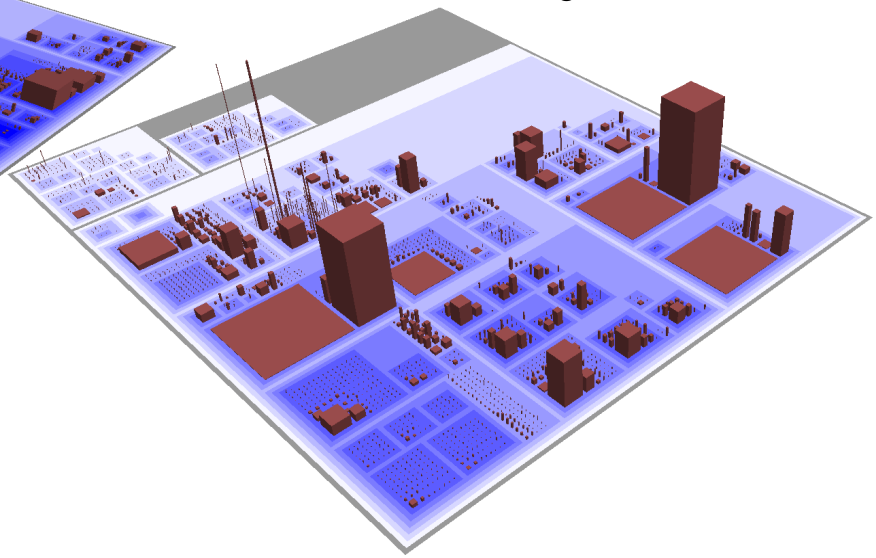
Azureus City
pop. 4'500+ classes

Crossing System Boundaries

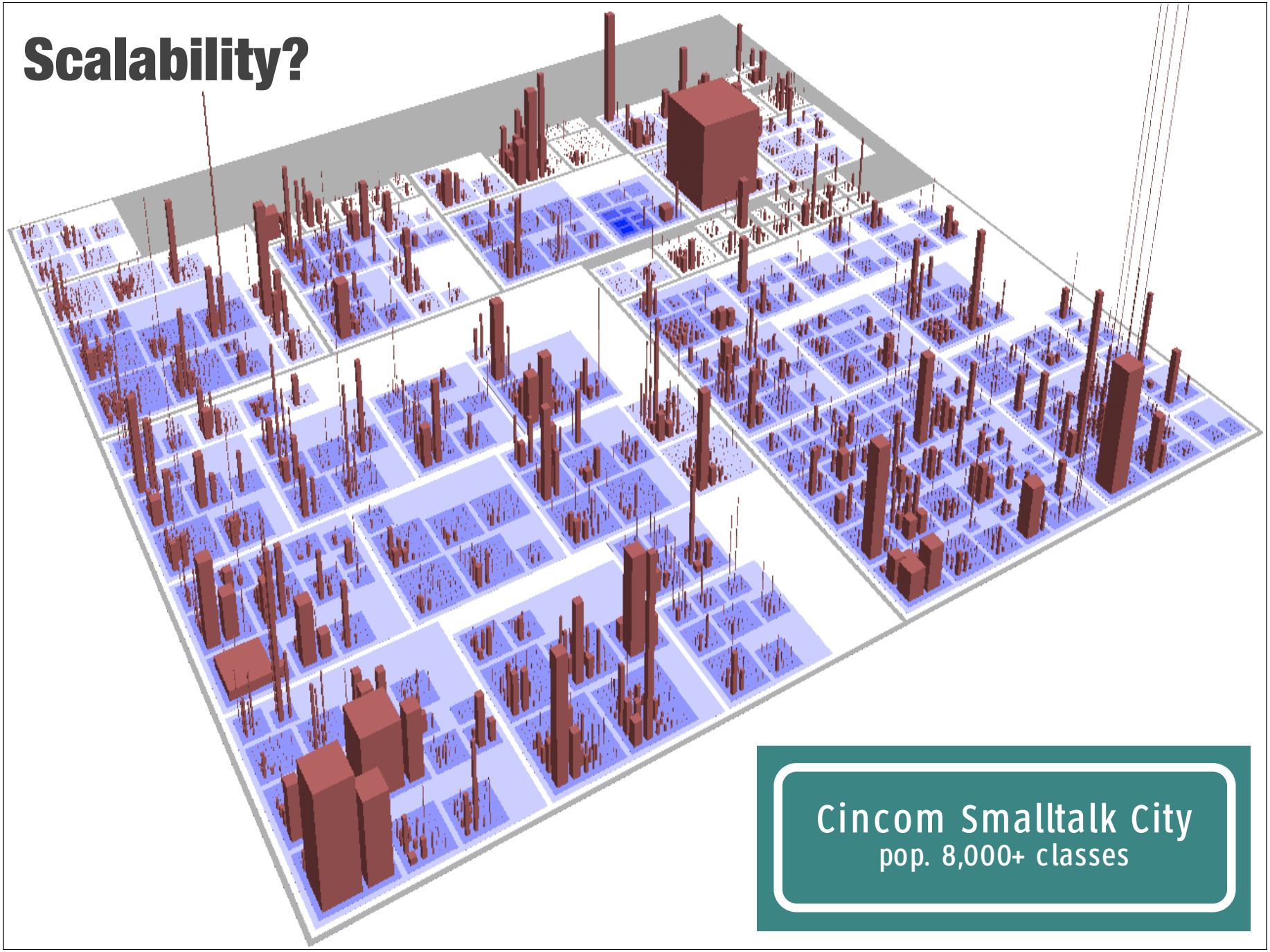
Azureus



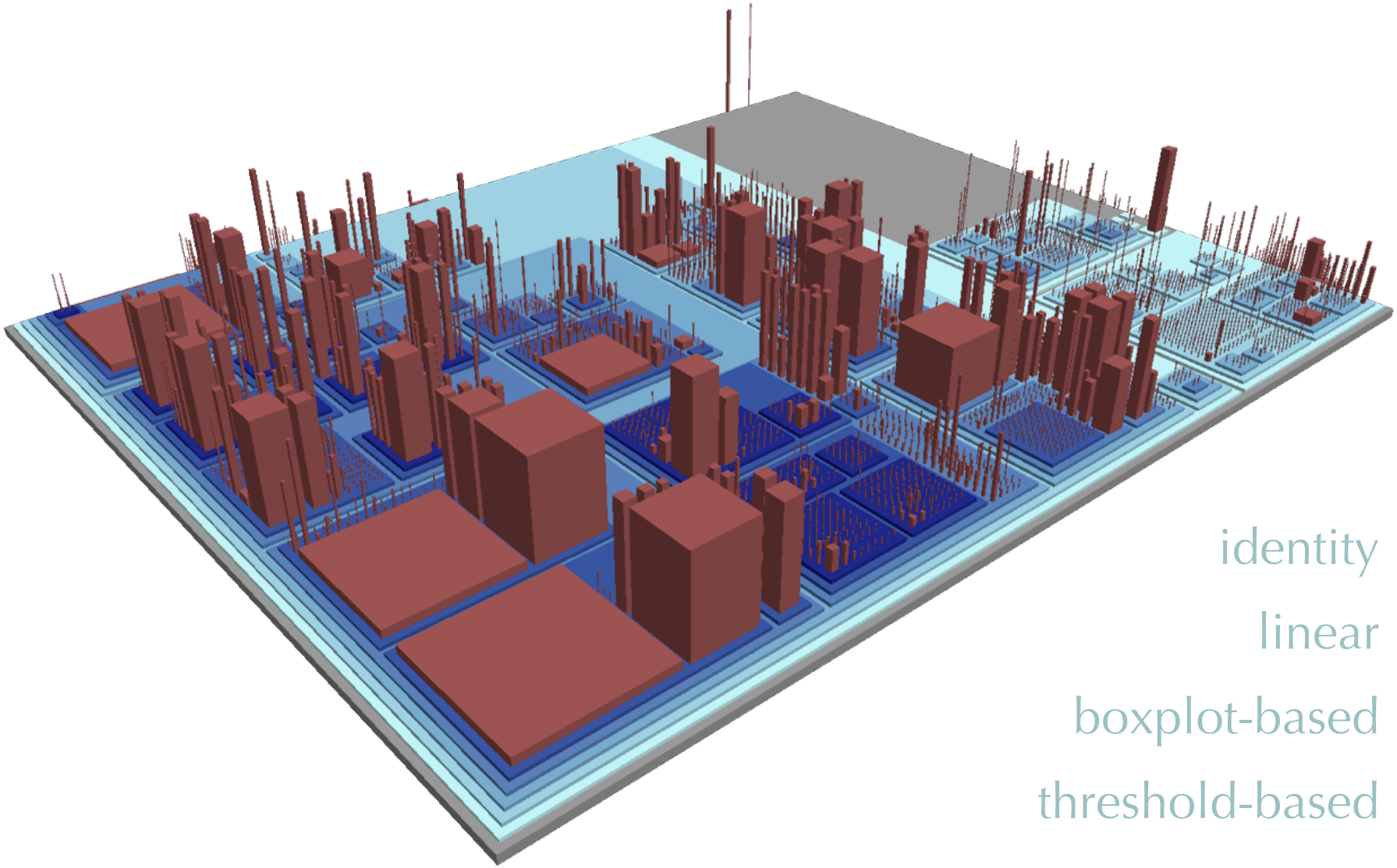
ArgoUML



Scalability?



Mapping Metrics

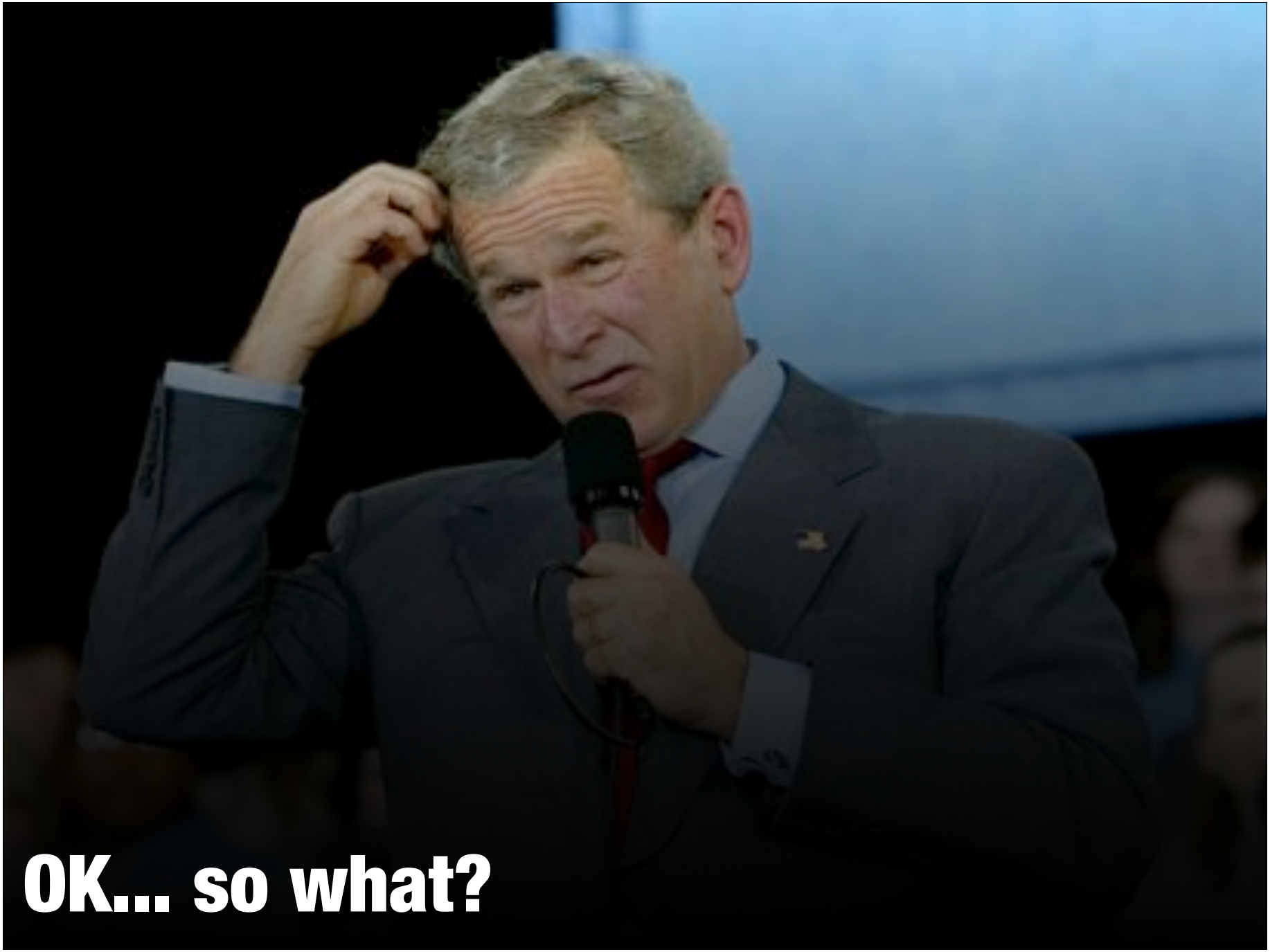


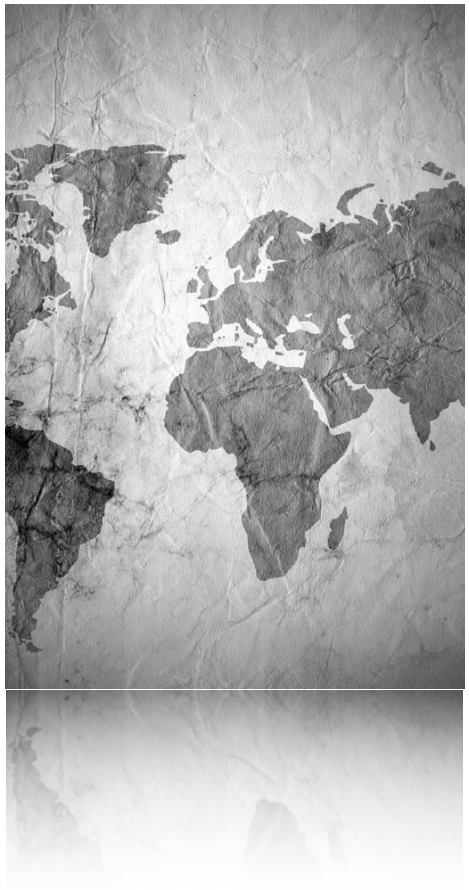
identity

linear

boxplot-based

threshold-based

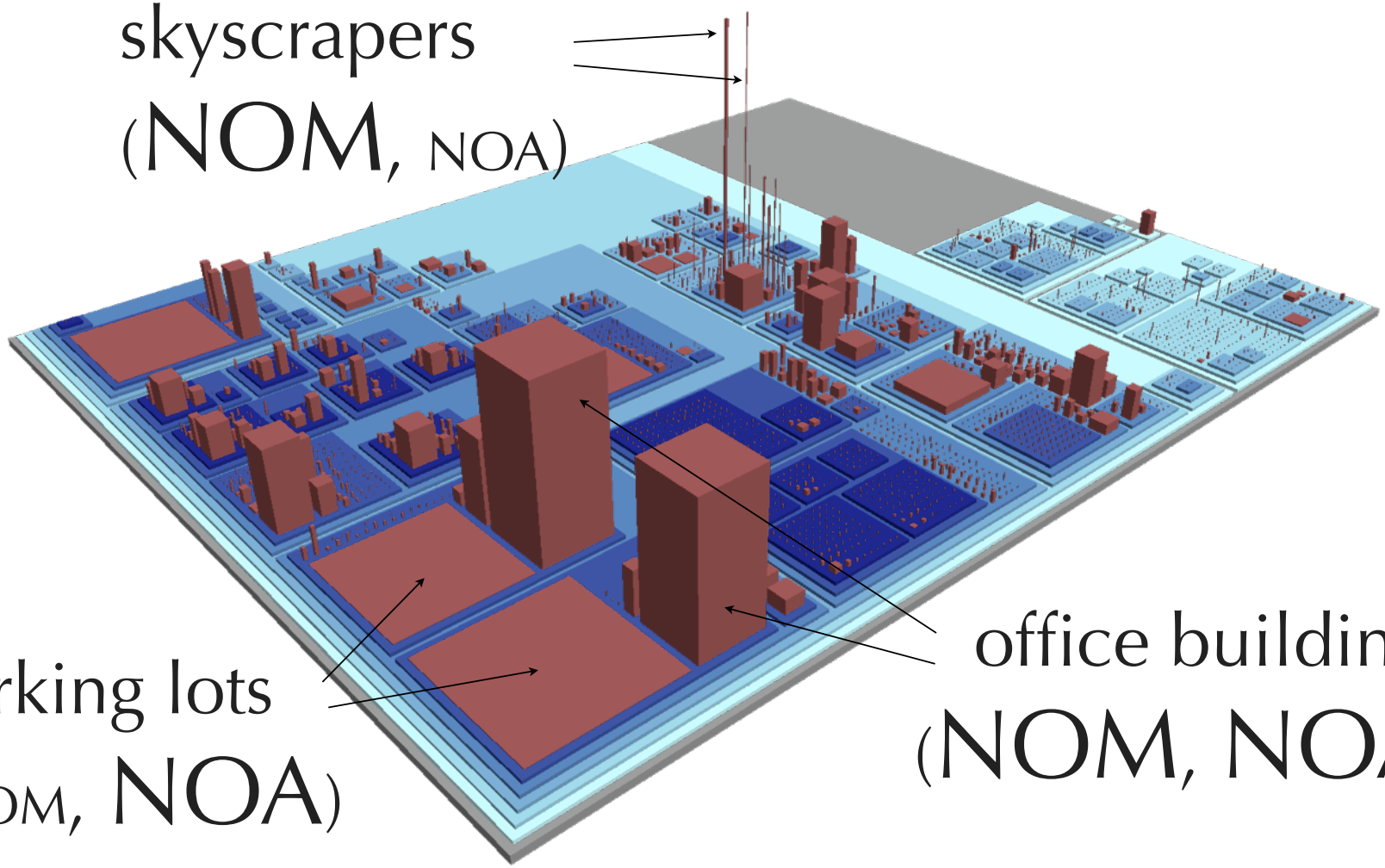

OK... so what?



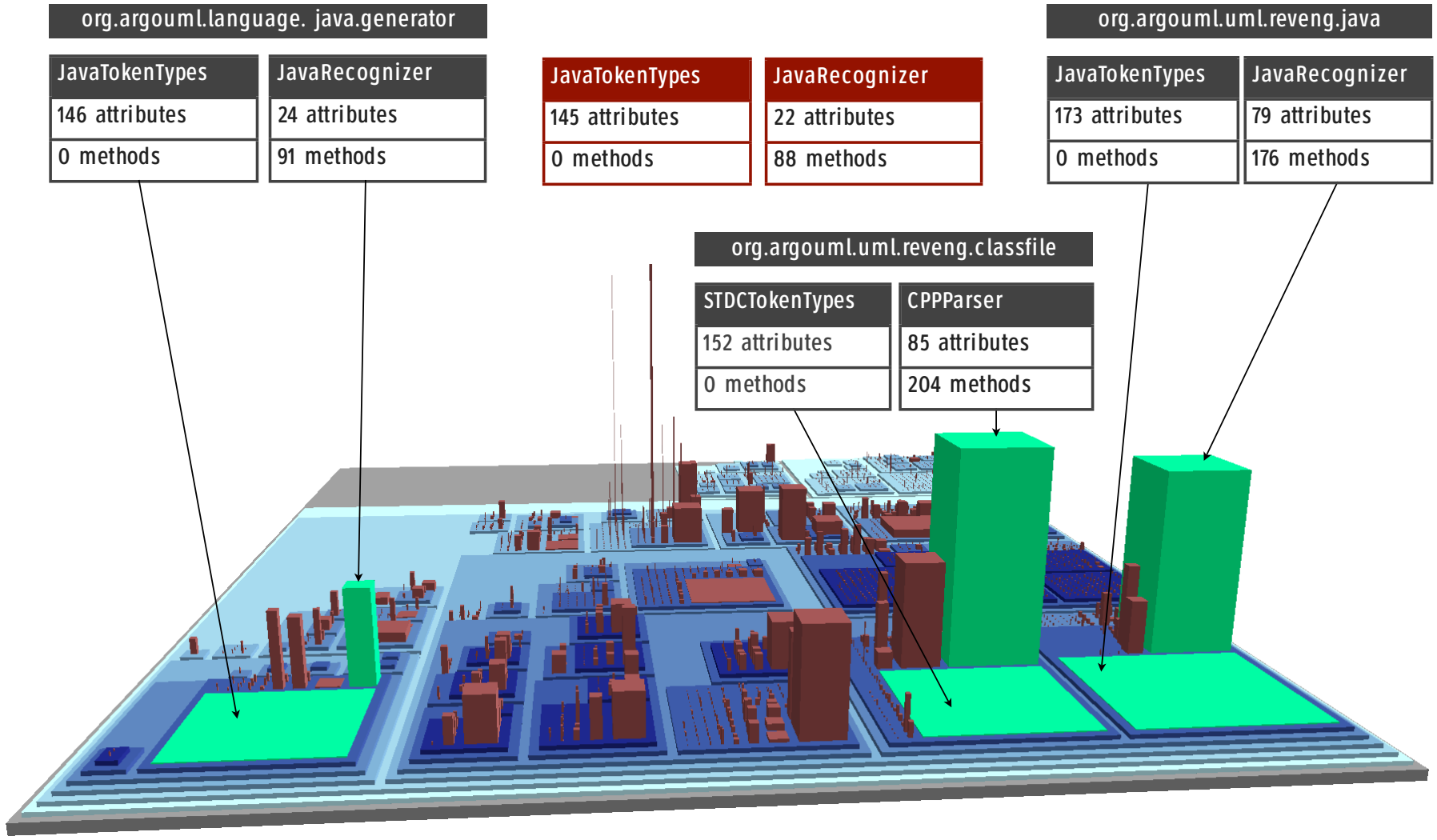
applications

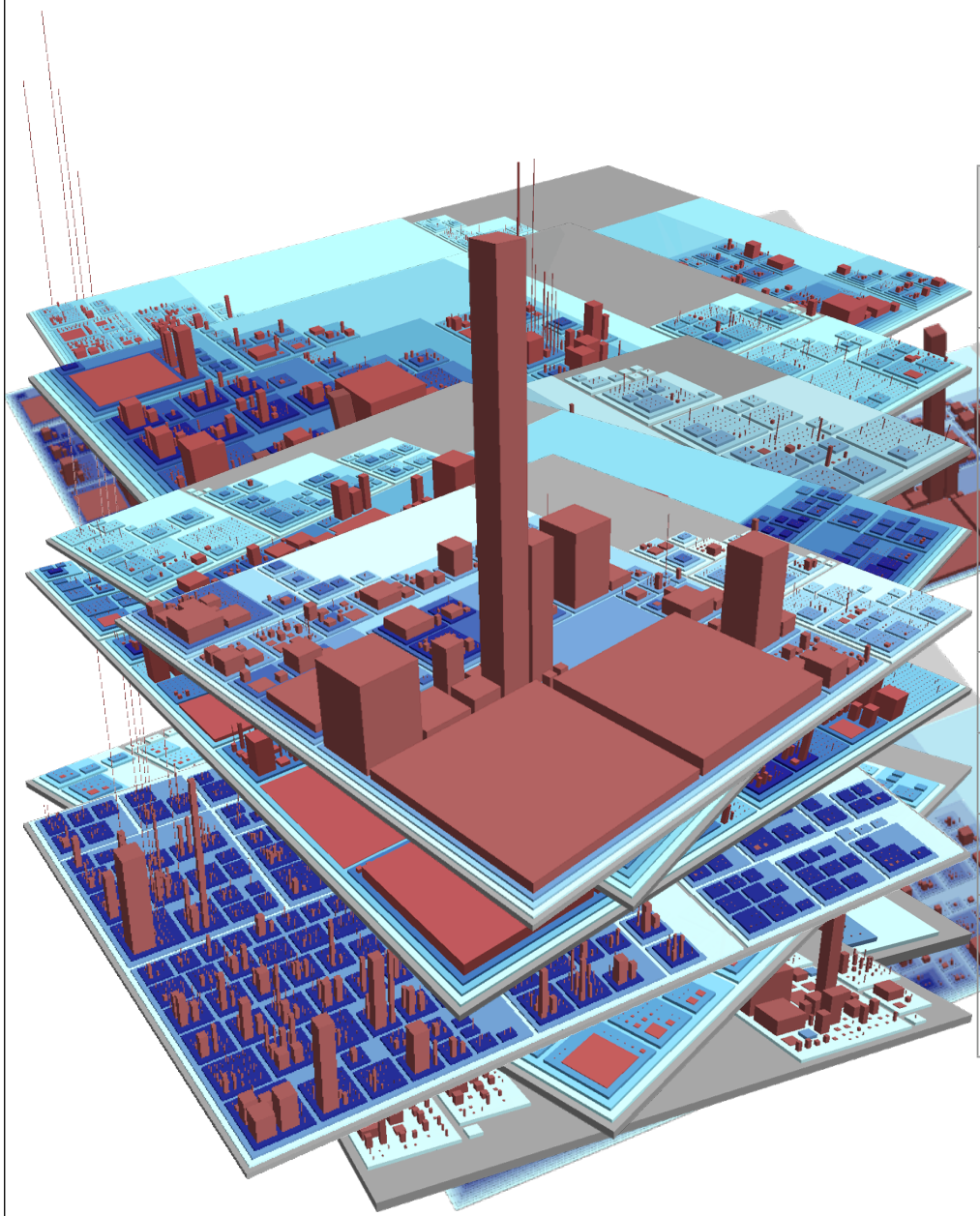
Large-scale Program Comprehension

City Lights..


skyscrapers
(NOM, NOA)
parking lots
(NOM, NOA)
office buildings
(NOM, NOA)

Sightseeing ArgoUML



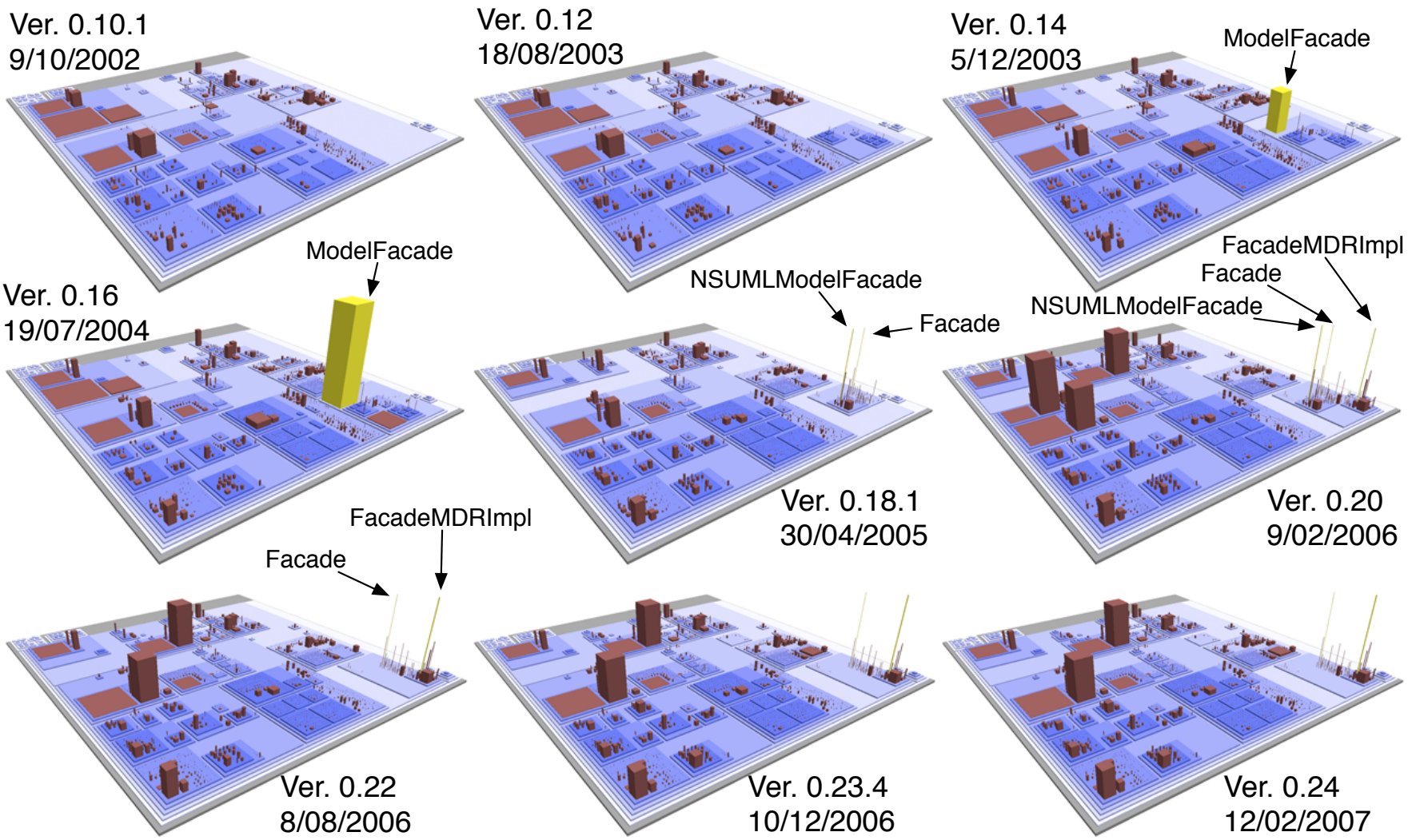


System	Language	NOP	NOC	kLOC
Azureus	Java	457	4734	274
ArgoUML	Java	144	2542	137
JHotDraw	Java	72	998	30
iText	Java	149	1250	80
Jmol	Java	105	1032	85
JDK 1.5 core	Java	137	4715	160
Moose	Smalltalk	278	961	32
Jun	Smalltalk	288	2236	351
CodeCity	Smalltalk	129	291	18
ScumVM	C++	18	1331	105

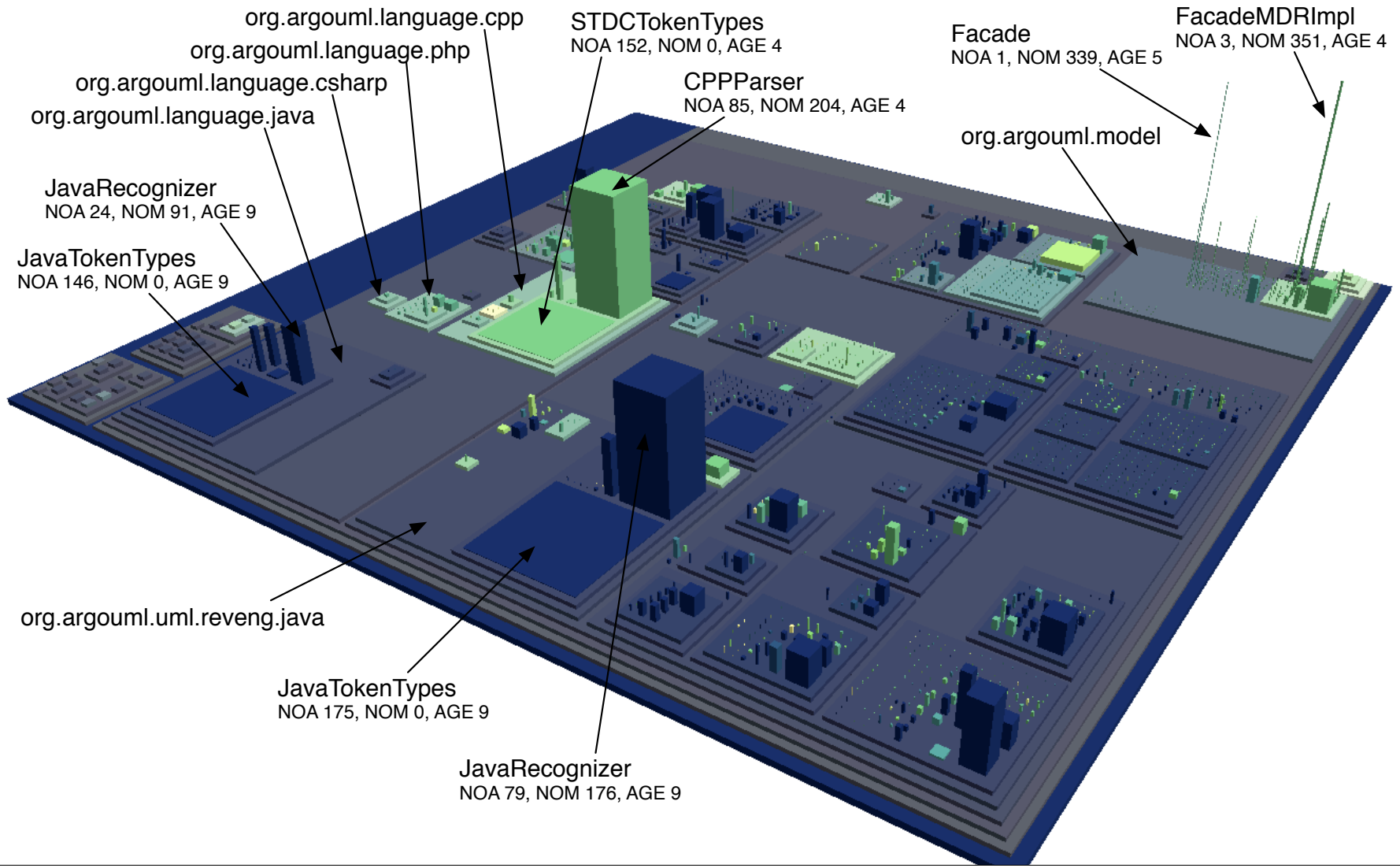
1,250,000 LOC

Evolution Analysis

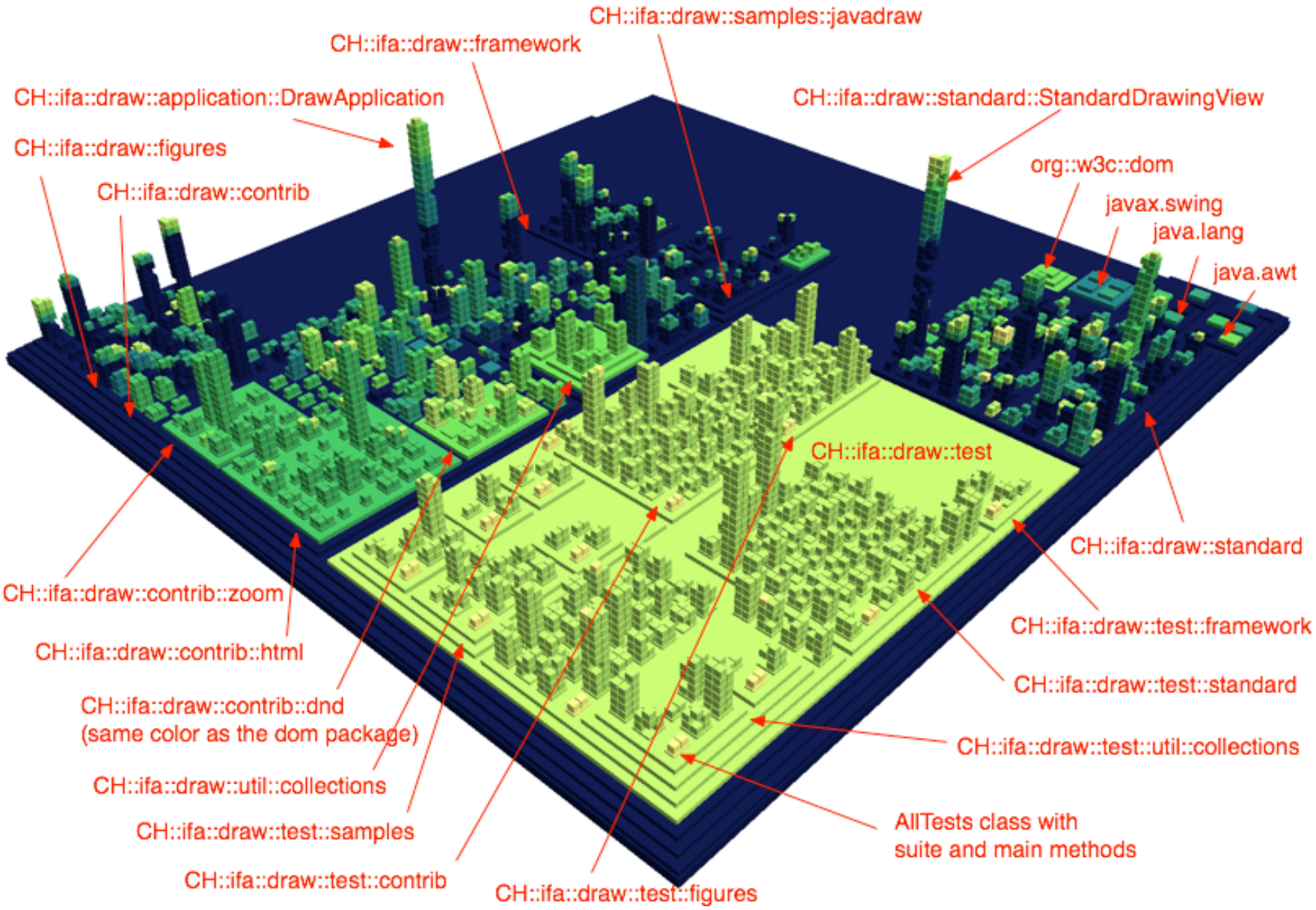
ArgoUML's filmstrip



ArgoUML Age Map

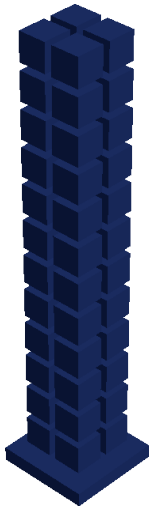


JHotDraw Age map



Age map interpretation

stable

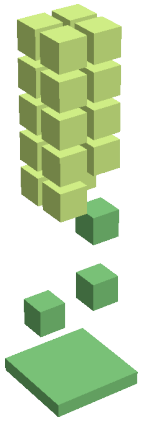


very old

rarely updated

old

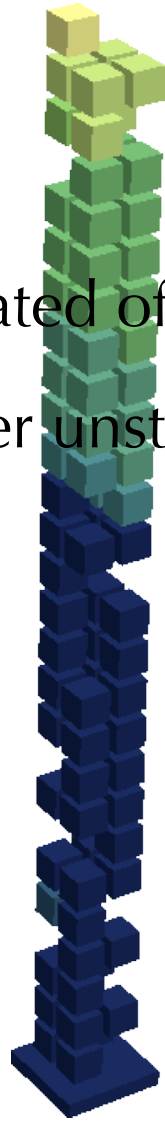
highly unstable



young

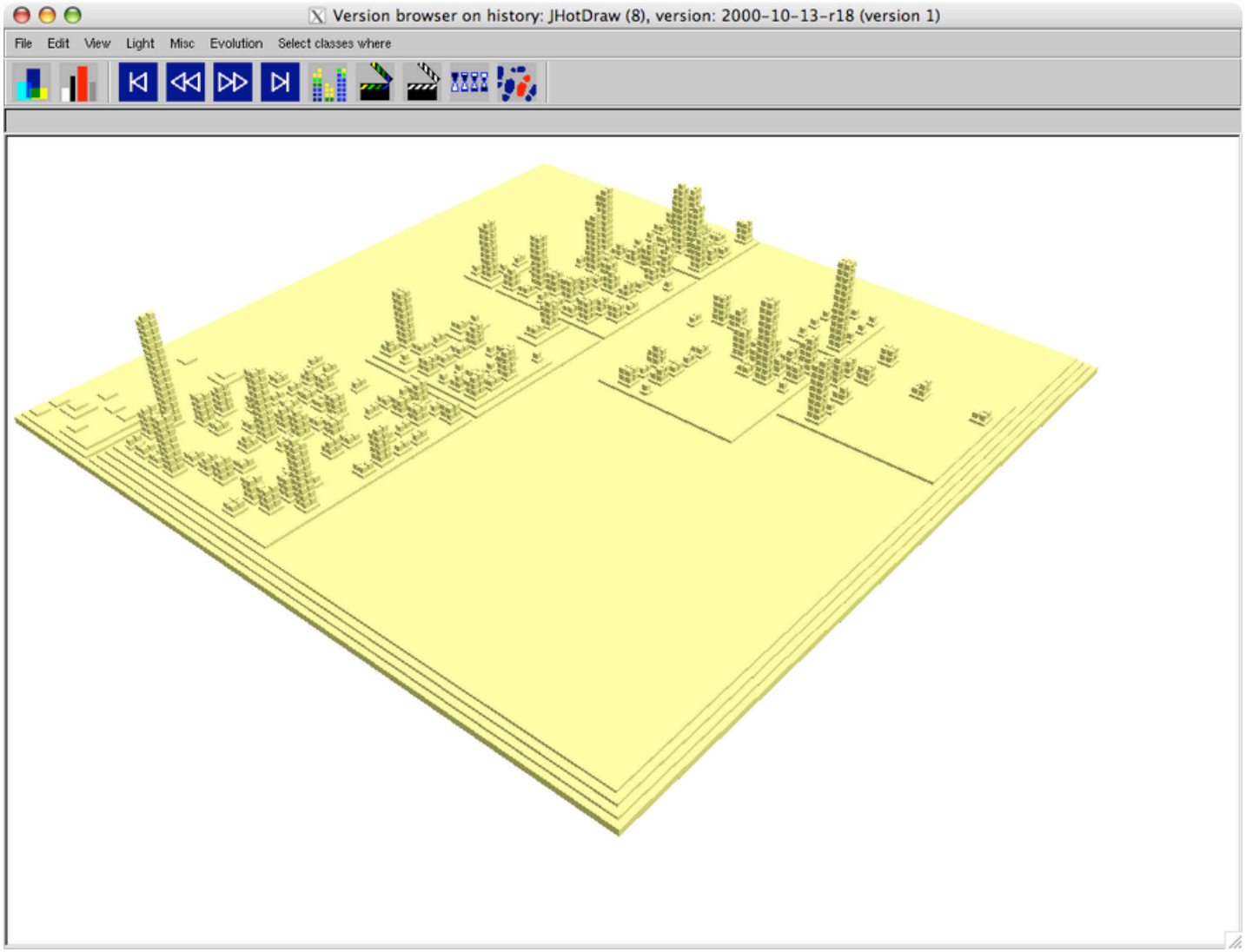
updated often,

rather unstable

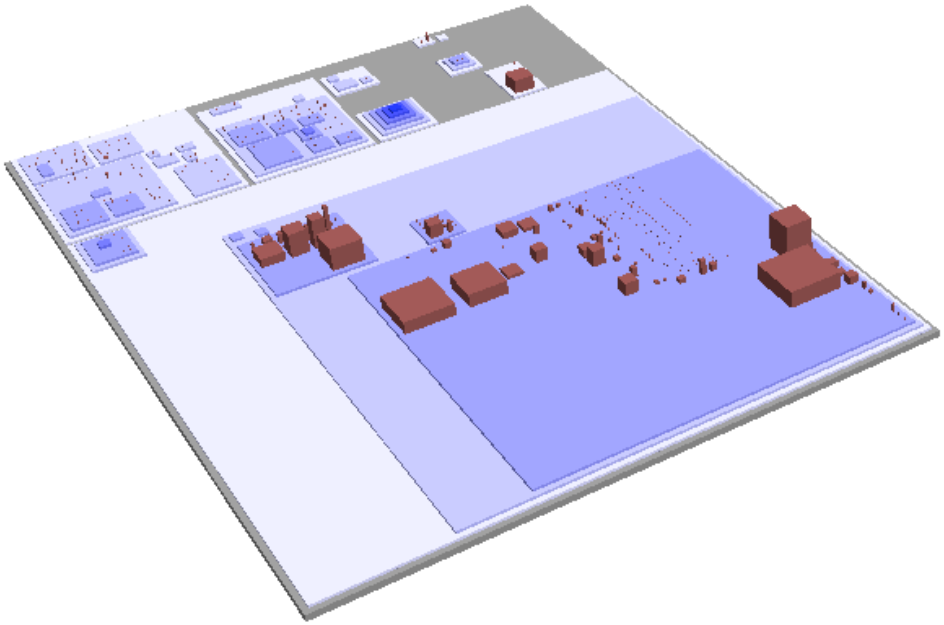


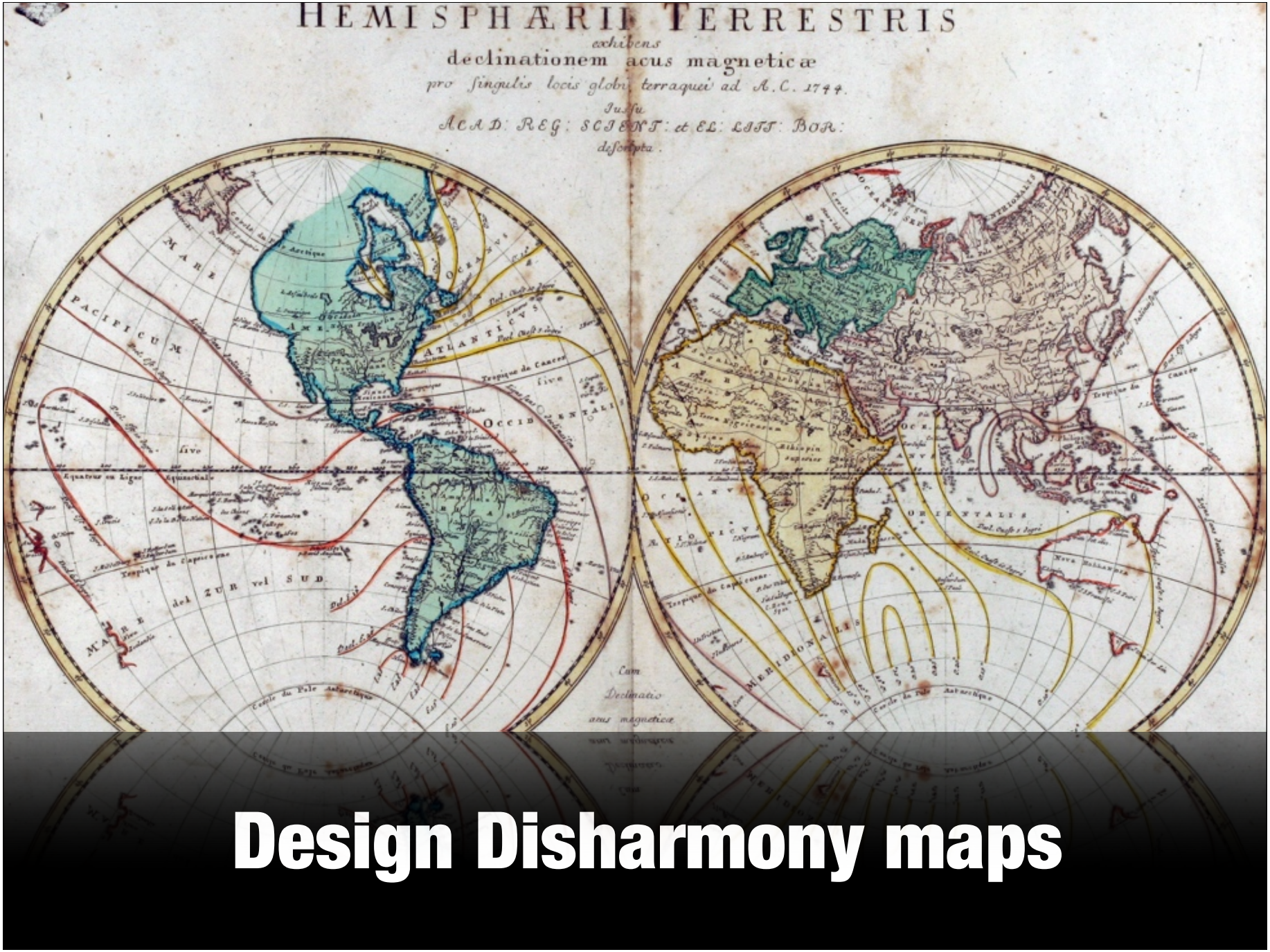
very old

JHotDraw's Travel through Time



Jmol’s Travel through Time



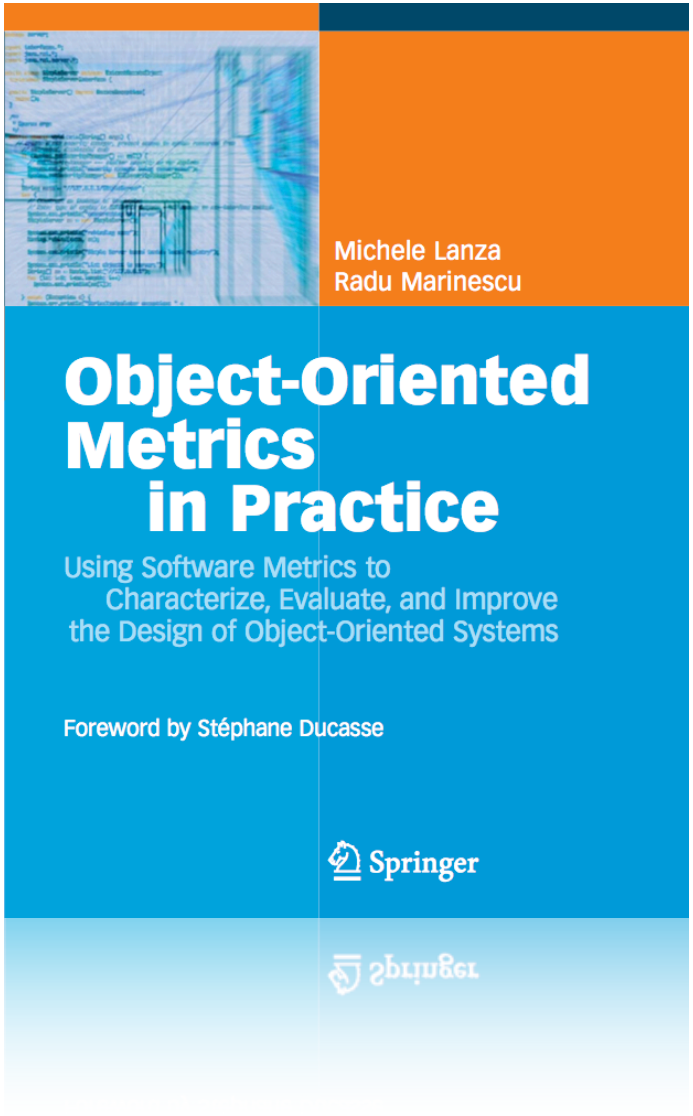


Design Disharmony maps

Shameless Plug

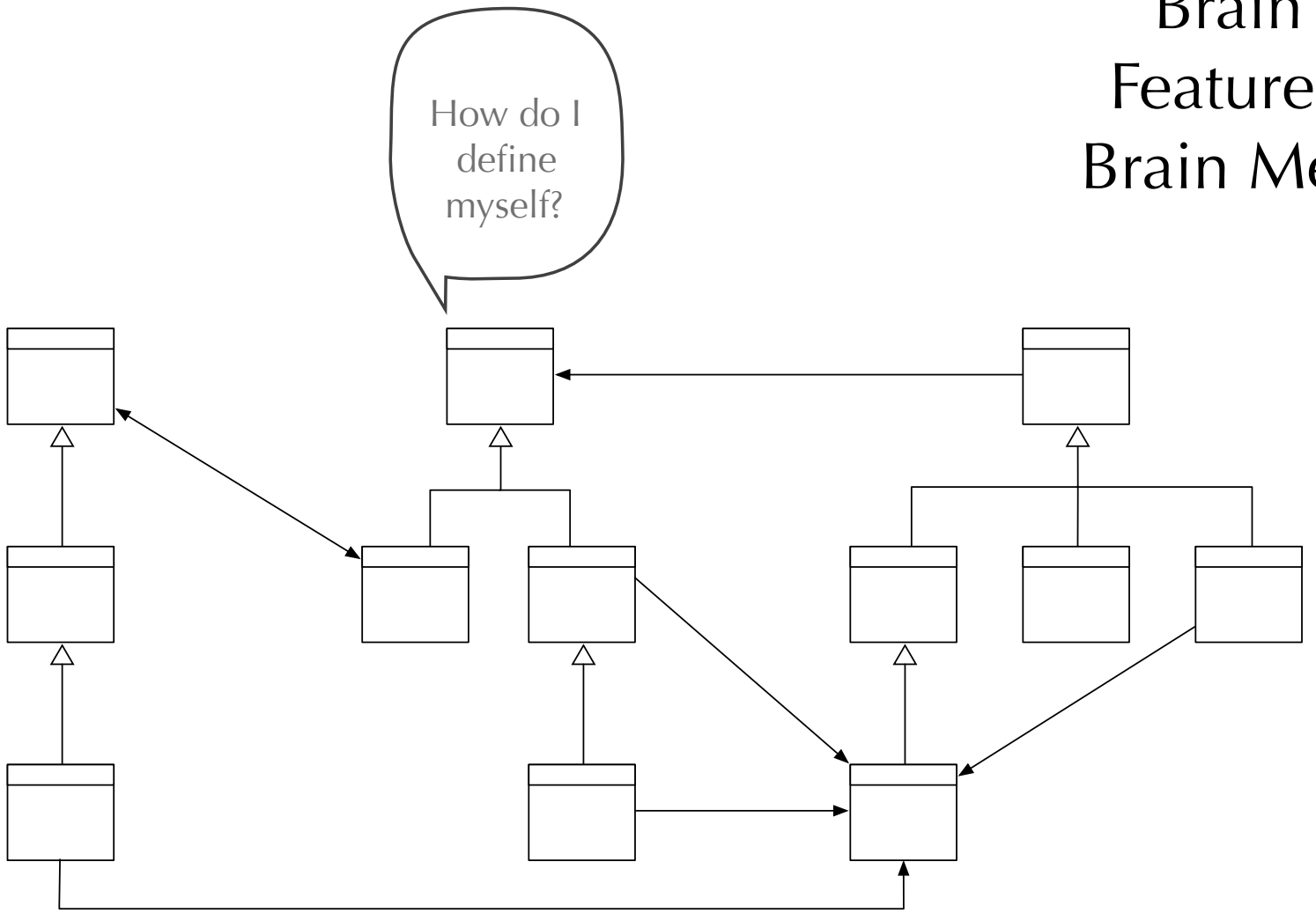
M. Lanza, R. Marinescu
"Object-Oriented Metrics in Practice"

Springer, 2006
ISBN 3-540-24429-8



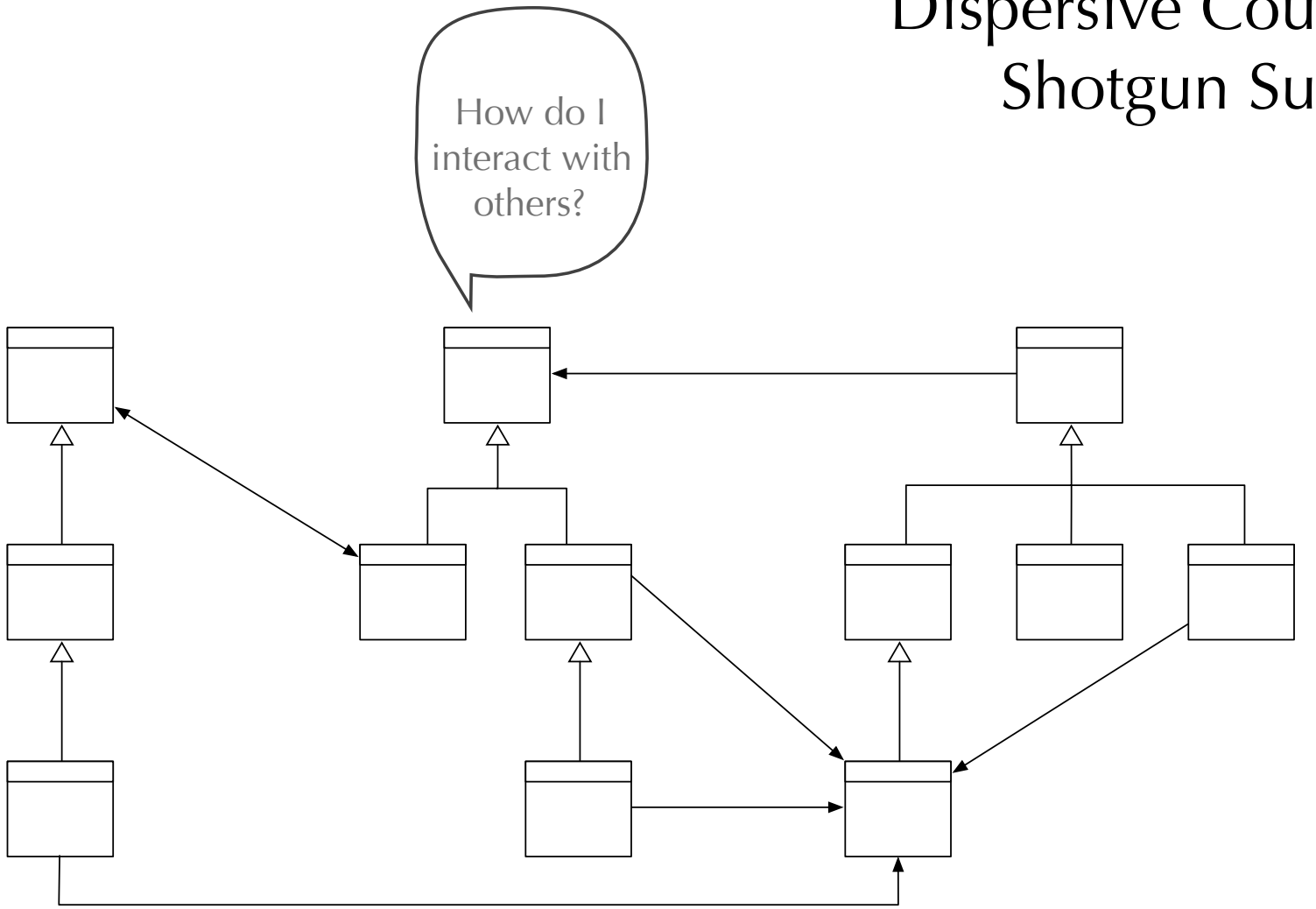
Identity Disharmony

God Class
Data Class
Brain Class
Feature Envy
Brain Method



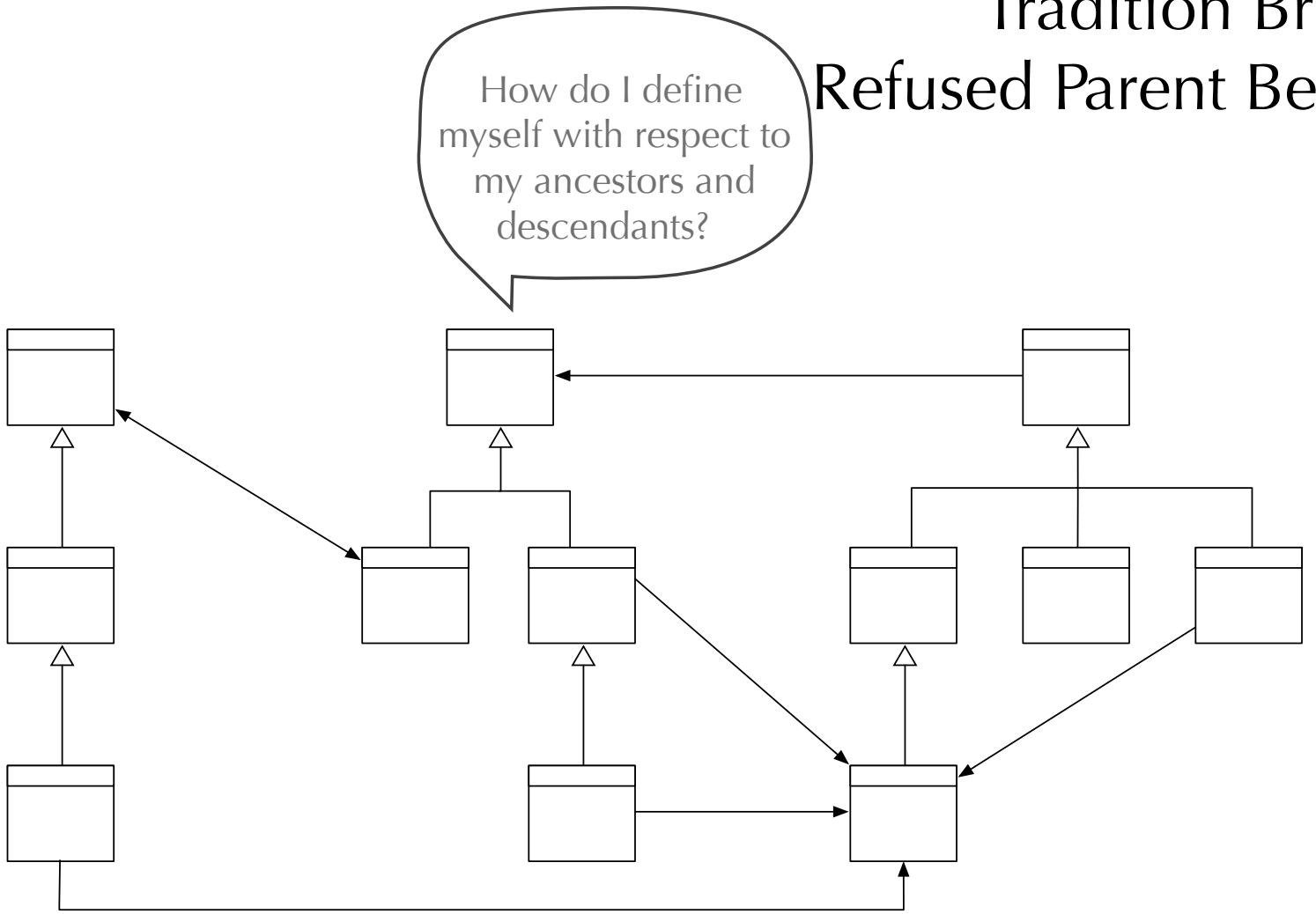
Collaboration Disharmony

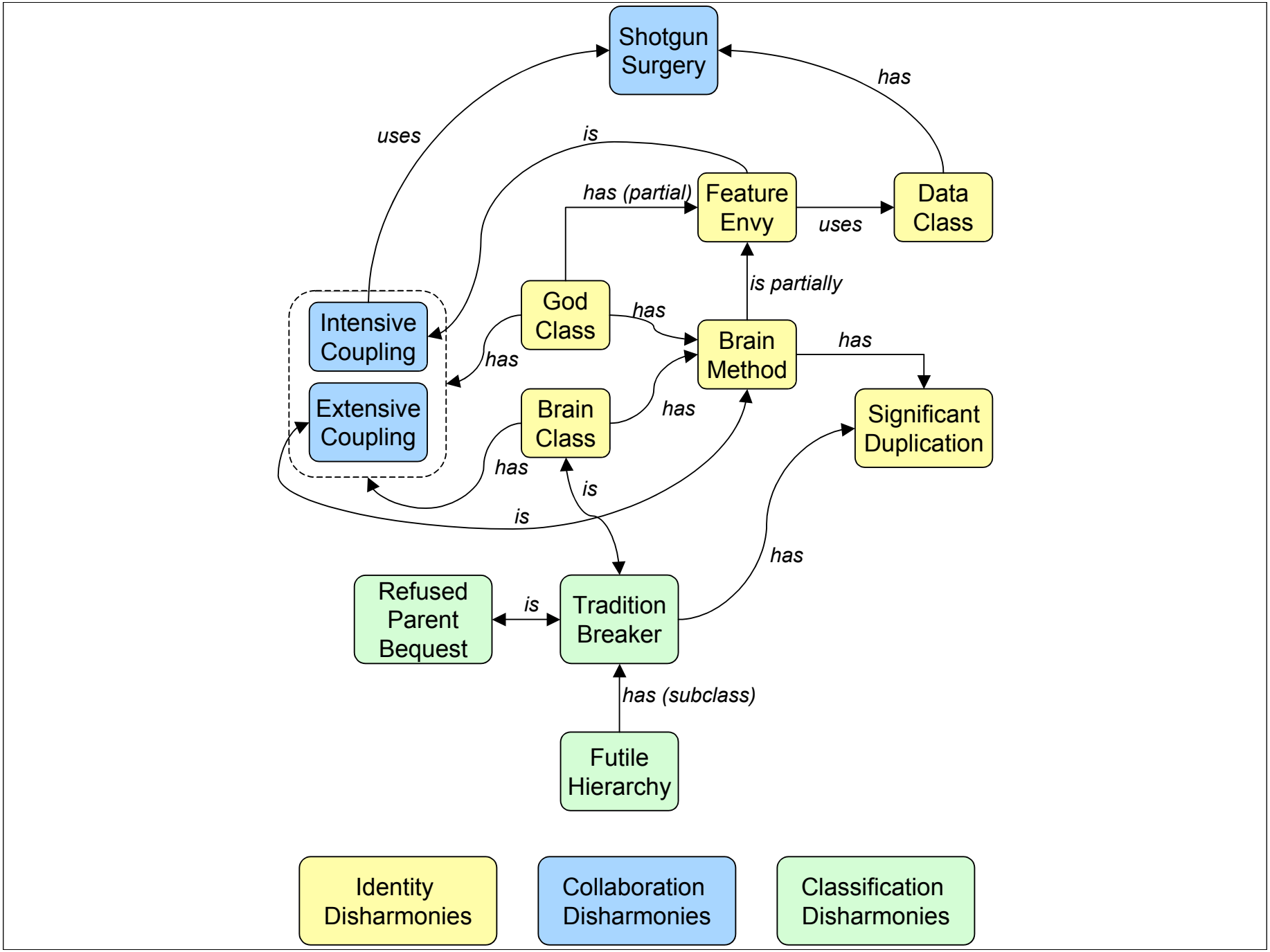
Intensive Coupling
Dispersive Coupling
Shotgun Surgery



Classification Disharmony

Futile Hierarchy
Tradition Breaker
Refused Parent Bequest

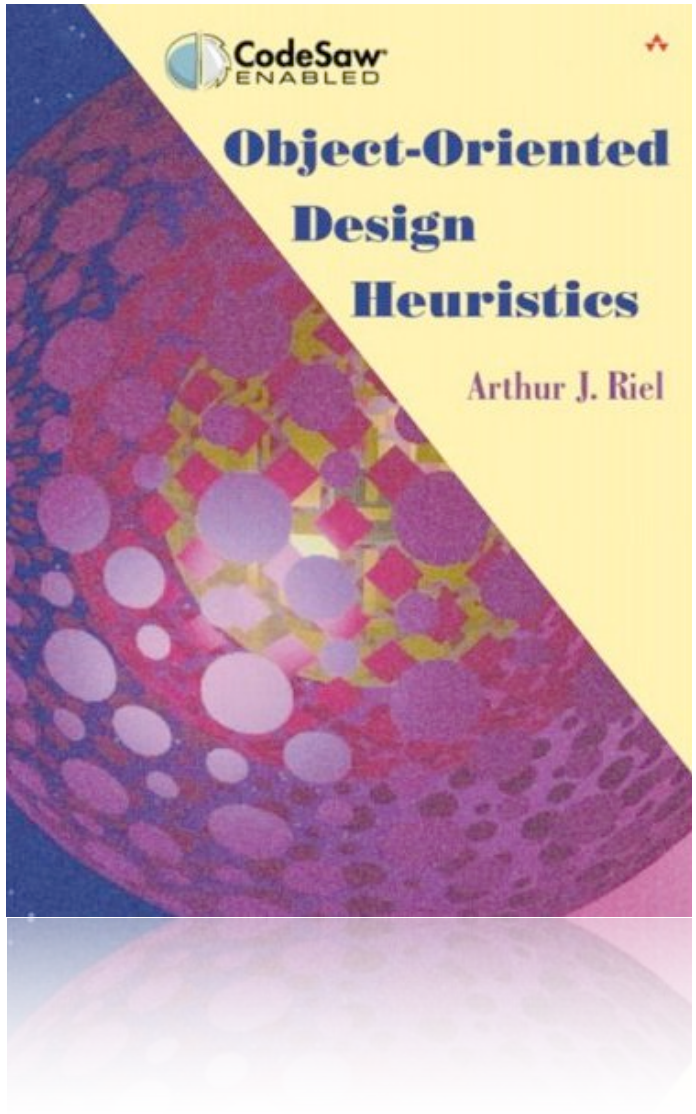



Shotgun Surgery
has
uses
is
has (partial)
Feature Envy
Data Class
uses
is partially
God Class
has
Brain Method
has
Intensive Coupling
has
Extensive Coupling
Brain Class
has
Significant Duplication
has
is
is
has
Refused Parent Bequest
is
Tradition Breaker
has (subclass)
Futile Hierarchy
Identity Disharmonies
Collaboration Disharmonies
Classification Disharmonies

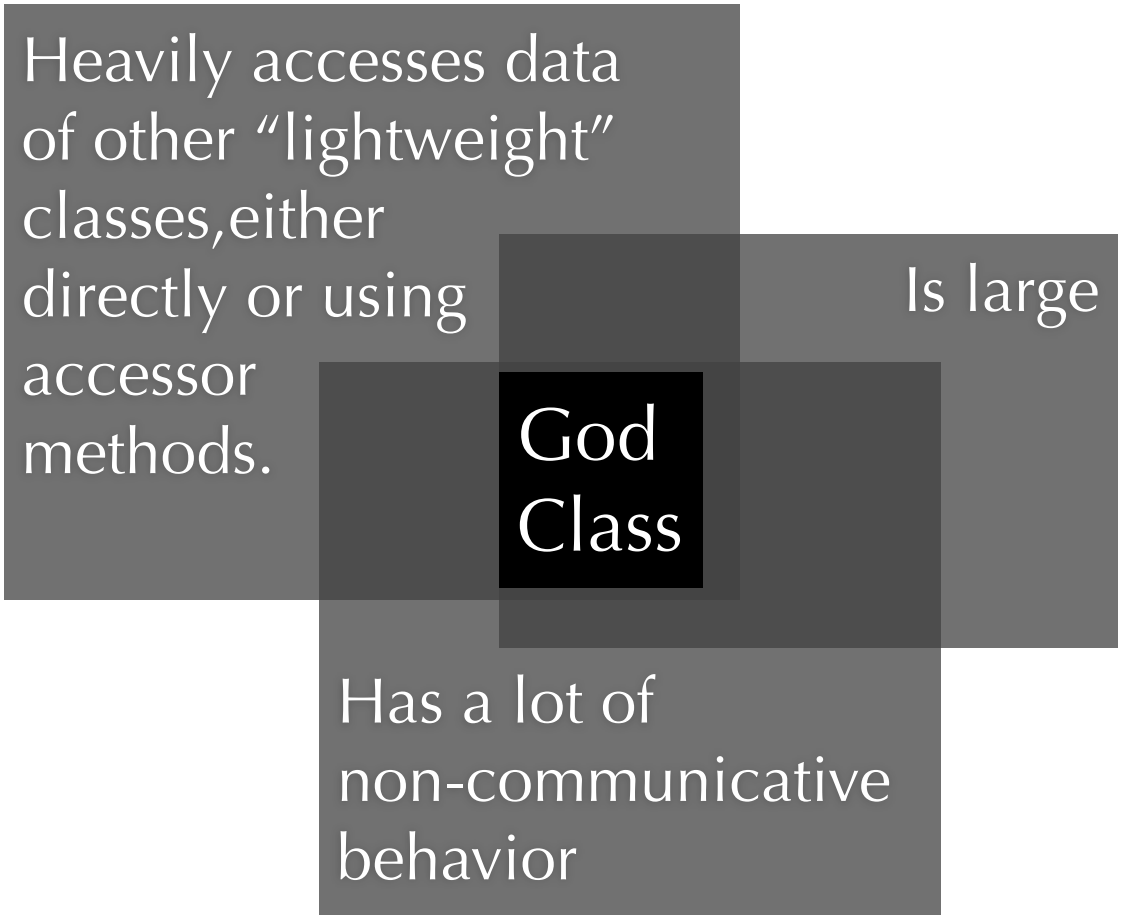
God Class

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

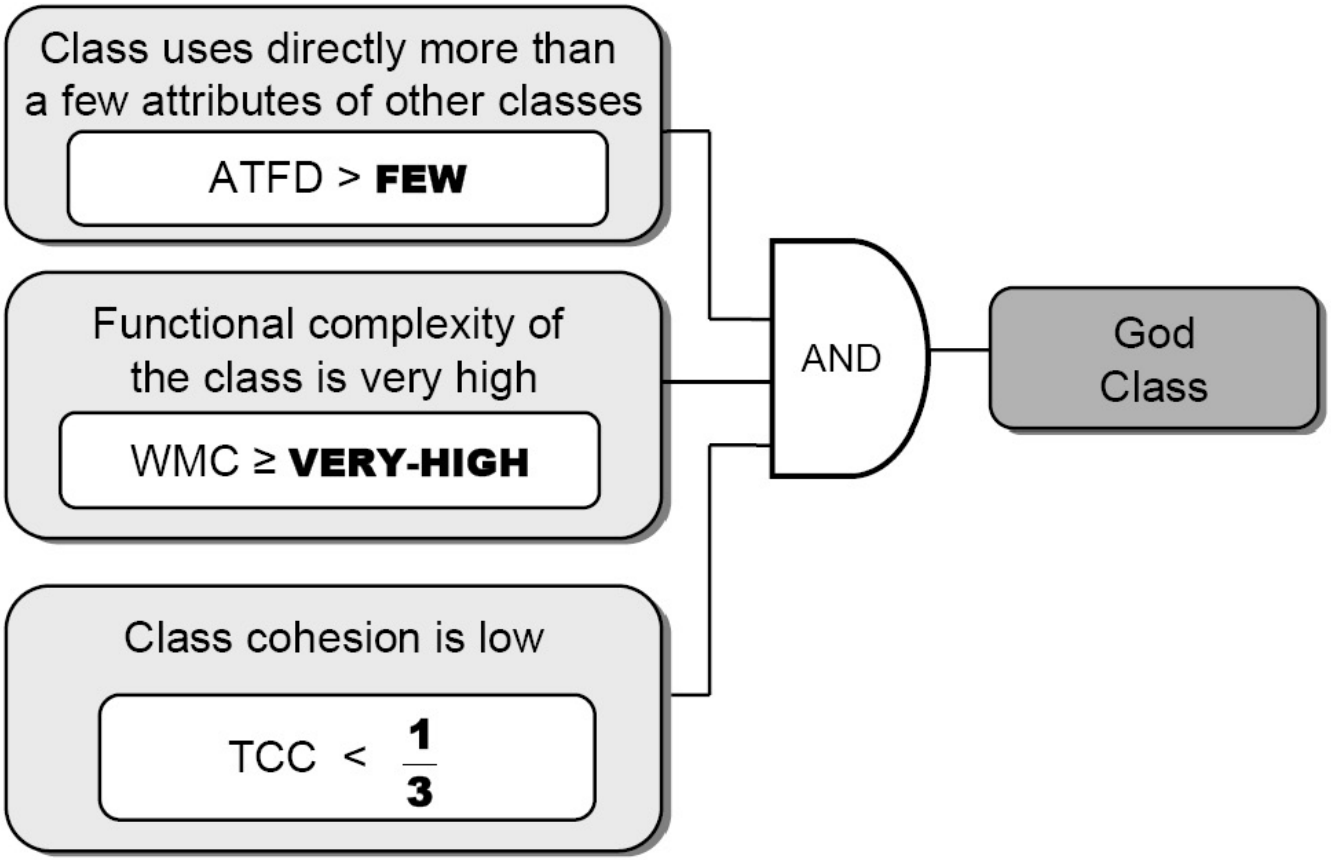
Arthur Riel, 1996



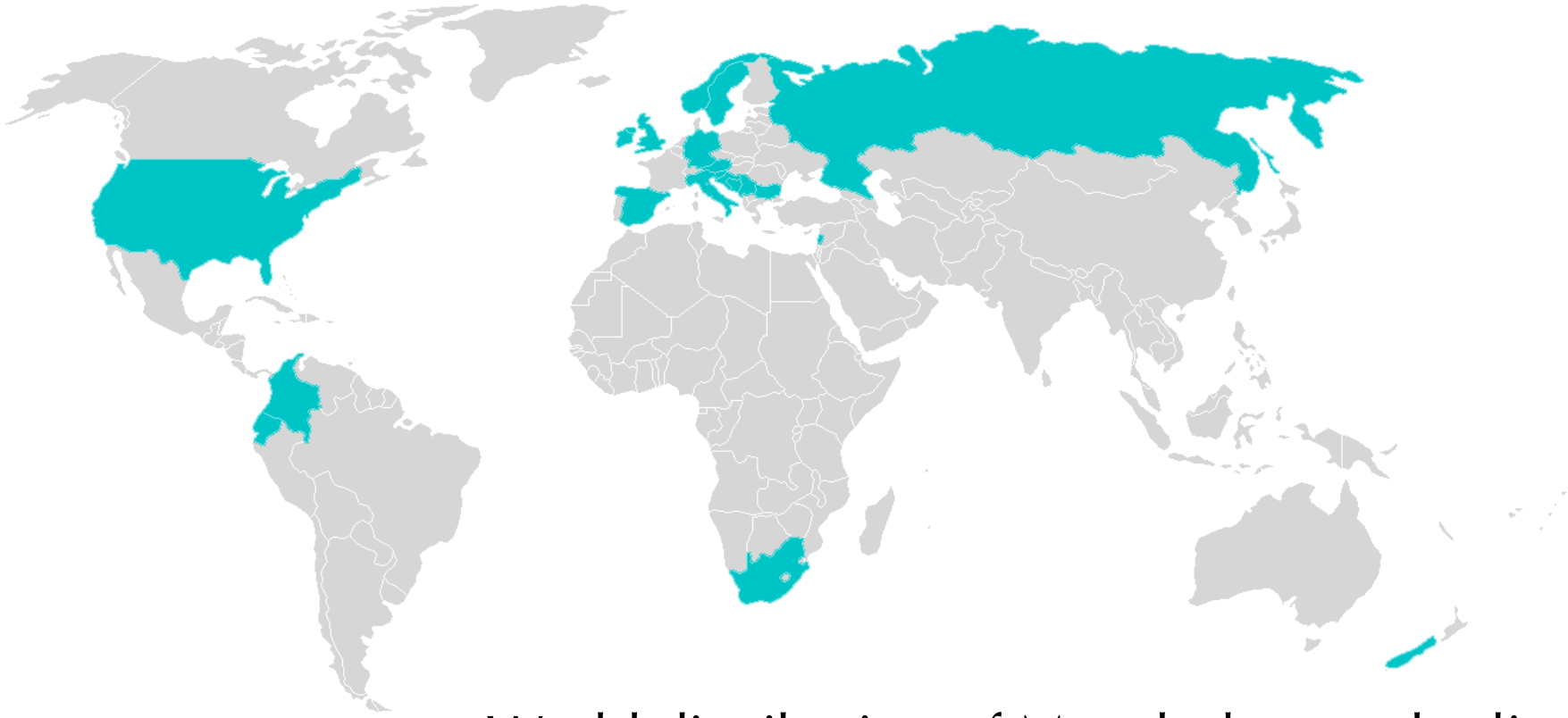
Characteristics of a God Class



The God Class detection strategy

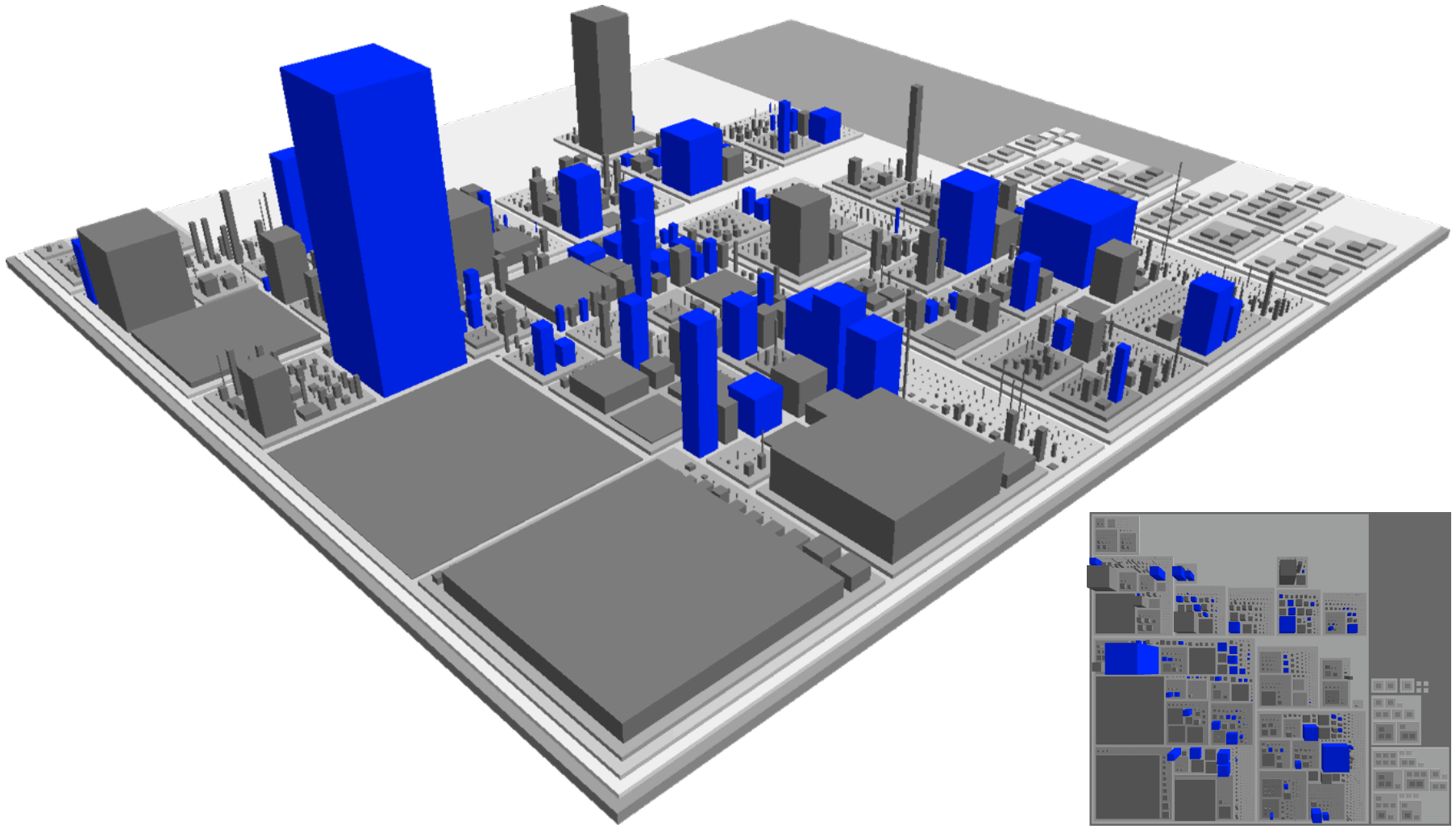


How should we display design problems?

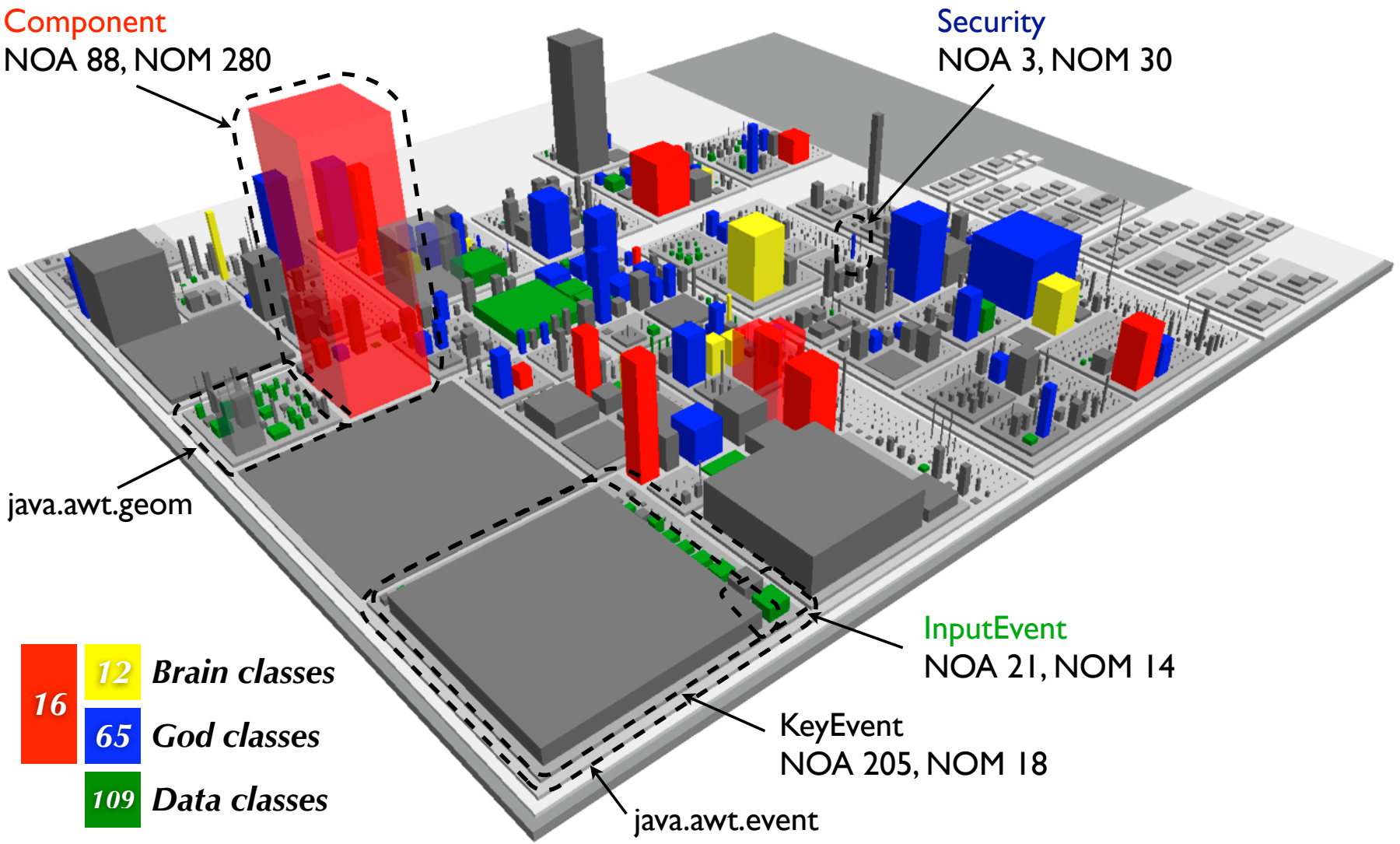


World distribution of Myxobolus cerebralis

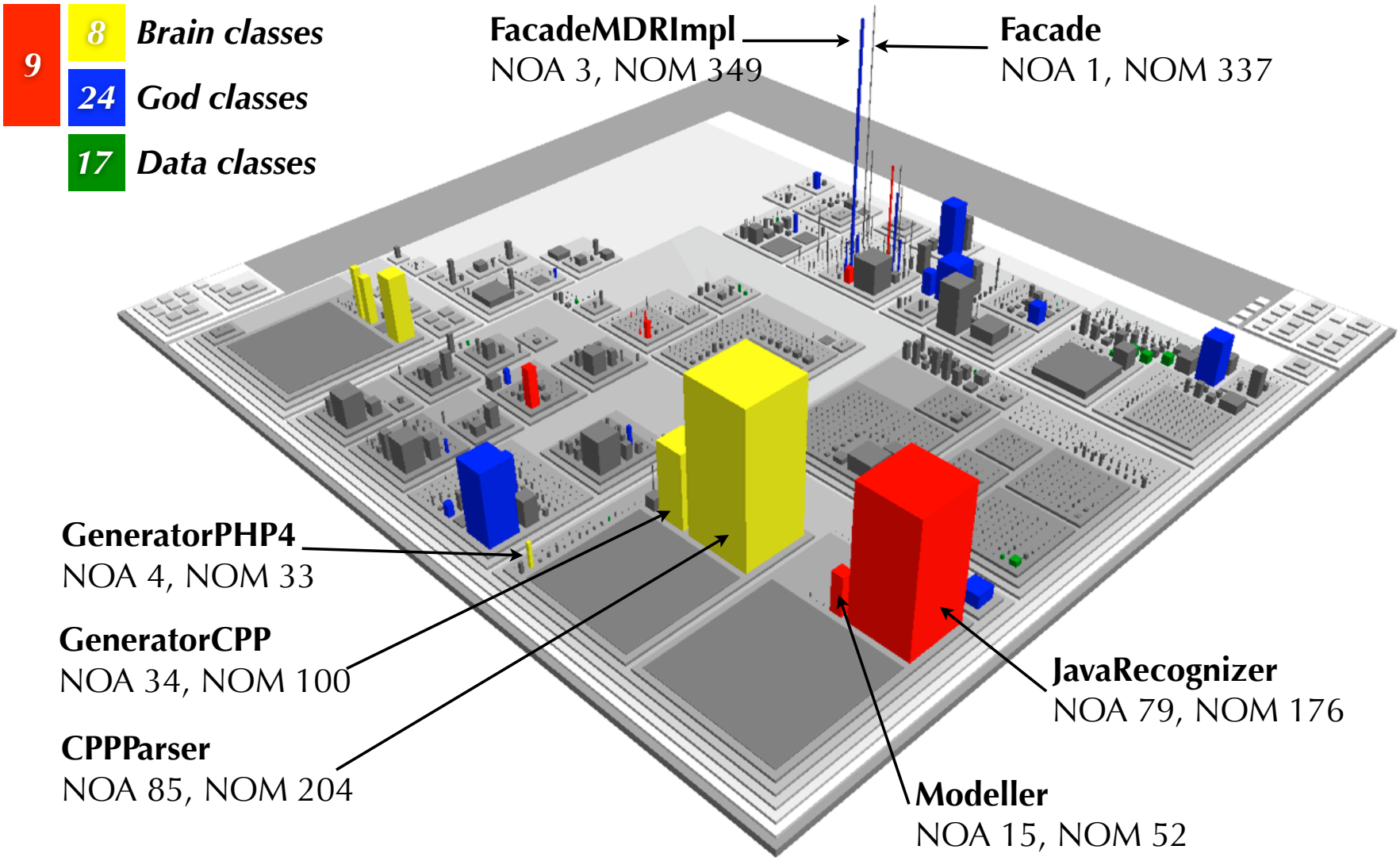
JDK 1.5 God Classes



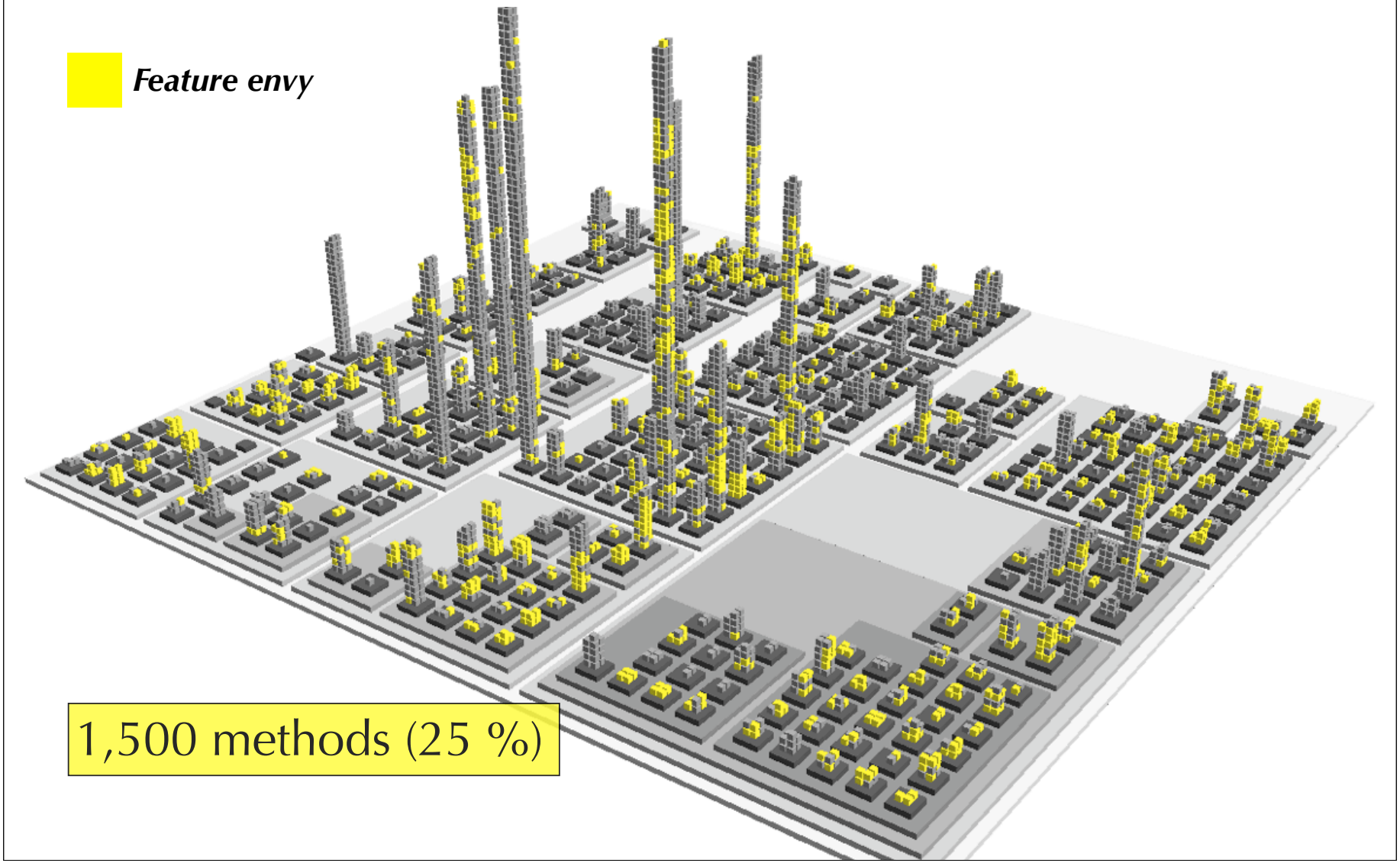
JDK's Idendity Disharmony map



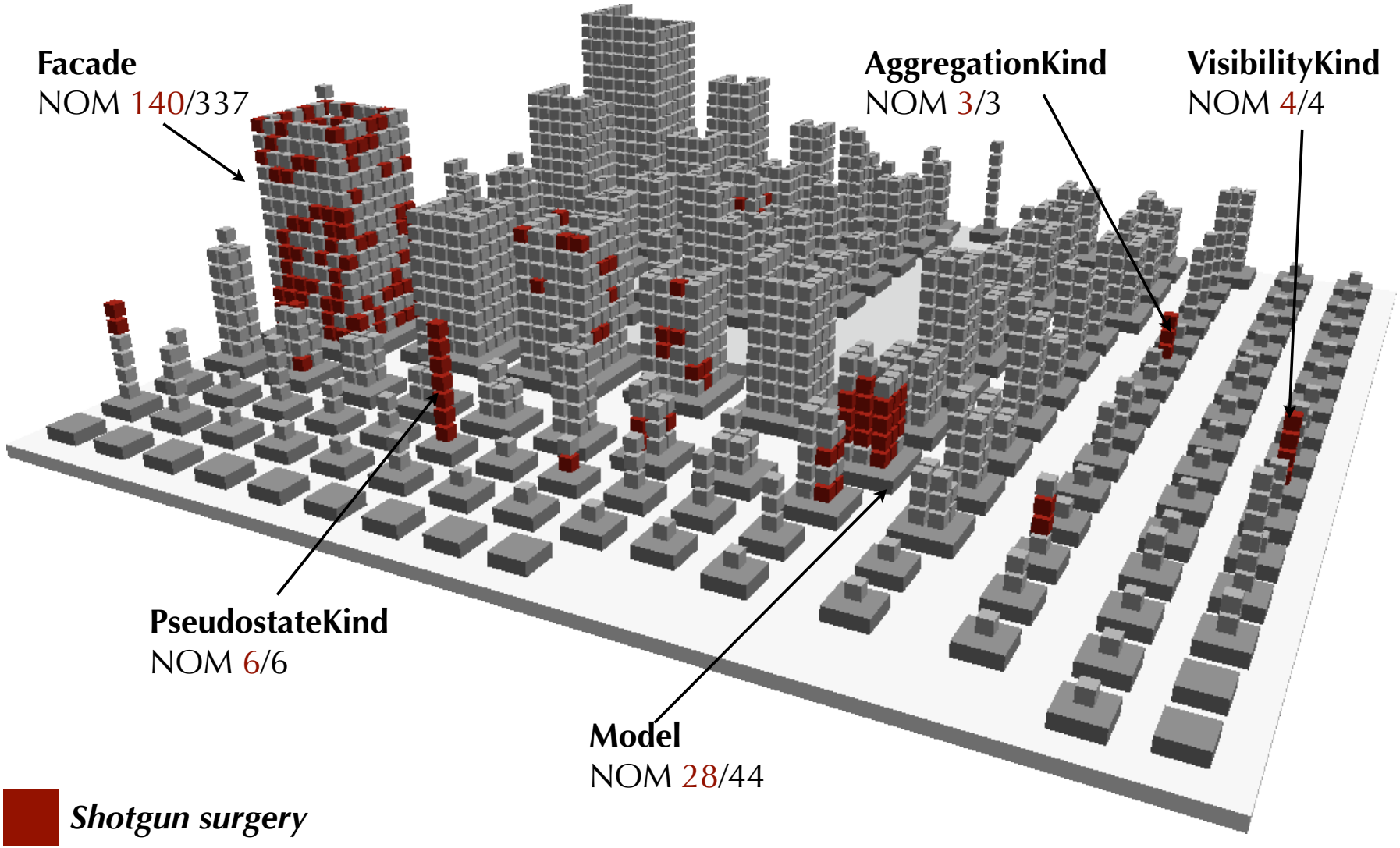
ArgoUML's Identity disharmony map

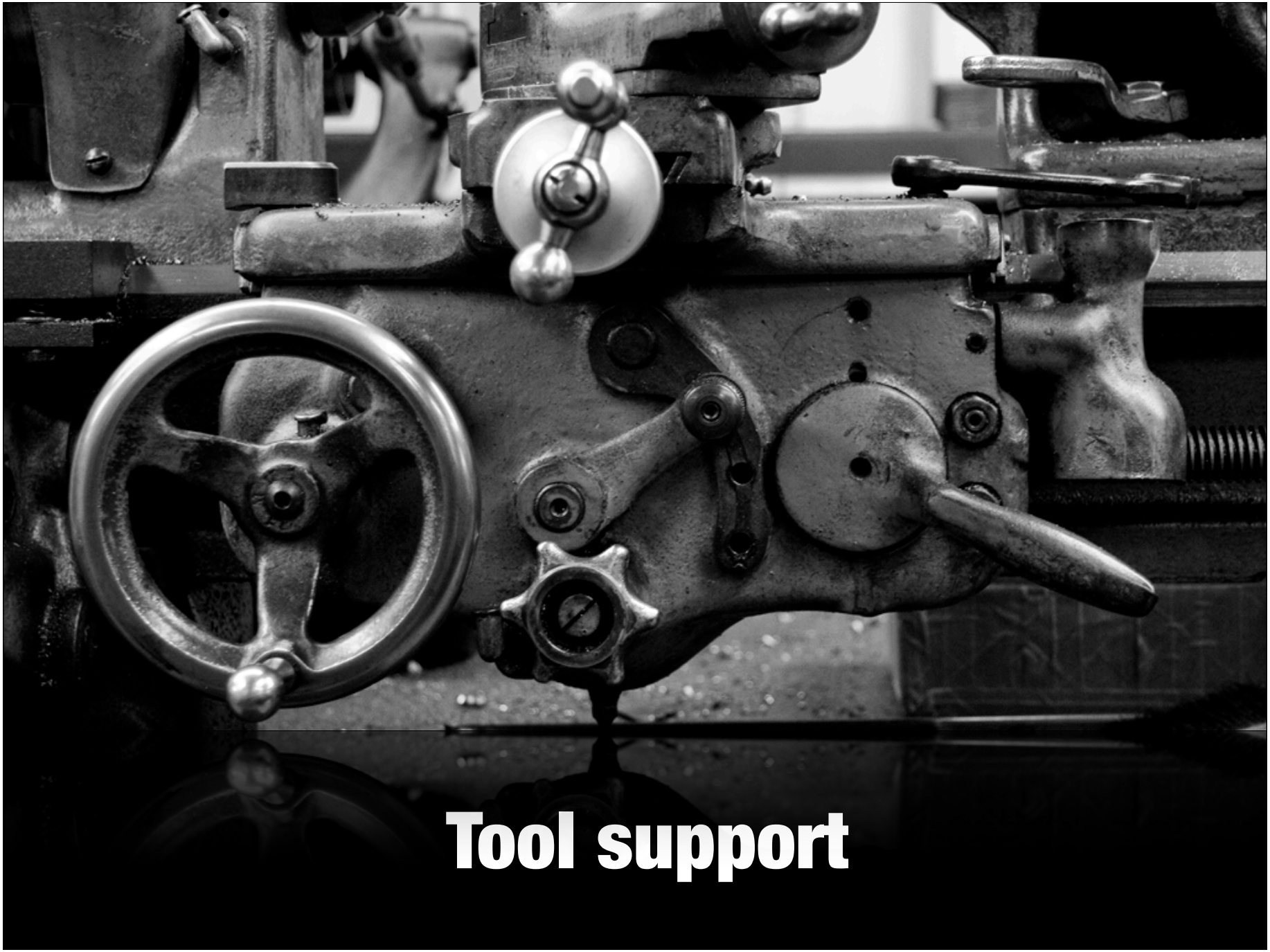


Jmol's Feature Envy

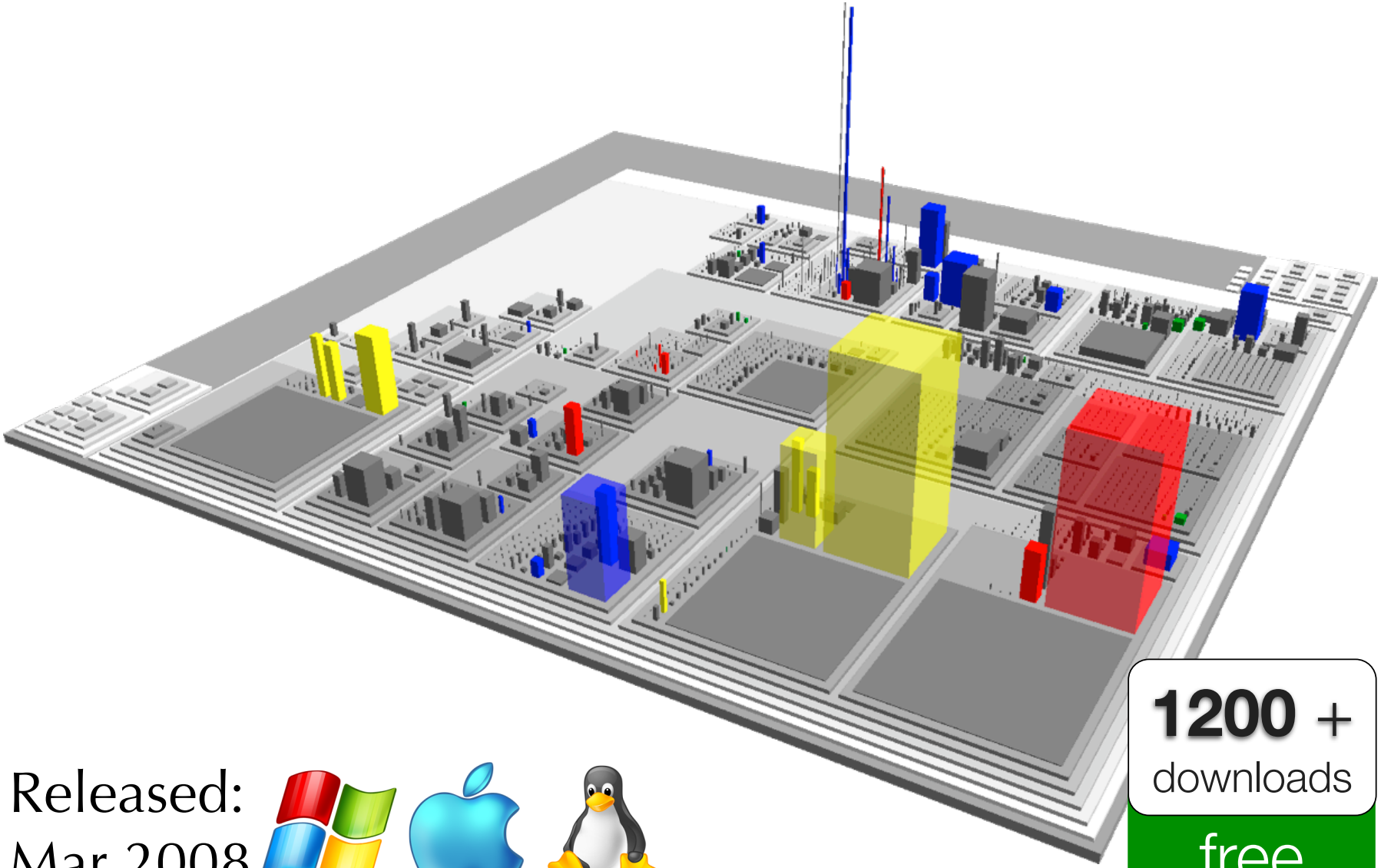


ArgoUML.Model's Shotgun Surgery Map




Tool support

http://www.inf.unisi.ch/phd/wettel/codecity.html



Released:
Mar 2008

1200 +
downloads

free

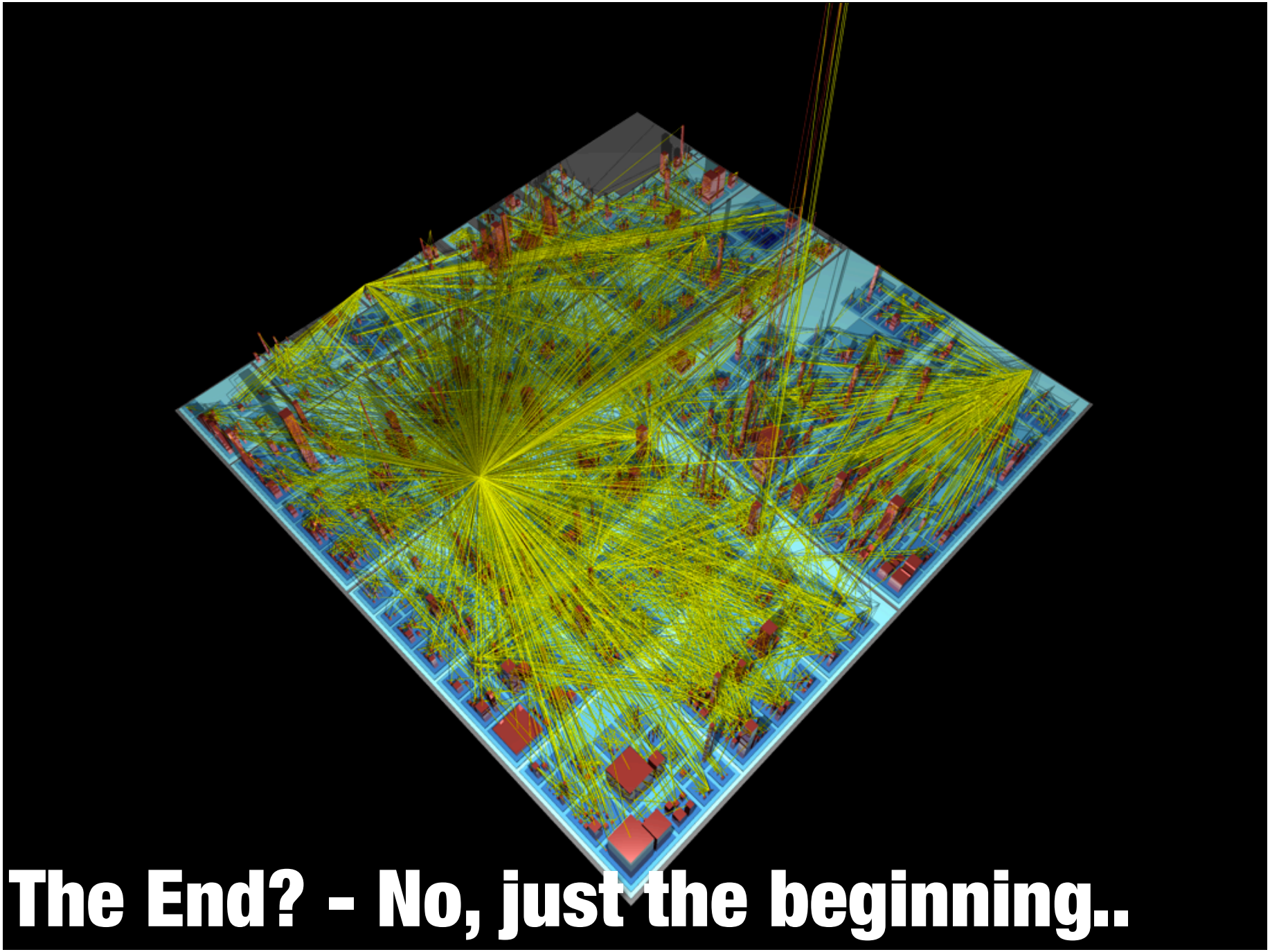
Epilogue

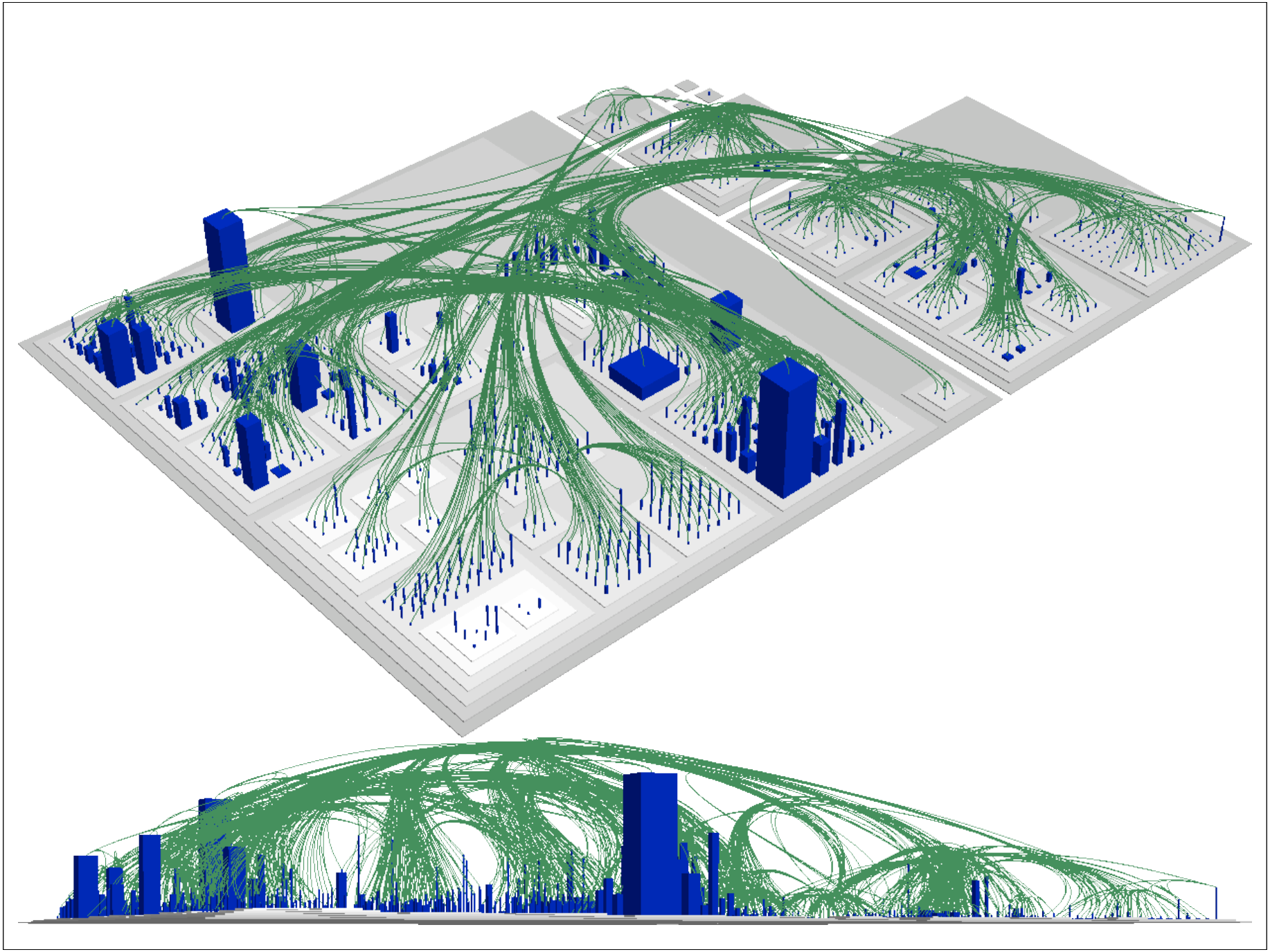
Huh?

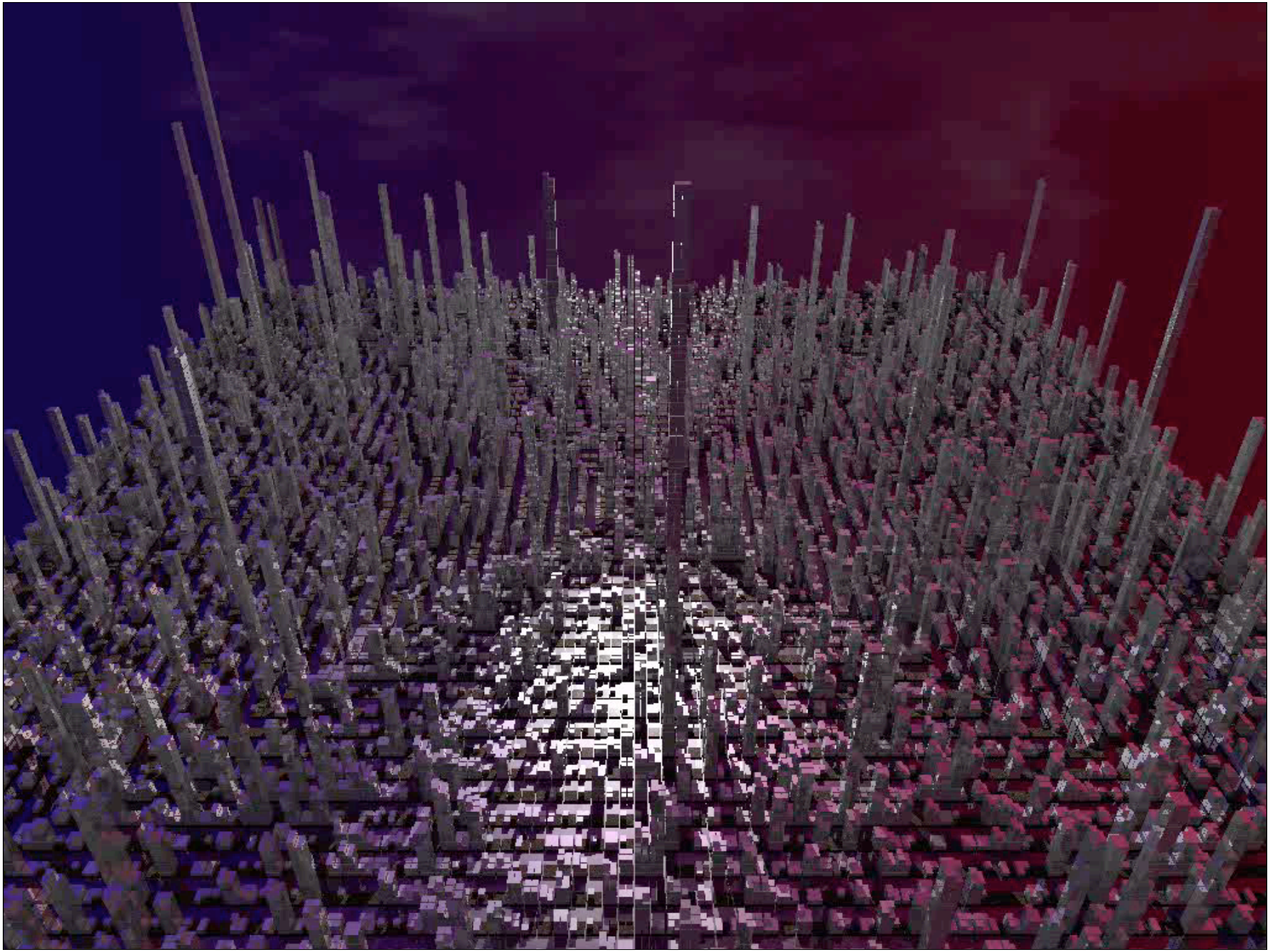
Reflections

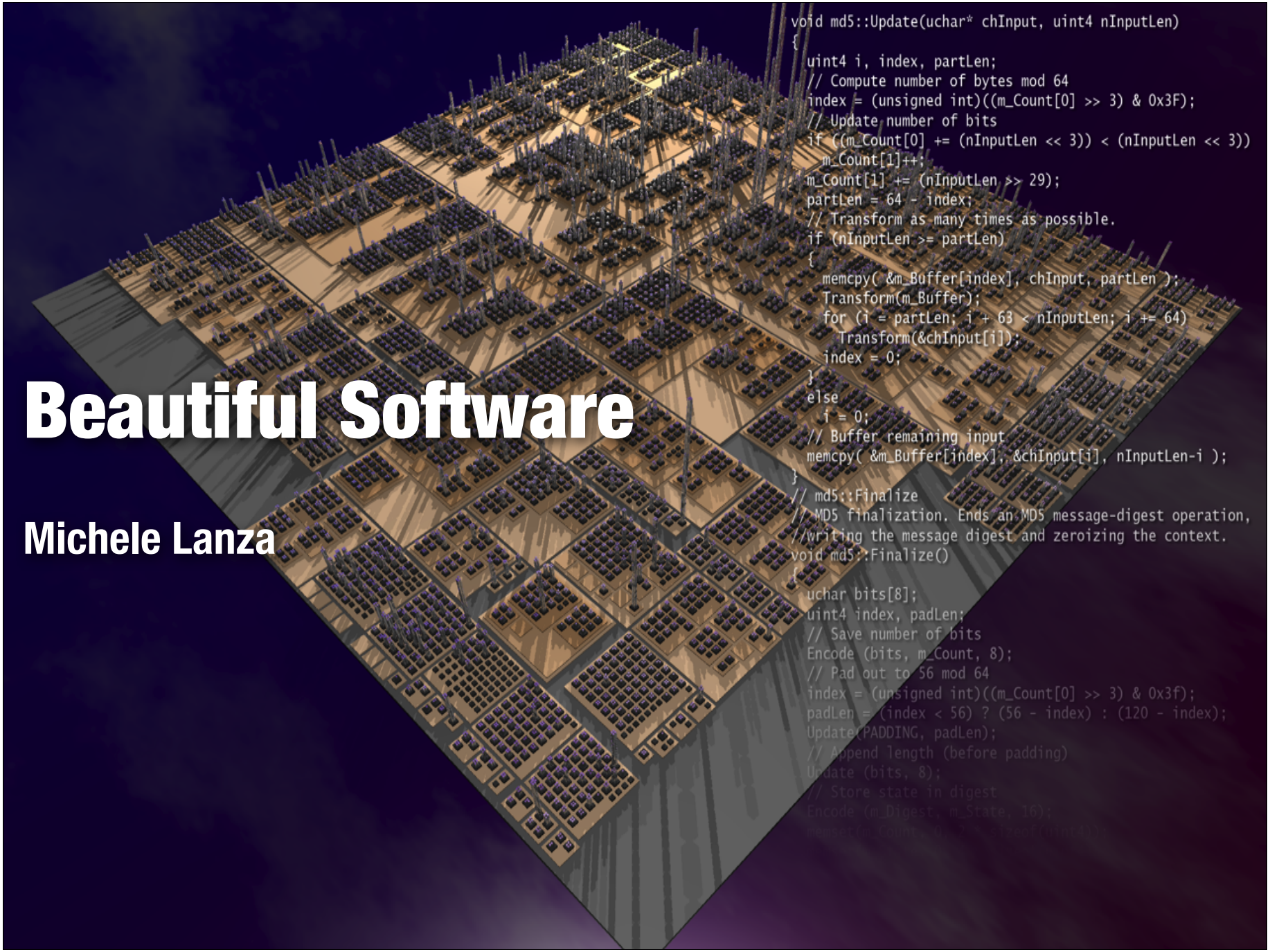
Visualizations are useless
..as pictures
..if not accessible

It'll take time for acceptance
..time is on my side


The End? - No, just the beginning..







Beautiful Software

Michele Lanza