Software Visualization 101+

Michele Lanza

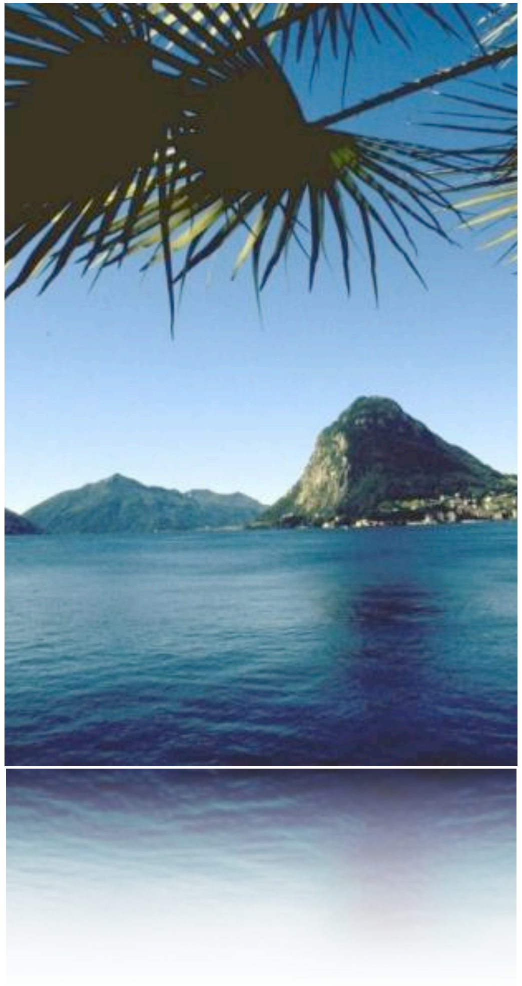
REVEAL @ Faculty of Informatics

University of Lugano, Switzerland

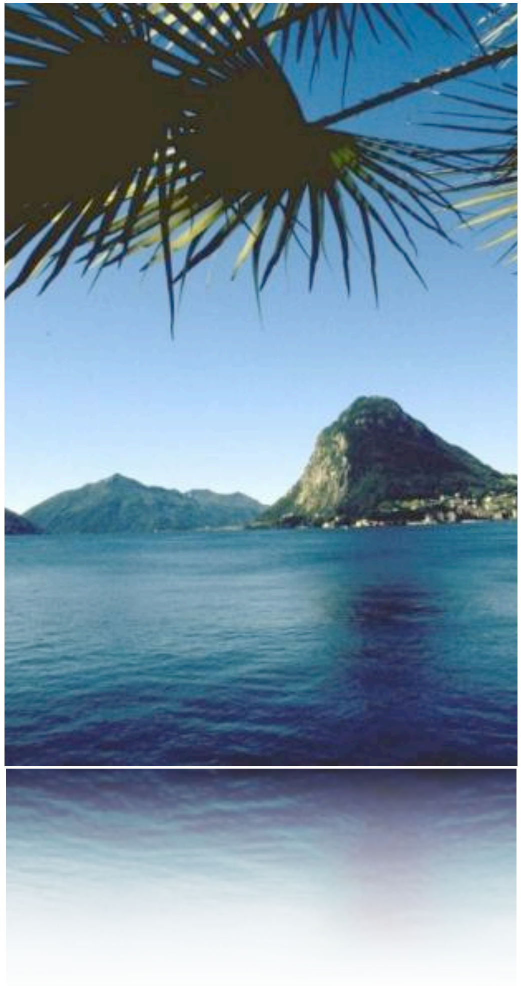
Part I

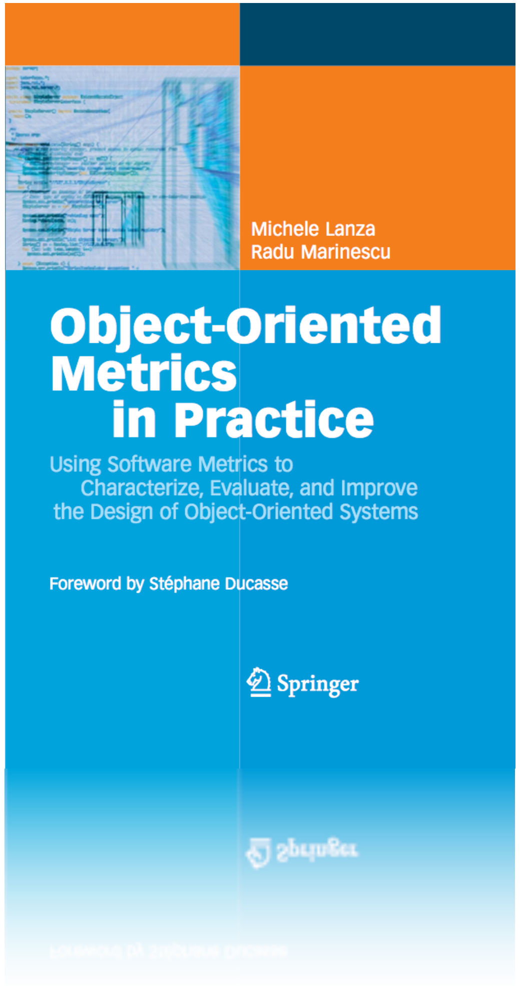
Prologue

Michele Lanza

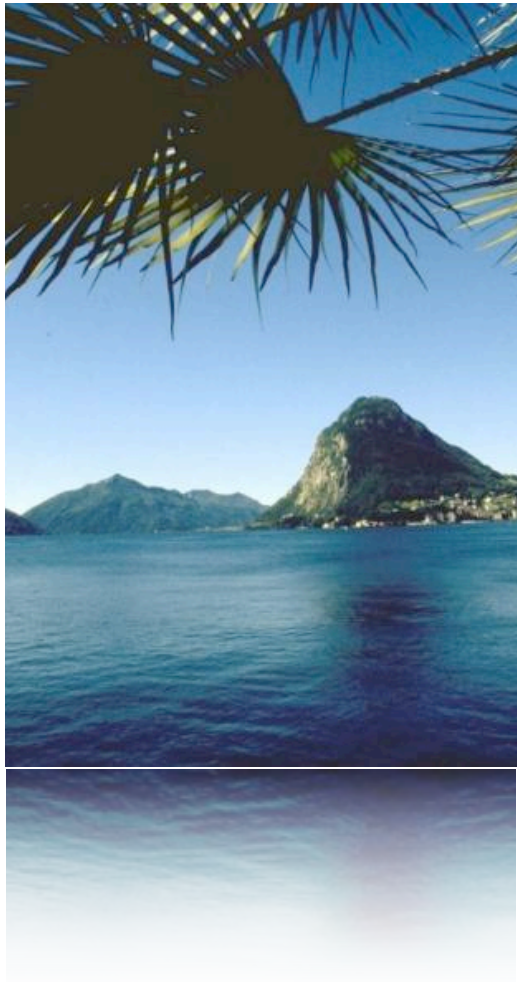


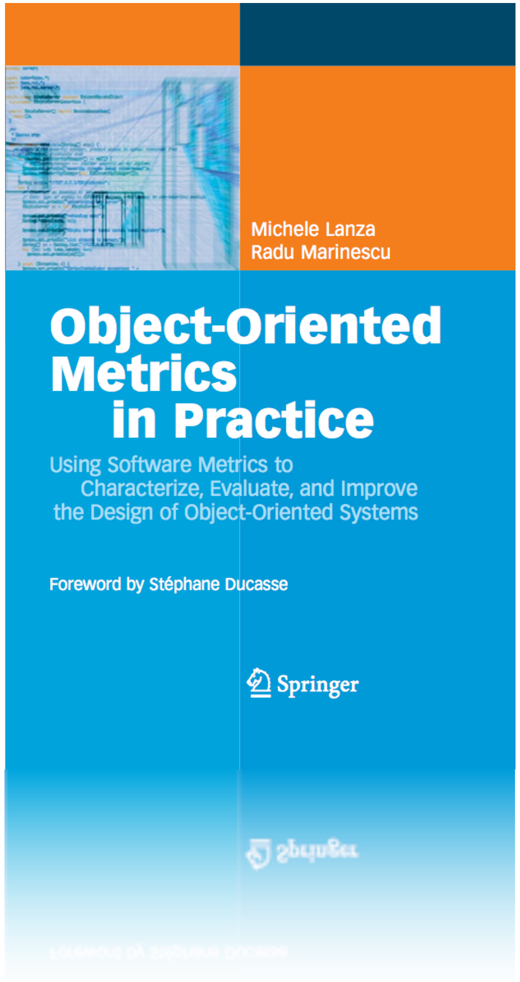
Michele Lanza





Michele Lanza



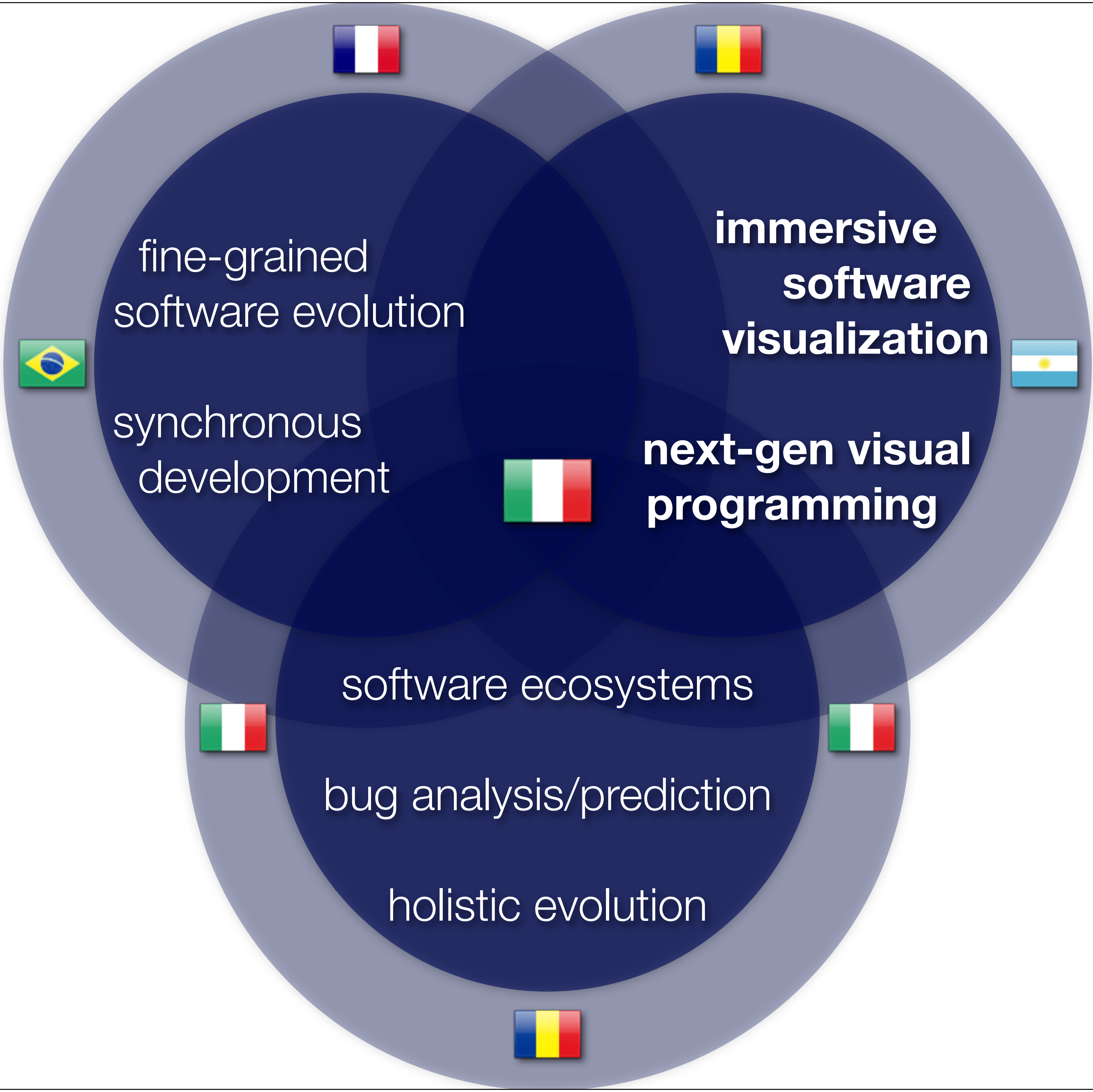


Michele Lanza

Academic Research

Industrial Reality

Military Fantasies



fine-grained software evolution
synchronous development
immersive software visualization
next-gen visual programming
software ecosystems
bug analysis/prediction
holistic evolution

Immersive Software Visualization

- R. Wettel, M. Lanza; Program Comprehension through Software Habitability. In **ICPC 2007** (15th IEEE International Conference on Program Comprehension), pp. 231 - 240, IEEE CS Press, 2007
- R. Wettel, M. Lanza; Visualizing Software Systems as Cities. In **VISSOFT 2007** (4th IEEE 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 **Softvis 2008** (4th ACM International Symposium on Software Visualization), pp. 155 - 164, ACM Press, 2008
- R. Wettel, M. Lanza; Visual Exploration of Large-scale System Evolution. In **WCRE 2008** (15th IEEE Working Conference on Reverse Engineering), pp. 219 - 228, IEEE CS Press, 2008
- R. Wettel, M. Lanza; CodeCity: 3D Visualization of Evolving Large-Scale Software. In **ICSE 2008** (30th ACM/IEEE International Conference on Software Engineering), pp. 921 - 922, ACM Press, 2008.

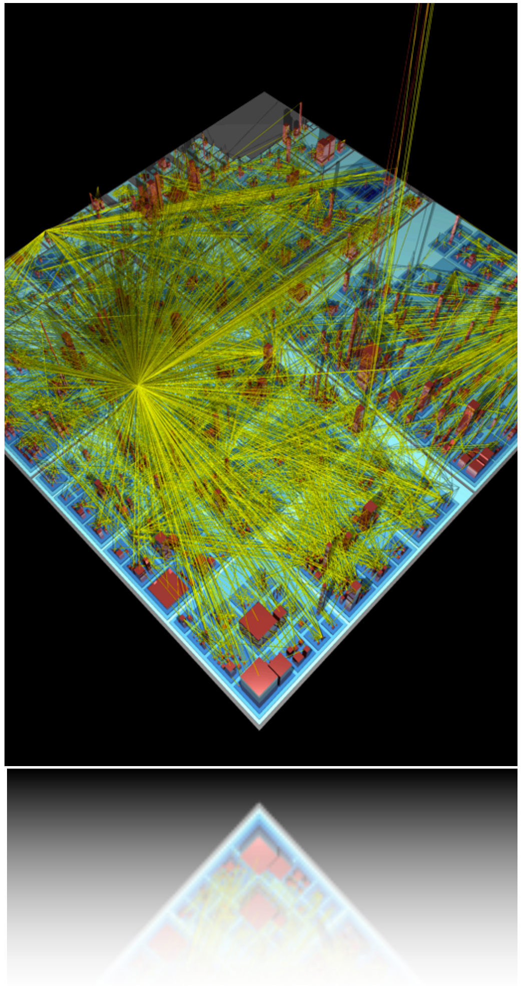
Richard Wettel

CodeCity

Goals

Goals

Goals

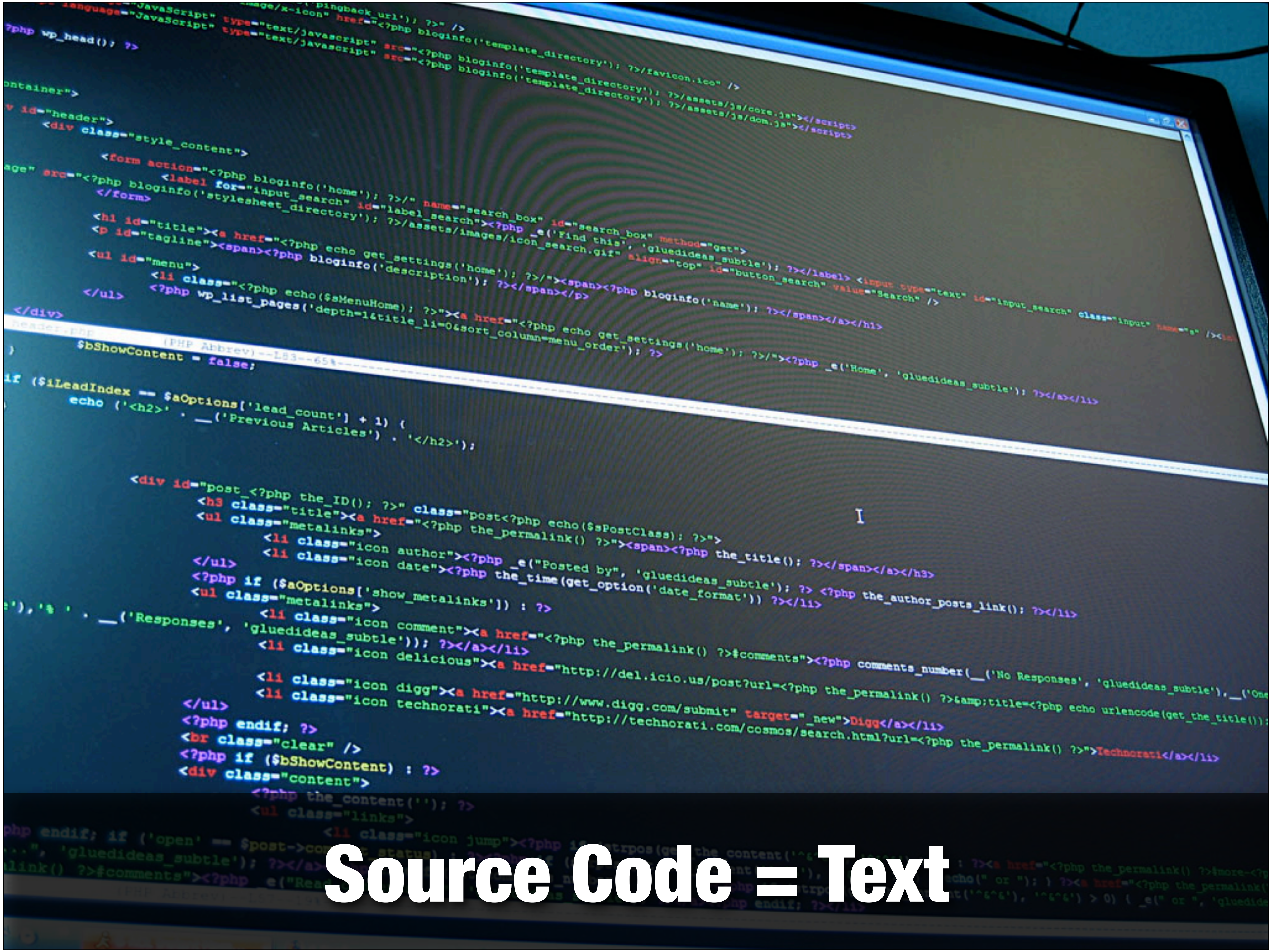


Goals

Part II

Software

[Software is] anything but hardware, [...] 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


Enjoy the Ride

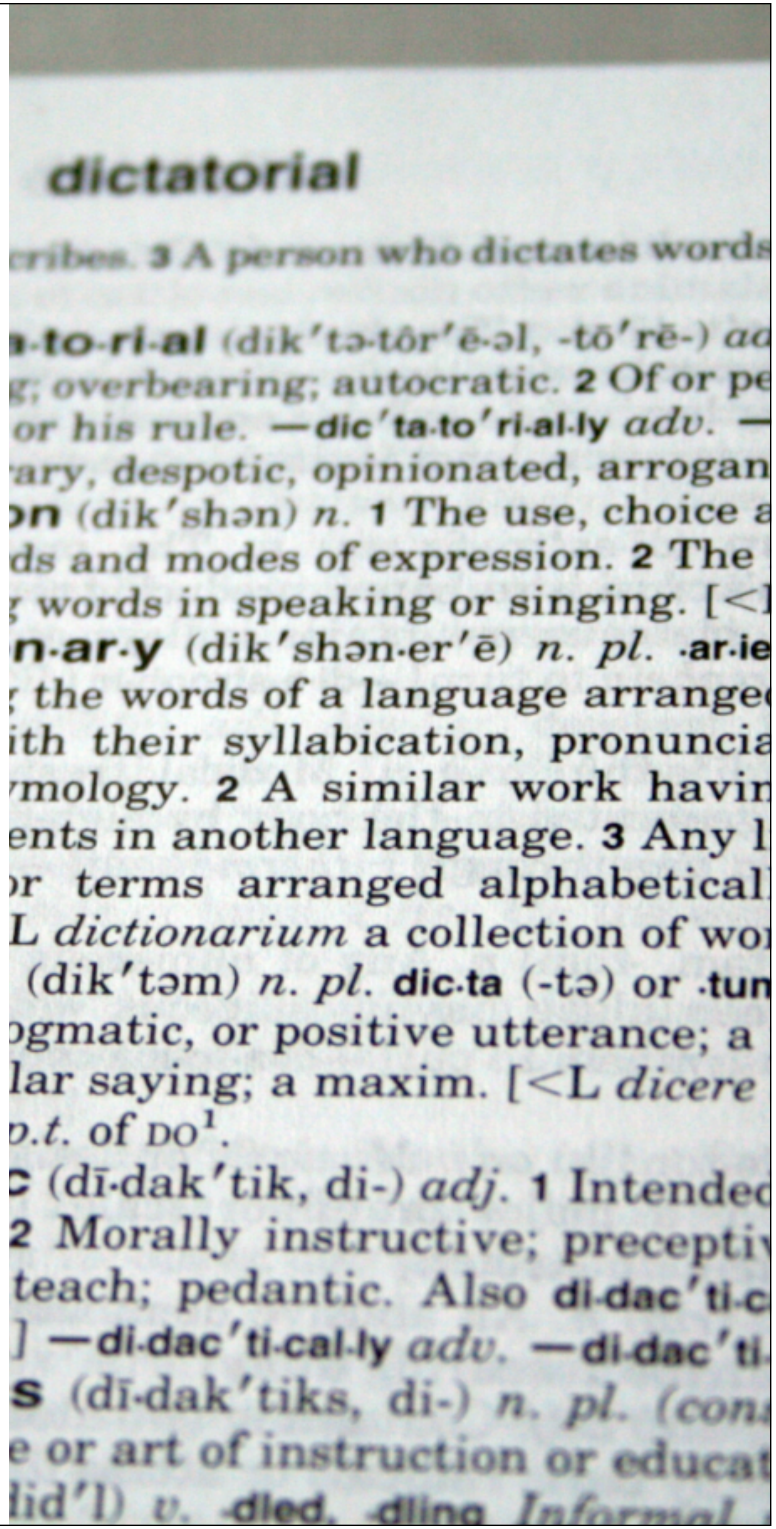
Part III

Software Visualization

Software Visualization

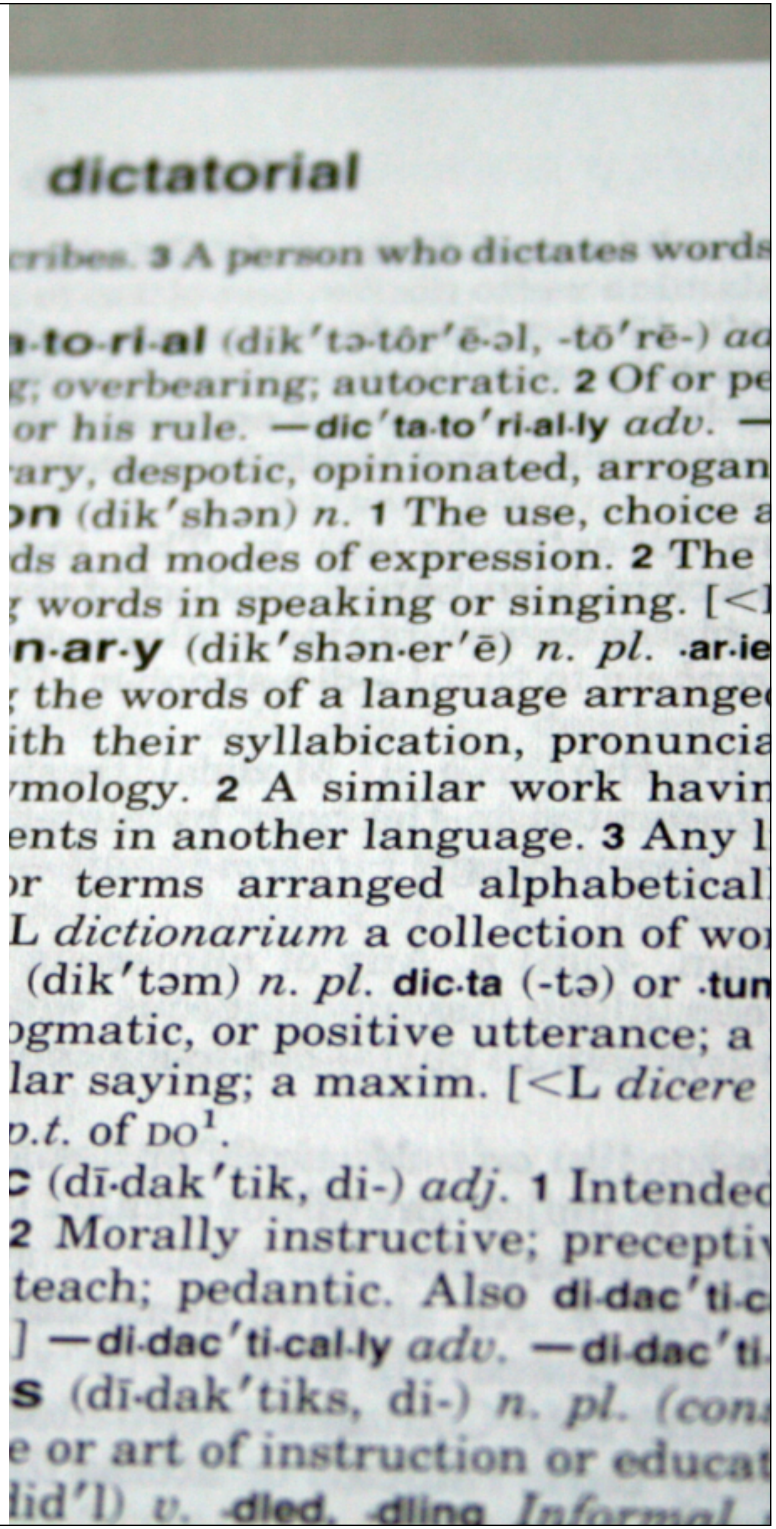
"The use of the crafts of typography, graphic design, animation, and cinematography with modern human-computer interaction and computer graphics technology to facilitate both the human understanding and effective use of computer software."

John Stasko, 1998



Software Visualization

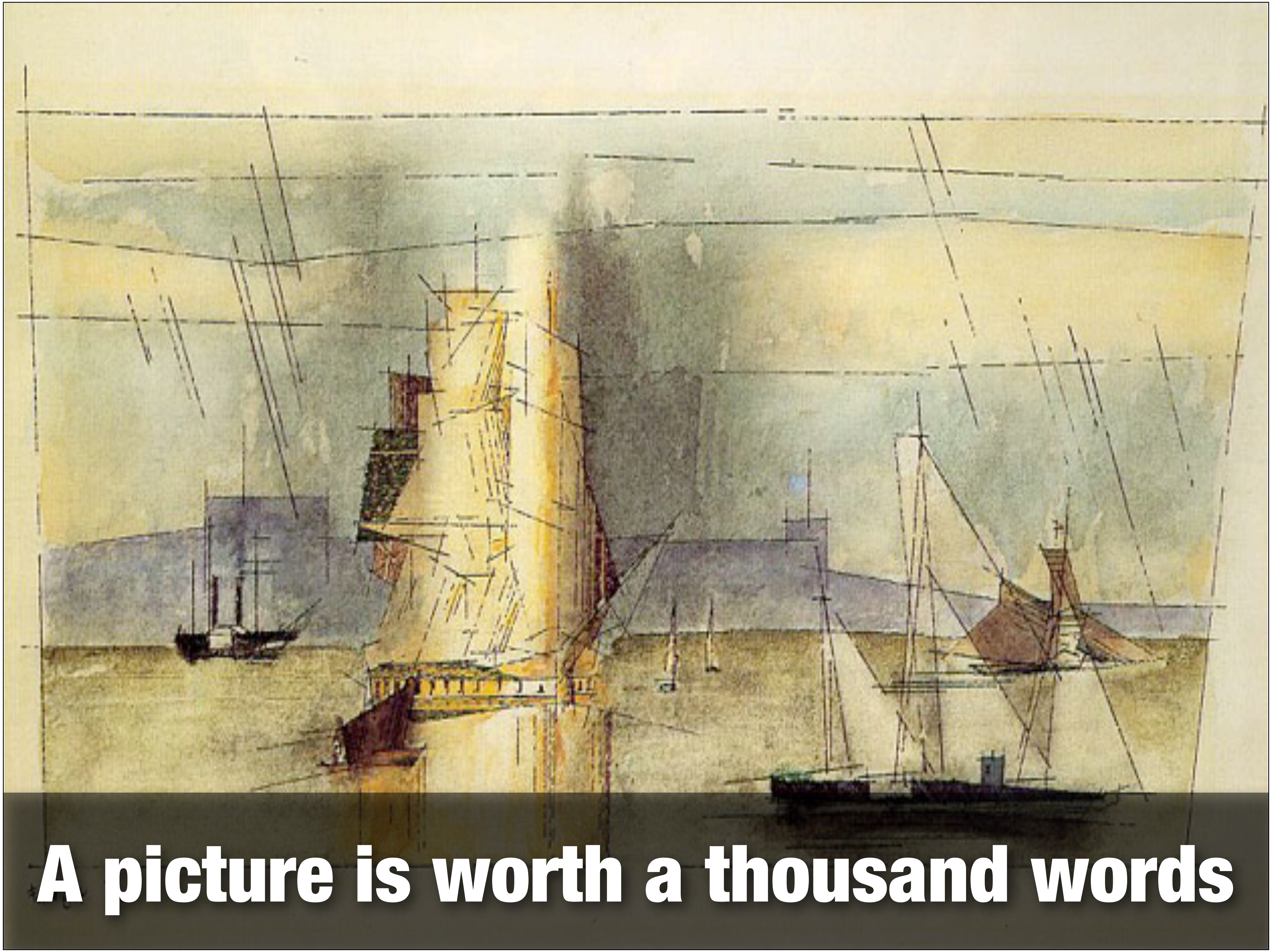
"The use of the crafts of typography, graphic design, animation, and cinematography with modern human-computer interaction and computer graphics technology to facilitate both the human understanding and effective use of computer software."

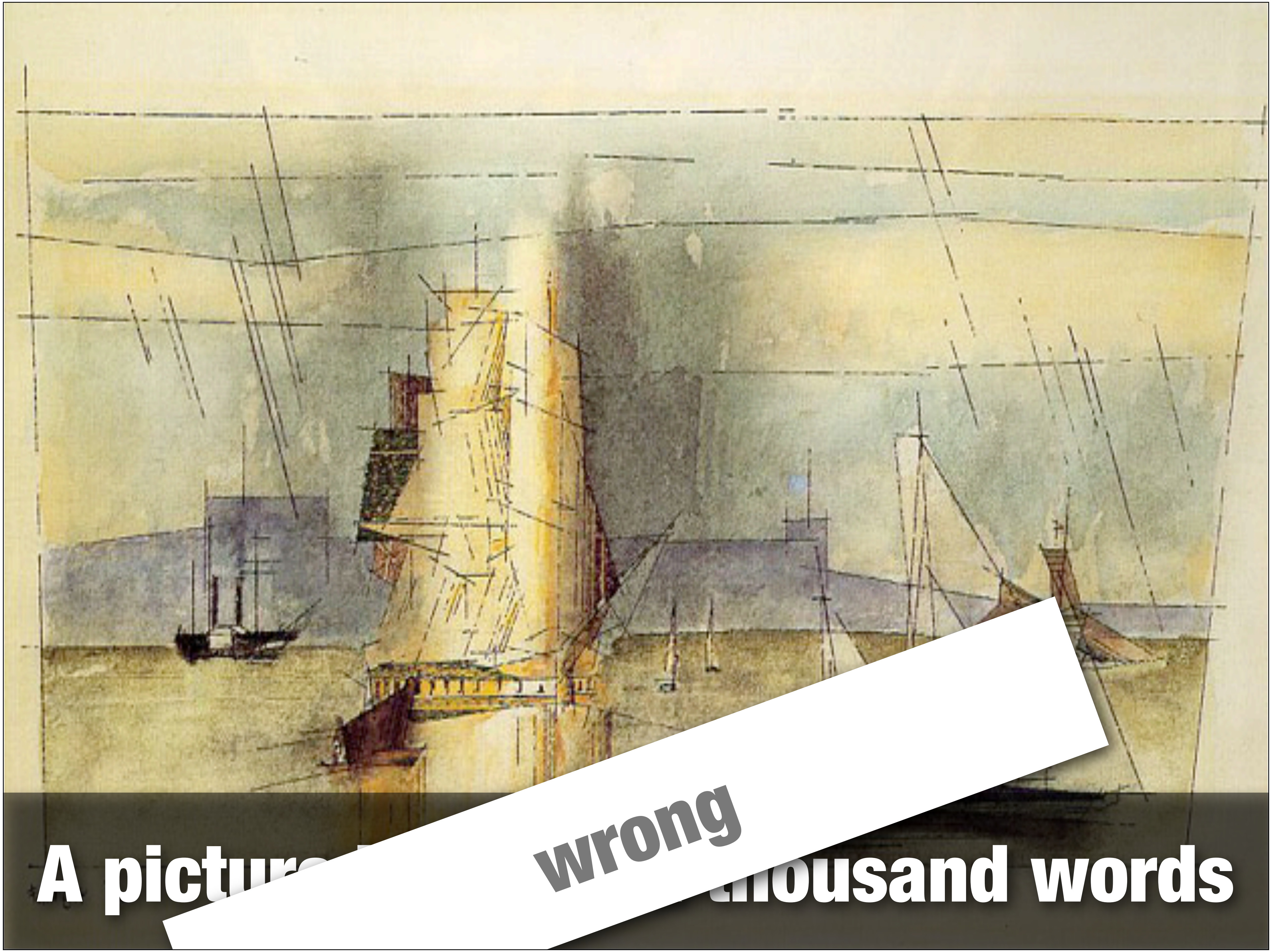


```
#include                                     <math.h>
#include                                   <sys/time.h>
#include                                   <X11/Xlib.h>
#include                                  <X11/keysym.h>
                                          double L ,o ,P
                                         ,_=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 (e,z,0,0),BlackPixel(e,0))
; scanf("%lf%lf%lf",y +n,w+y, y+s)+1; y ++); XSelectInput(e,z= XCreateSimpleWindow(e,z,0,0,400,400,
0,0,WhitePixel(e,0) ),KeyPressMask); for(XMapWindow(e,z); ; T=sin(O)){ struct timeval G={ 0,dt*1e6}
; K= cos(j); N=1e4; M+= H*_; Z=D*K; F+=_*P; r=E*K; W=cos( O); m=K*W; H=K*T; O+=D*_*F/ K+d/K*E*_; B=
sin(j); a=B*T*D-E*W; XClearWindow(e,z); t=T*E+ D*B*W; j+=d*_*D-_*F*E; P=W*E*B-T*D; for (o+=(I=D*W+E
*T*B,E*d/K *B+v+B/K*F*D)*_; p<y; ){ T=p[s]+i; E=c-p[w]; D=n[p]-L; K=D*m-B*T-H*E; if(p [n]+w[ p]+p[s
]== 0|K <fabs(W=T*r-I*E +D*P) |fabs(D=t *D+Z *T-a *E)> K)N=1e4; else{ q=W/K *4E2+2e2; C= 2E2+4e2/ K
 *D; N-1E4&& XDrawLine(e ,z,k,N ,U,q,C); N=q; U=C; } ++p; } L+=_* (X*t +P*M+m*l); T=X*X+ l*l+M *M;
  XDrawString(e,z,k ,20,380,f,17); D=v/l*15; i+=(B *l-M*r -X*Z)*_; for(; XPending(e); u *=CS!=N){
                                   XEvent z; XNextEvent(e ,&z);
                                       ++*((N=XLookupKeysym
                                         (&z.xkey,0))-IT?
                                         N-LT? UP-N?& E:&
                                         J:& u: &h); --*(
                                         DN -N? N-DT ?N==
                                         RT?&u: & W:&h:&J
                                          ); } m=15*F/l;
                                          c+=(I=M/ l,l*H
                                          +I*M+a*X)*_; H
                                          =A*r+v*X-F*l+(
                                          E=.1+X*4.9/l,t
                                          =T*m/32-I*T/24
                                           )/S; K=F*M+(
                                           h* 1e4/l-(T+
                                           E*5*T*E)/3e2
                                           )/S-X*d-B*A;
                                           a=2.63 /l*d;
                                           X+=( d*l-T/S
                                            *(.19*E +a
                                            *.64+J/1e3
                                            )-M* v +A*
                                            Z)*_; l +=
                                            K *_; W=d;
                                            sprintf(f,
                                            "%5d  %3d"
                                            "%7d",p =l
                                           /1.7,(C=9E3+
                              O*57.3)%0550,(int)i); d+=T*(.45-14/l*
                             X-a*130-J* .14)*_/125e2+F*_*v; P=(T*(47
                             *I-m* 52+E*94 *D-t*.38+u*.21*E) /1e2+W*
                             179*v)/2312; select(p=0,0,0,0,&G); v-=(
                              W*F-T*(.63*m-I*.086+m*E*19-D*25-.11*u
                               )/107e2)*_; D=cos(o); E=sin(o); } }
```

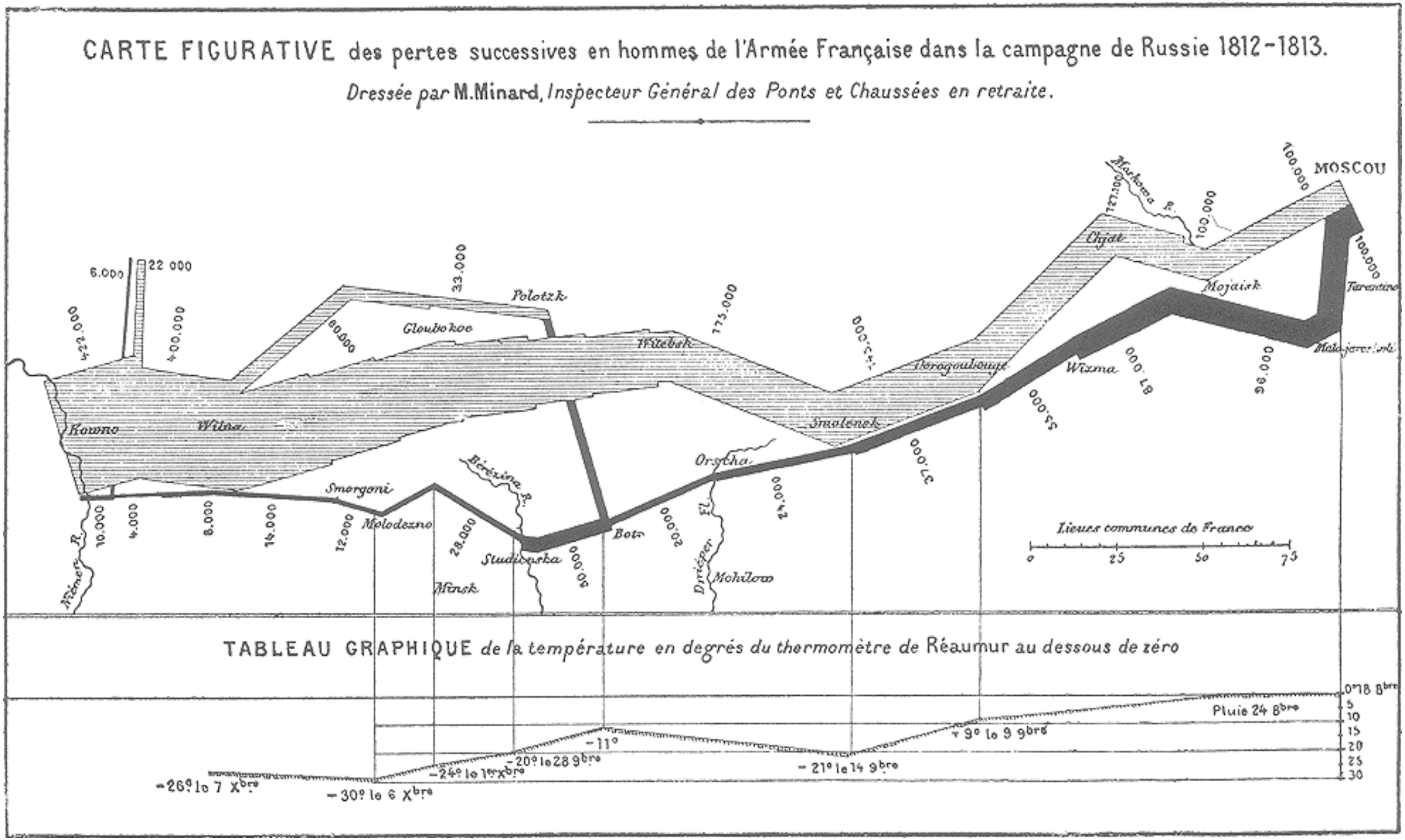
```
#include                                        <math.h>
#include                                      <sys/time.h>
#include                                      <X11/Xlib.h>
#include                                     <X11/keysym.h>
                                             double L ,o ,P
                                            ,_=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                        ,0))
; scanf("%lf%lf%lf",y +n,w+y, y+s)+1; y ++); XSelectInput(e,z= XCrea                          400,
0,0,WhitePixel(e,0) ),KeyPressMask); for(XMapWindow(e,z); ; T=si                              1e6}
; K= cos(j); N=1e4; M+= H*_; Z=D*K; F+=_*P; r=E*K; W=cos( O)                                  B=
sin(j); a=B*T*D-E*W; XClearWindow(e,z); t=T*E+ D*B*W; j                                       W+E
*T*B,E*d/K *B+v+B/K*F*D)*_; p<y; ){ T=p[s]+i; E=c-p                                           [s
]== 0|K <fabs(W=T*r-I*E +D*P) |fabs(D=t *D+Z *T                                           e2/ K
 *D; N-1E4&& XDrawLine(e ,z,k,N ,U,q,C); N=                                           1+M *M;
  XDrawString(e,z,k ,20,380,f,17); D=v/                                          u *=CS!=N){

                                               /l;
                                              / l,l*H
                                           M+a*X)*_; H
                                            =A*r+v*X-F*l+(
                                            E=.1+X*4.9/l,t
                                            =T*m/32-I*T/24
                                             )/S; K=F*M+(
                                             h* 1e4/l-(T+
                                             E*5*T*E)/3e2
                                             )/S-X*d-B*A;
                                             a=2.63 /l*d;
                                             X+=( d*l-T/S
                                              *(.19*E +a
                                              *.64+J/1e3
                                              )-M* v +A*
                                              Z)*_; l +=
                                              K *_; W=d;
                                              sprintf(f,
                                              "%5d  %3d"
                                              "%7d",p =l
                                             /1.7,(C=9E3+
                                O*57.3)%0550,(int)i); d+=T*(.45-14/l*
                               X-a*130-J* .14)*_/125e2+F*_*v; P=(T*(47
                               *I-m* 52+E*94 *D-t*.38+u*.21*E) /1e2+W*
                               179*v)/2312; select(p=0,0,0,0,&G); v-=(
                                W*F-T*(.63*m-I*.086+m*E*19-D*25-.11*u
                                 )/107e2)*_; D=cos(o); E=sin(o); } }
```

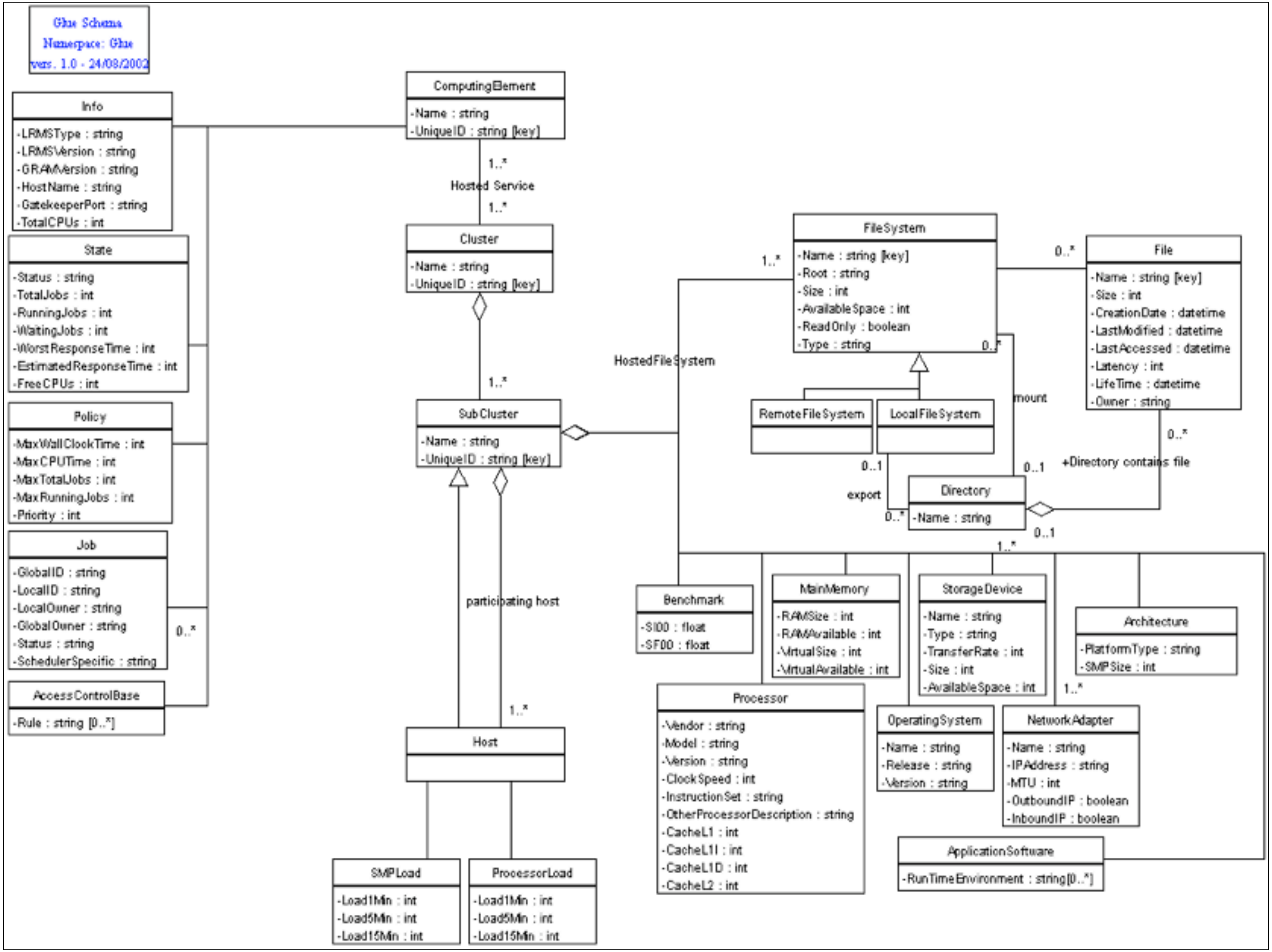
not software visualization


A picture is worth a thousand words

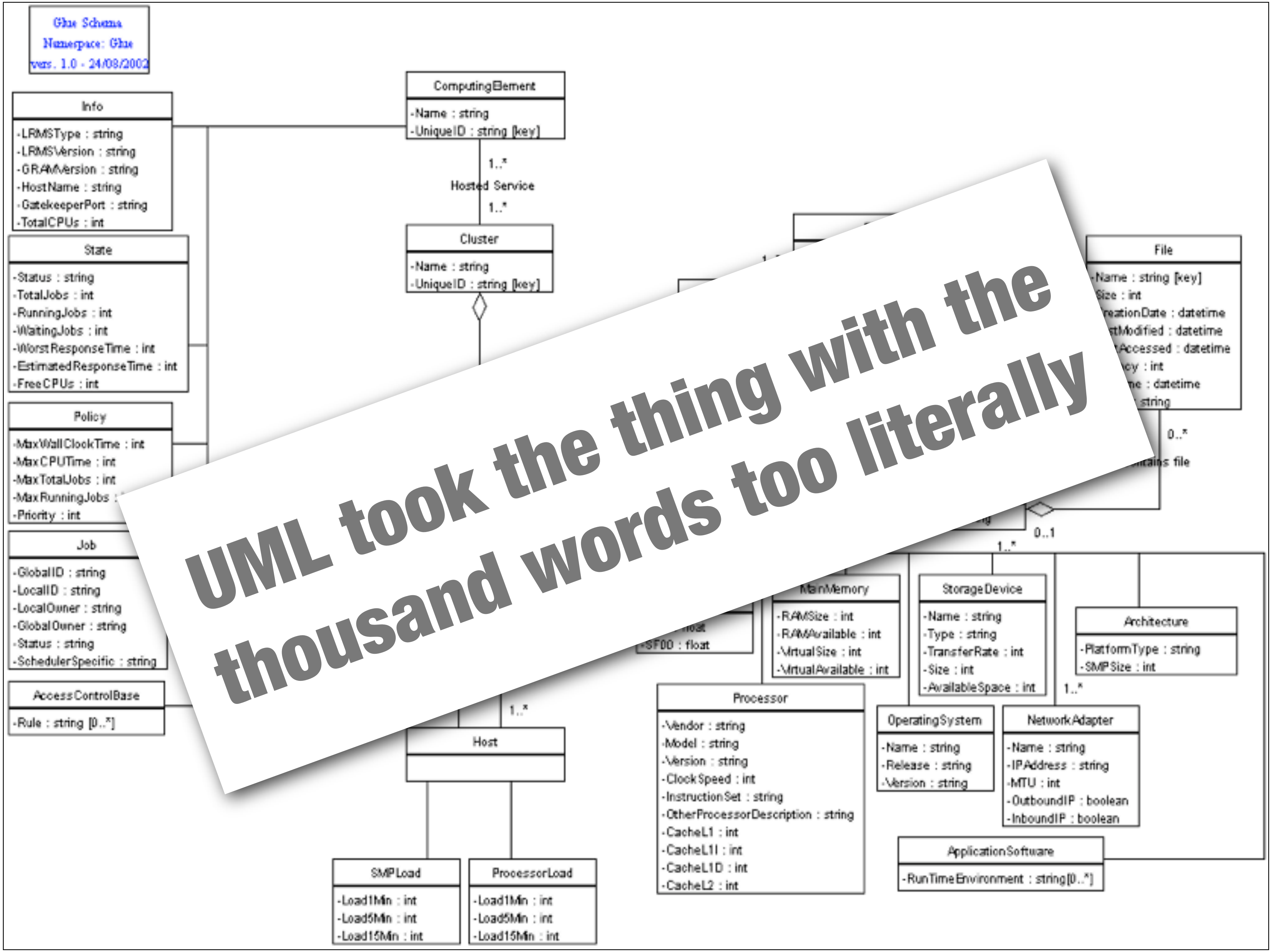

A pictur
wrong
thousand words


Visualization is about stories


CARTE FIGURATIVE des pertes successives en hommes de l'Armée Française dans la campagne de Russie 1812-1813.
Dressée par M. Minard, Inspecteur Général des Ponts et Chaussées en retraite.
MOSCOU
100.000
127.100
Moskowa R.
100.000
Chjat
100.000
Tarentino
Mojaisk
Malo-jarosewli
96.000
87.000
Wixma
55.000
Dorogobouge
145.000
Smolensk
37.000
175.000
Witebsk
Polotzk
33.000
Gloubokoe
60.000
6.000
22 000
422.000
400.000
Kowno
Wilna
Orscha
24.000
Smorgoni
Molodezno
12.000
14.000
8.000
4.000
10.000
Niemen R.
28.000
Bérézina R.
Studienska
50.000
Botr
20.000
Dniéper Fl.
Mohilow
Minsk
Lieues communes de France
0
25
50
75
TABLEAU GRAPHIQUE de la température en degrés du thermomètre de Réaumur au dessous de zéro
0° le 18 8bre
Pluie 24 8bre
5
10
15
20
25
30
+ 9° le 9 9bre
− 21° le 14 9bre
− 11°
− 20° le 28 9bre
− 24° le 1er Xbre
− 30° le 6 Xbre
− 26° le 7 Xbre



Glue Schema
Namespace: Glue
vers. 1.0 - 24/08/2002
ComputingElement
-Name : string
-UniqueID : string [key]
Info
-LRMSType : string
-LRMSVersion : string
-GRAMVersion : string
-HostName : string
-GatekeeperPort : string
-TotalCPUs : int
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
AccessControlBase
-Rule : string [0..*]
0..*
1..*
Hosted Service
1..*
Cluster
-Name : string
-UniqueID : string [key]
1..*
SubCluster
-Name : string
-UniqueID : string [key]
participating host
1..*
Host
SMPLoad
-Load1Min : int
-Load5Min : int
-Load15Min : int
ProcessorLoad
-Load1Min : int
-Load5Min : int
-Load15Min : int
HostedFileSystem
1..*
FileSystem
-Name : string [key]
-Root : string
-Size : int
-AvailableSpace : int
-ReadOnly : boolean
-Type : string
0..*
0..*
File
-Name : string [key]
-Size : int
-CreationDate : datetime
-LastModified : datetime
-LastAccessed : datetime
-Latency : int
-LifeTime : datetime
-Owner : string
RemoteFileSystem
LocalFileSystem
mount
0..1
0..1
export
0..*
Directory
-Name : string
0..*
+Directory contains file
0..1
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 the thing with the thousand words too literally

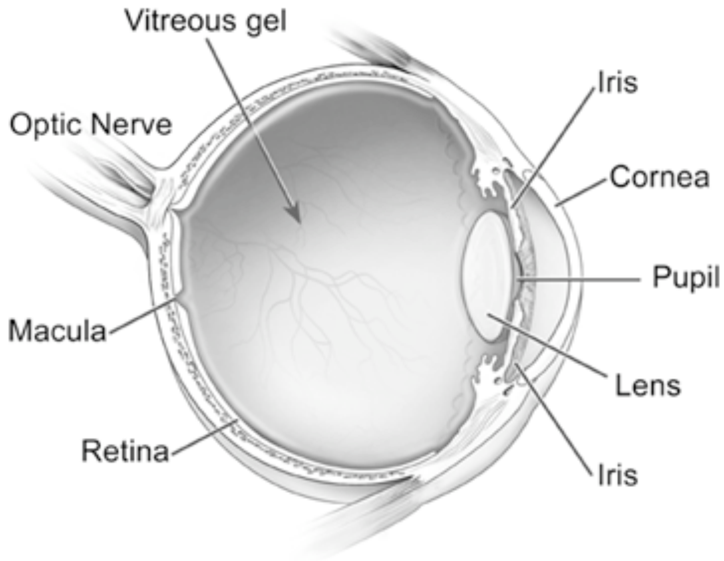
Part IV

Seeing


Seeing is Understanding

We are Visual Beings

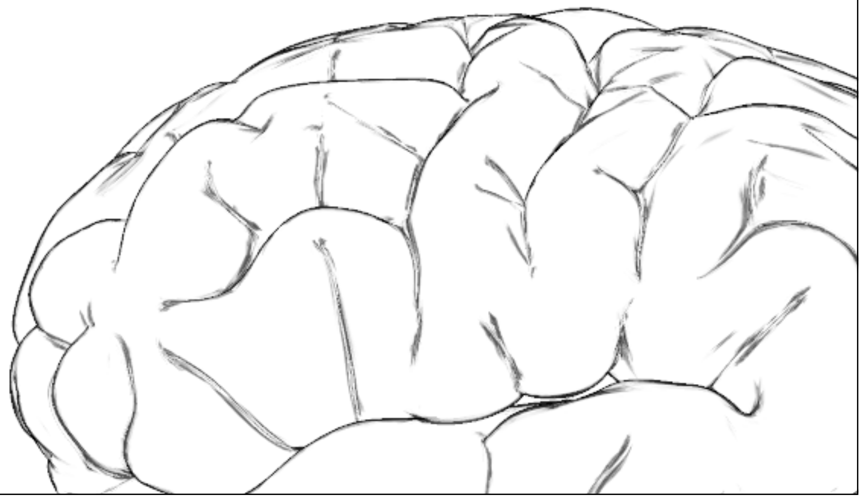
70% of all brain inputs
come through the eyes



We see with our Brain

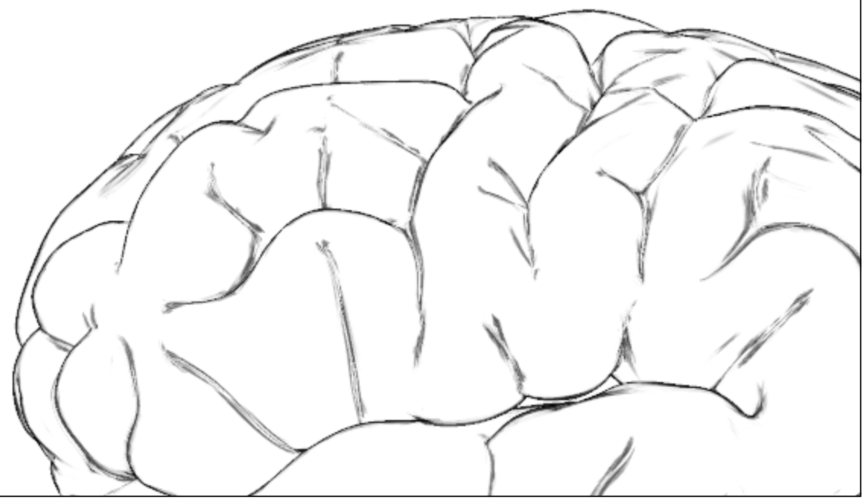
- 3 types of memory to process visual information
  - Iconic, the visual sensory register
  - Short-term, the working memory
  - (Long-term)

Iconic and Short-term Memory



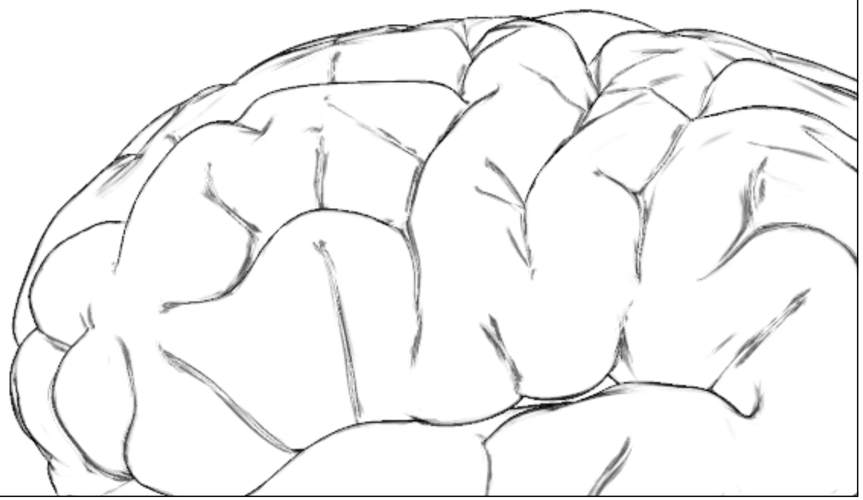
Iconic and Short-term Memory

- *Iconic Memory* is a buffer that retains information for less than 1 second before passing it to short-term memory



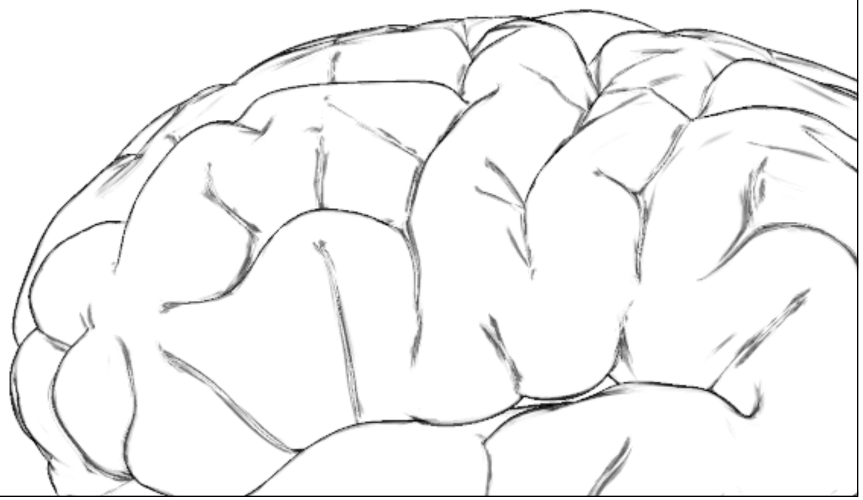
Iconic and Short-term Memory

- *Iconic Memory* is a buffer that retains information for less than 1 second before passing it to short-term memory
  - Perception of a limited set of attributes is very fast, automatic & subconscious, therefore called **pre-attentive**



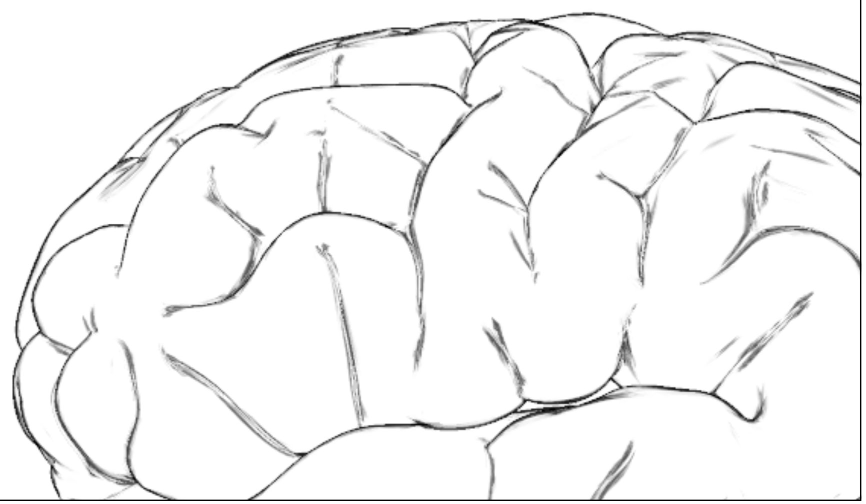
Iconic and Short-term Memory

- *Iconic Memory* is a buffer that retains information for less than 1 second before passing it to short-term memory
  - Perception of a limited set of attributes is very fast, automatic & subconscious, therefore called **pre-attentive**
- *Short-term Memory* processes information as “chunks”



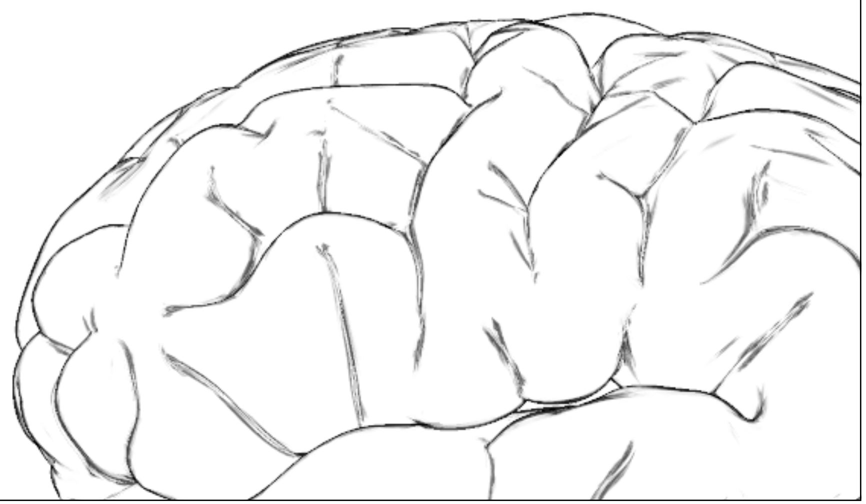
Iconic and Short-term Memory

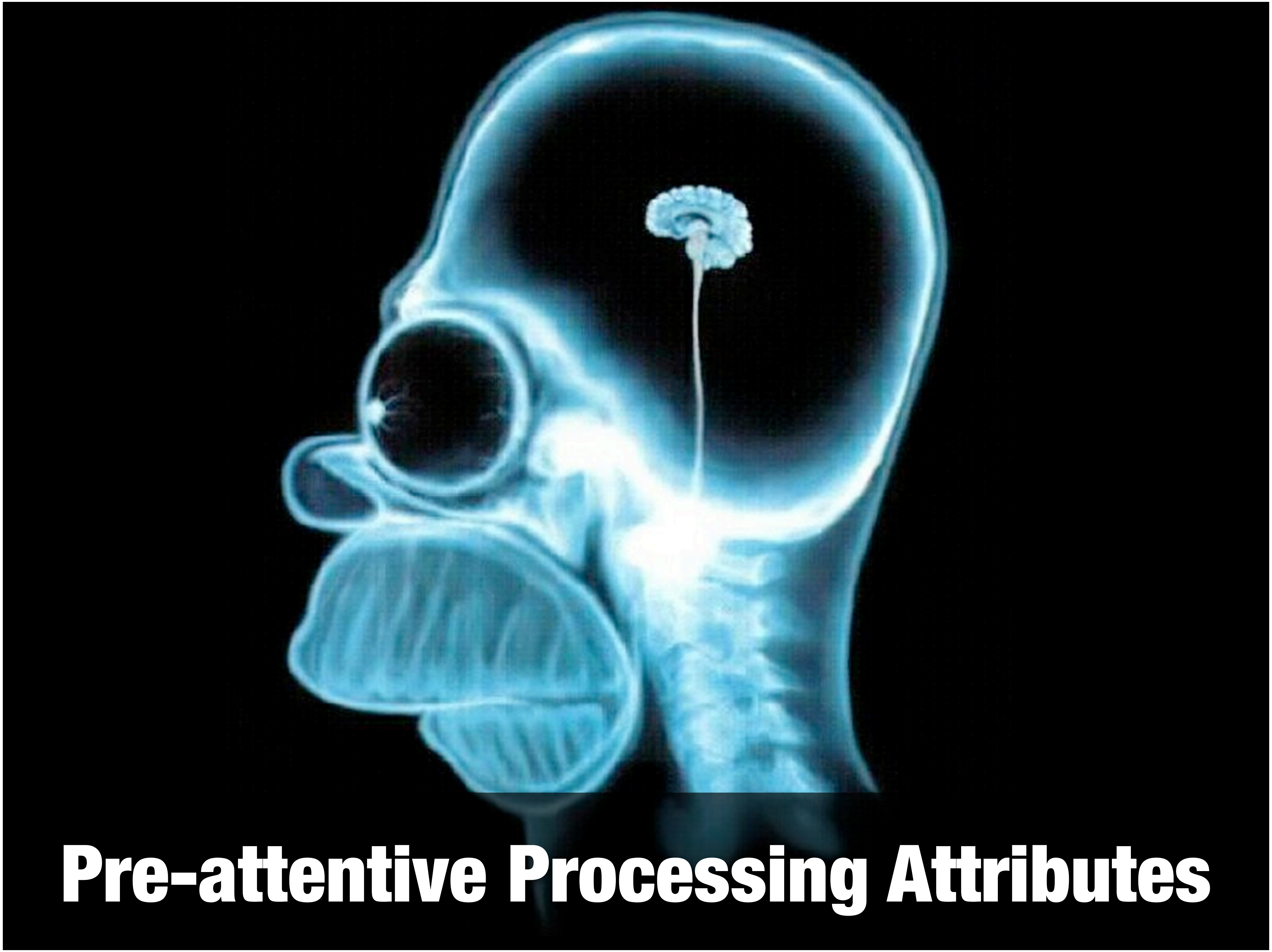
- *Iconic Memory* is a buffer that retains information for less than 1 second before passing it to short-term memory
  - Perception of a limited set of attributes is very fast, automatic & subconscious, therefore called **pre-attentive**
- *Short-term Memory* processes information as “chunks”
  - Storage is temporary and of limited capacity (3-9 chunks)



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 of a limited set of attributes is very fast, automatic & subconscious, therefore called **pre-attentive**
- *Short-term Memory* processes information as “chunks”
  - Storage is temporary and of limited capacity (3-9 chunks)
    - This explains why charts are more expressive than tables

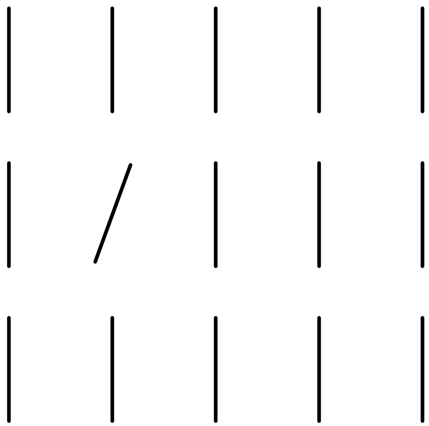



Pre-attentive Processing Attributes

Pre-attentive Attributes of Form

Orientation	Line Length	Line Width	Size
Shape	Curvature	Added Marks	Enclosure

Pre-attentive Attributes of Form



Line Length

Line Width

Size

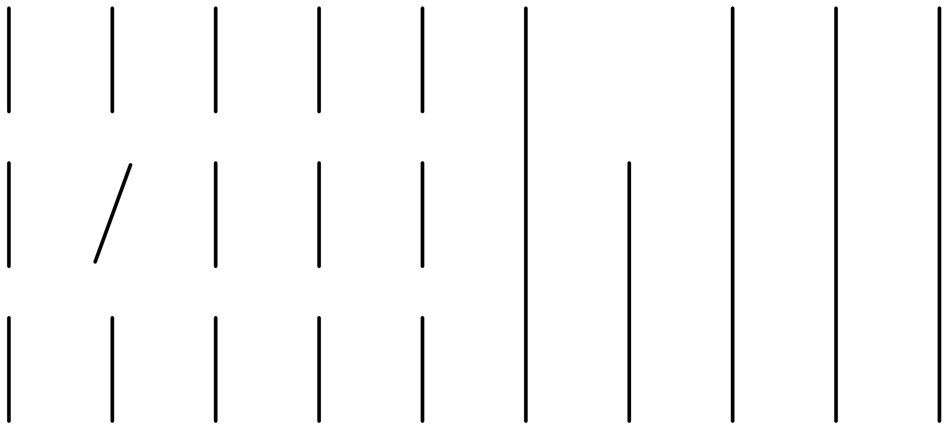
Shape

Curvature

Added Marks

Enclosure

Pre-attentive Attributes of Form



Line Width

Size

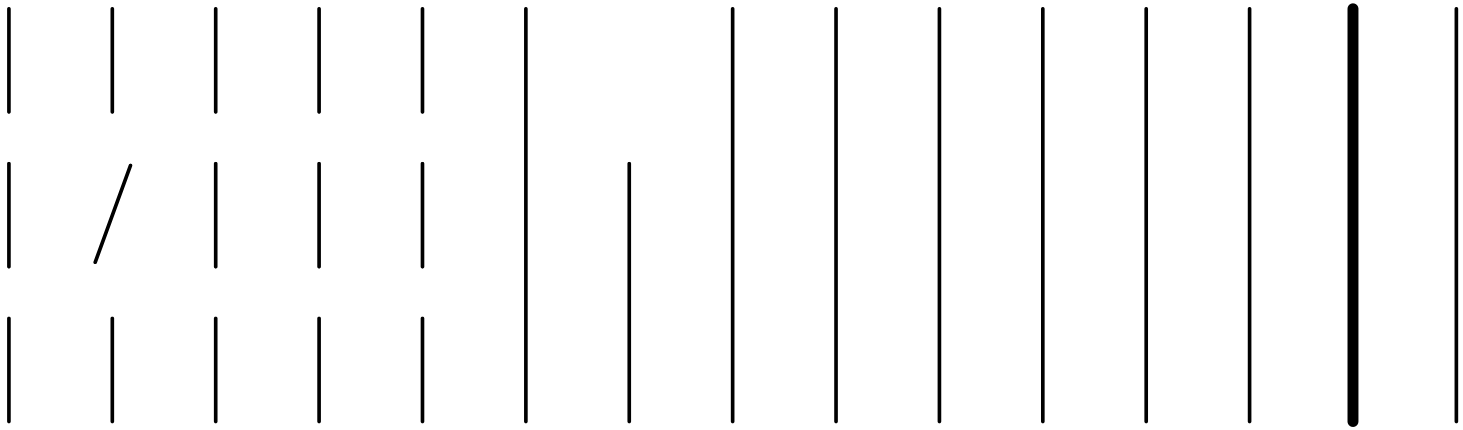
Shape

Curvature

Added Marks

Enclosure

Pre-attentive Attributes of Form



Size

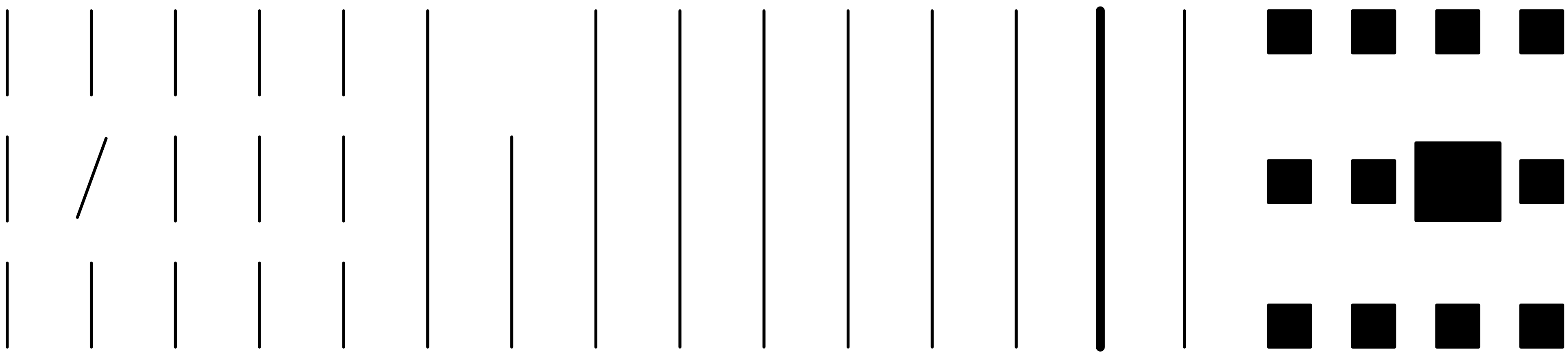
Shape

Curvature

Added Marks

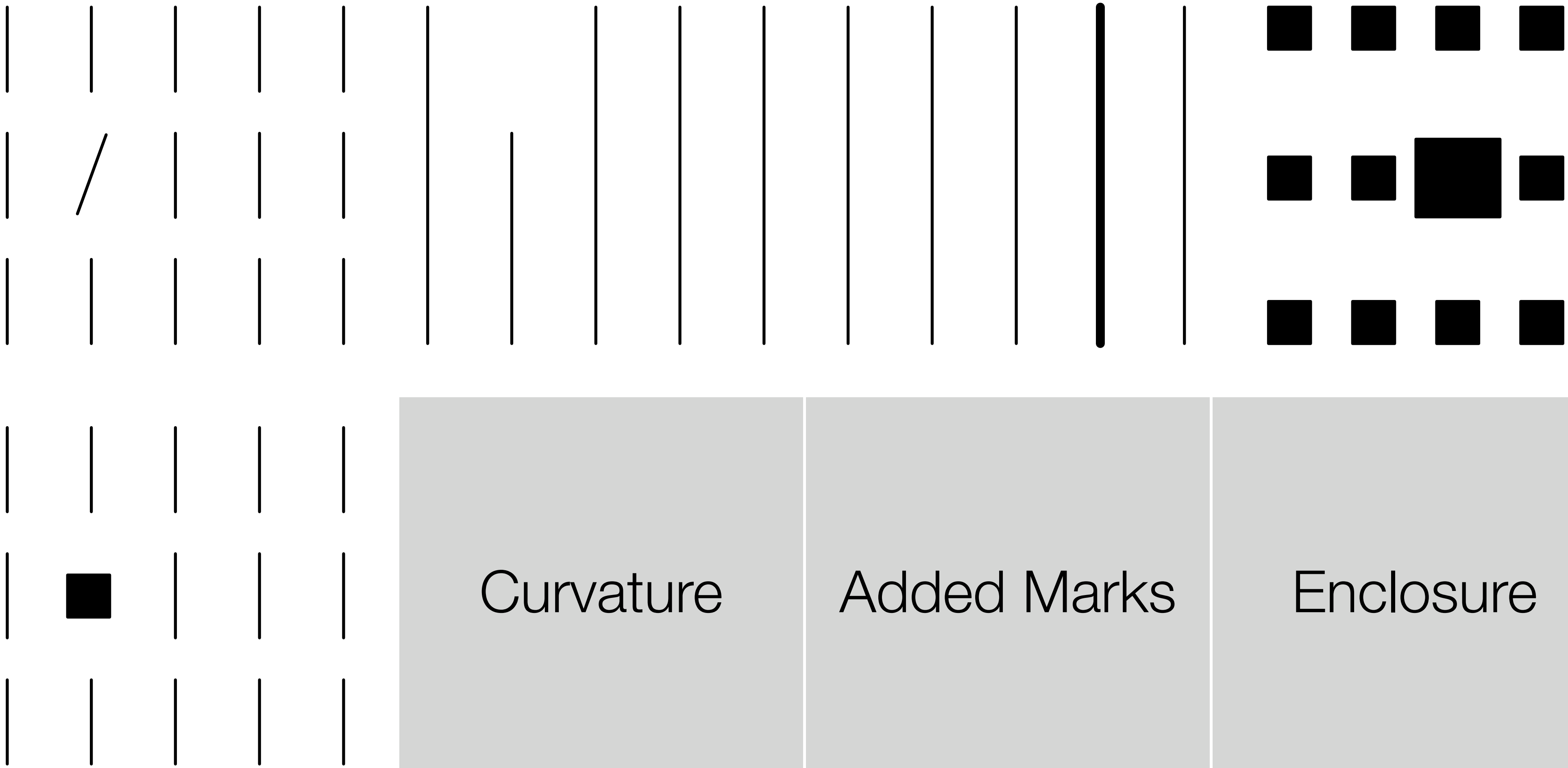
Enclosure

Pre-attentive Attributes of Form



Shape	Curvature	Added Marks	Enclosure

Pre-attentive Attributes of Form

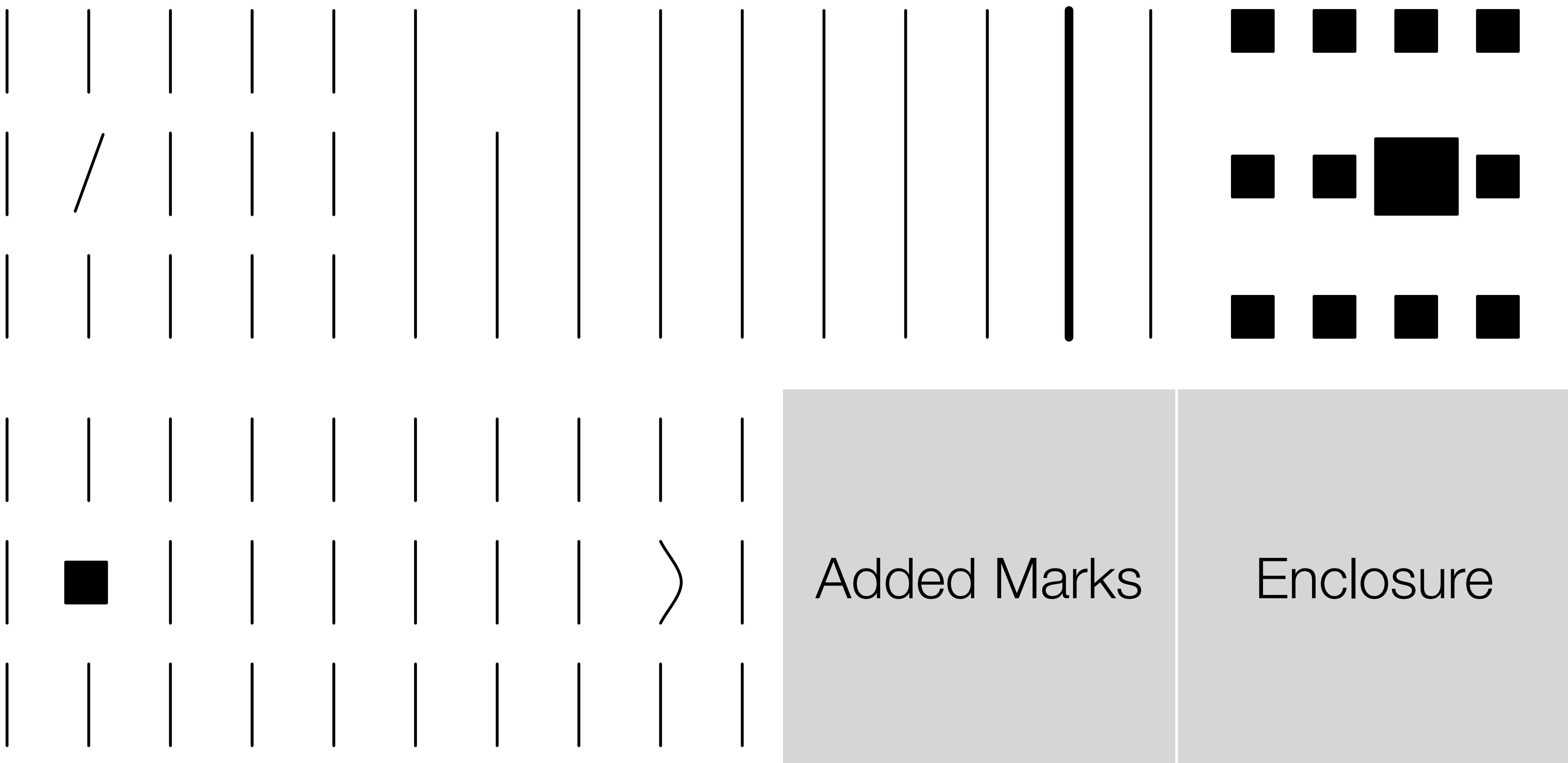


Curvature

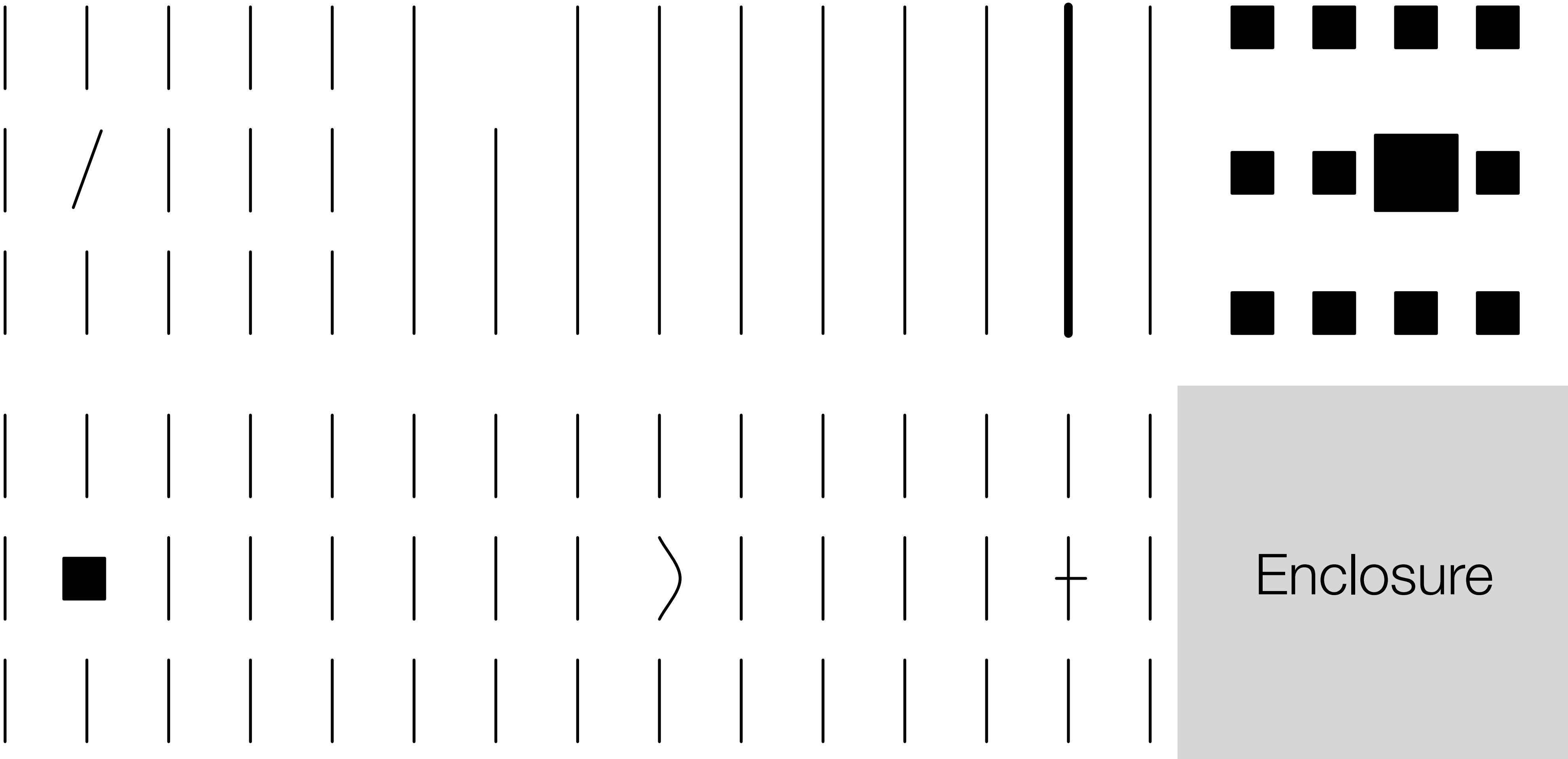
Added Marks

Enclosure

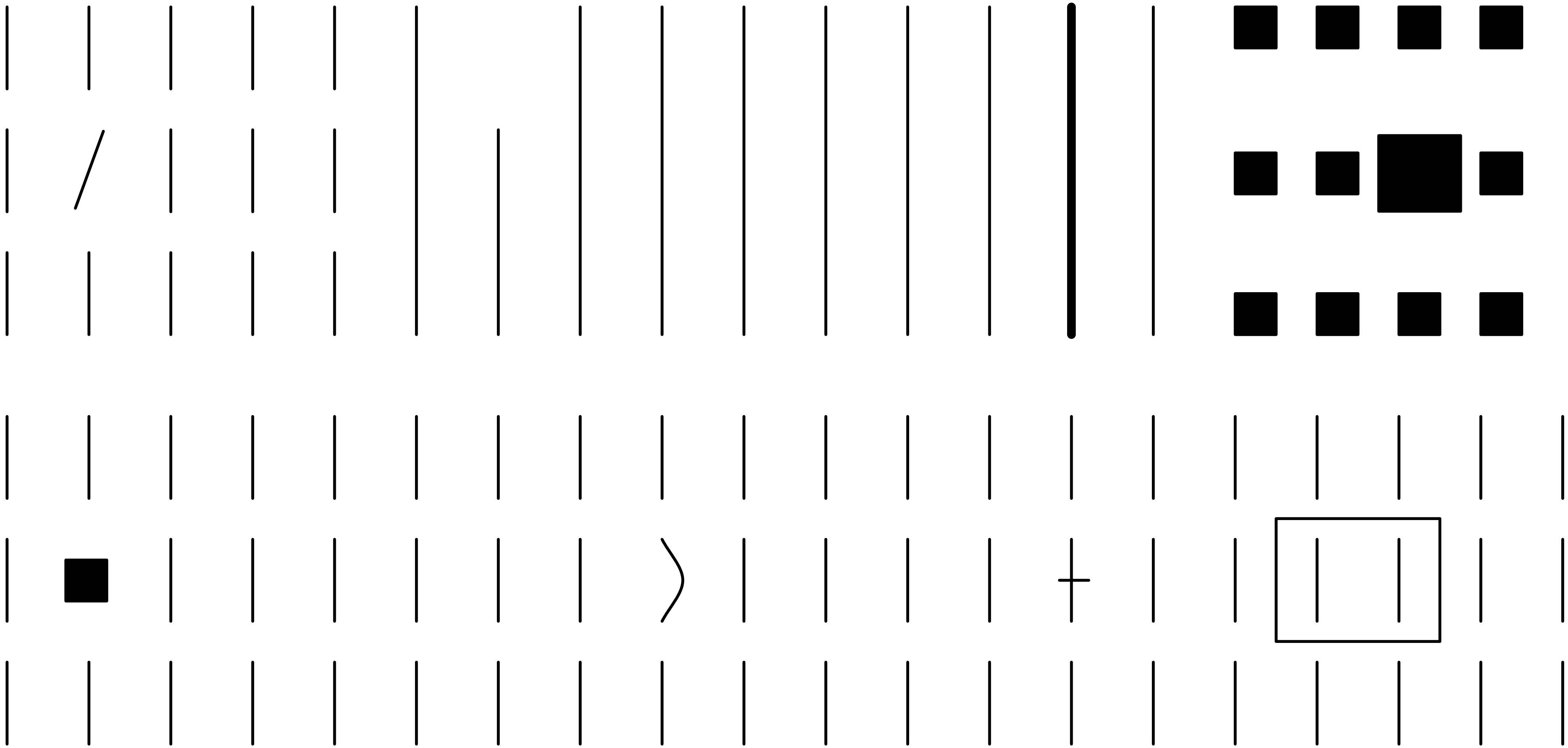
Pre-attentive Attributes of Form

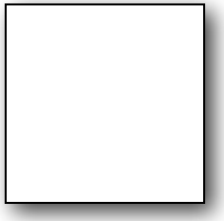


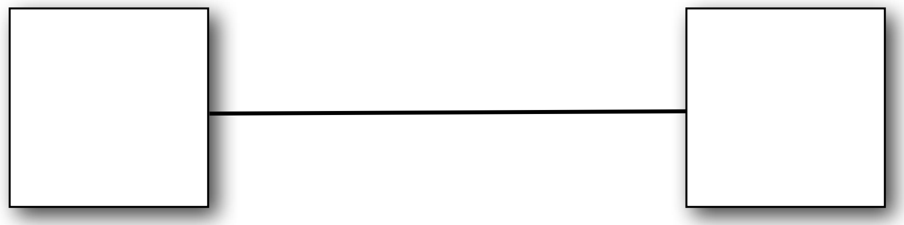
Pre-attentive Attributes of Form

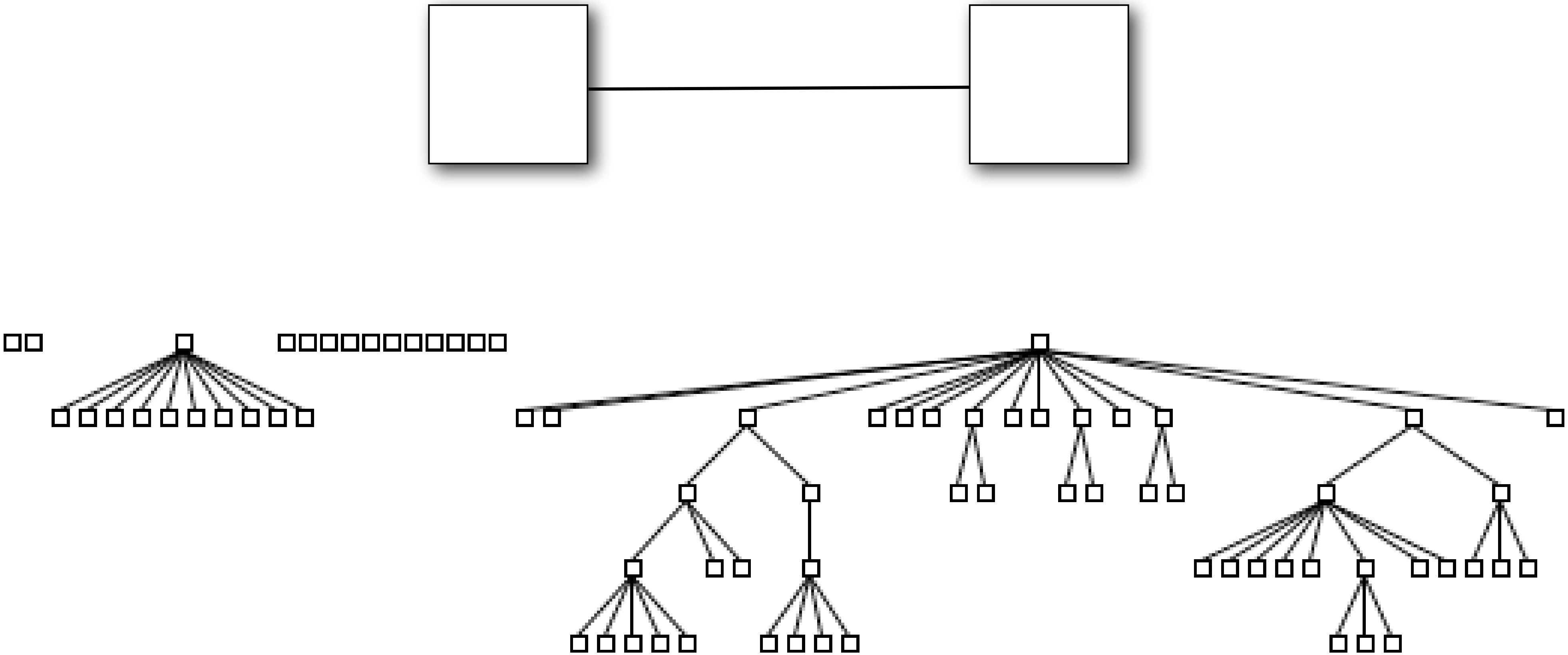


Pre-attentive Attributes of Form

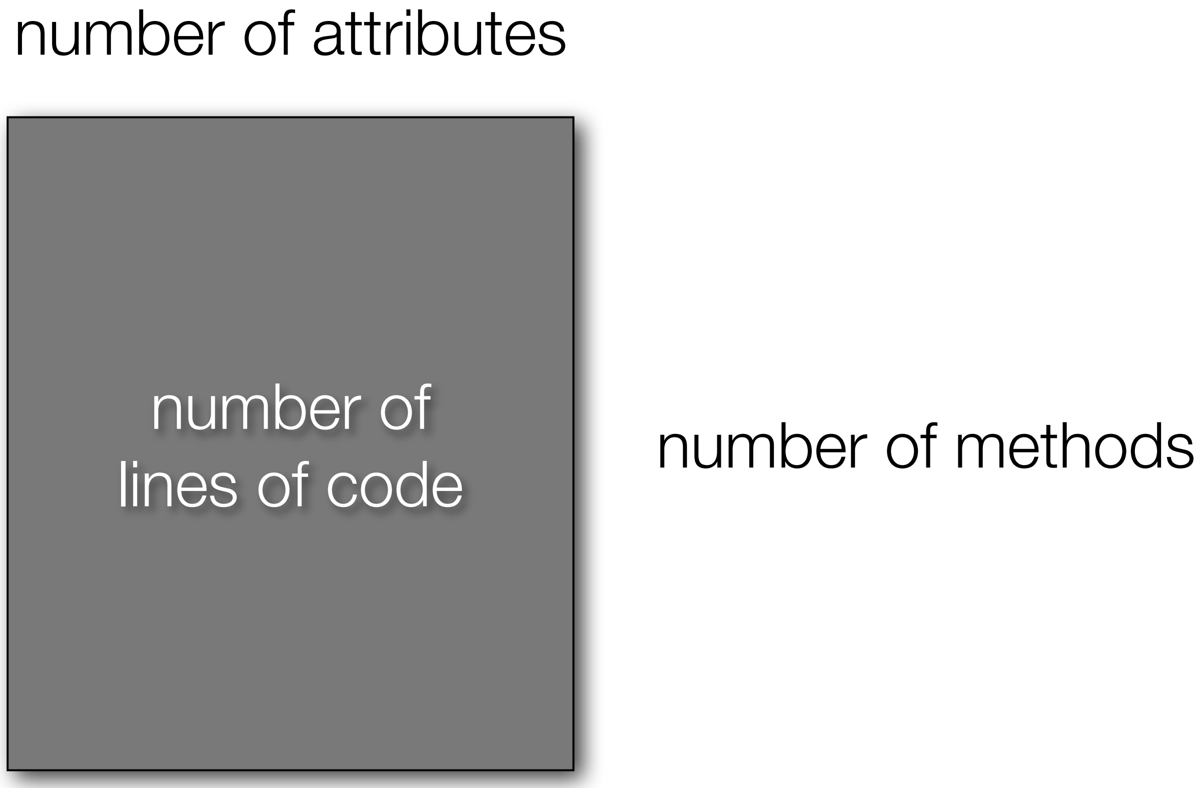


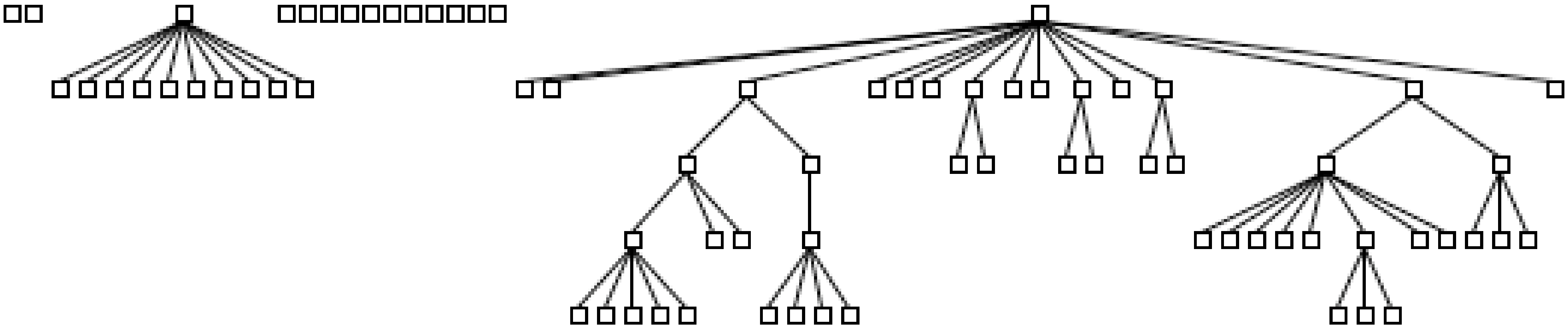


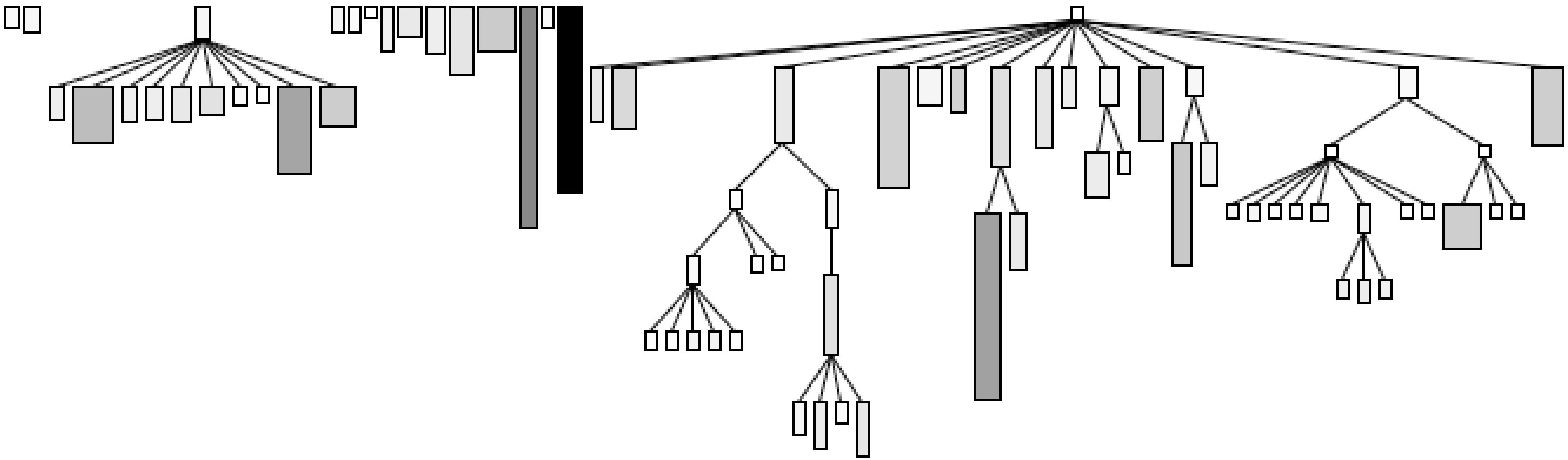


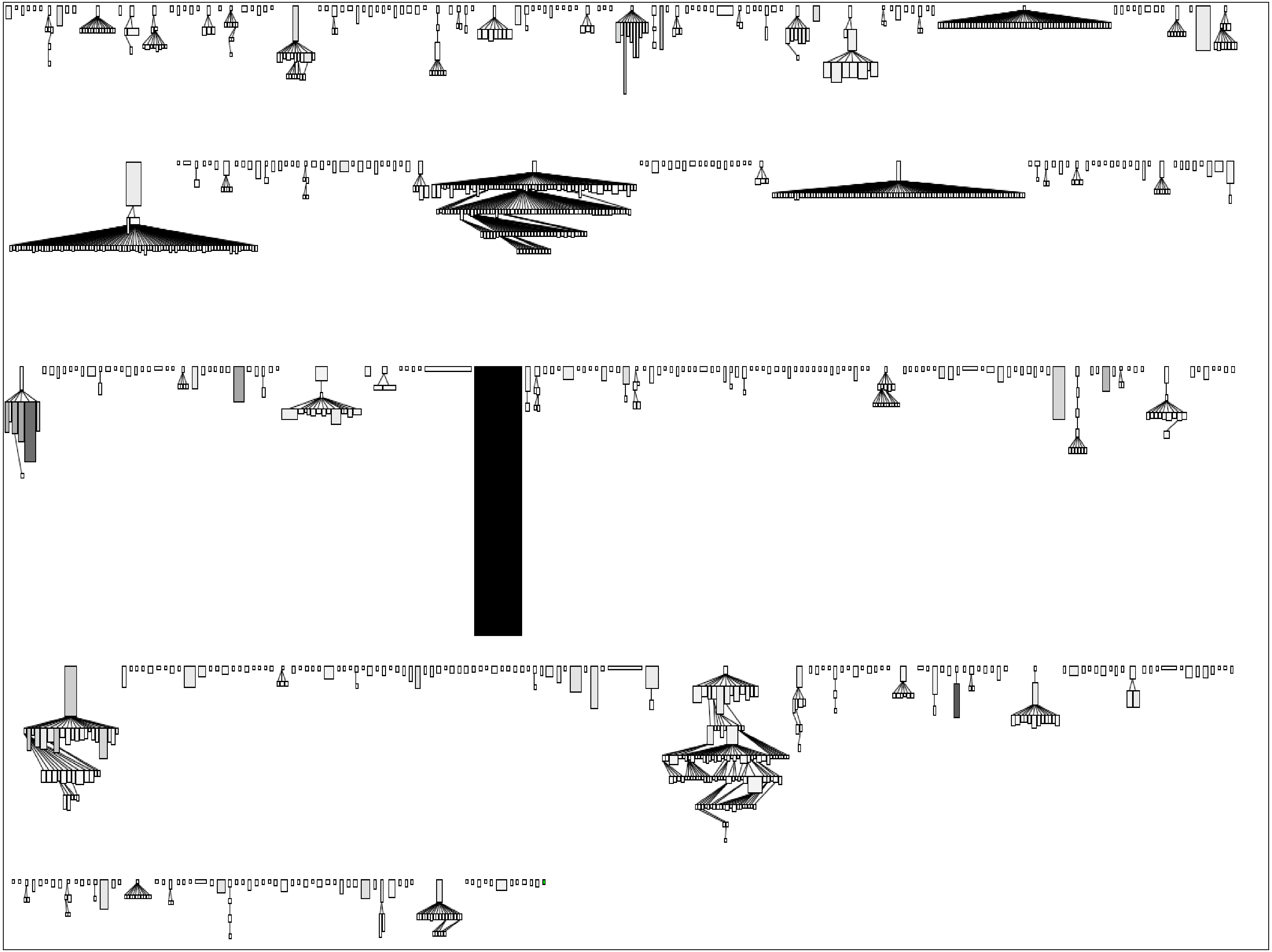


The Polymetric View Principle



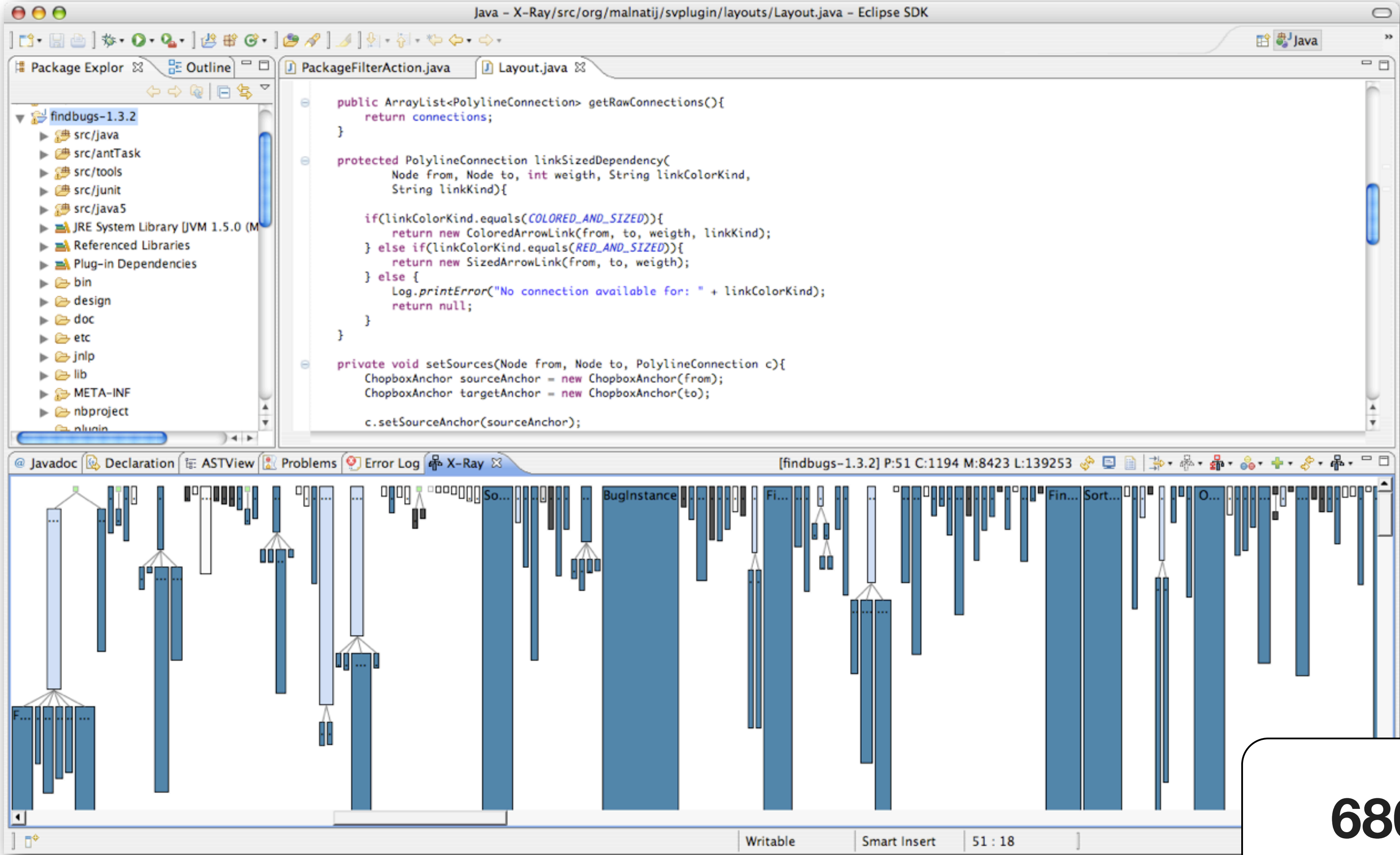






The X-Ray Eclipse Plugin

xray.inf.usi.ch



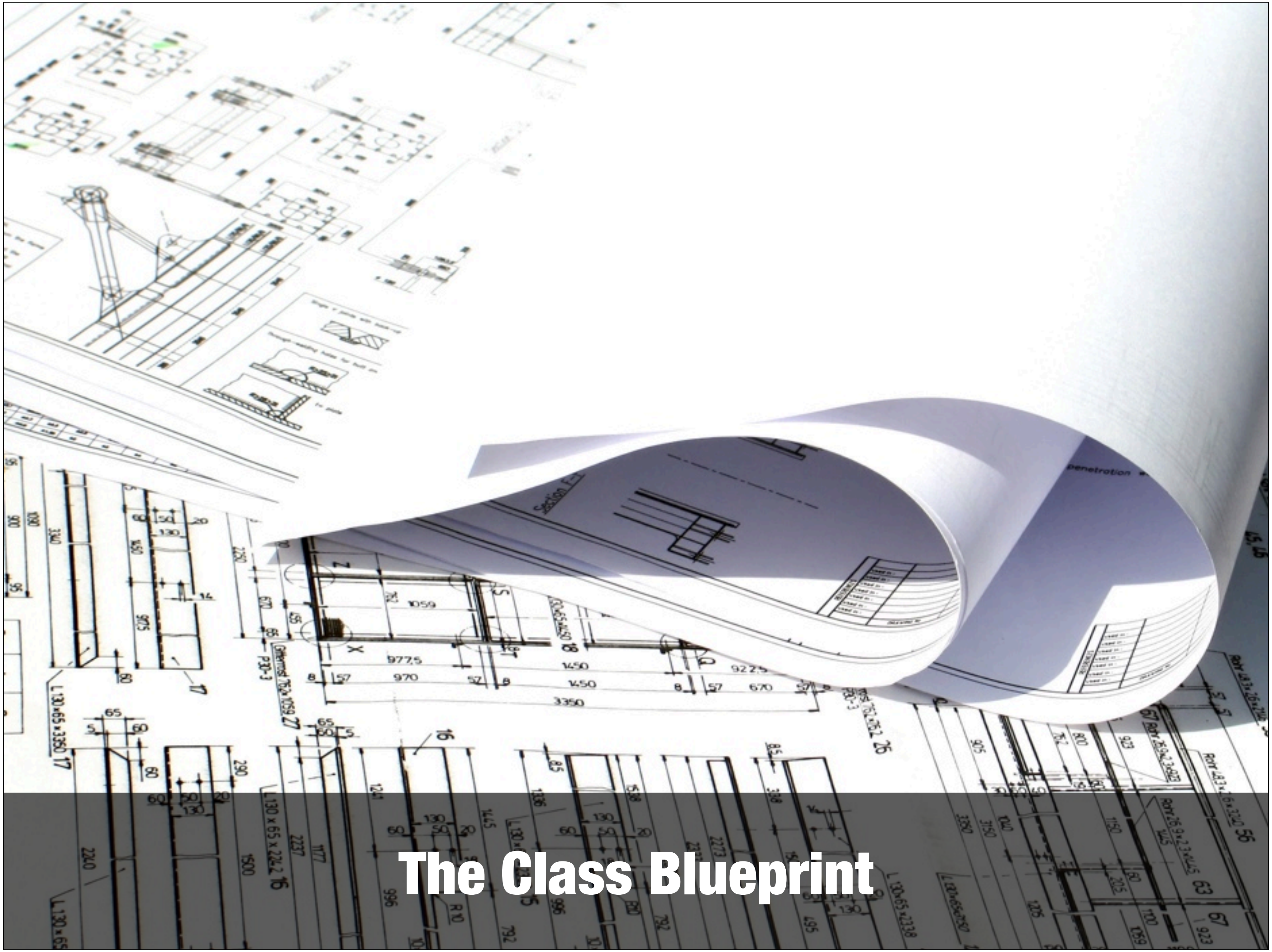
6800+ downloads

free

Released Nov 2007

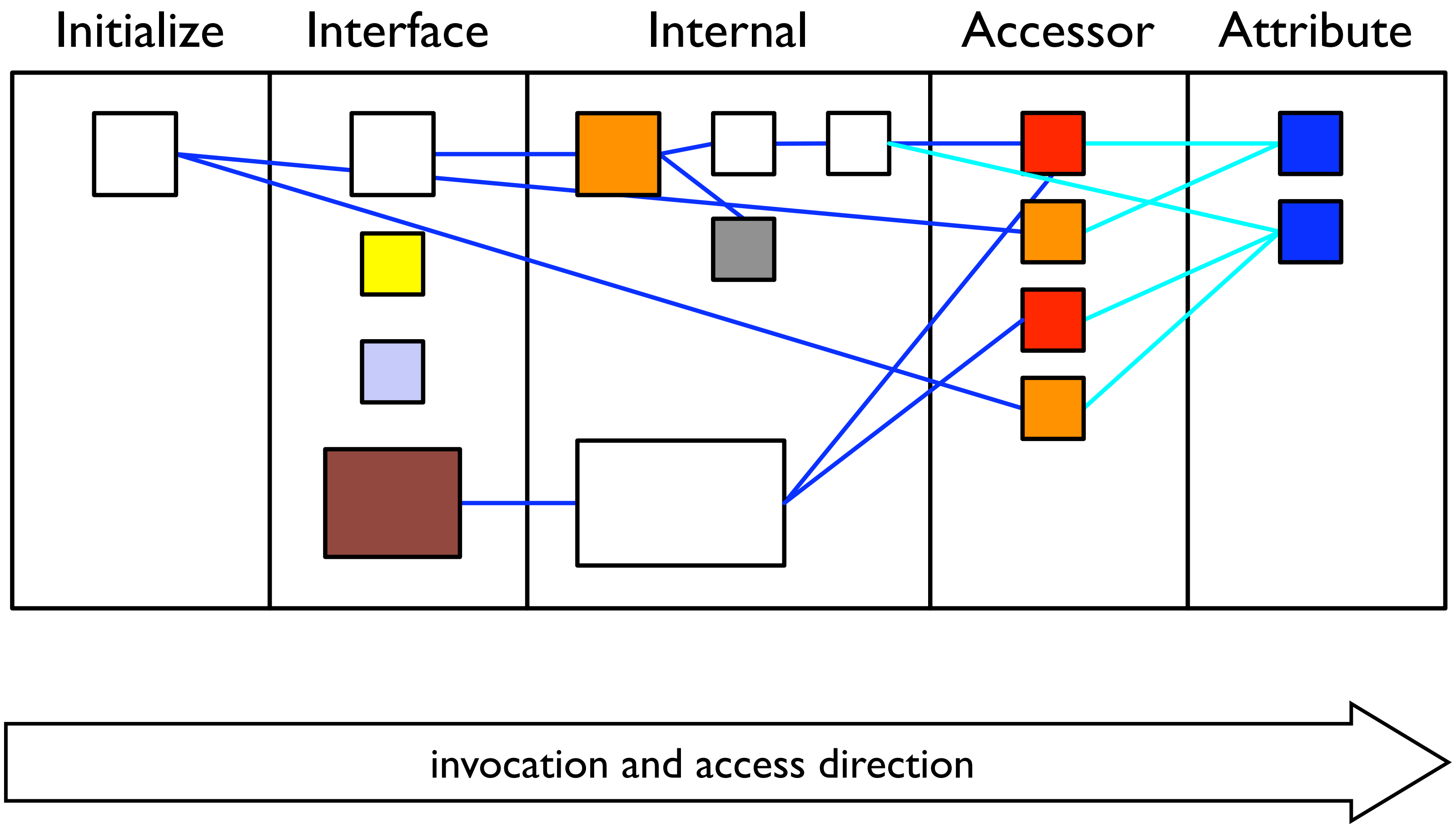
Part V

Software Visualization++

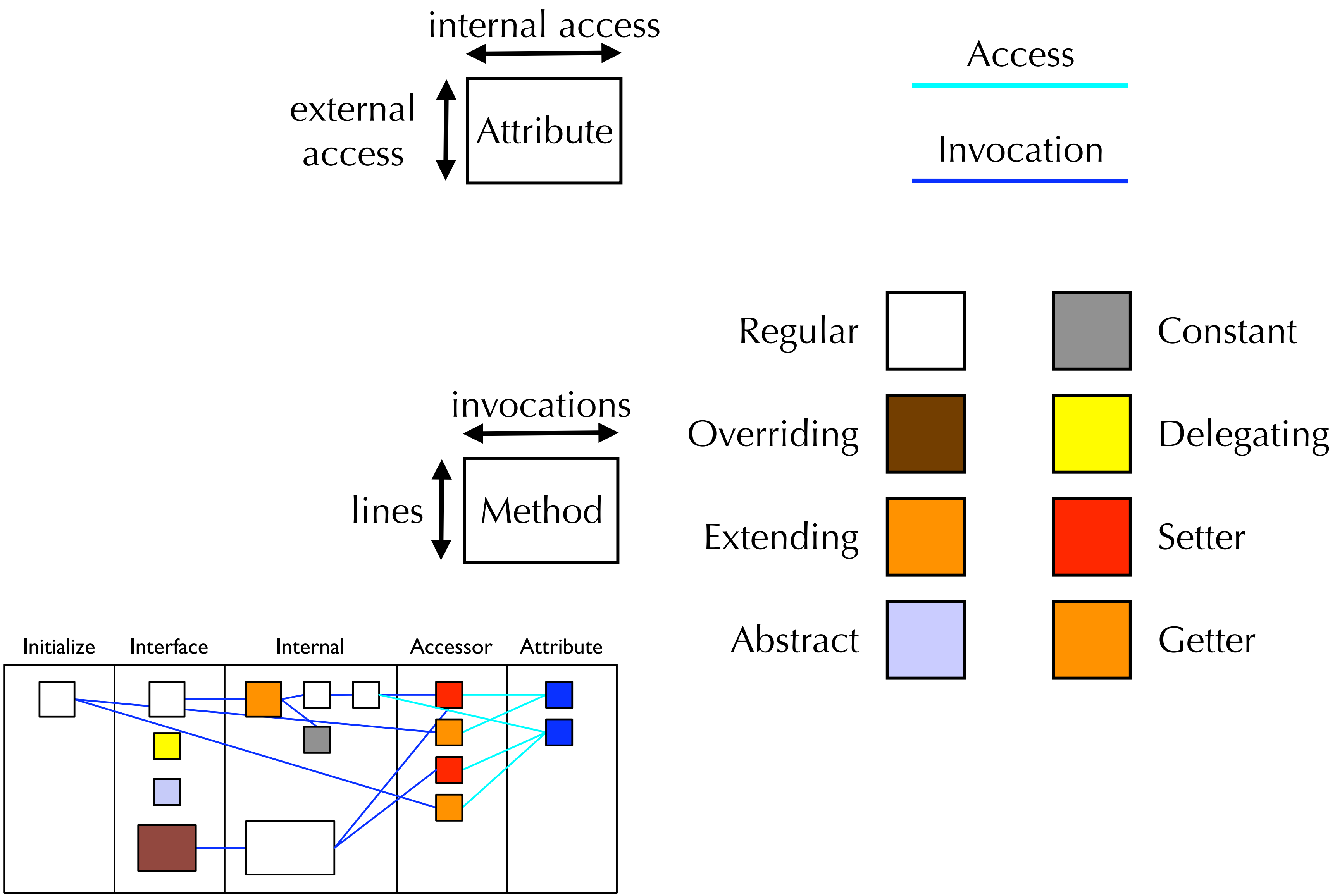


The Class Blueprint

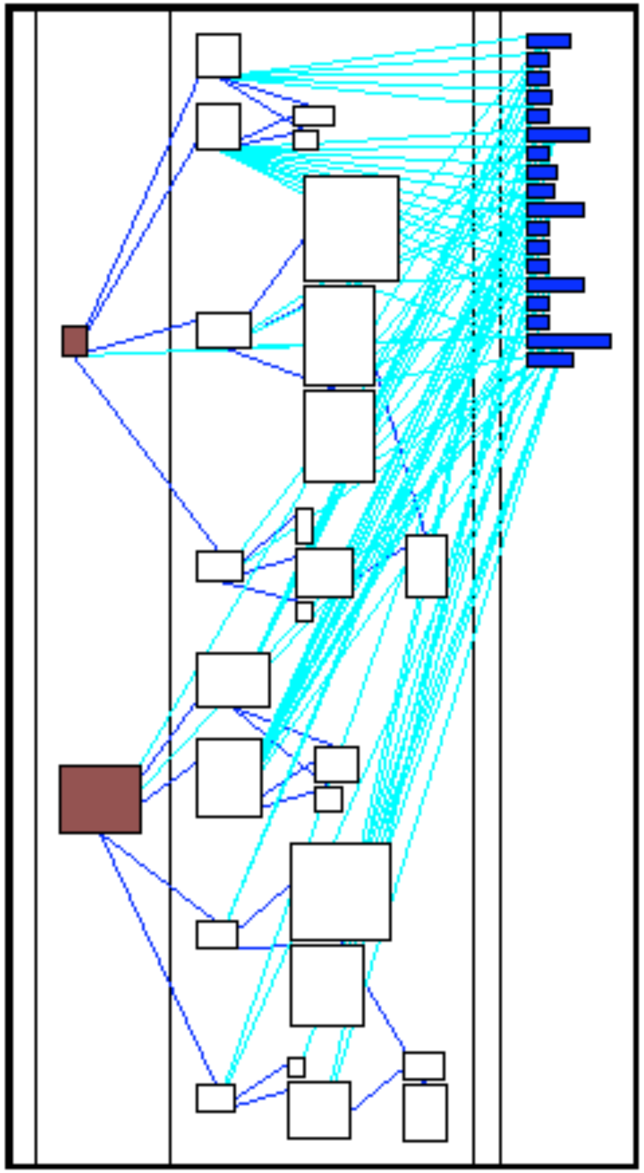
The Class Blueprint

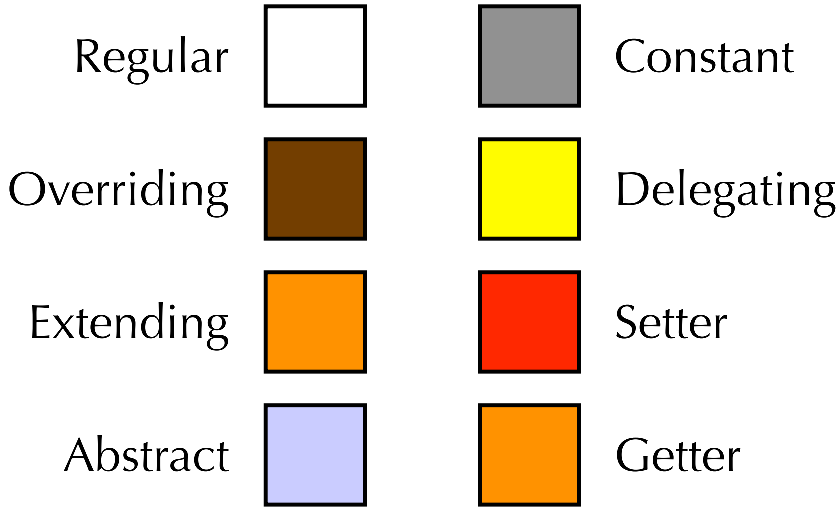


Detailing the Class Blueprint

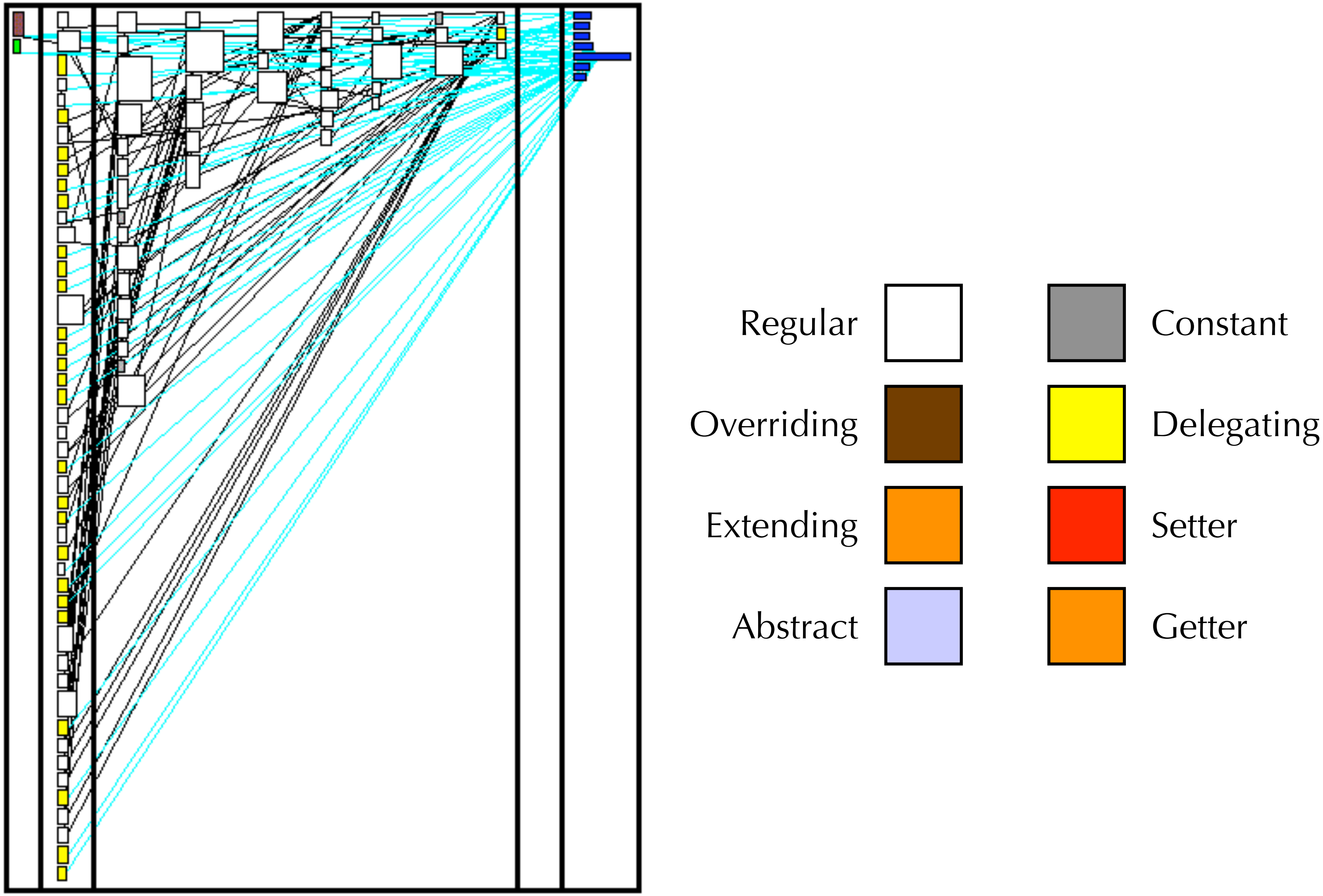


Schizophrenia

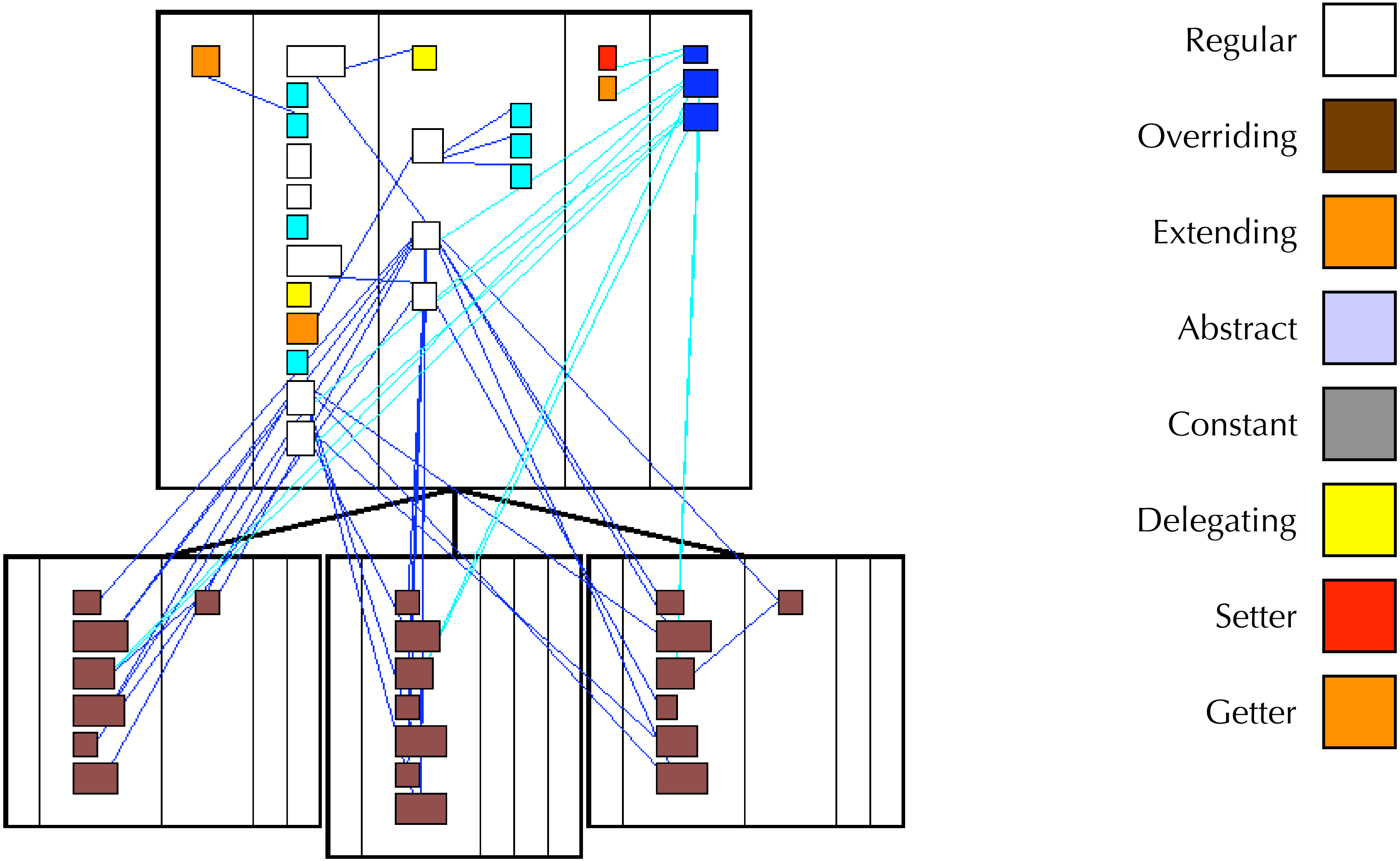




Wannabe


Regular
Constant
Overriding
Delegating
Extending
Setter
Abstract
Getter

Gory Details



Where's the Beauty?

“Software is intangible,
having no physical shape or size.”

Thomas Ball, Stephen Eick

“Software Visualization in the Large”

In *Computer*, vol. 29, no.4, pp. 33-43,
IEEE Computer Society Press, 1996


The Best Defense is Attack

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

“Patterns of Software: Tales from the Software Community”, Oxford University Press, 1998.


Visualizing Software as Cities

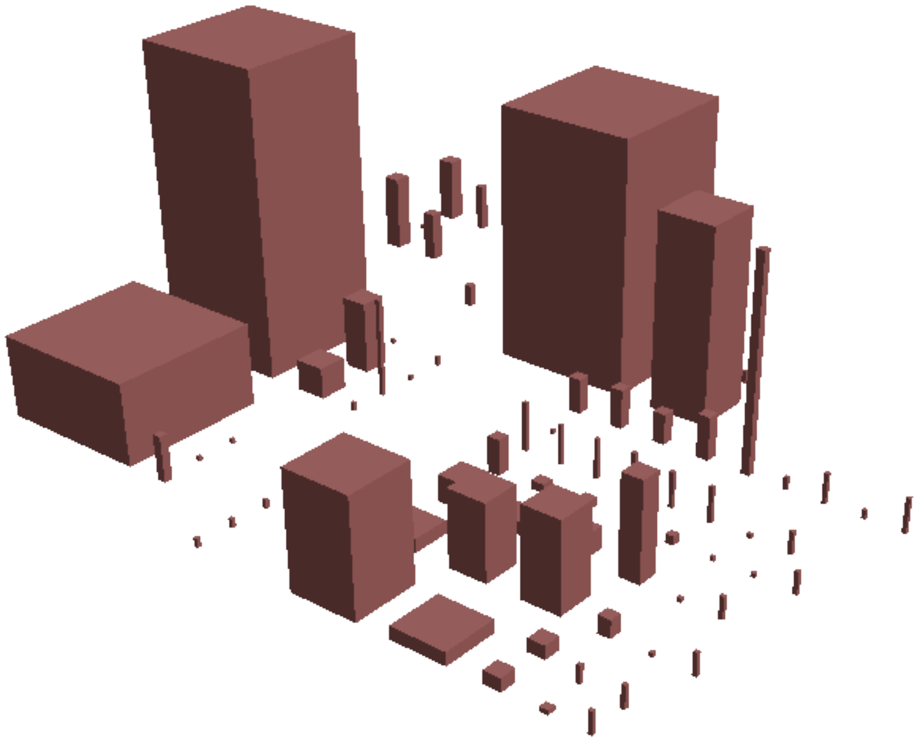
The City Metaphor

The City Metaphor

domain mapping	

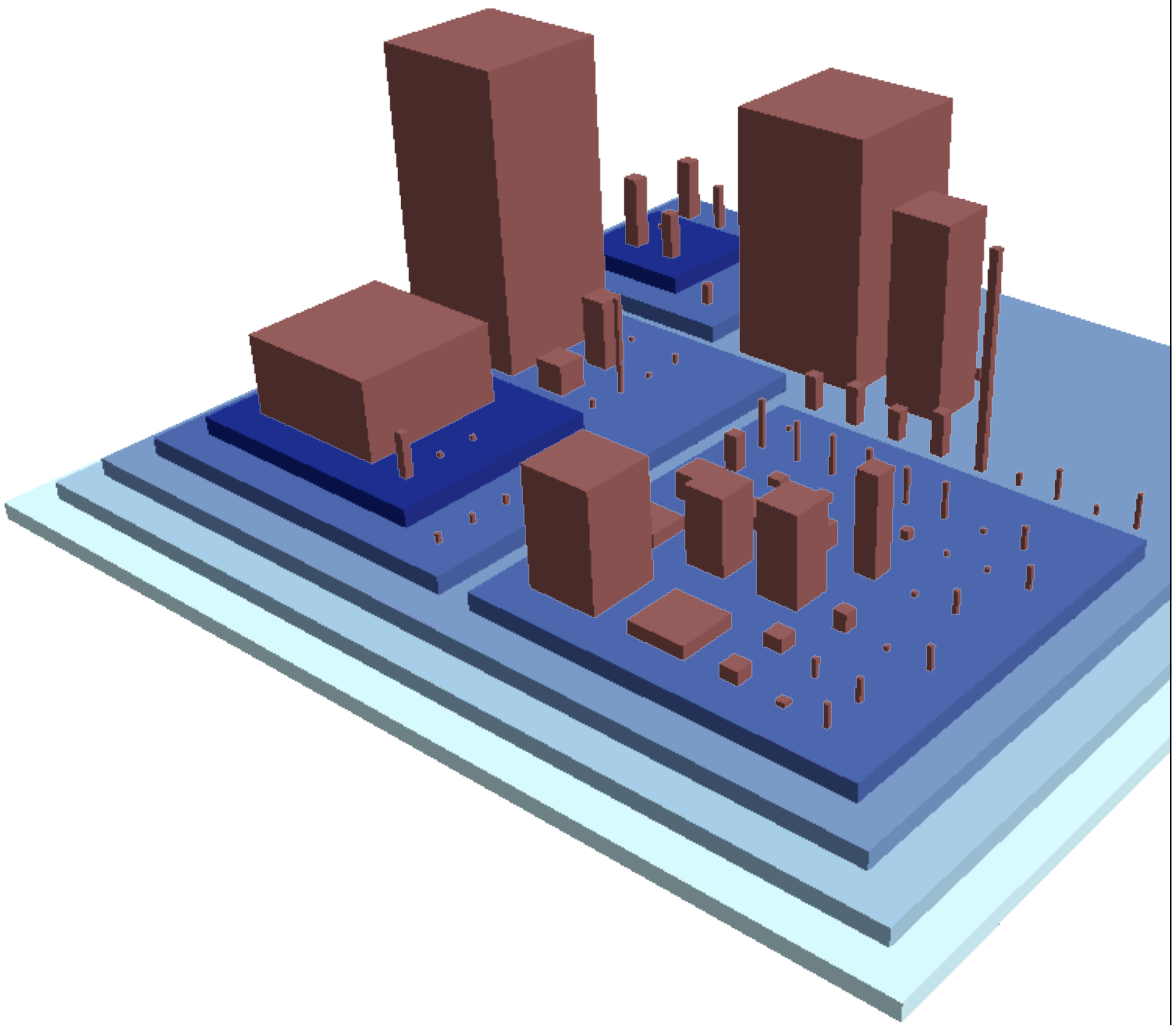
The City Metaphor

domain mapping	
classes	buildings



The City Metaphor

domain mapping	
classes	buildings
packages	districts

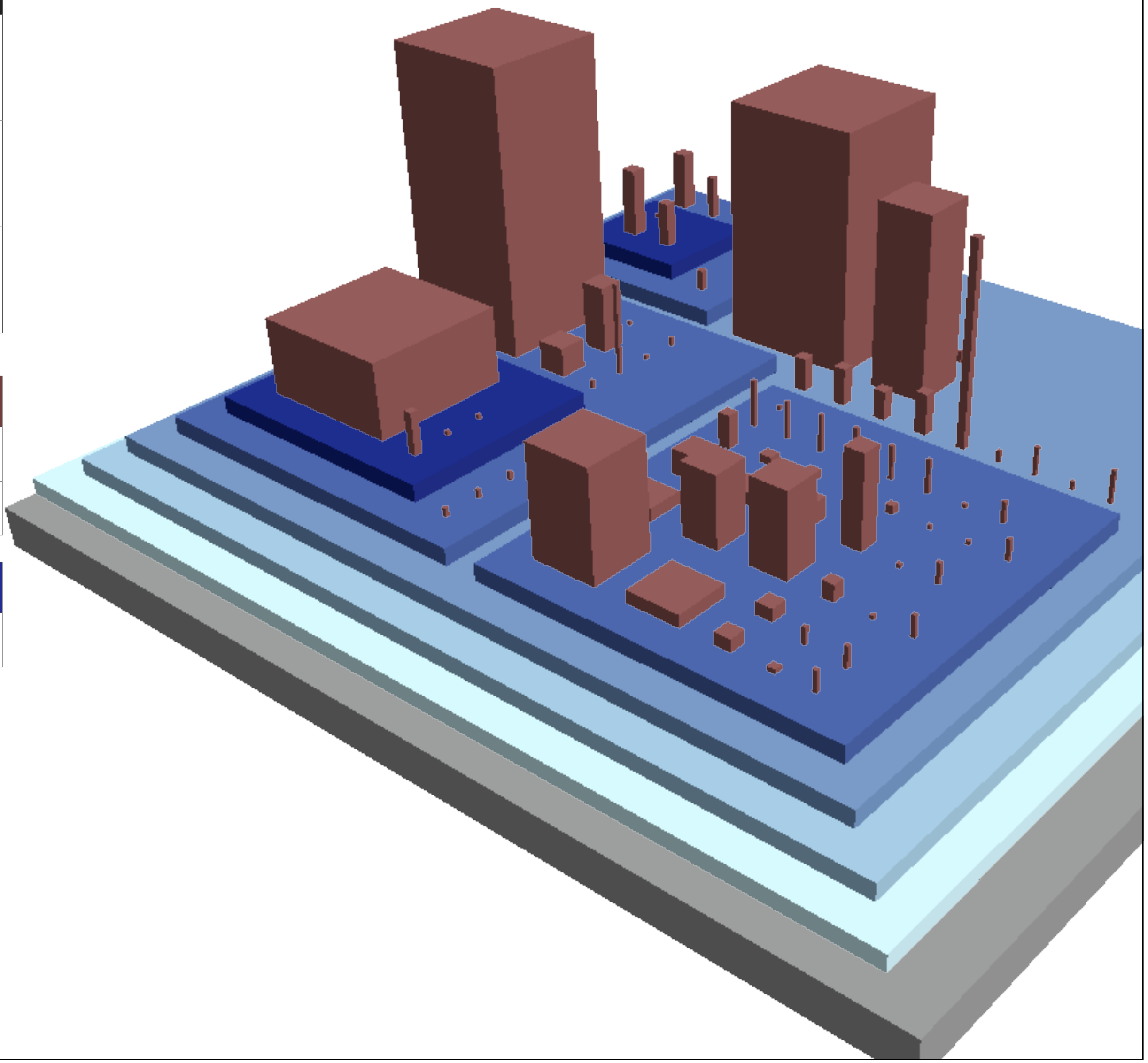


The City Metaphor

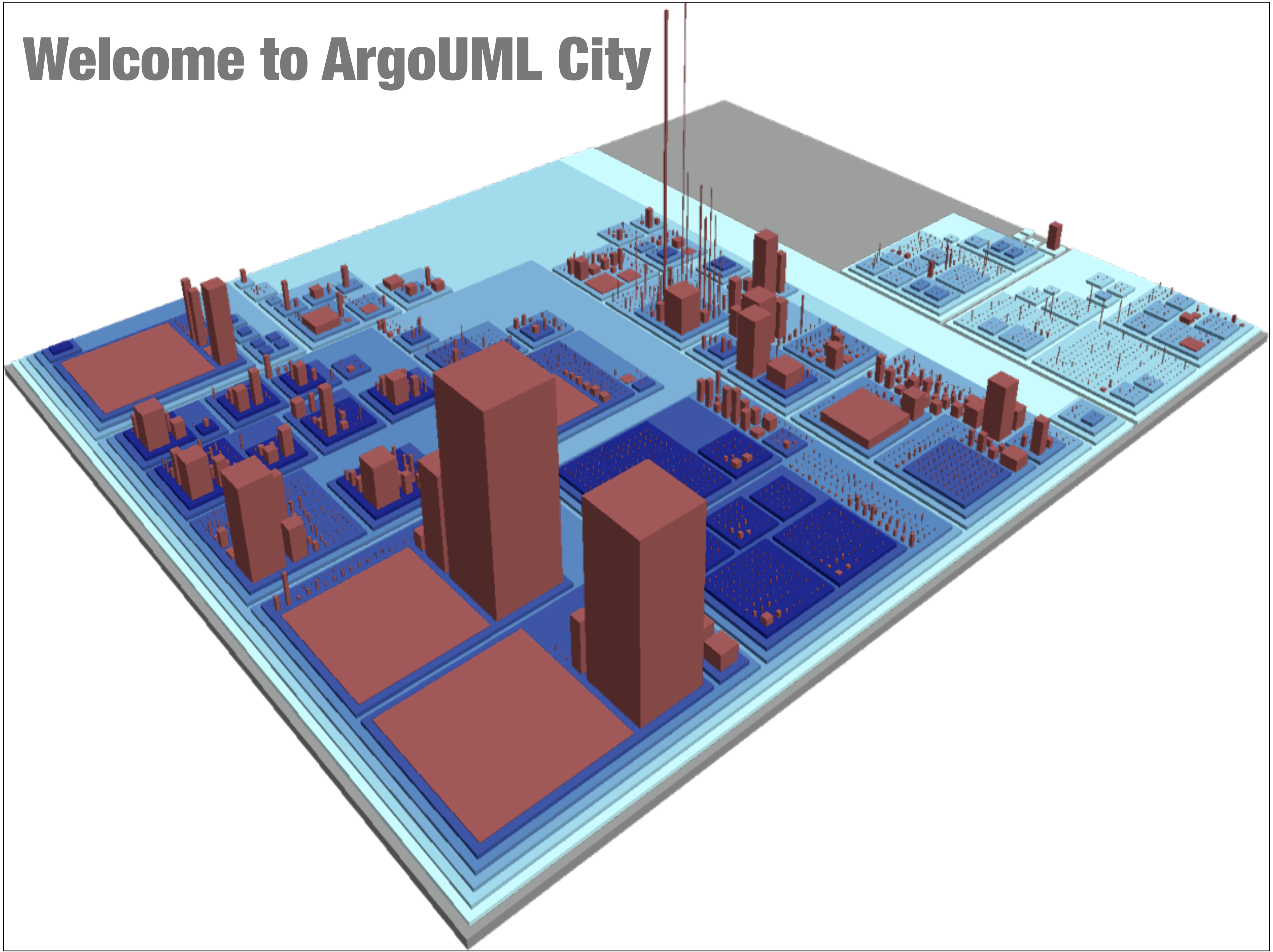
domain mapping	
classes	buildings
packages	districts
system	city

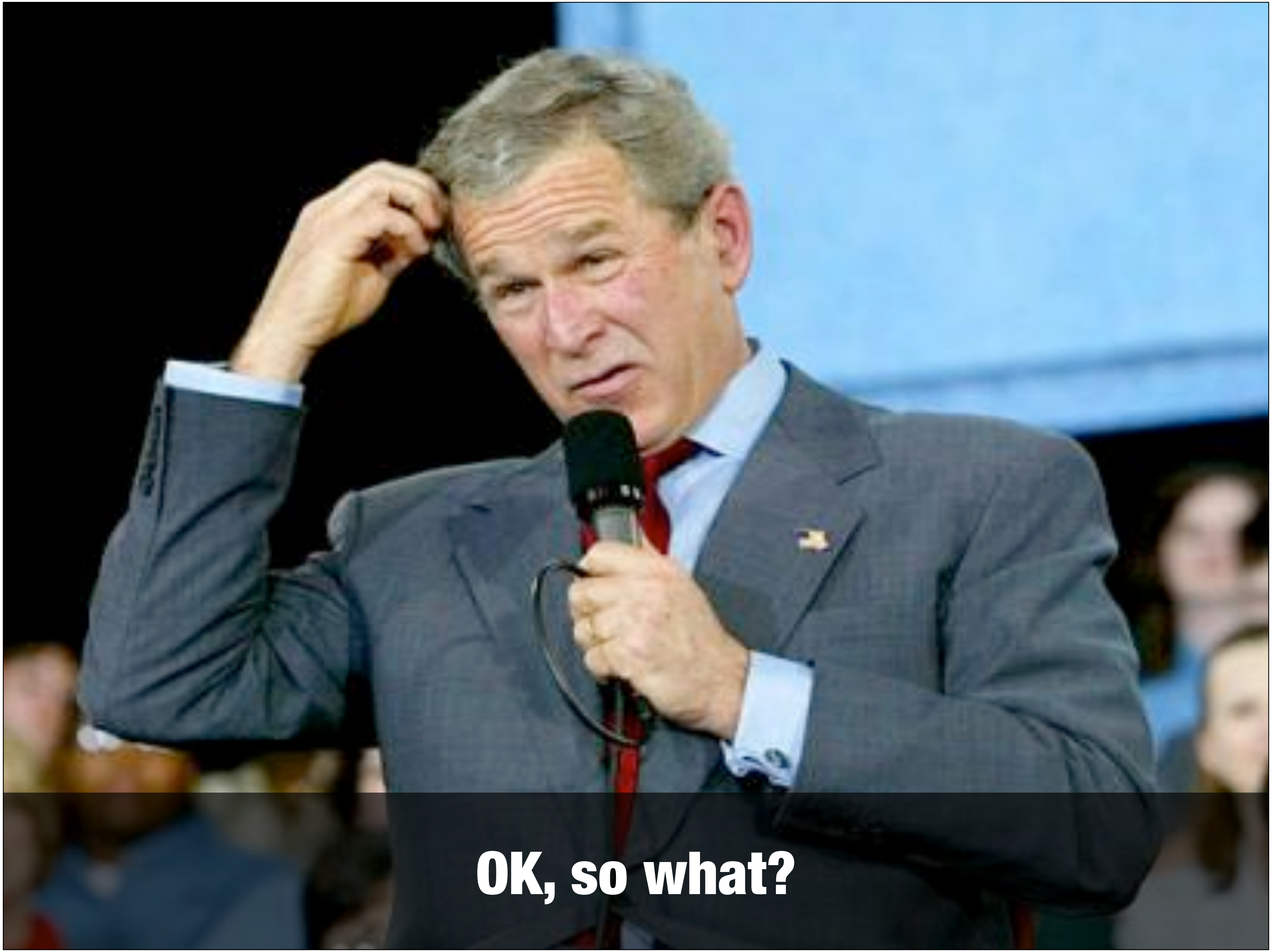
class metric	building property
number of methods (NOM)	height
number of attributes (NOA)	width, length

package metric	district property
nesting level	color



Welcome to ArgoUML City



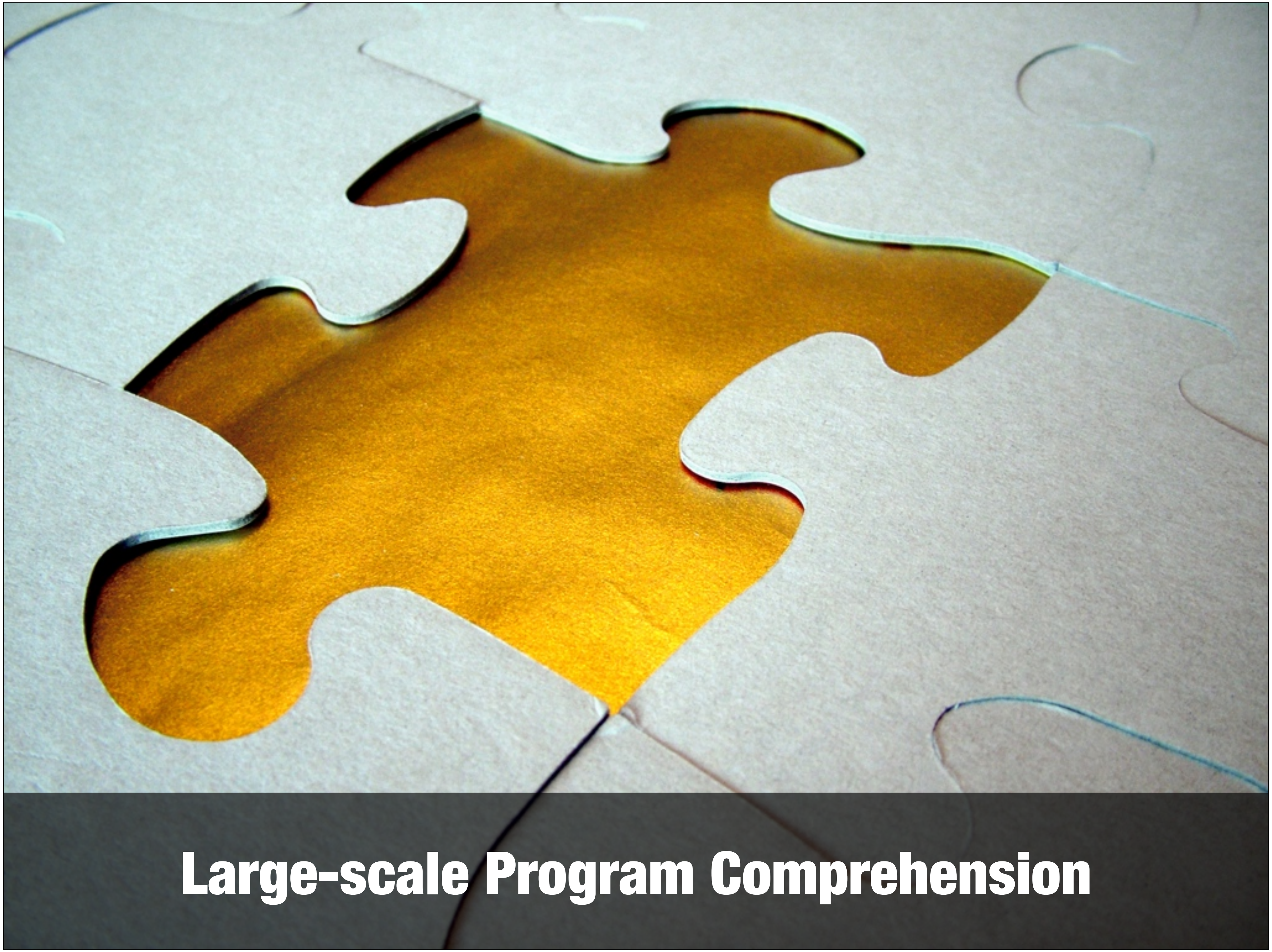

OK, so what?

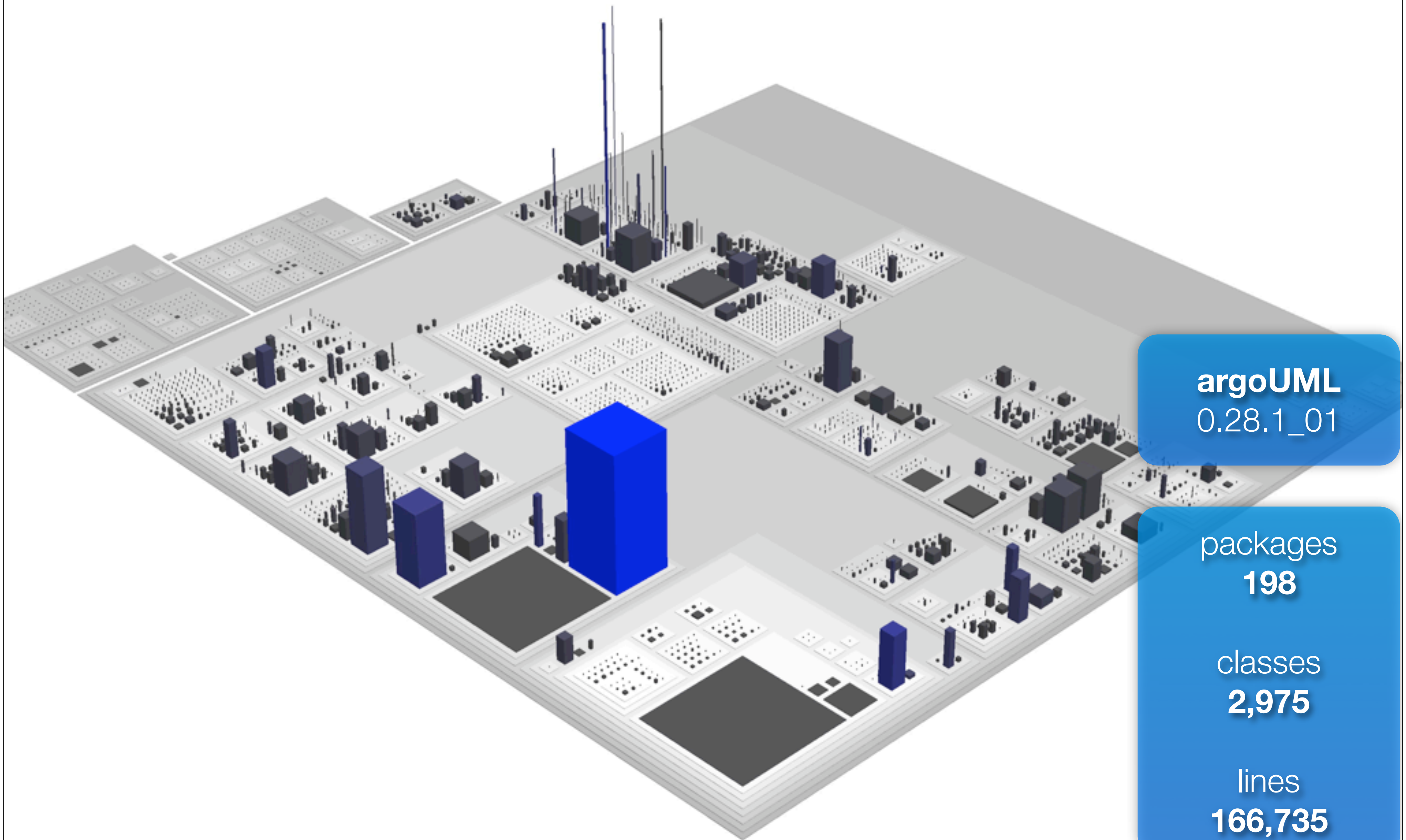
applications

applications

applications

applications


Large-scale Program Comprehension

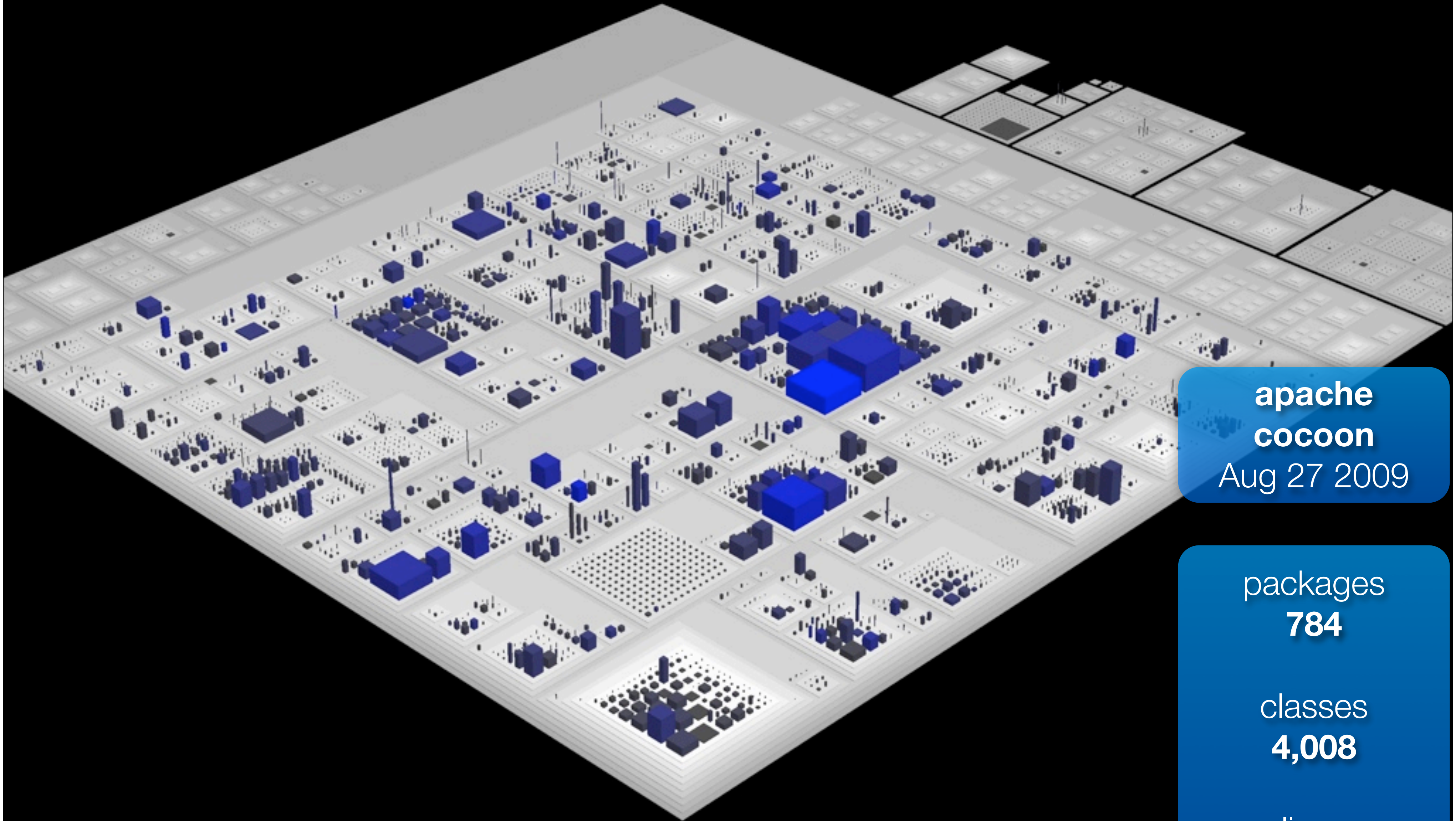


argoUML
0.28.1_01

packages
198

classes
2,975

lines
166,735

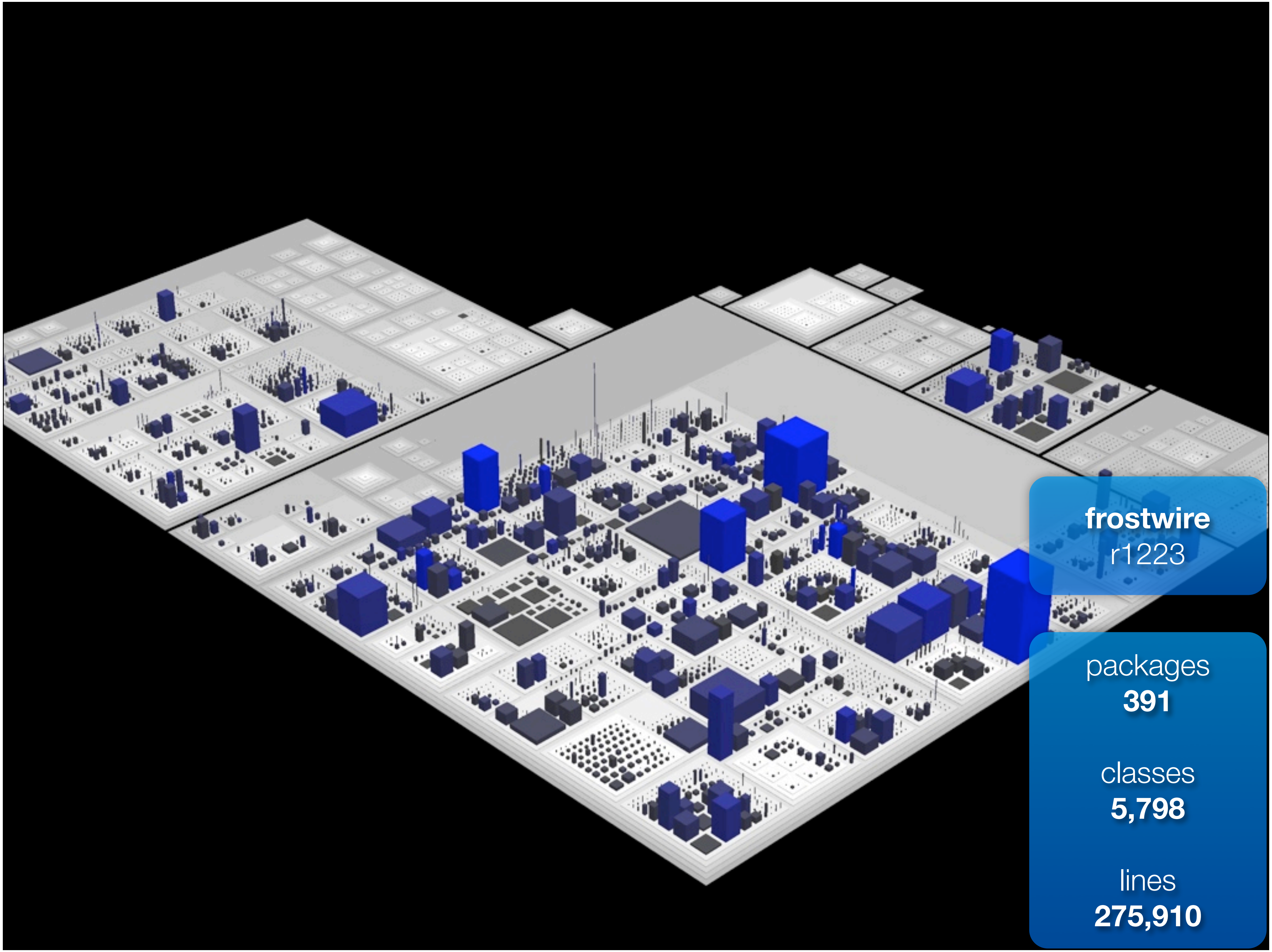


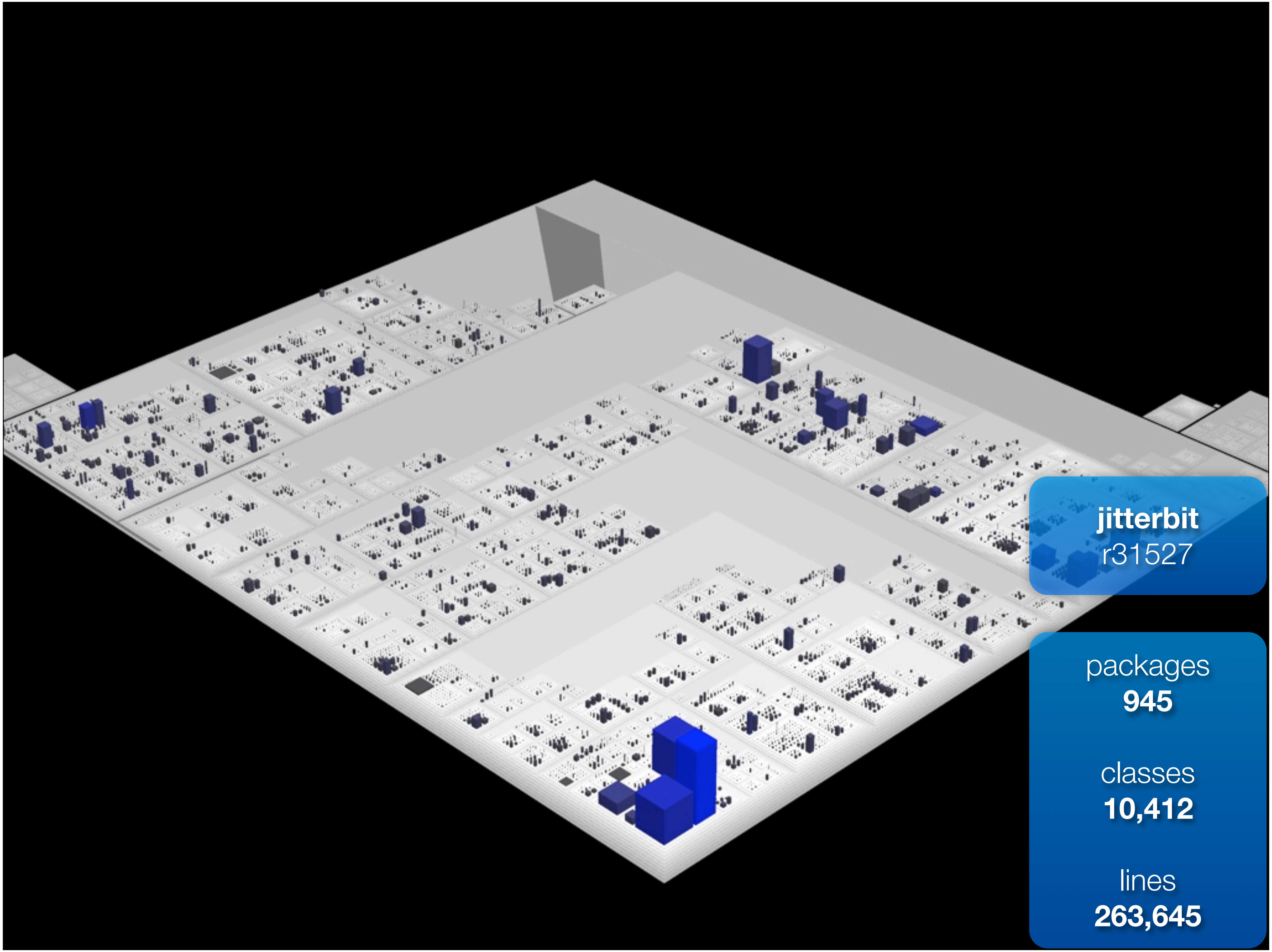
apache cocoon
Aug 27 2009

packages
784

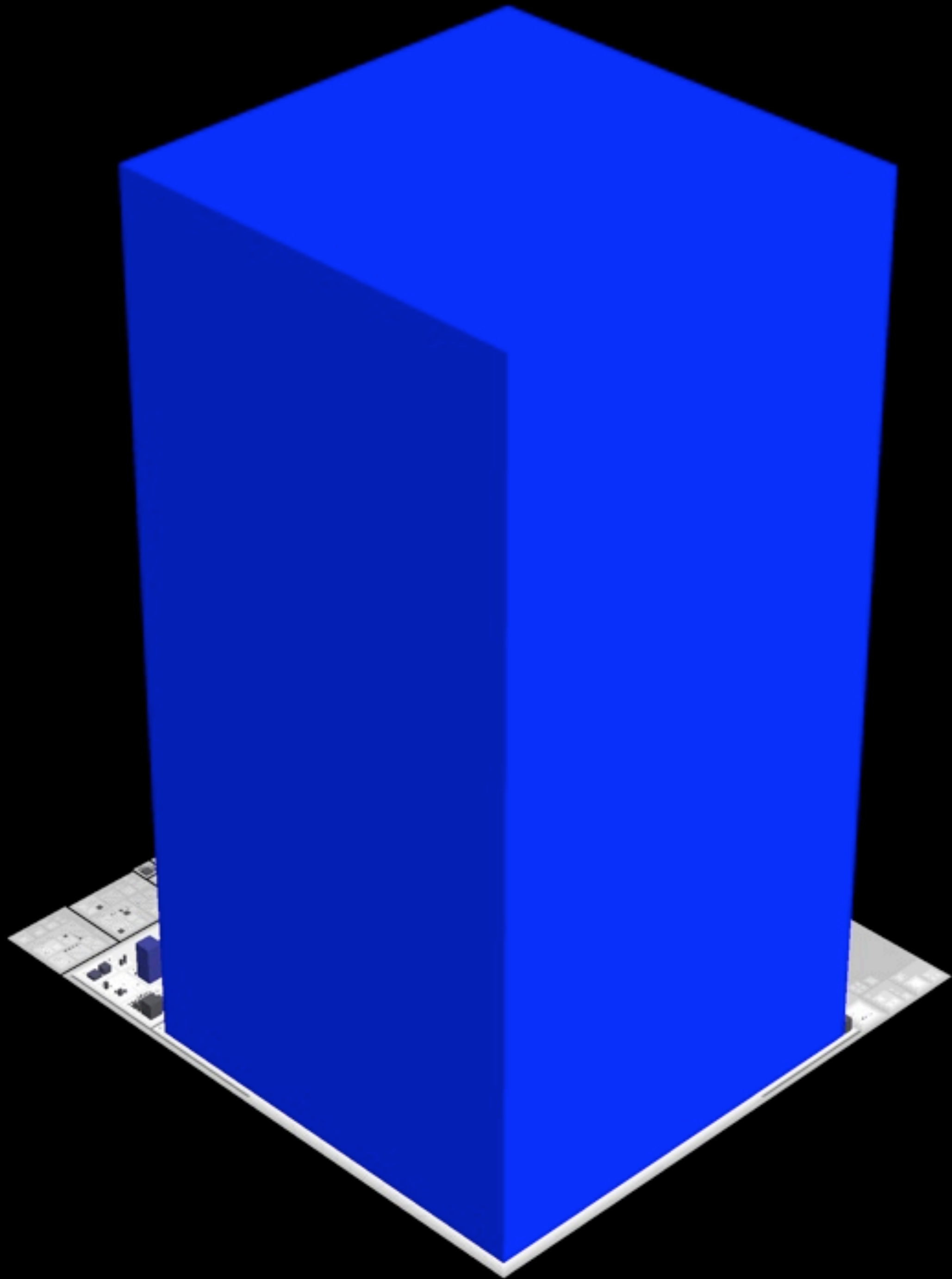
classes
4,008

lines
173,436


frostwire
r1223
packages
391
classes
5,798
lines
275,910



jitterbit
r31527
packages
945
classes
10,412
lines
263,645

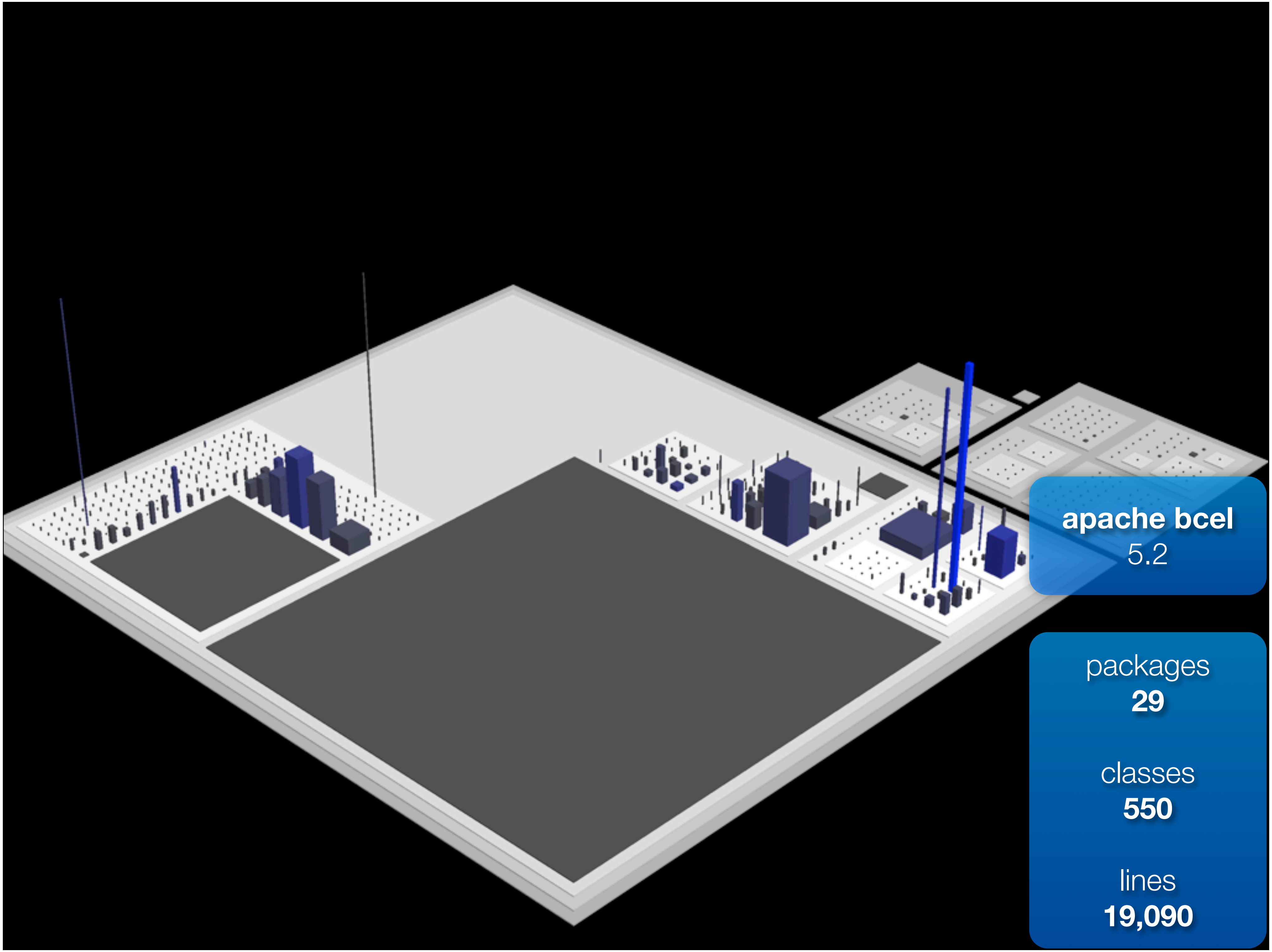


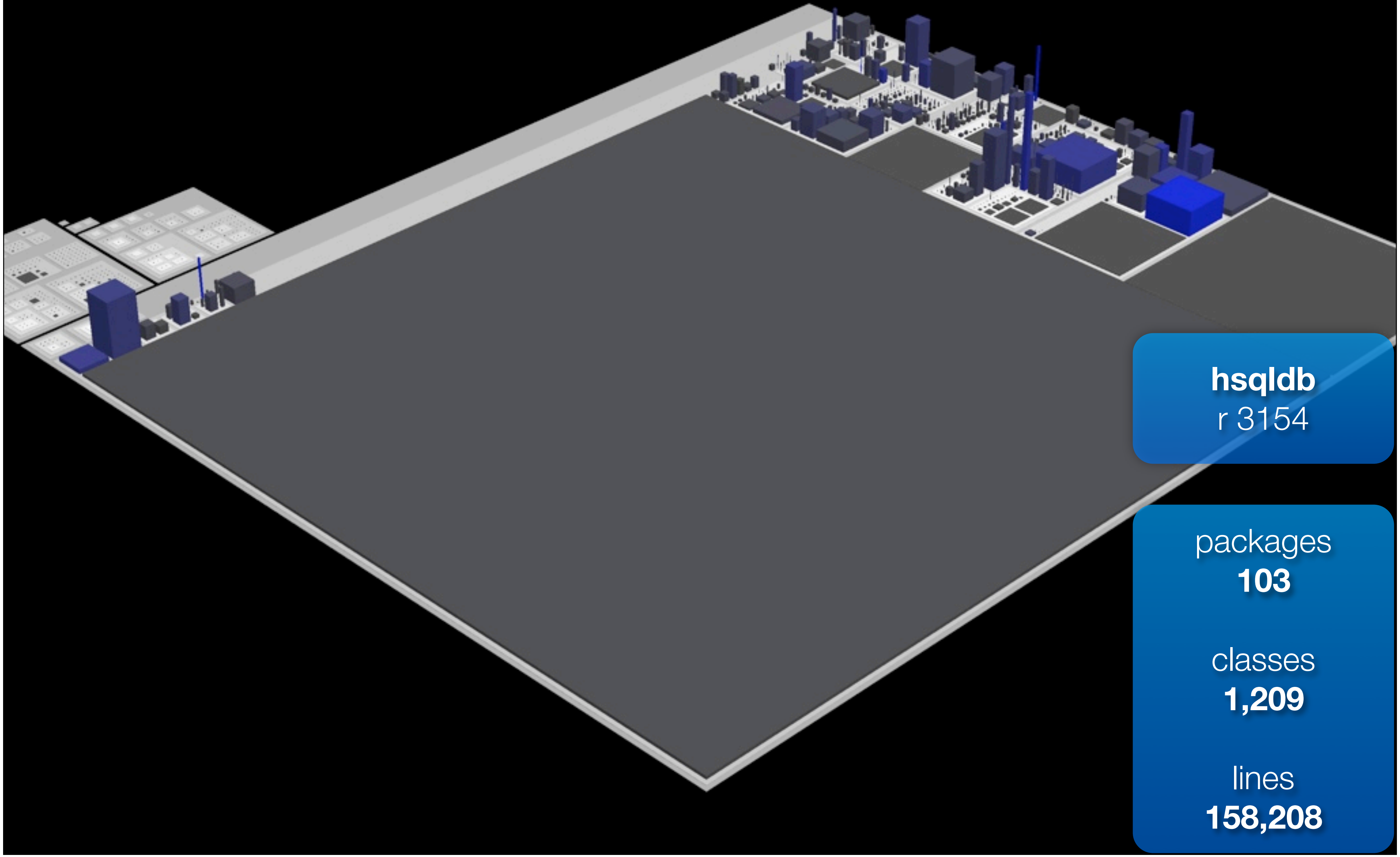
openswing
r296

packages
268

classes
1,754

lines
112,495


apache bcel
5.2
packages
29
classes
550
lines
19,090

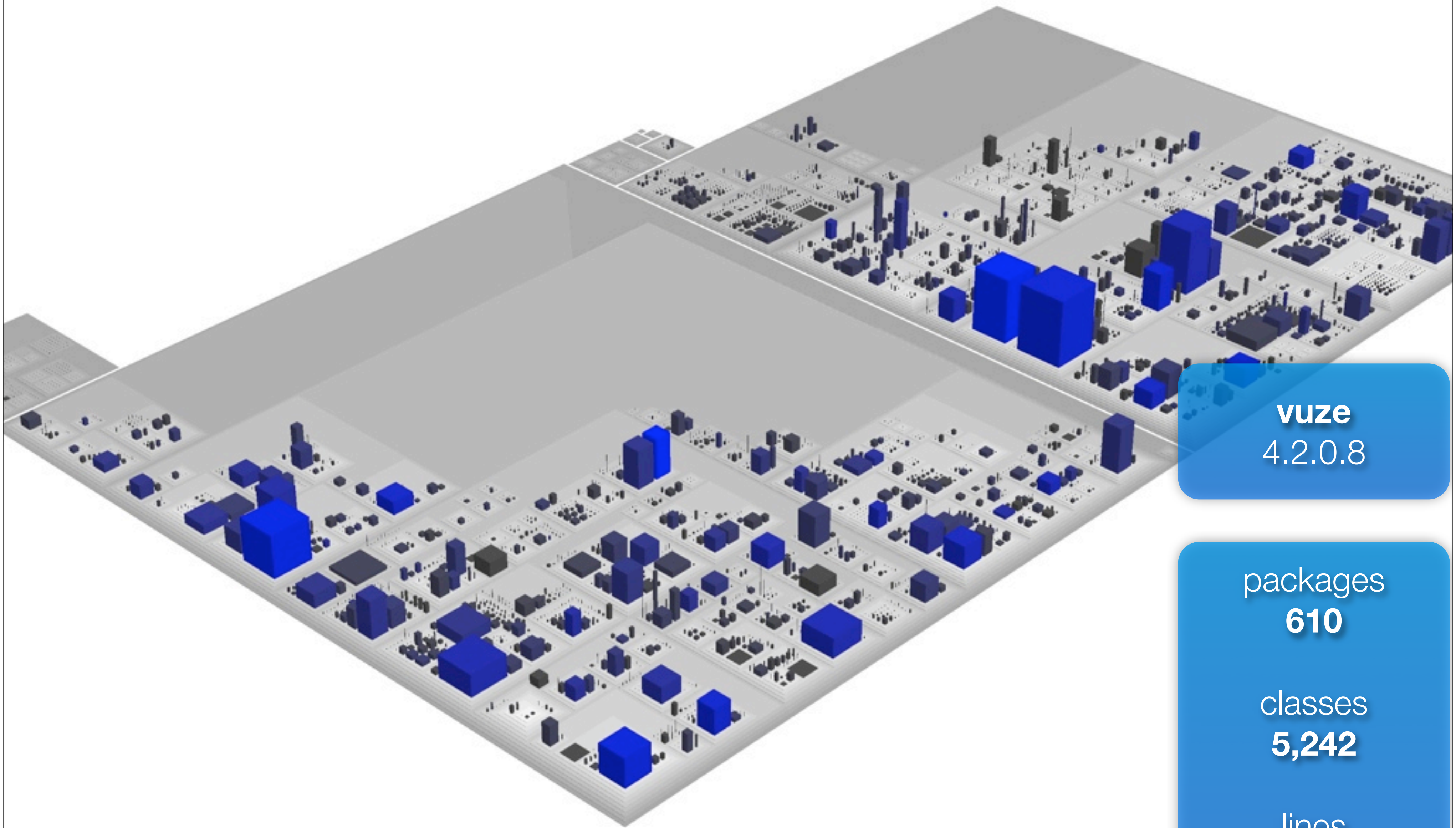


hsqldb
r 3154

packages
103

classes
1,209

lines
158,208

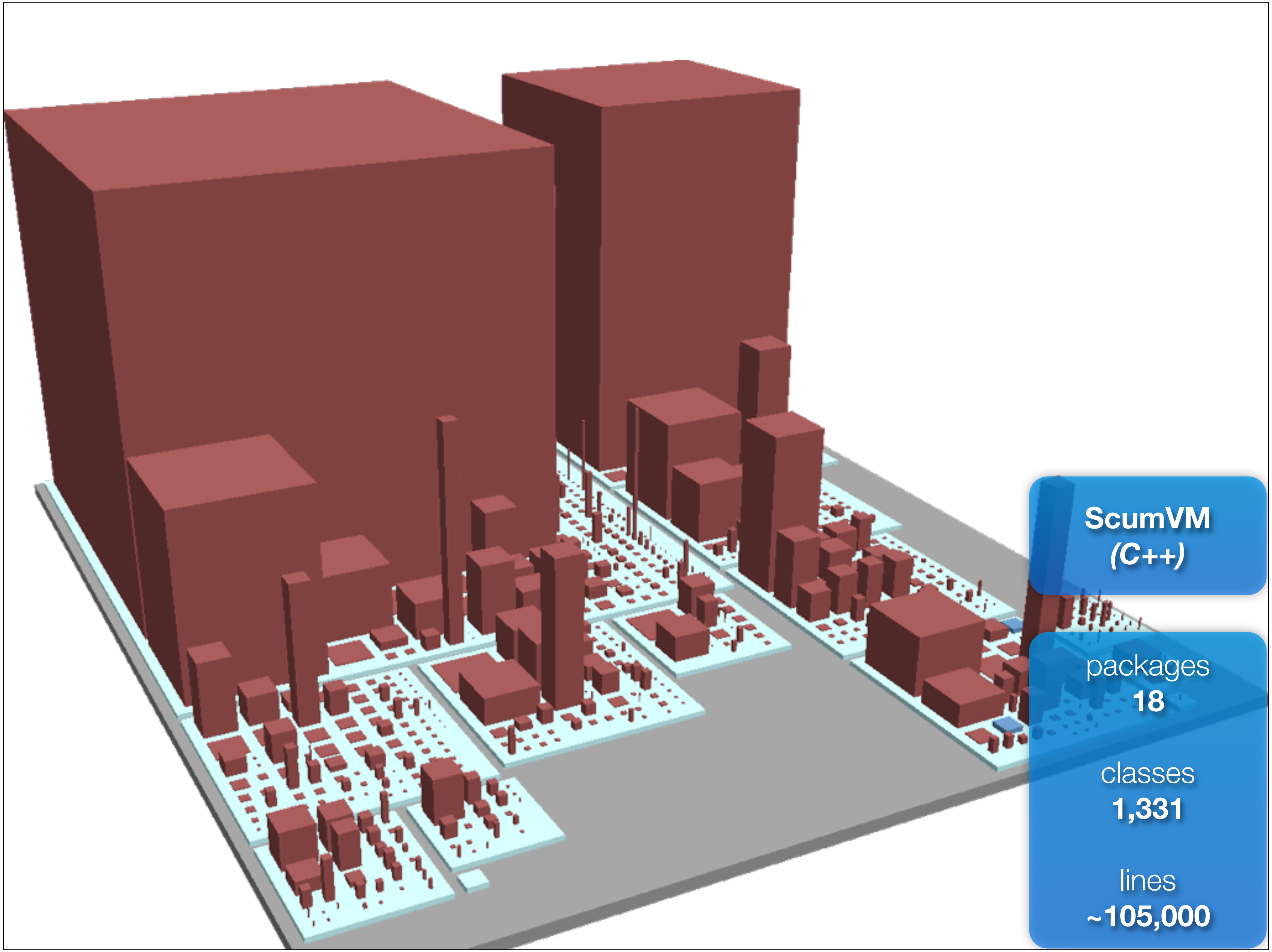


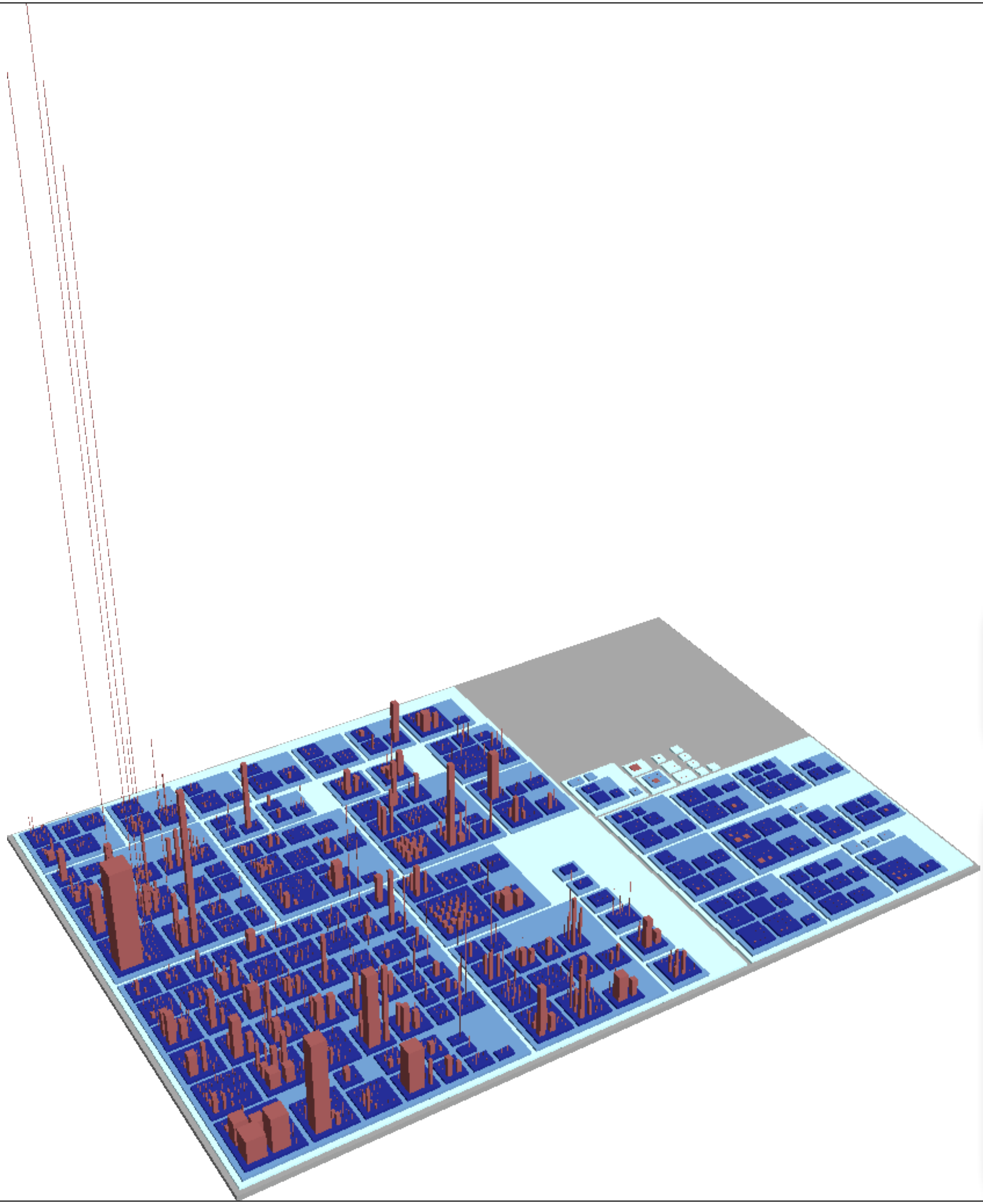
vuze
4.2.0.8

packages
610

classes
5,242

lines
456,064


ScumVM
(C++)
packages
18
classes
1,331
lines
~105,000

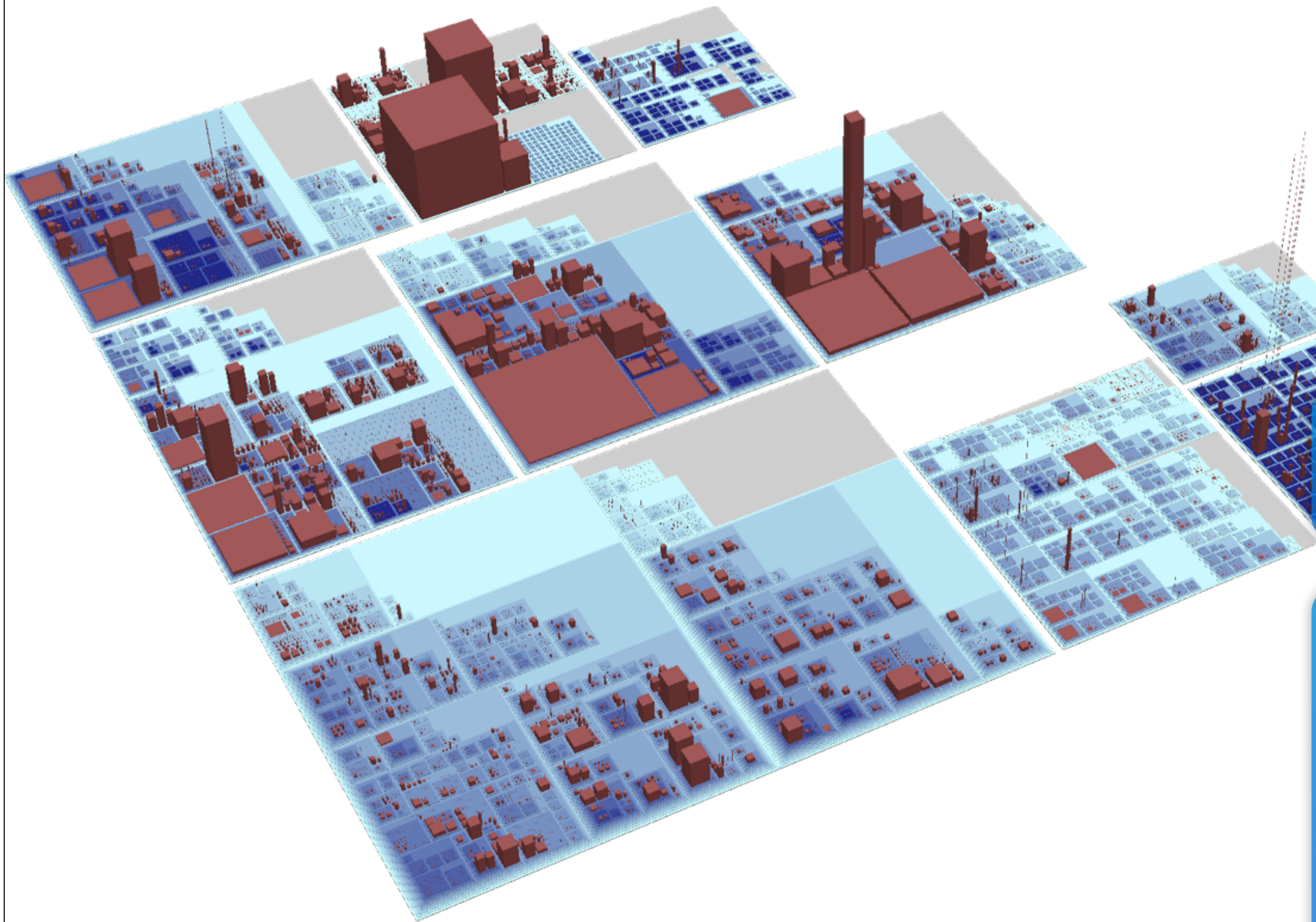


Jun
(Smalltalk)

packages
288

classes
2,2236

lines
~351,000



10 systems
(Java,Smalltalk, C++)

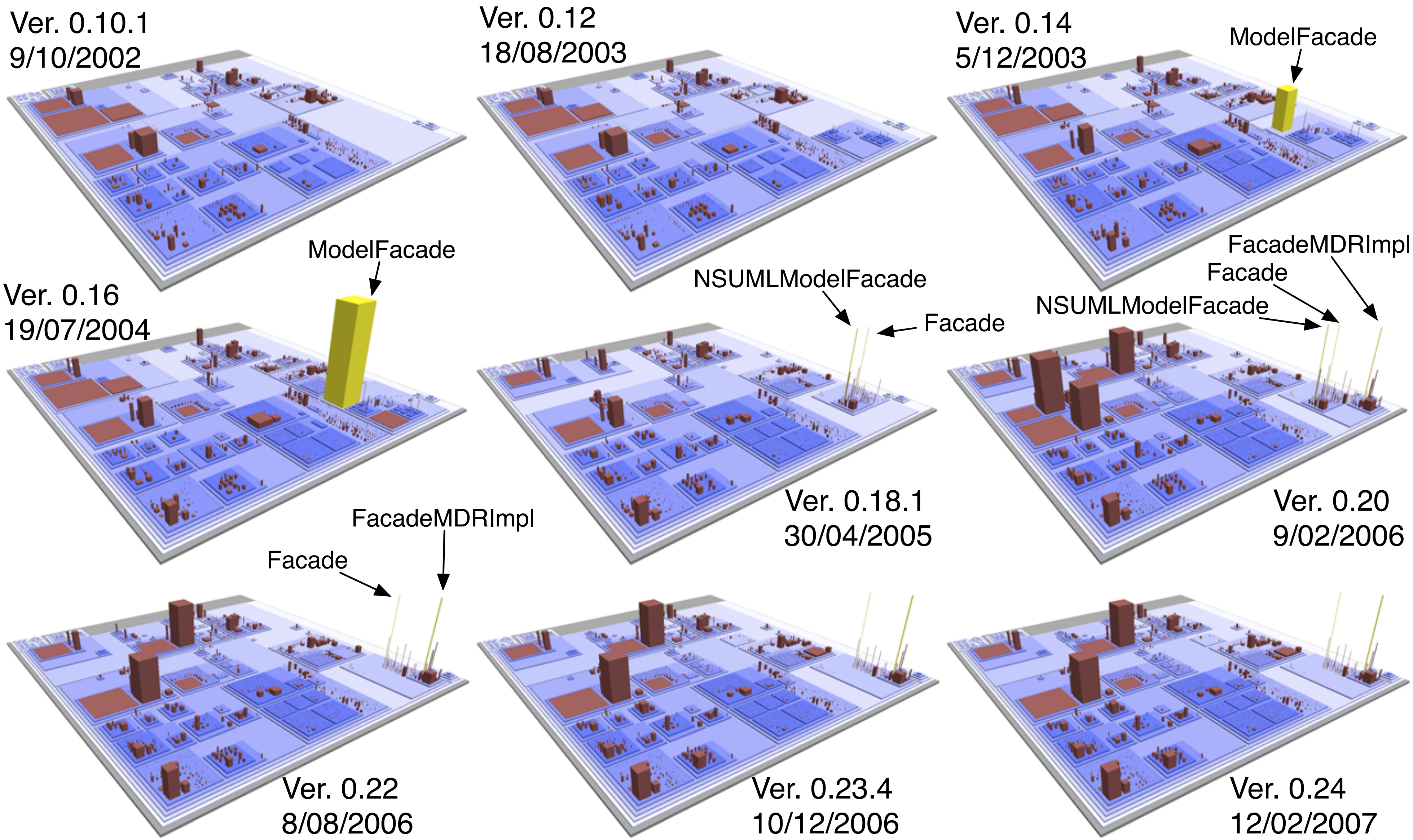
packages
1,777

classes
17,090

lines
~1,272,000


Understanding Evolution

ArgoUML's filmstrip



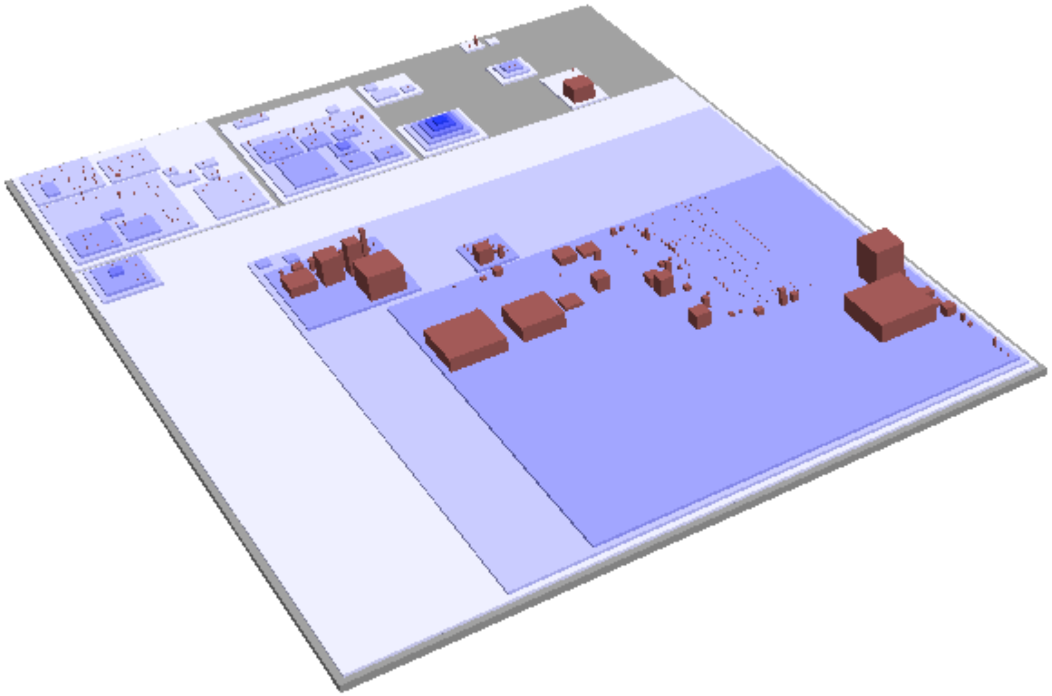
The Time Machine

JMol

versions
57 (bi-monthly snapshots)

time
1999-2007

The Time Machine

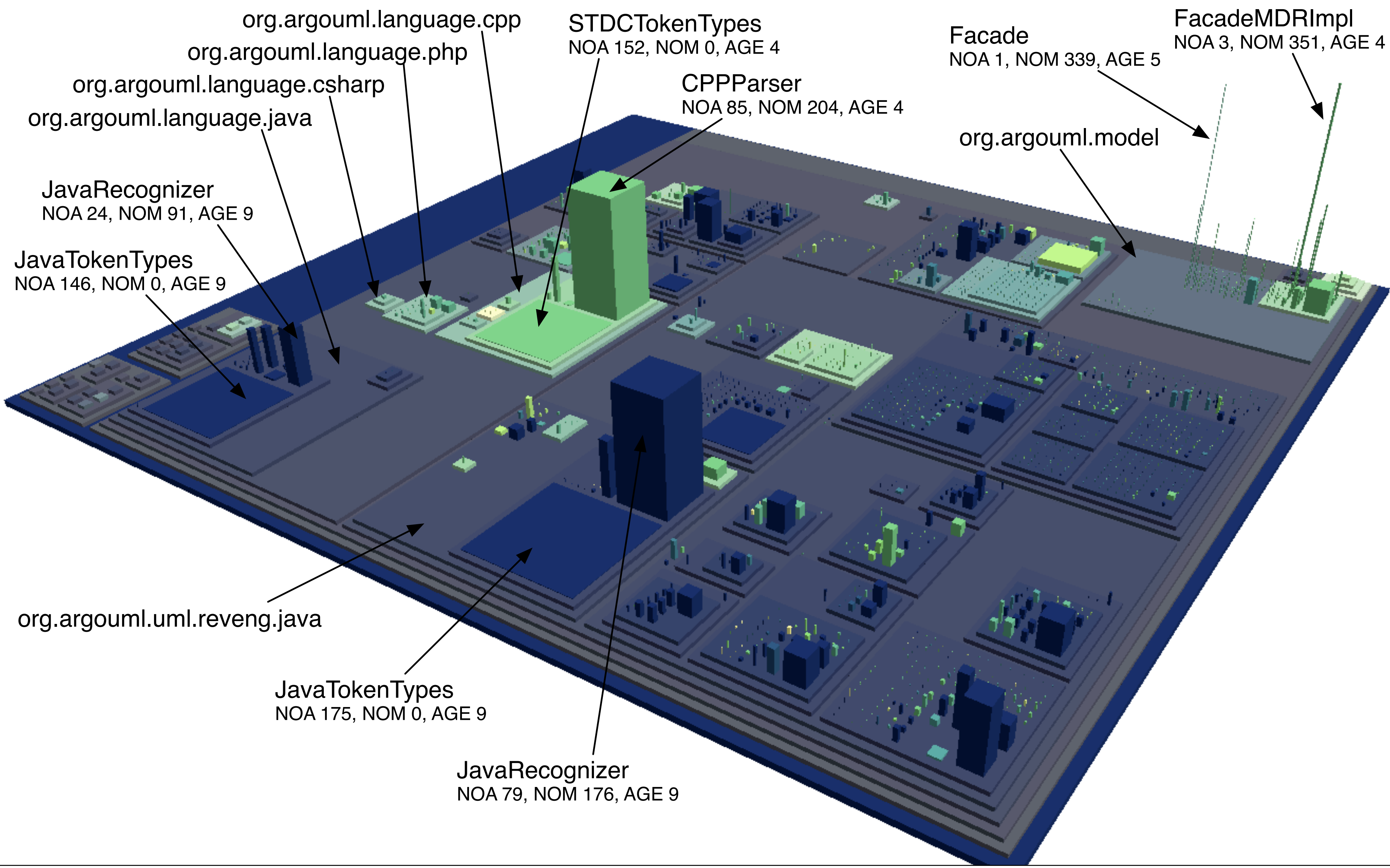


JMol

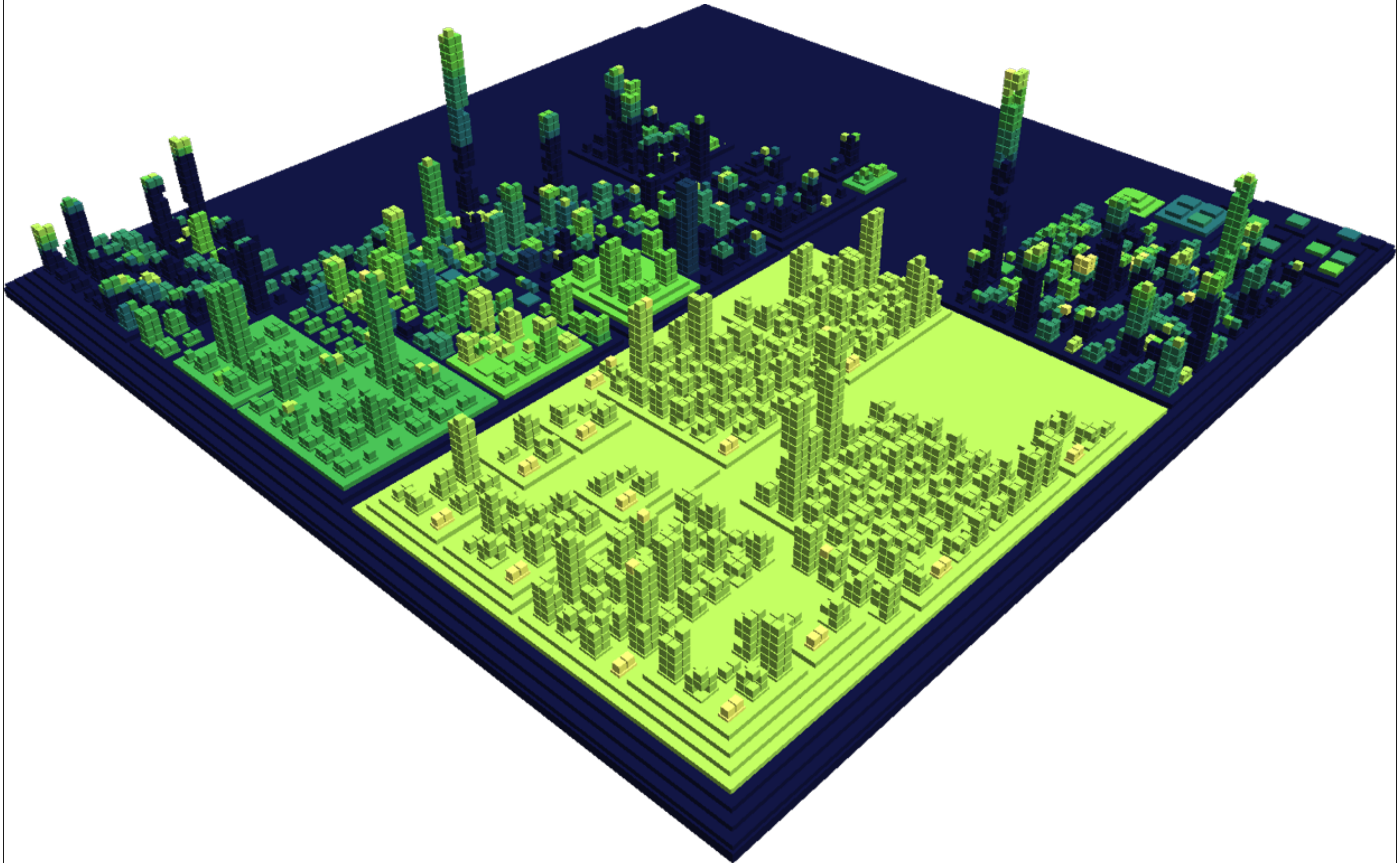
versions
57 (bi-monthly snapshots)

time
1999-2007

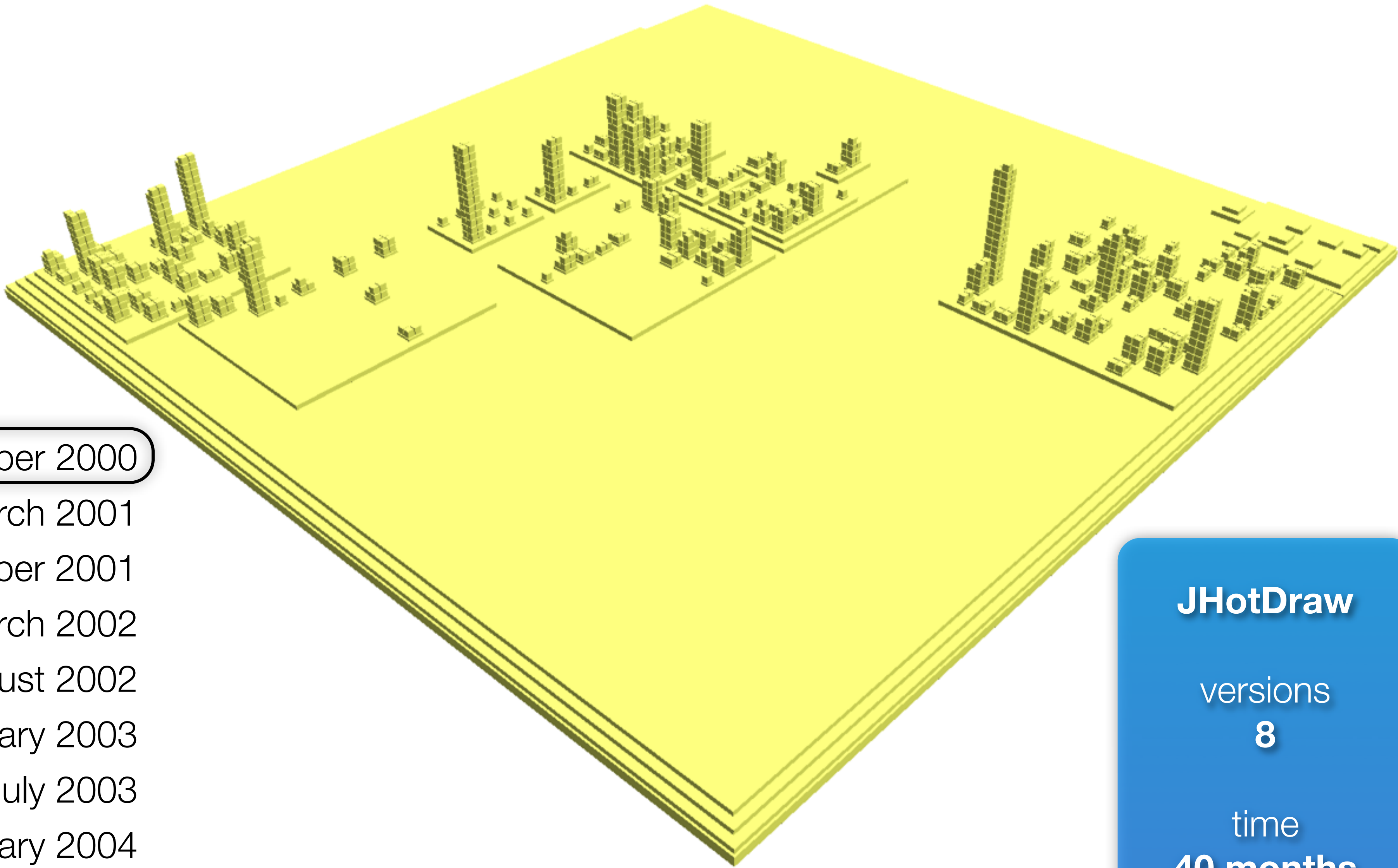
ArgoUML Age Map



JHotDraw Fine-grained Age map



Time Travel + Age Map



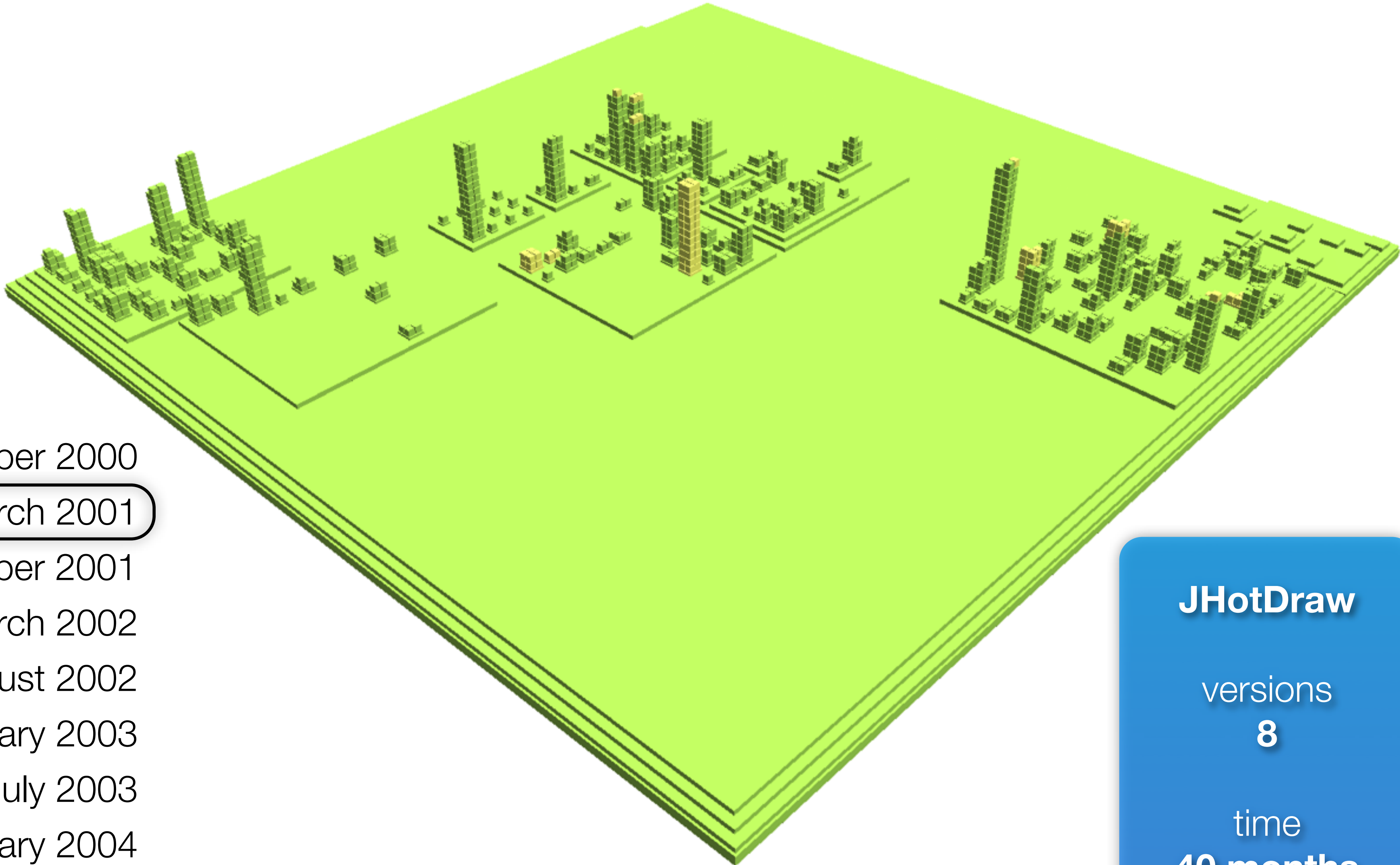
October 2000
March 2001
September 2001
March 2002
August 2002
January 2003
July 2003
January 2004

JHotDraw

versions
8

time
40 months

Time Travel + Age Map



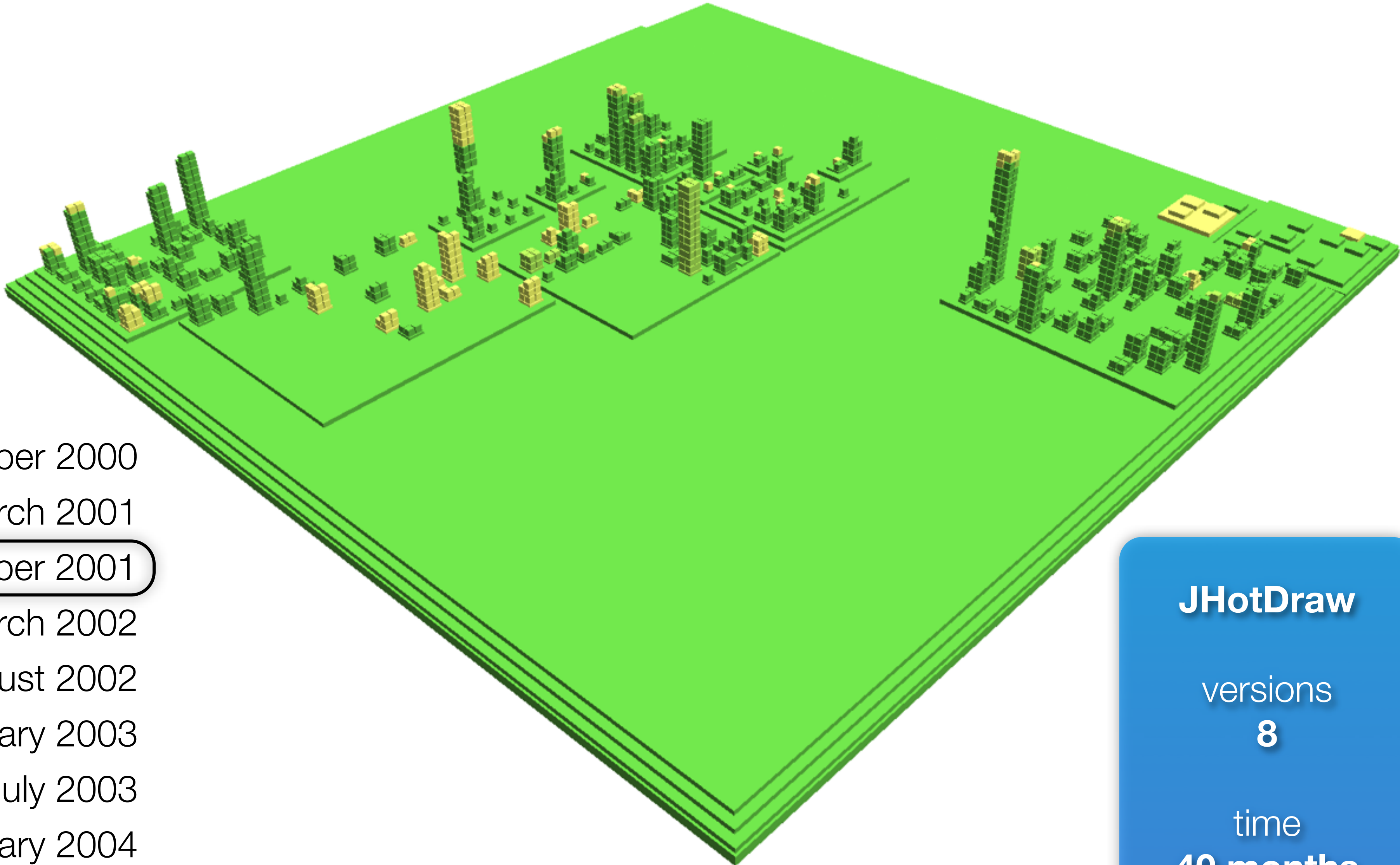
October 2000
March 2001
September 2001
March 2002
August 2002
January 2003
July 2003
January 2004

JHotDraw

versions
8

time
40 months

Time Travel + Age Map



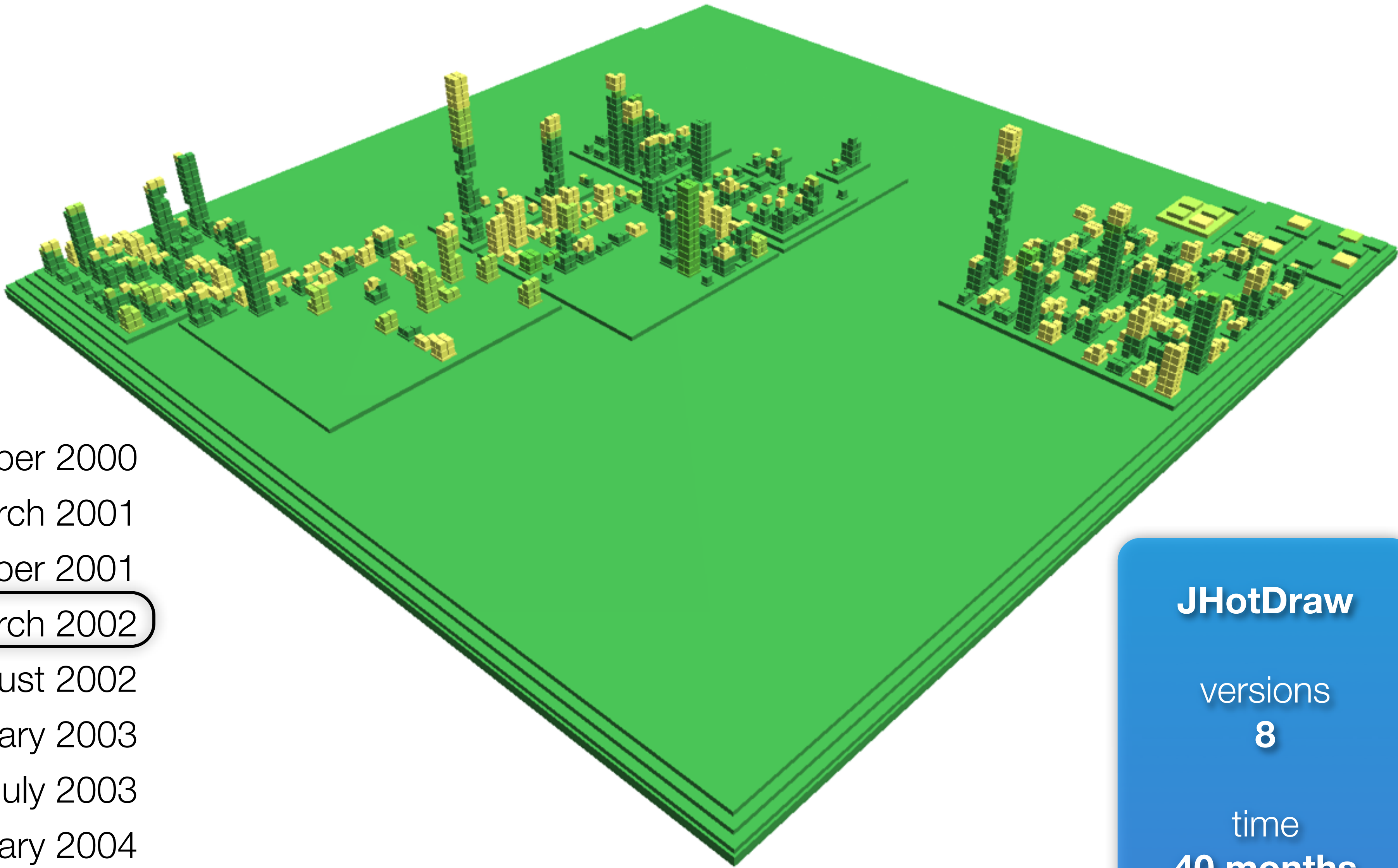
October 2000
March 2001
September 2001
March 2002
August 2002
January 2003
July 2003
January 2004

JHotDraw

versions
8

time
40 months

Time Travel + Age Map



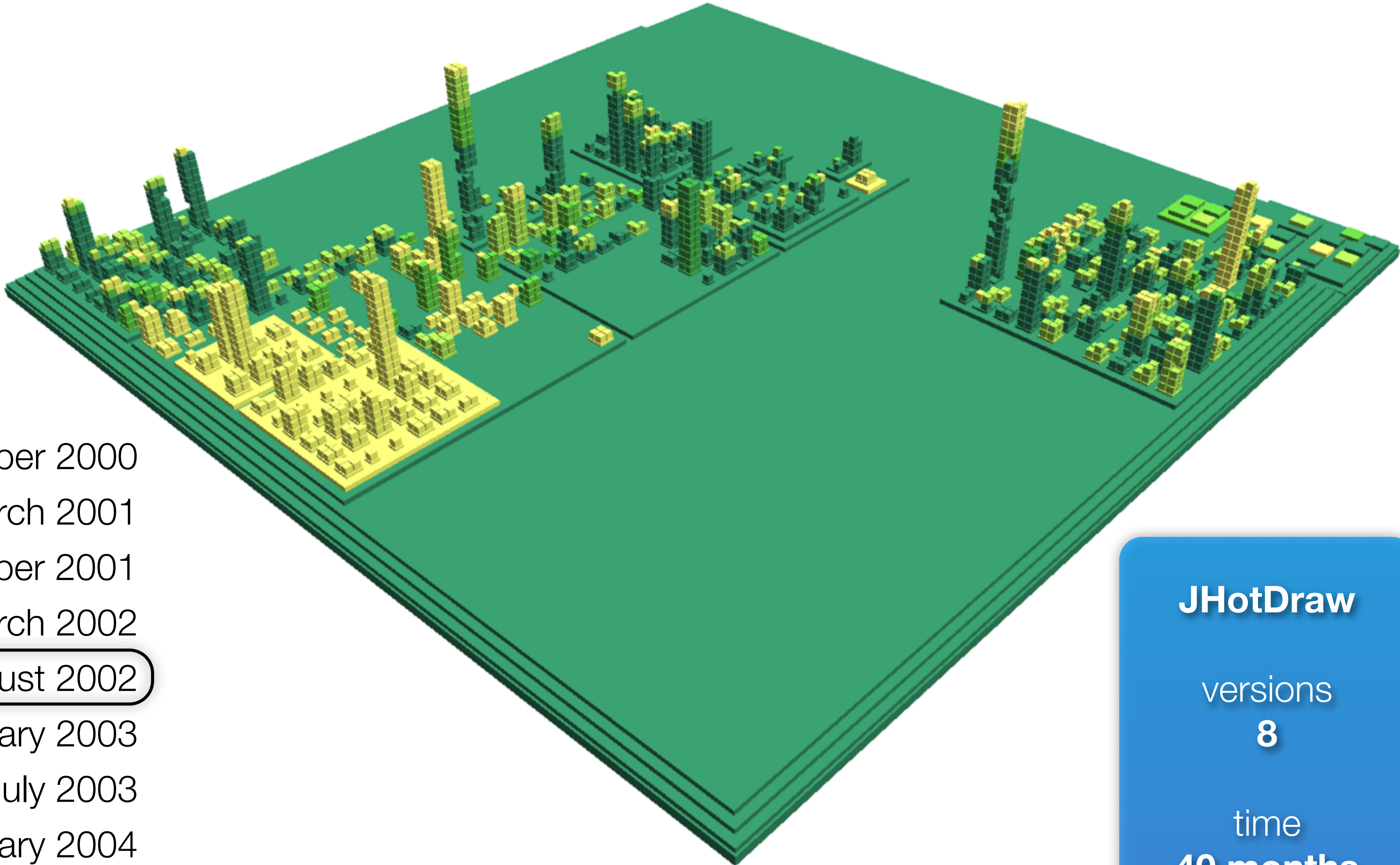
October 2000
March 2001
September 2001
March 2002
August 2002
January 2003
July 2003
January 2004

JHotDraw

versions
8

time
40 months

Time Travel + Age Map



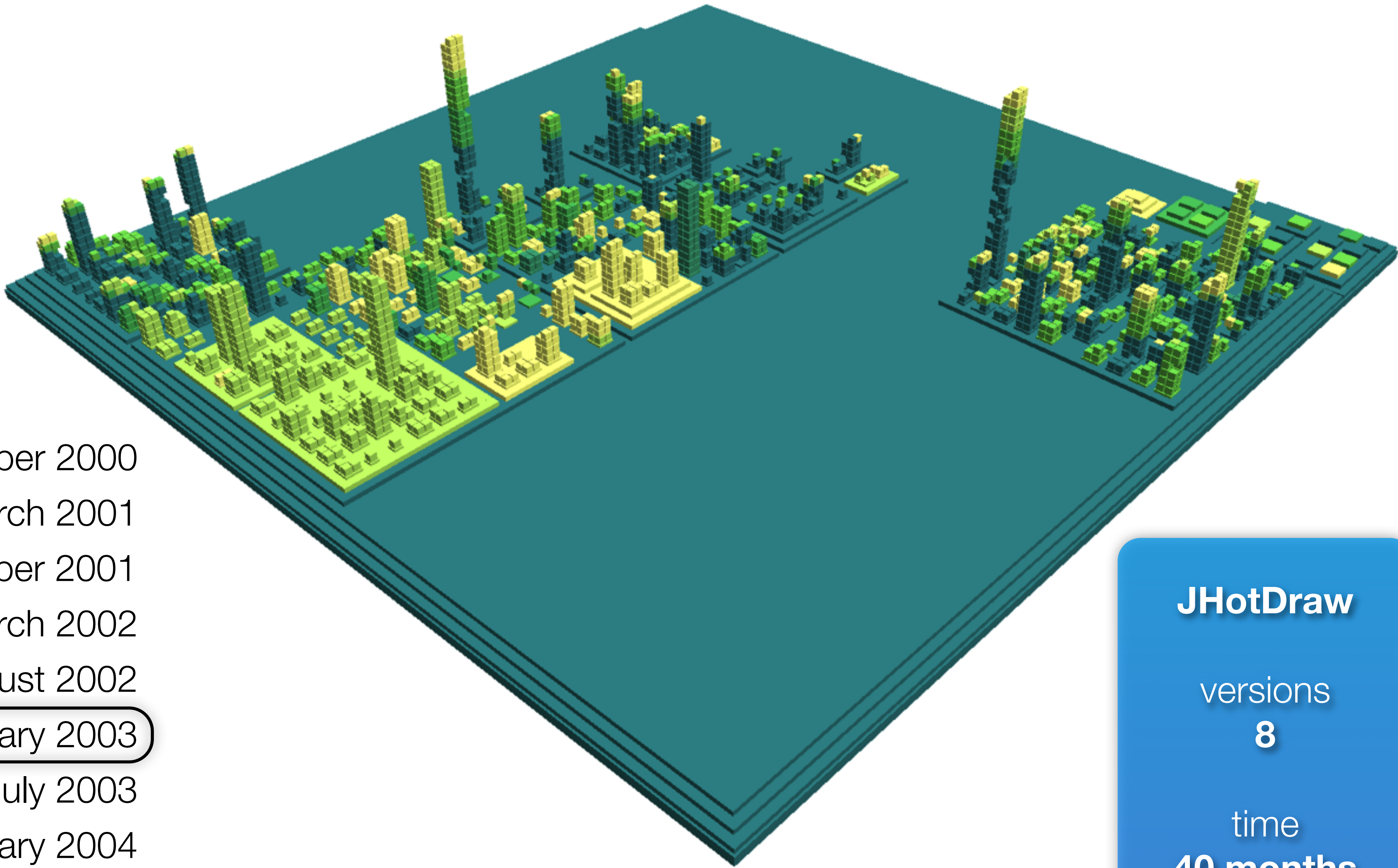
October 2000
March 2001
September 2001
March 2002
August 2002
January 2003
July 2003
January 2004

JHotDraw

versions
8

time
40 months

Time Travel + Age Map



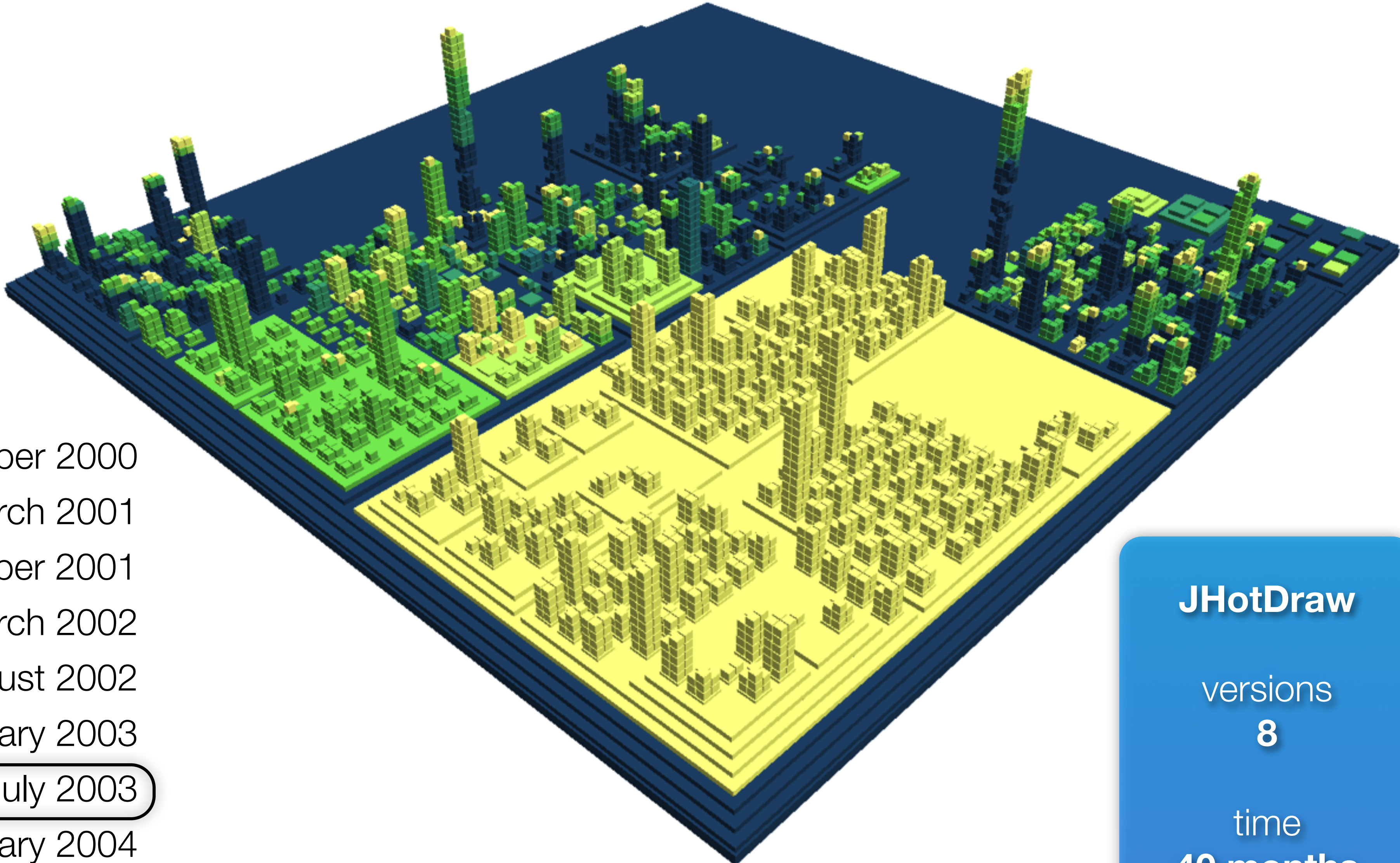
October 2000
March 2001
September 2001
March 2002
August 2002
January 2003
July 2003
January 2004

JHotDraw

versions
8

time
40 months

Time Travel + Age Map



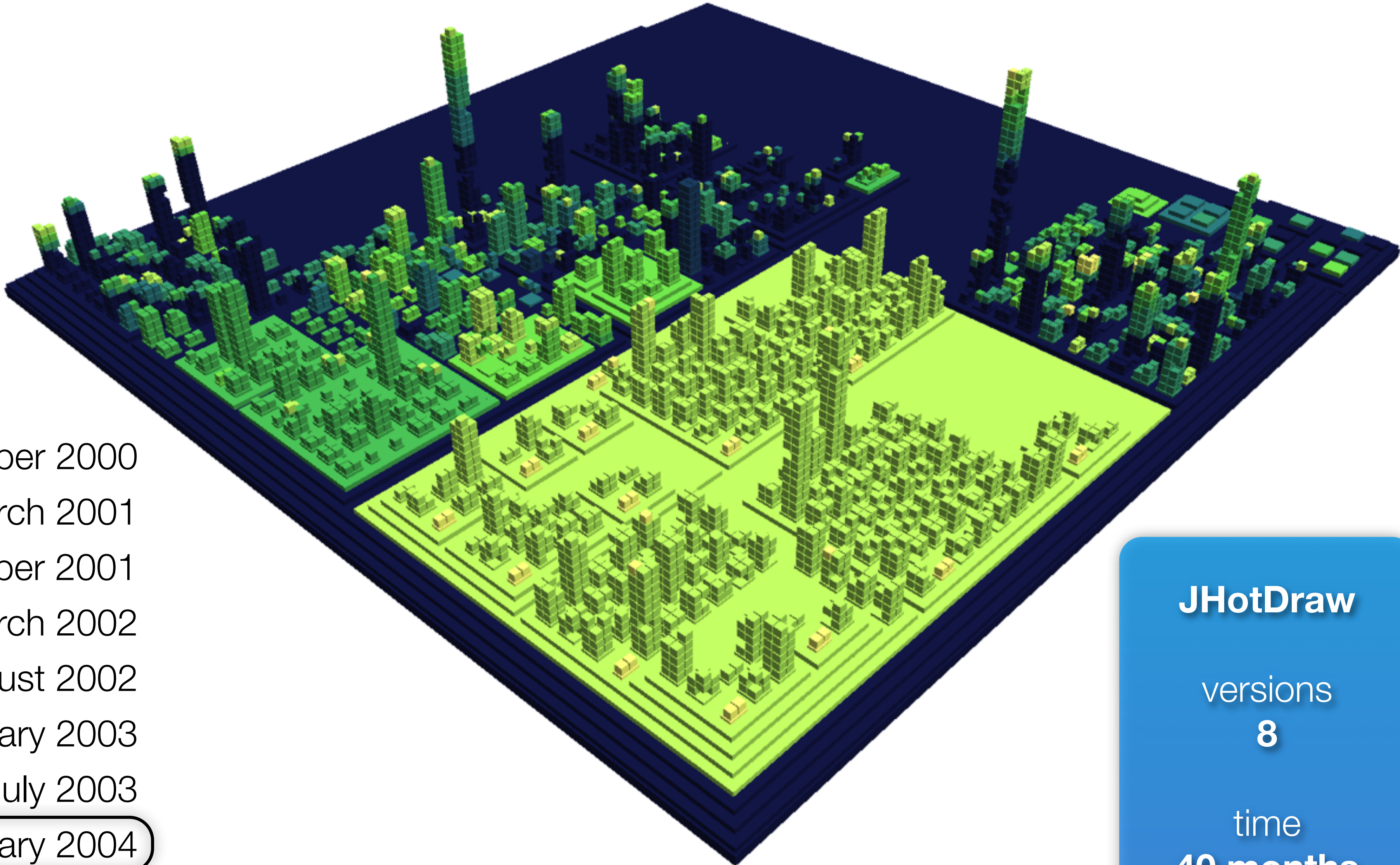
October 2000
March 2001
September 2001
March 2002
August 2002
January 2003
July 2003
January 2004

JHotDraw

versions
8

time
40 months

Time Travel + Age Map

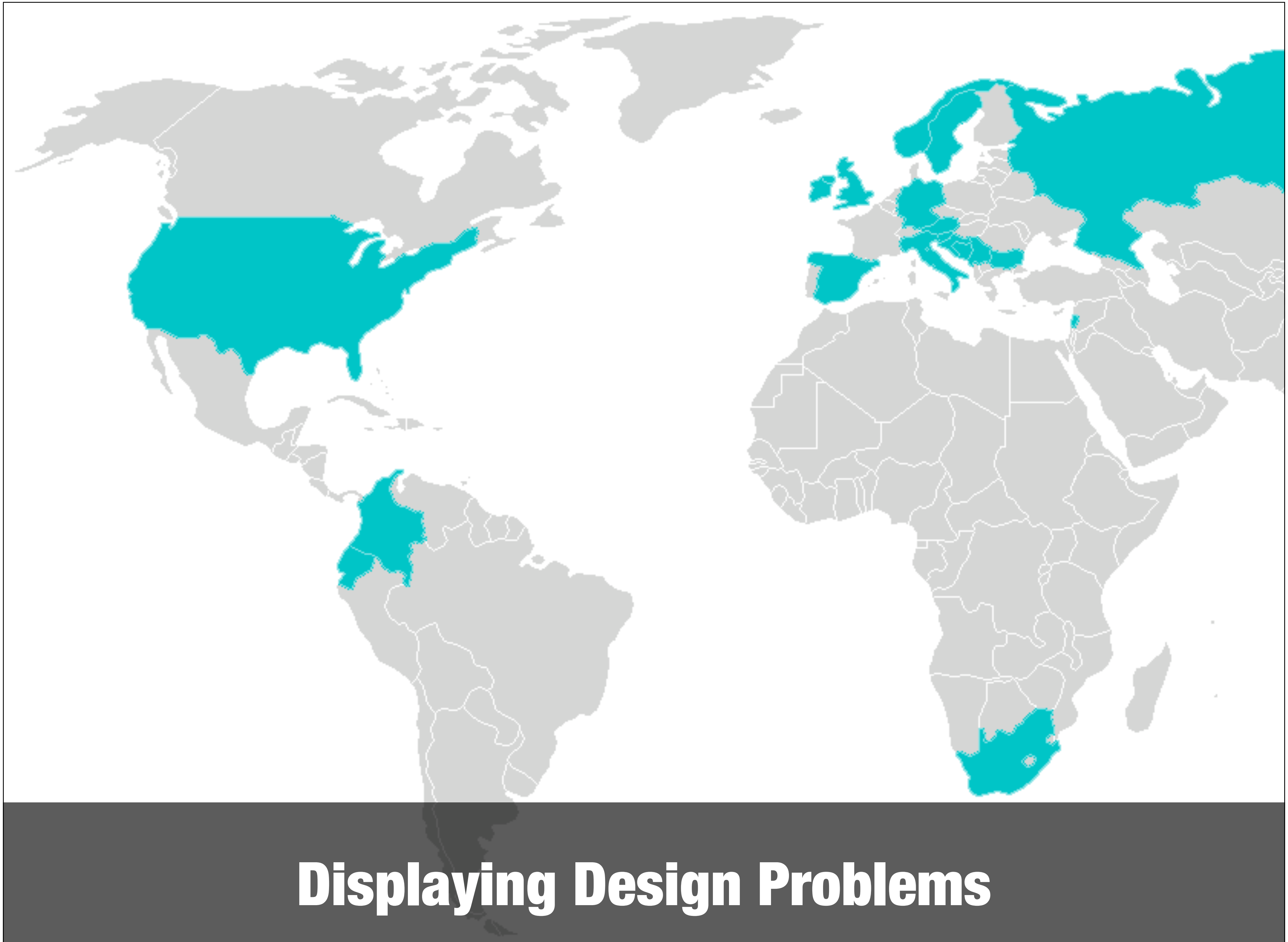


October 2000
March 2001
September 2001
March 2002
August 2002
January 2003
July 2003
January 2004

JHotDraw

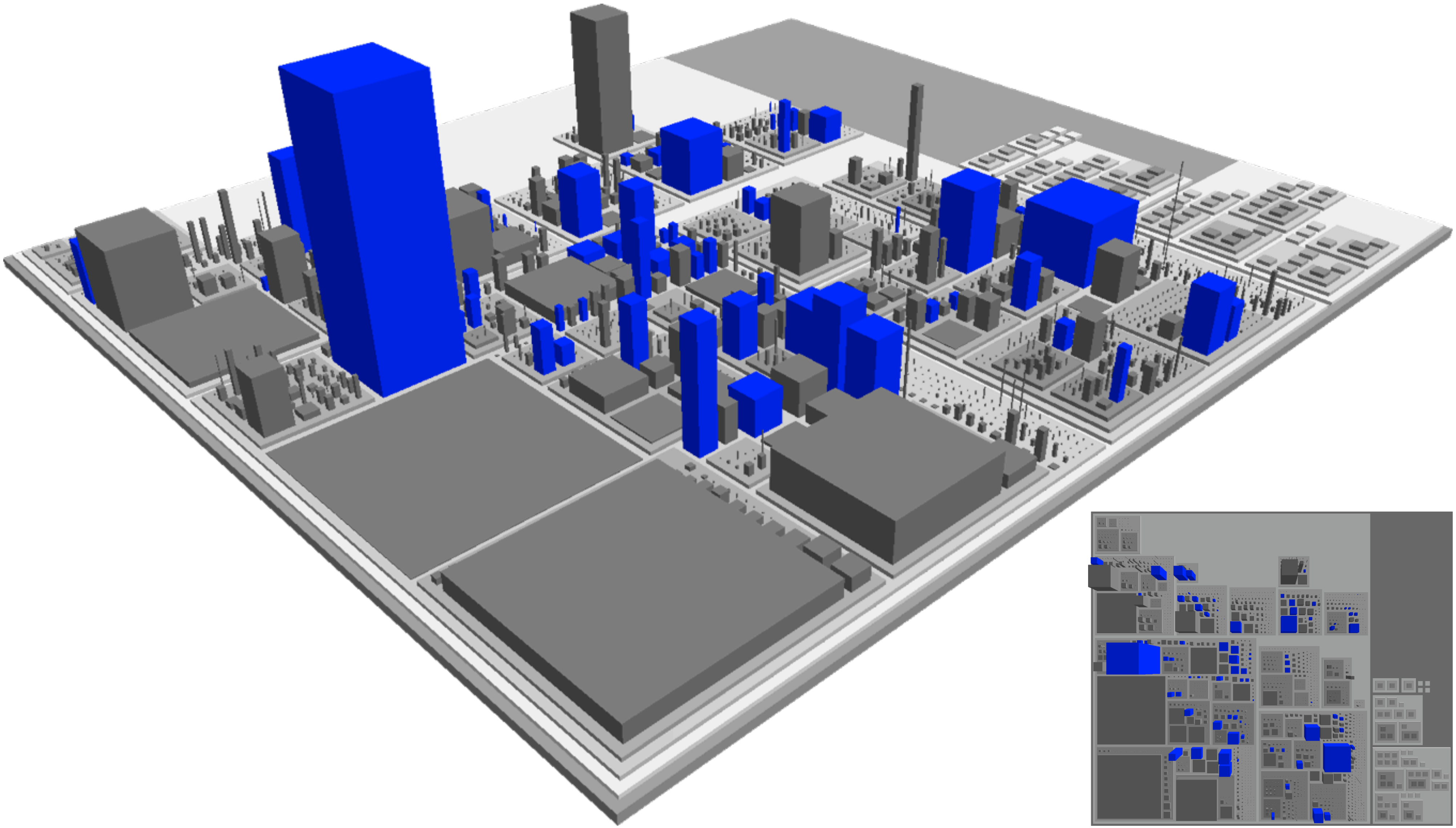
versions
8

time
40 months

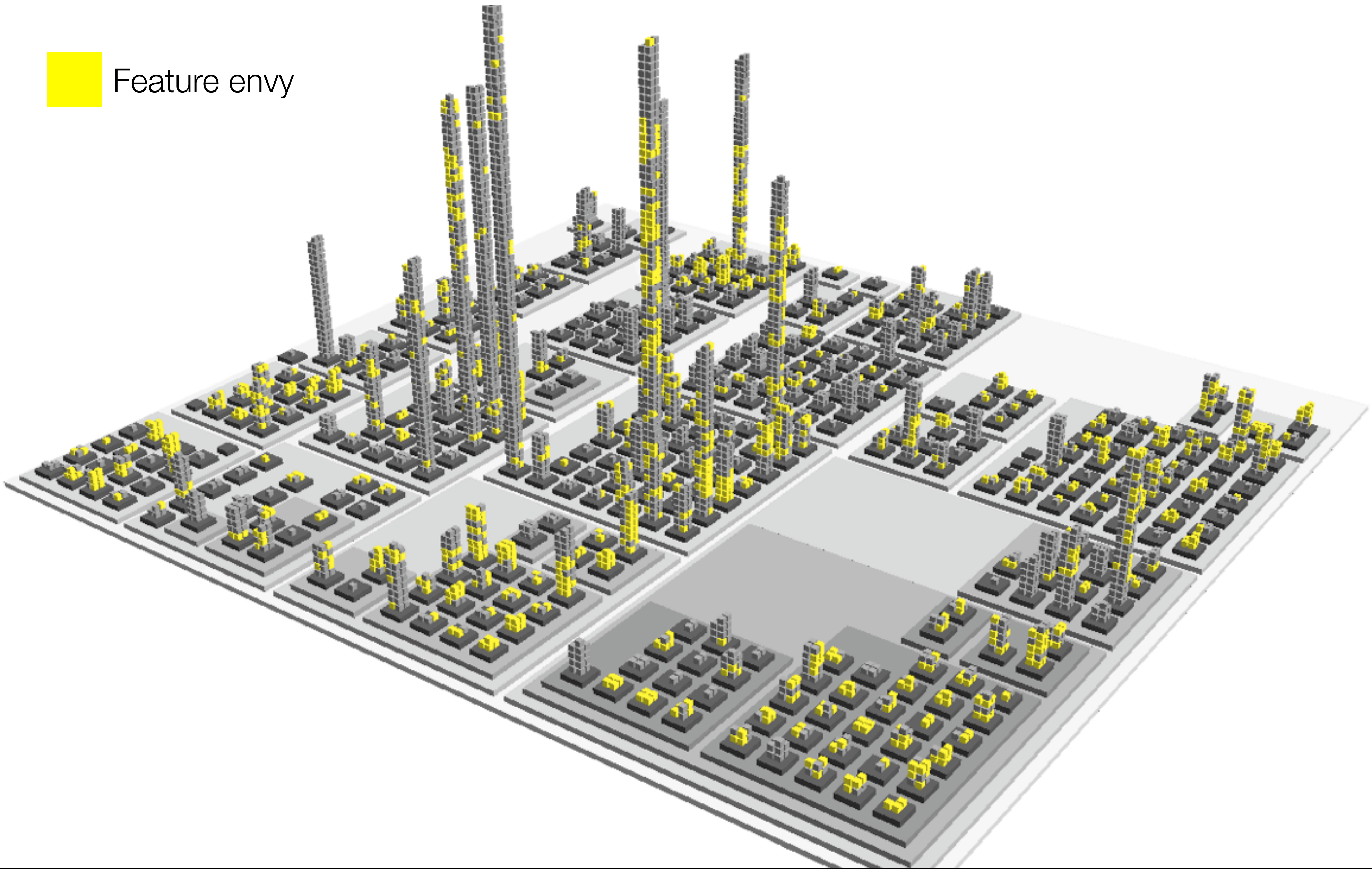


Displaying Design Problems

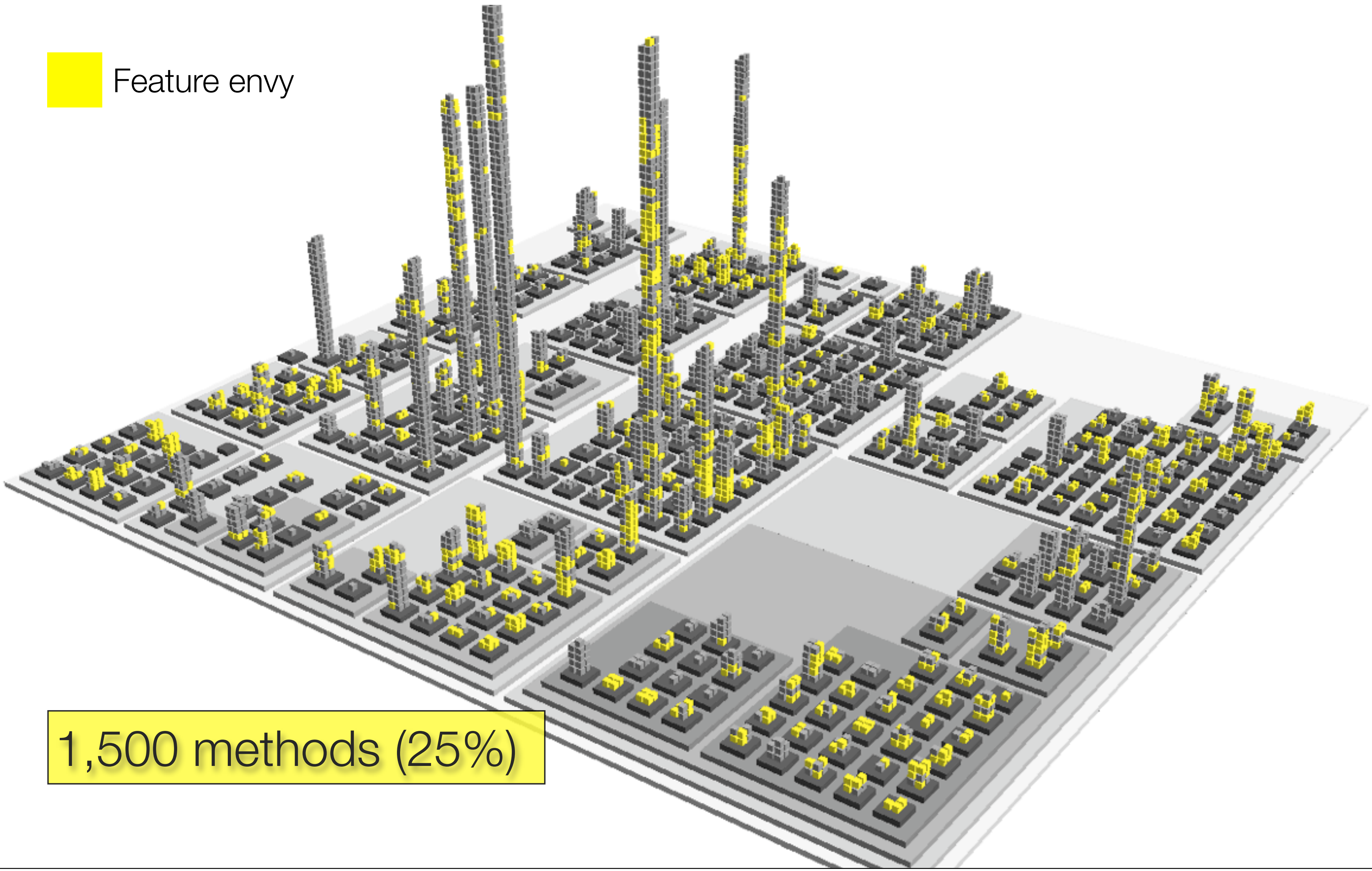
JDK 1.5 God Classes



Jmol's Feature Envy

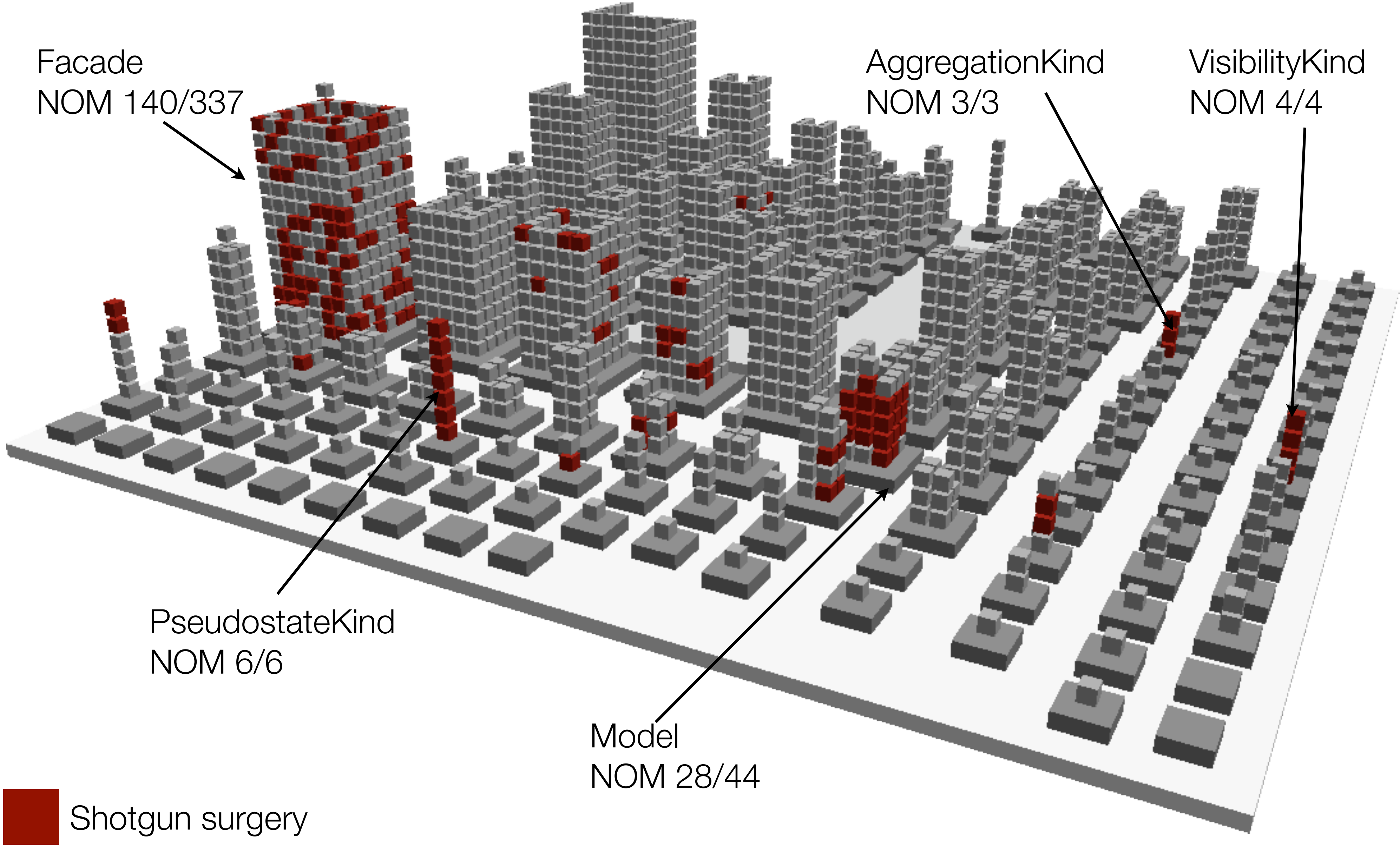


Jmol's Feature Envy



1,500 methods (25%)

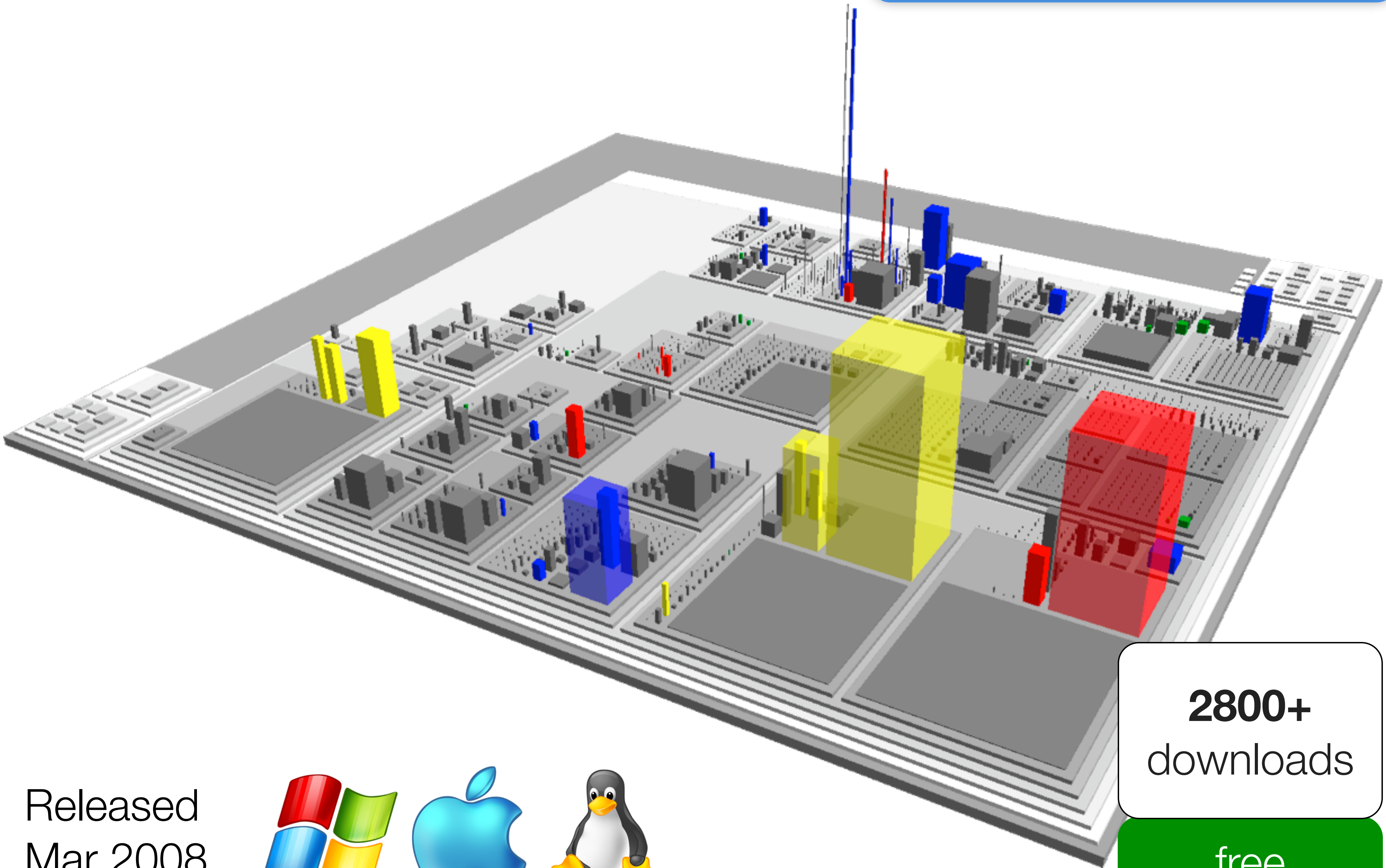
ArgoUML.Model's Shotgun Surgery Map




Tool Support

CodeCity

codecity.inf.usi.ch



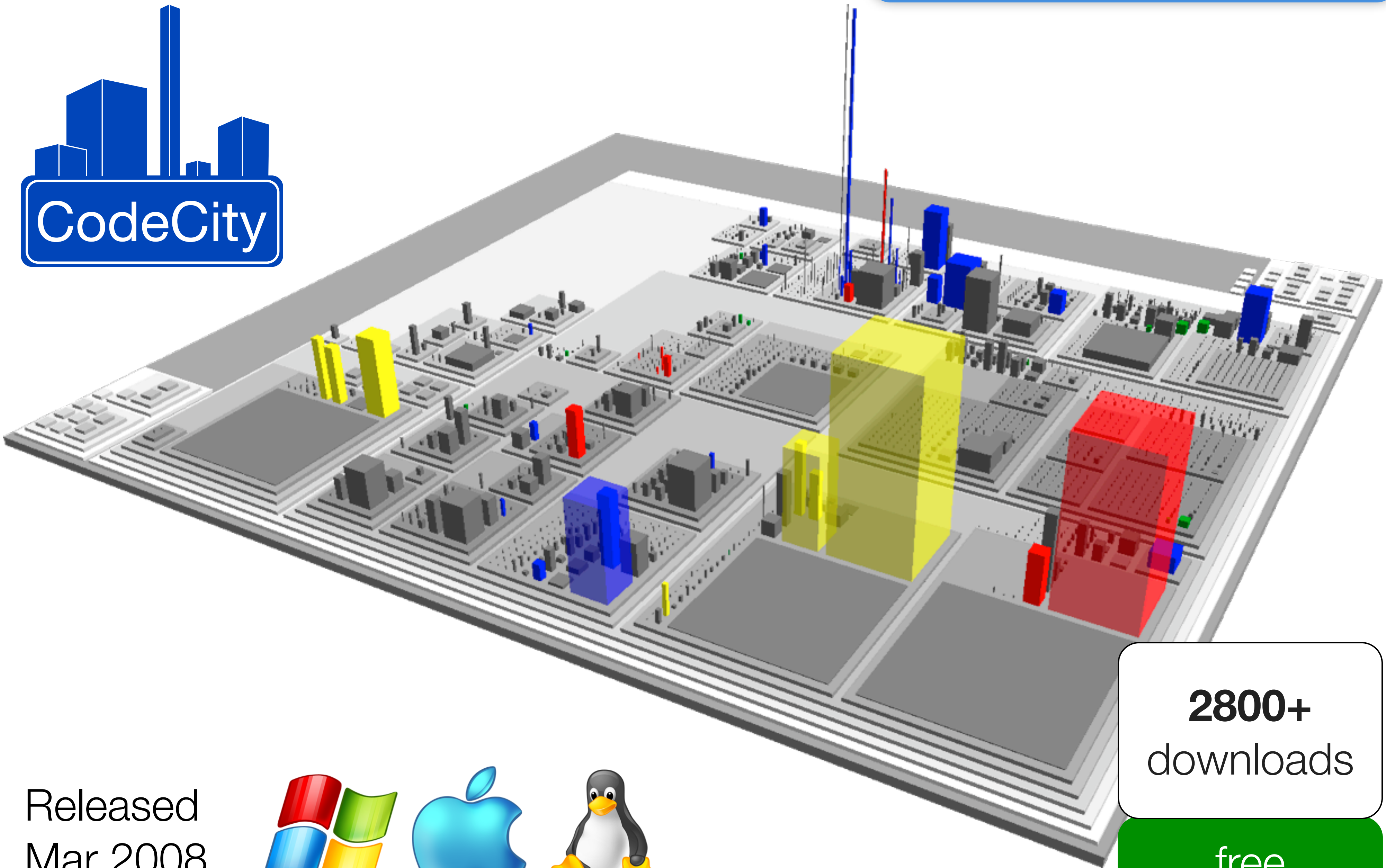
Released
Mar 2008

2800+
downloads

free

CodeCity

codecity.inf.usi.ch



Released
Mar 2008

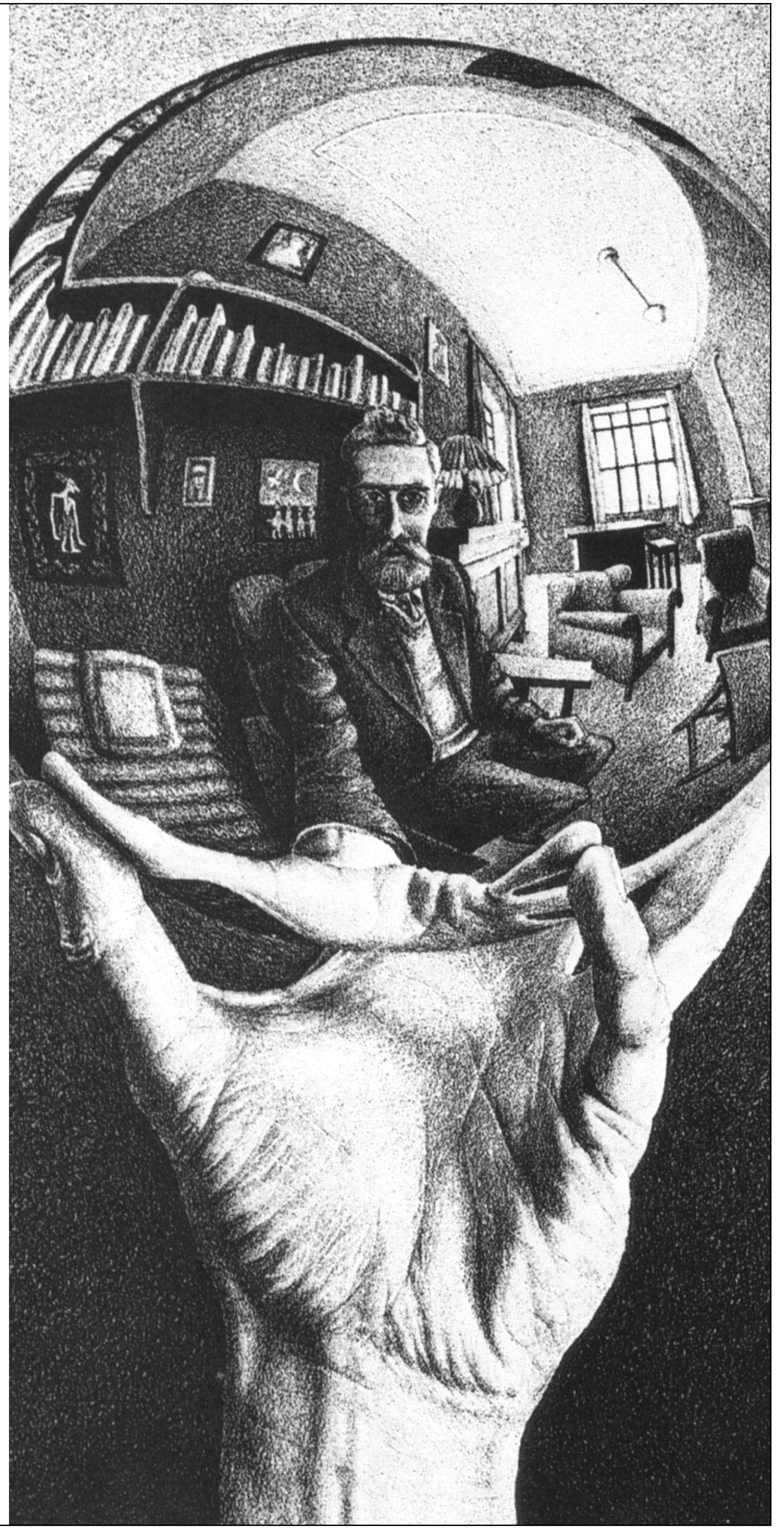
2800+
downloads

free

Part VI

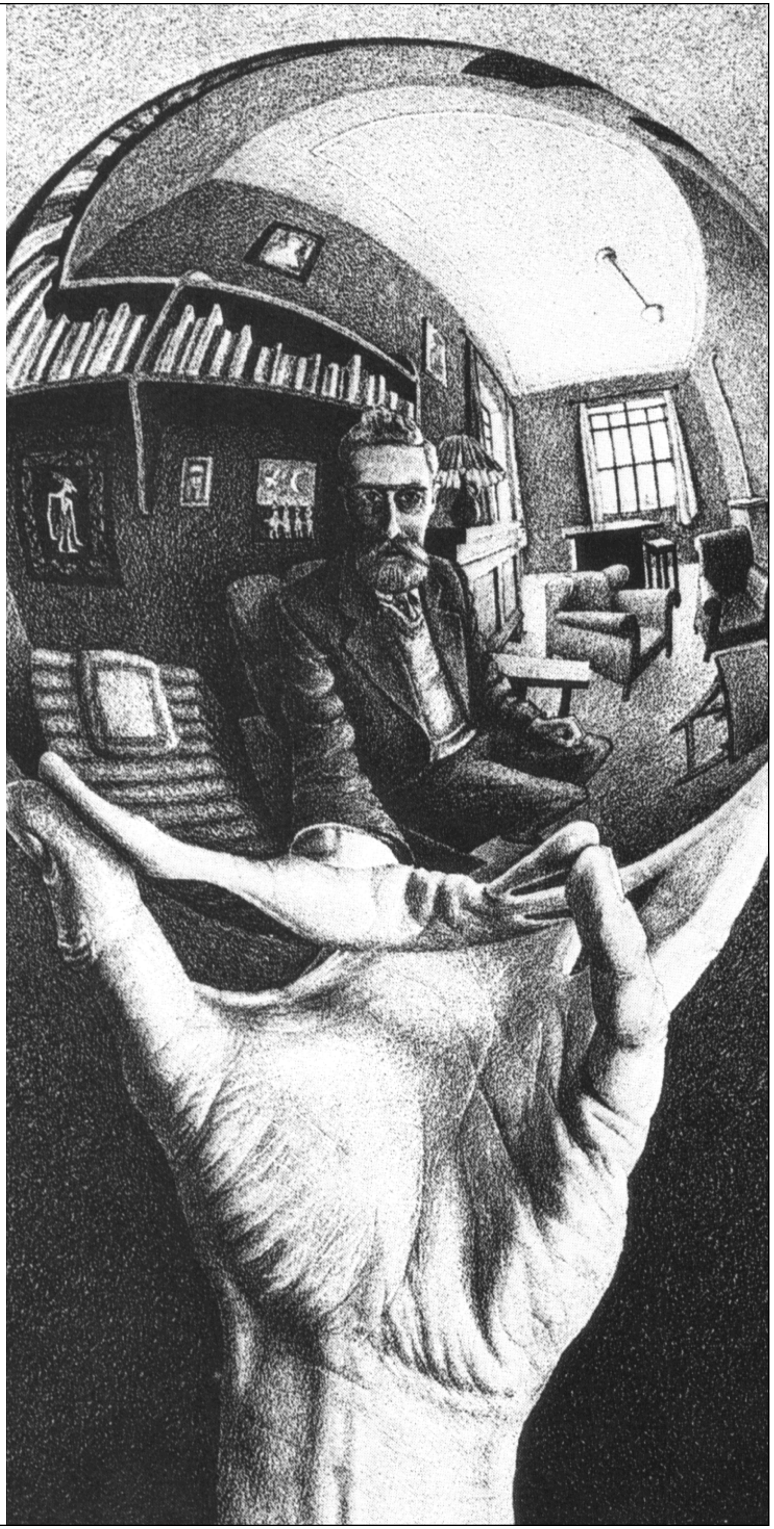
Epilogue

Reflections



Reflections

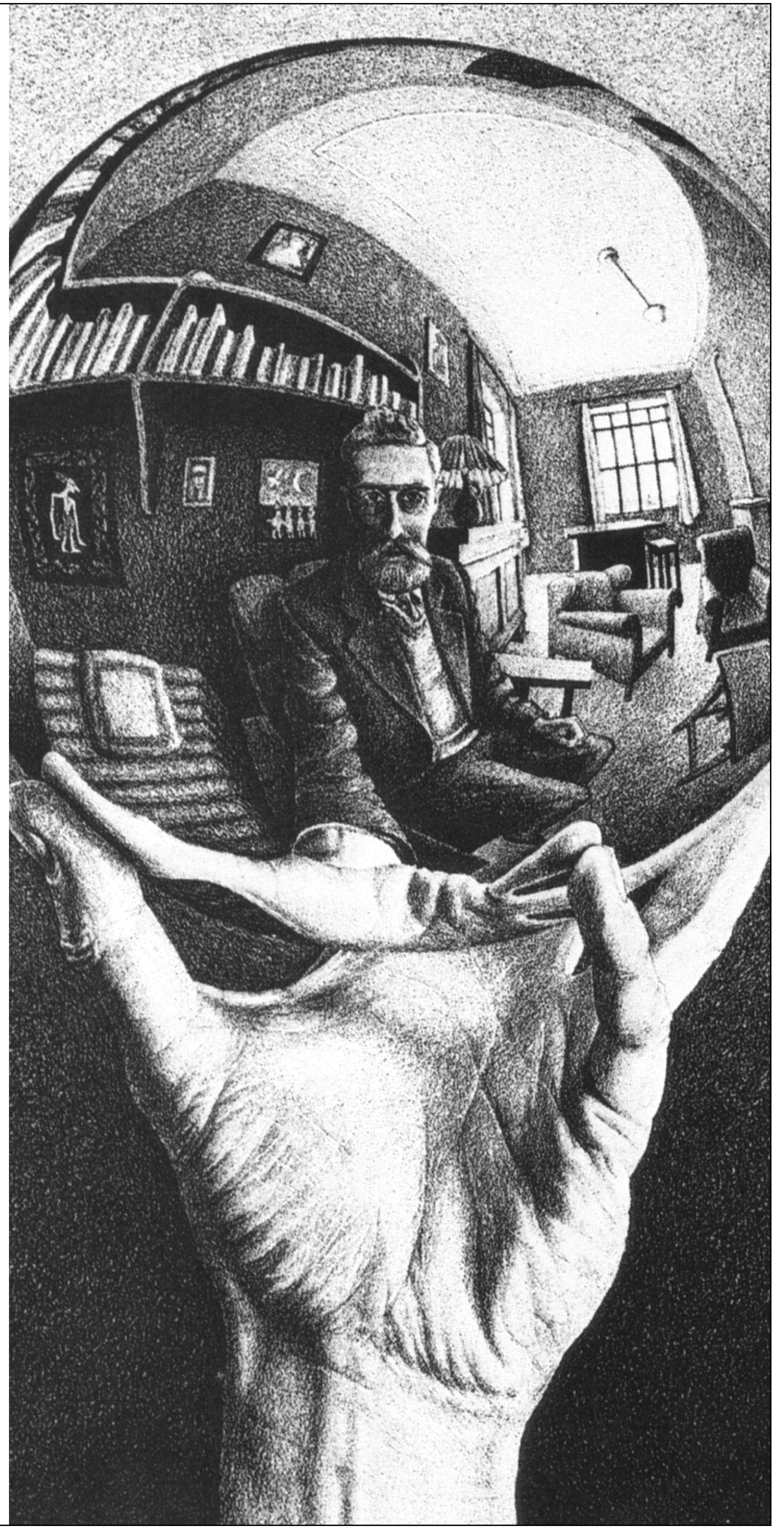
Software Visualization is



Reflections

Software Visualization is

a means to make the intangible tangible

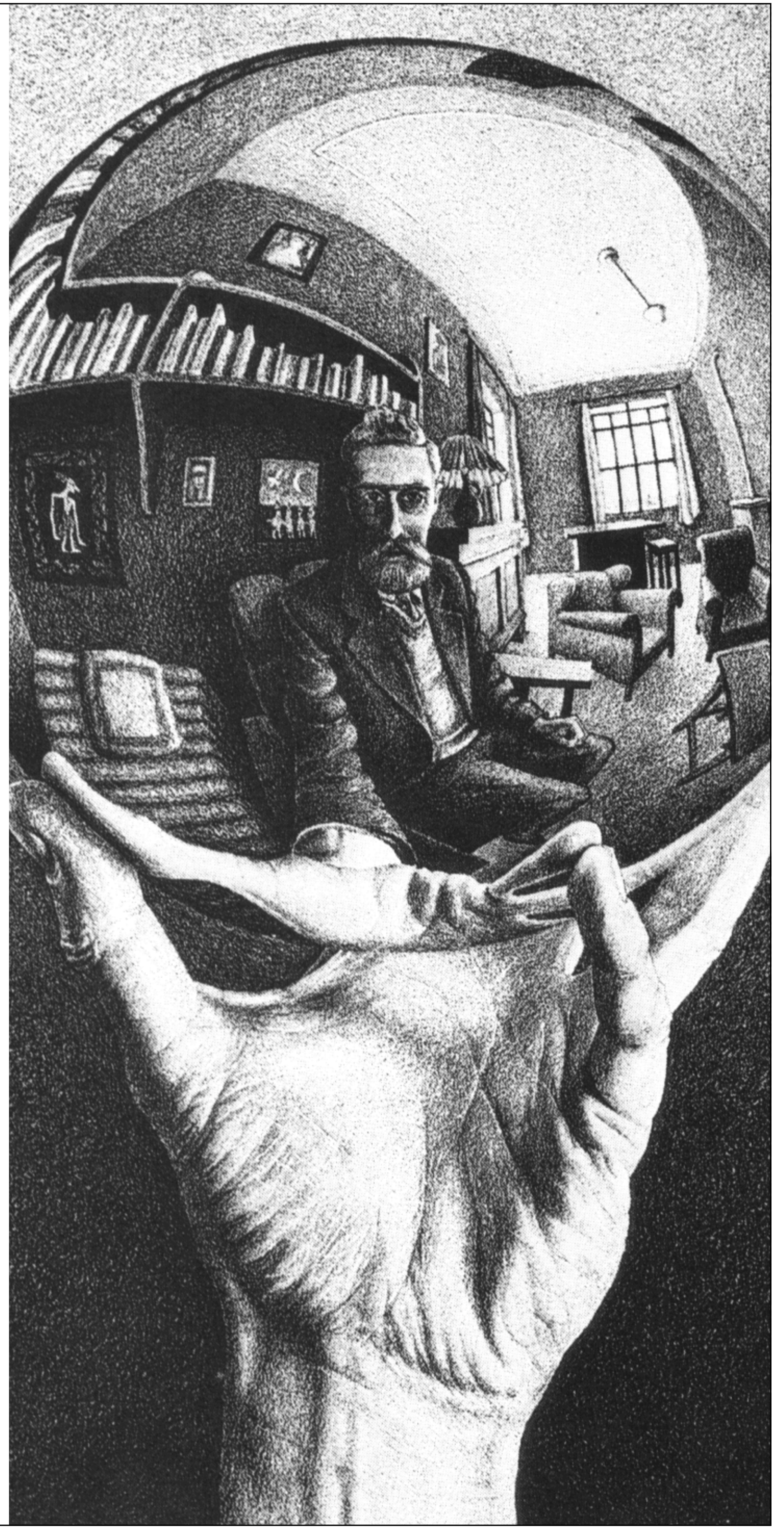


Reflections

Software Visualization is

a means to make the intangible tangible

not so difficult after all



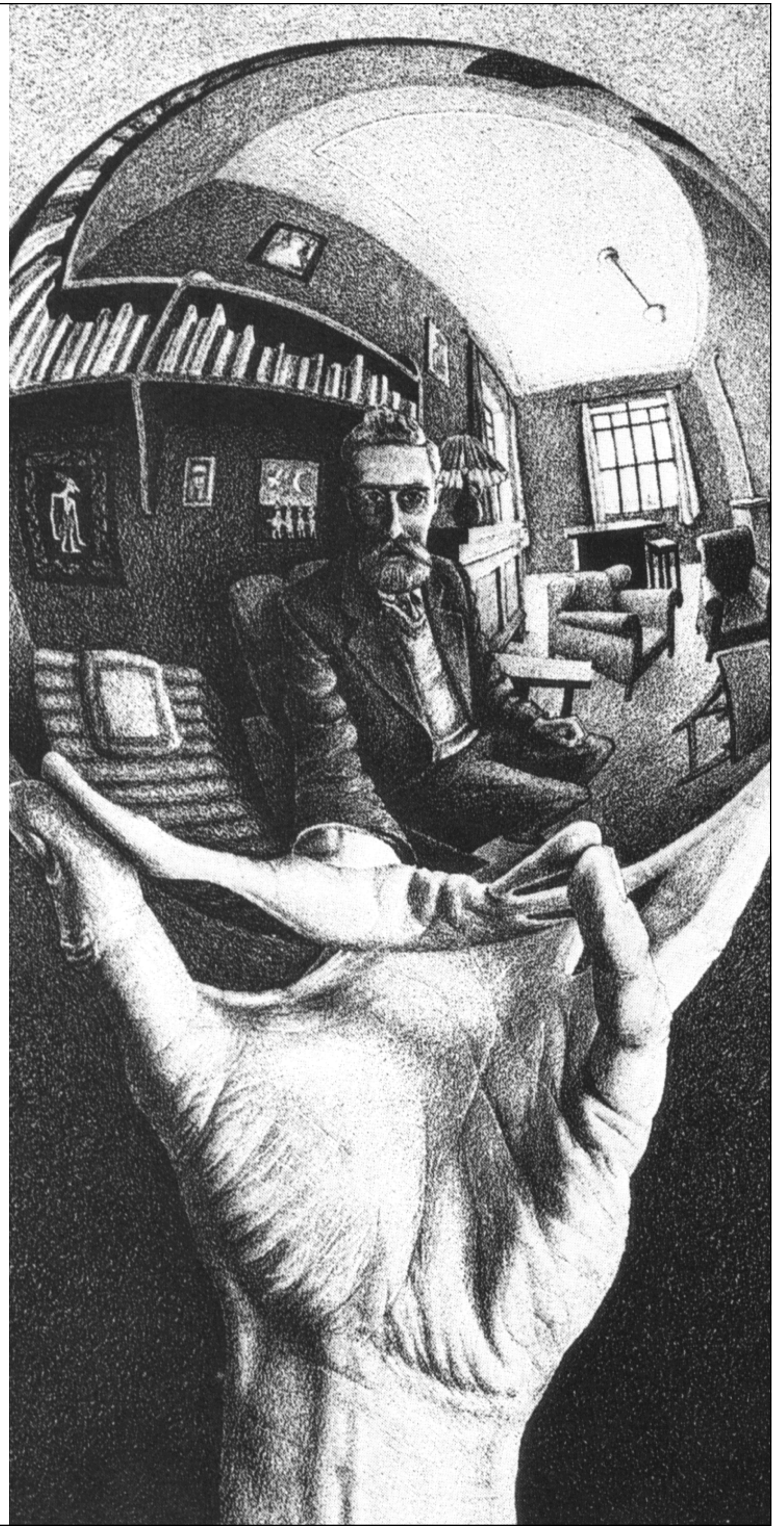
Reflections

Software Visualization is

a means to make the intangible tangible

not so difficult after all

still in its infancy



Reflections

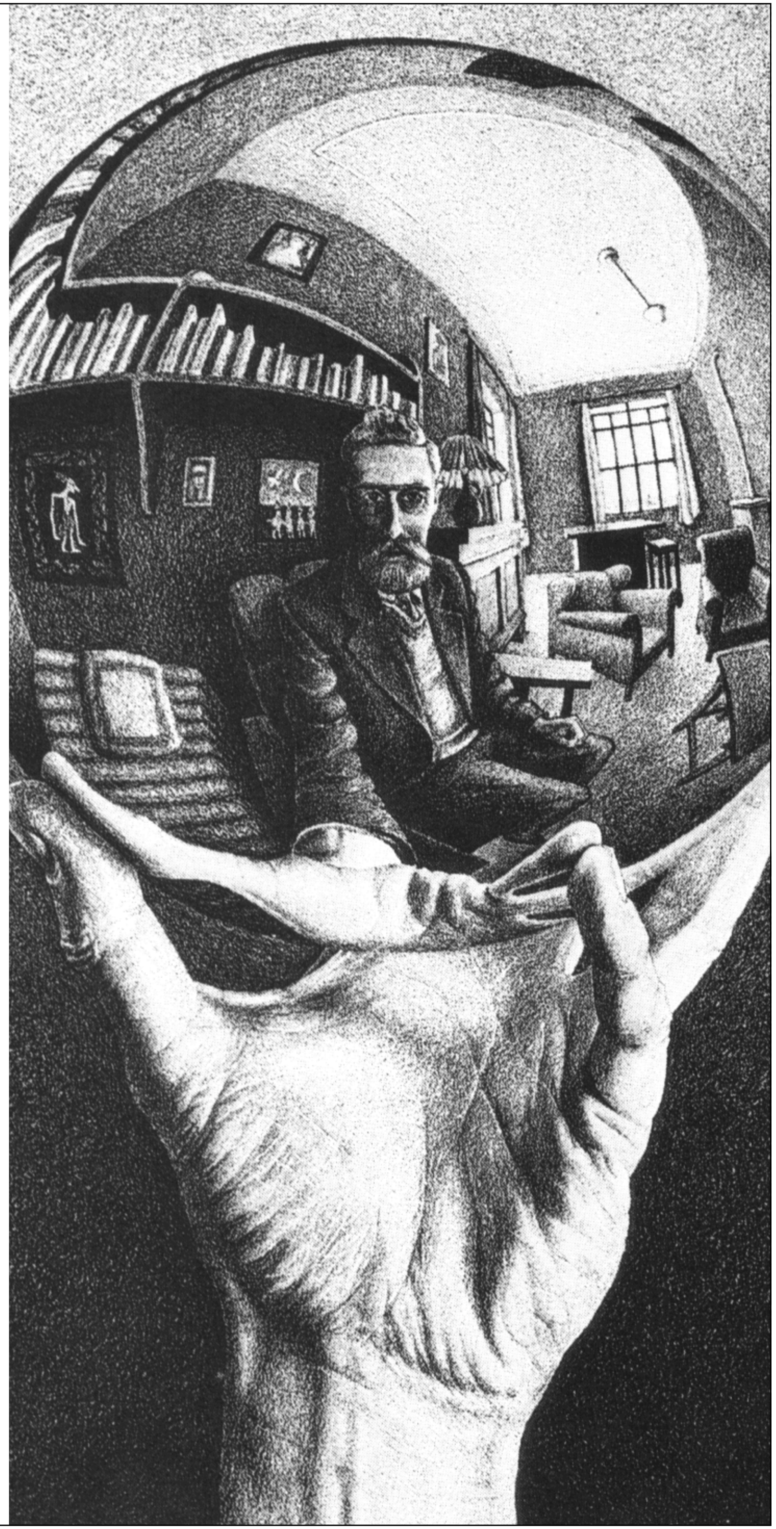
Software Visualization is

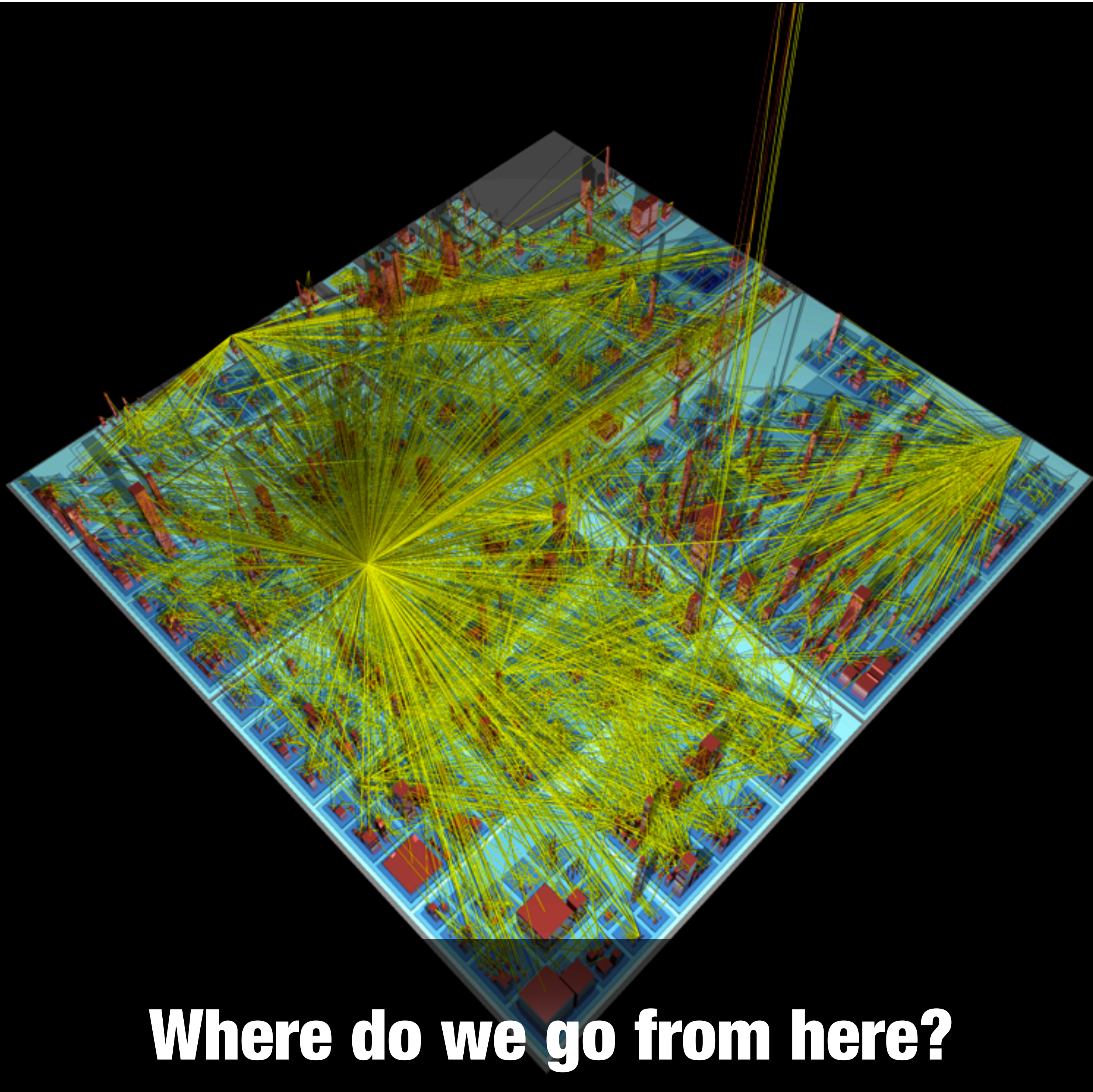
a means to make the intangible tangible

not so difficult after all

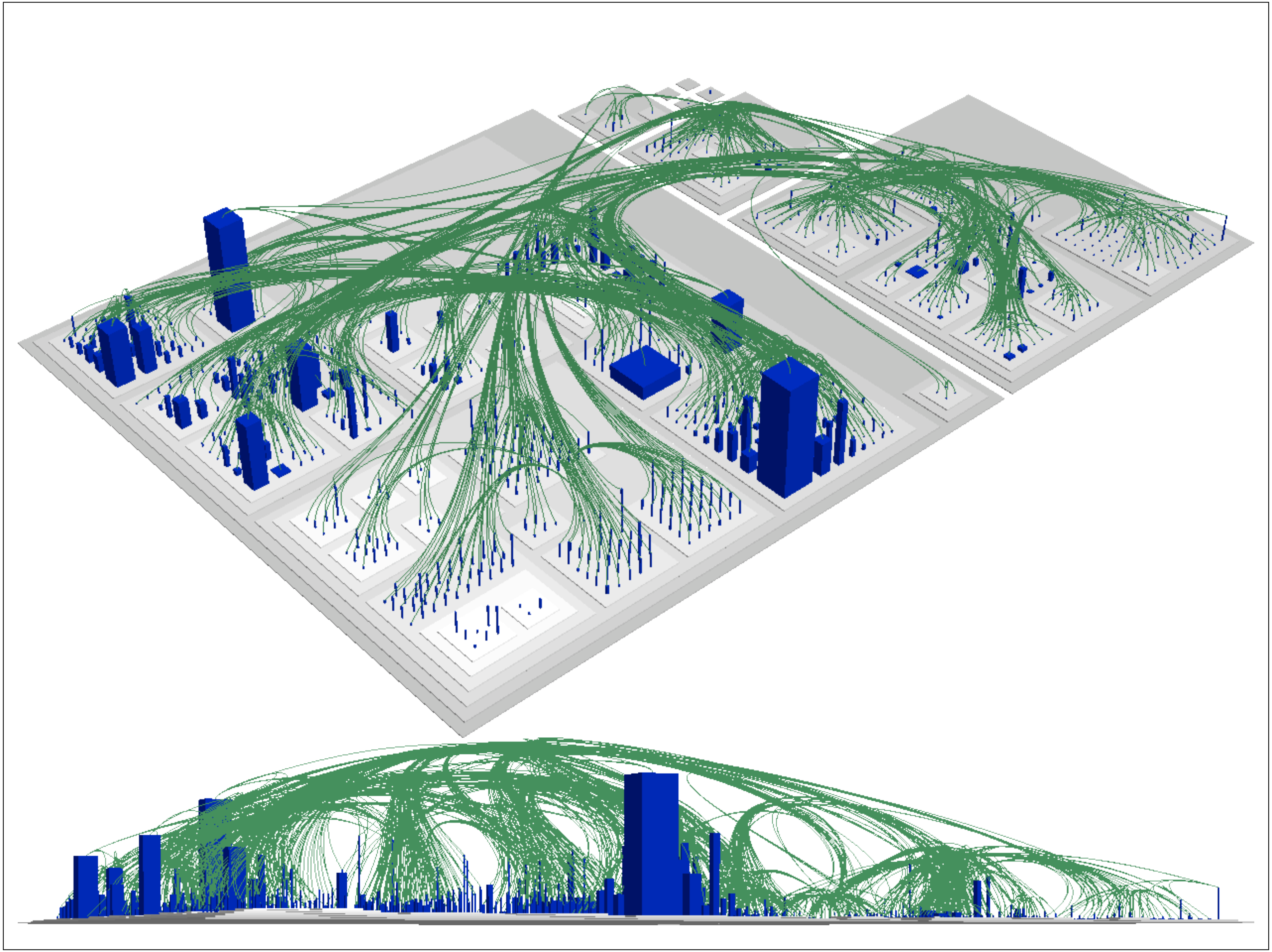
still in its infancy

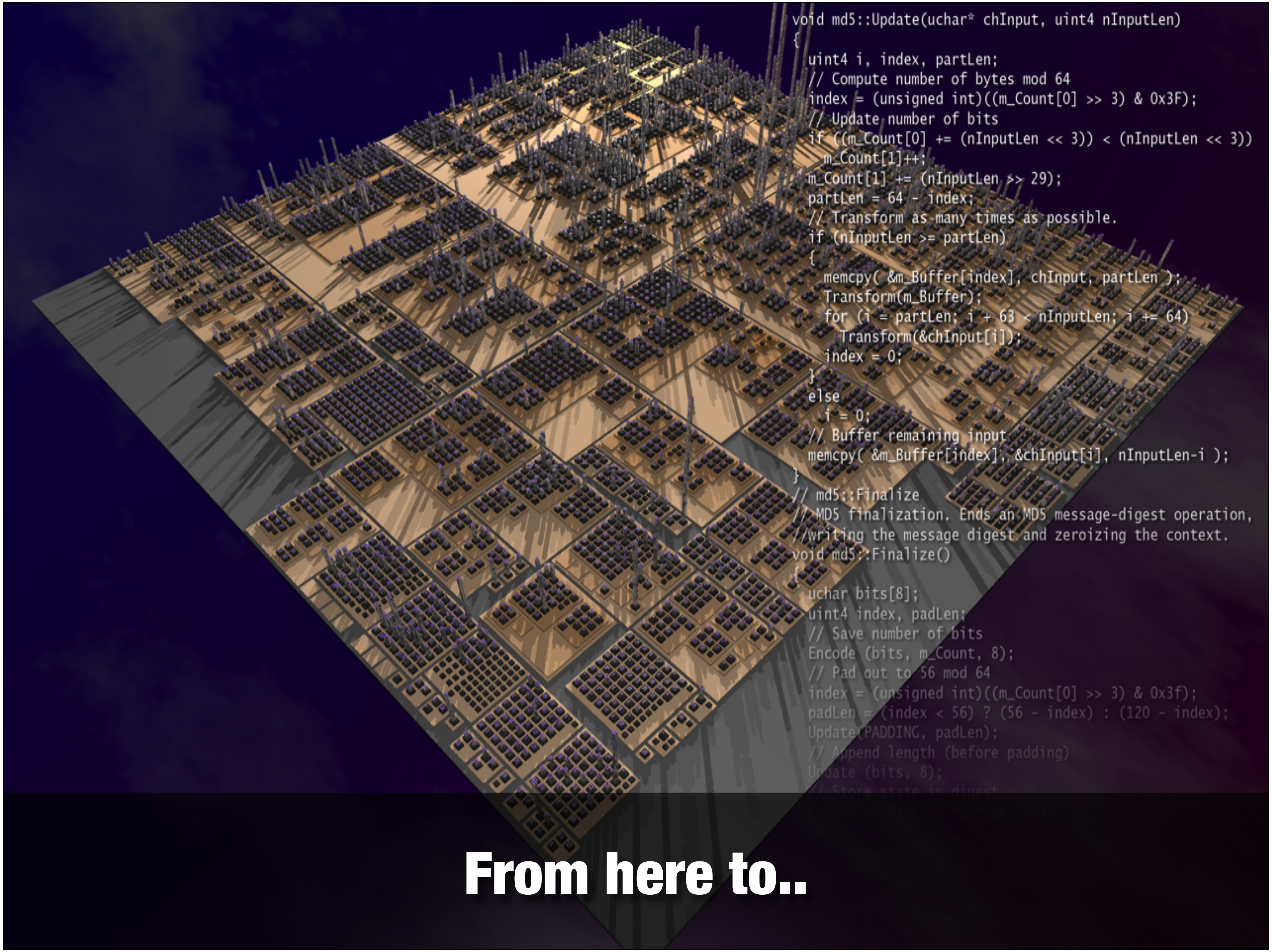
an exciting research area





Where do we go from here?




void md5::Update(uchar* chInput, uint4 nInputLen)
{
uint4 i, index, partLen;
// Compute number of bytes mod 64
index = (unsigned int)((m_Count[0] >> 3) & 0x3F);
// Update number of bits
if ((m_Count[0] += (nInputLen << 3)) < (nInputLen << 3))
m_Count[1]++;
m_Count[1] += (nInputLen >> 29);
partLen = 64 - index;
// Transform as many times as possible.
if (nInputLen >= partLen)
{
memcpy(&m_Buffer[index], chInput, partLen);
Transform(m_Buffer);
for (i = partLen; i + 63 < nInputLen; i += 64)
Transform(&chInput[i]);
index = 0;
}
else
i = 0;
// Buffer remaining input
memcpy(&m_Buffer[index], &chInput[i], nInputLen-i);
}
// md5::Finalize
// MD5 finalization. Ends an MD5 message-digest operation,
//writing the message digest and zeroizing the context.
void md5::Finalize()
{
uchar bits[8];
uint4 index, padLen;
// Save number of bits
Encode (bits, m_Count, 8);
// Pad out to 56 mod 64
index = (unsigned int)((m_Count[0] >> 3) & 0x3f);
padLen = (index < 56) ? (56 - index) : (120 - index);
Update(PADDING, padLen);
// Append length (before padding)
Update (bits, 8);
From here to..

Software Visualization 101+

Michele Lanza

REVEAL @ Faculty of Informatics

University of Lugano, Switzerland