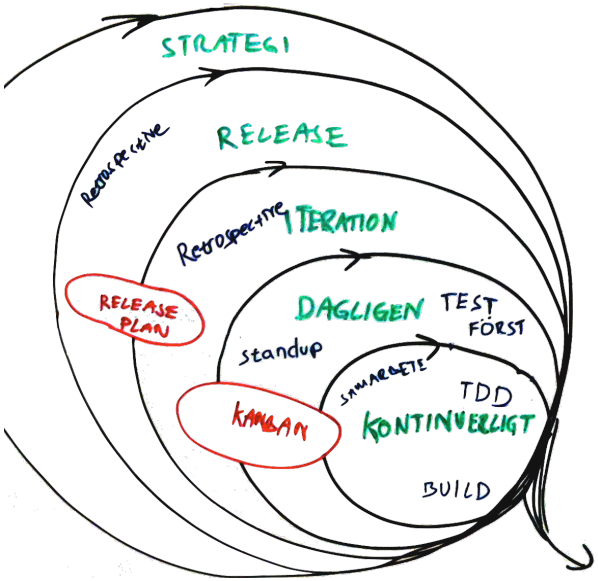
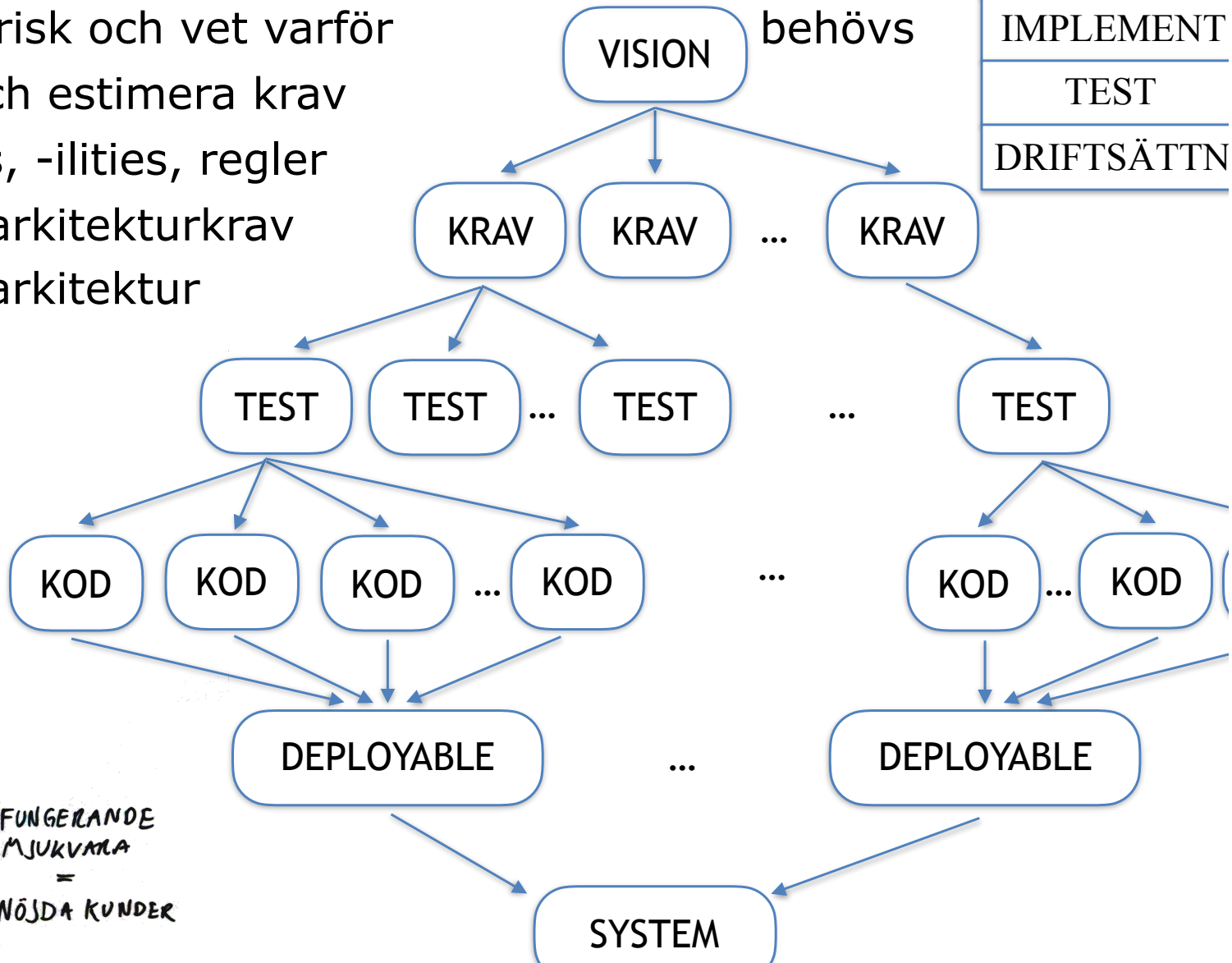
Feedback?

Den Röda Tråden

- Vi kan välja utvecklingsmodell
- Vi kan hantera risk och vet varför
- Vi kan skriva och estimera krav
  - User stories, -ilities, regler
- Vi kan ta fram arkitekturkrav
- Vi kan ta fram arkitektur



FUNGERANDE MJUKVARA = NÖJDA KUNDER

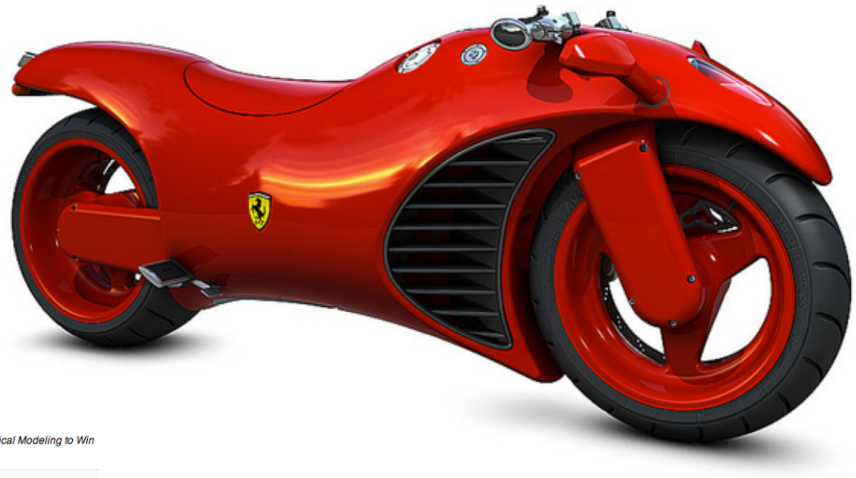


google define: design

google define: design

“the act of working out the form of something”

Vad är design?

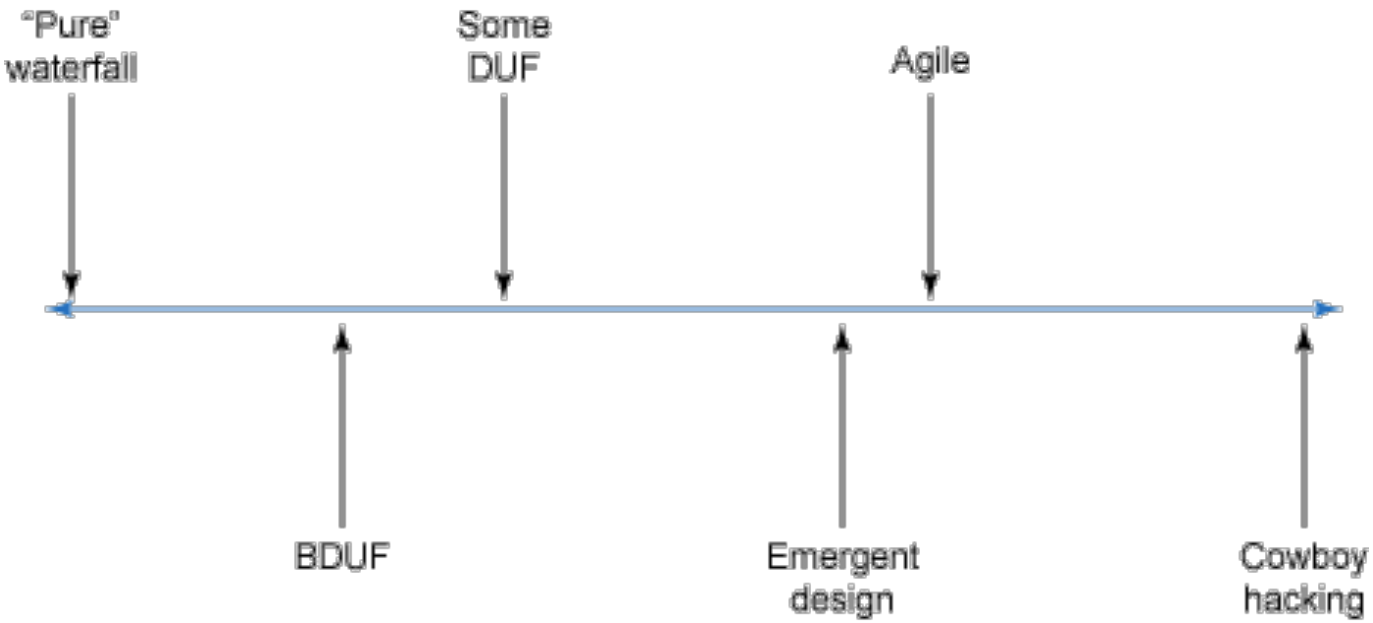


© Glinik - 2008

Another example is the following flight simulator, the winner of the 1998 IOCCC,[20] as listed and described in *Calculated Bets: Computers, Gambling, and Mathematical Modeling to Win* (1991)[21] and shown below:

```
#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,
                                     B[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*P/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 =1
                                        /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); } }
```

Incremental Design
Evolutionary Design
Emergent Design



http://www.ibm.com/developerworks/library/j-eaed1/

Incremental Design

1. Välj ett problem
2. Lös det på **enklast möjliga sätt**
3. Förenkla och tydliggör (Abstrahera, Generalisera)
4. Gå till 1

Hur tar vi fram en arkitektur?

Utgå från vetenskapligt sätt

1. Sätt upp hypotes - kandidatarkitektur!
2. Testa hypotesen utifrån arkitekturellt signifikanta user stories och -ilities-scenarion.
3. Förändra hypotesen om den inte håller!

Designelement

Metod

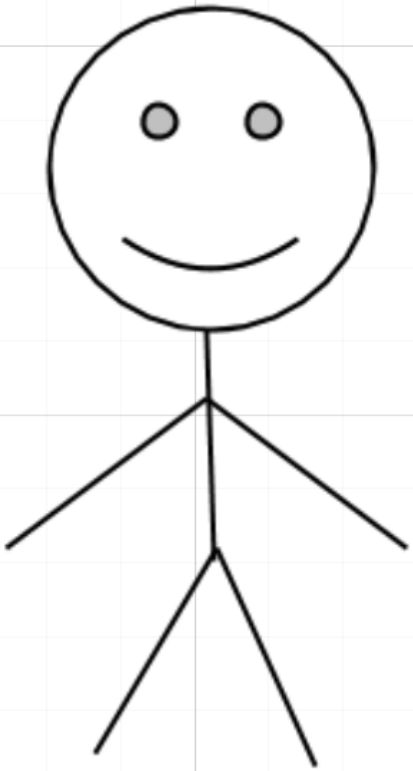
Klass

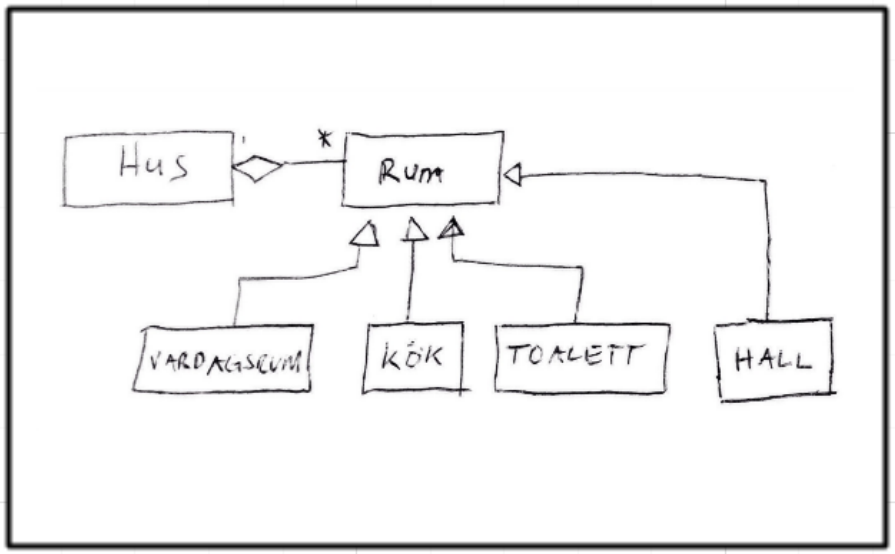
Komponent

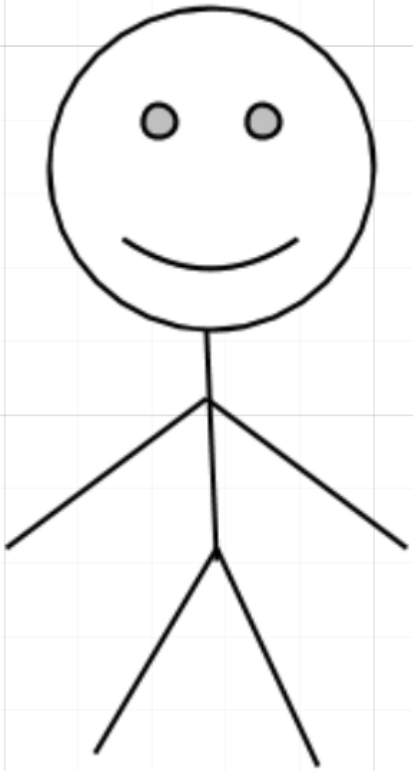
Tjänst

Etc…

För vem?







Designprinciper

- Abstraction
- Decomposition and modularization
- Coupling and Cohesion
- Encapsulation and information hiding
- Consistency (Ubiquituous language)

Encapsulation vs information hiding

Encapsulation

- Bundling methods with data
- Does not mandate private data

Information hiding (David Parnas 1972)

- Private data
- Hiding not only data but changes in design and/or implementation

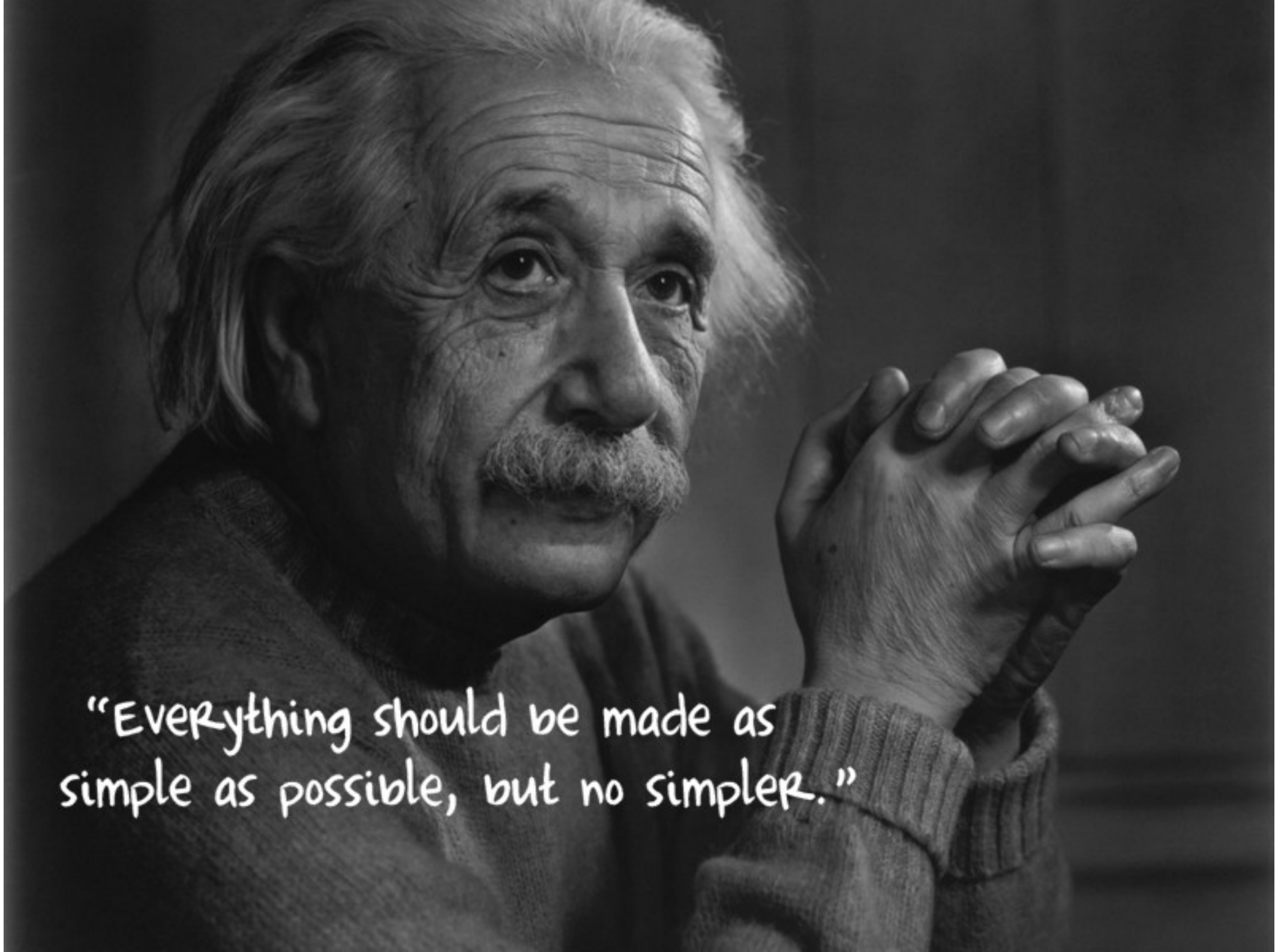
http://www.javaworld.com/article/2075271/core-java/encapsulation-is-not-information-hiding.html

Designprinciper

- Reveal Intent - *Naming*
- SRP: Single Responsibility Principle
  - One and one reason only to change
- DRY: Don’t Repeat Yourself
  - Duplication: Missed opportunity for abstraction!
- KISS: Keep it simple, stupid!
- YAGNI: You ain’t gonna need it!

Analysis Paralysis

No one gets it right the first time!


"Everything should be made as
simple as possible, but no simpler."

Design pattern

“Each pattern describes a problem that occurs over and over again in our environment, and then describes the core of the solution to that problem...”

Christopher Alexander

Design pattern

A pattern is a **solution** to a **problem** within a certain **context** that has a set of predictable **consequences**.

Ted Neward

https://dzone.com/articles/reclaiming-design-patterns-20-years-later

Software Design Pattern

- “a general reusable solution to a commonly occurring problem within a given context”

http://en.wikipedia.org/wiki/Software_design_pattern

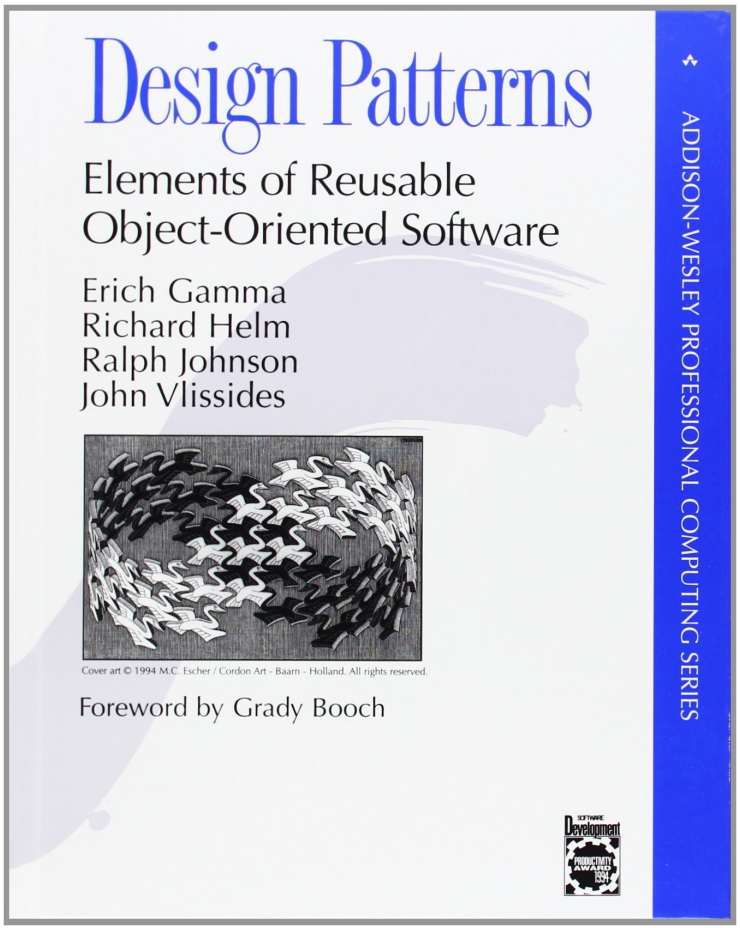
Men framförallt:

- Design patterns ger oss ett precist språk!

- “Begreppsmodell för domänen software engineering”

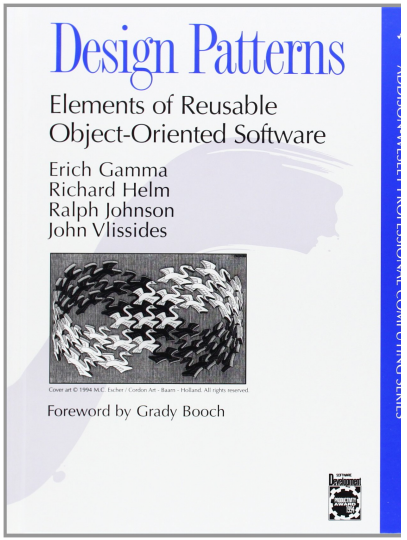
GoF

- Creational
  - Builder,
  - Factory method, …
- Structural
  - Adapter, Decorator, …
- Behavioral
  - Iterator, Observer, …



"workarounds for missing features in C++"?

GoF



“Design patterns provide a common vocabulary for designers to use to communicate, document, and explore design alternatives. Design patterns make a system seem less complex by letting you talk about it at a higher level of abstraction than that of a design notation or programming language. Design patterns raise the level at which you design and discuss design with your colleagues.”

Design patterns

Factory vs Builder

Static Factory Method

*Static Factory Method is **not** Factory, Factory method or Abstract Factory*

When: Creation is a bit complex for constructor.

Then: Create a static method that returns an instance of a class

"Static factory method is simply a static method that returns an instance of a class" Effective Java by Joshua Bloch

Se also:

- http://stackoverflow.com/questions/929021/what-are-static-factory-methods-in-java
- http://stackoverflow.com/questions/13029261/design-patterns-factory-vs-factory-method-vs-abstract-factory

Builder

When:

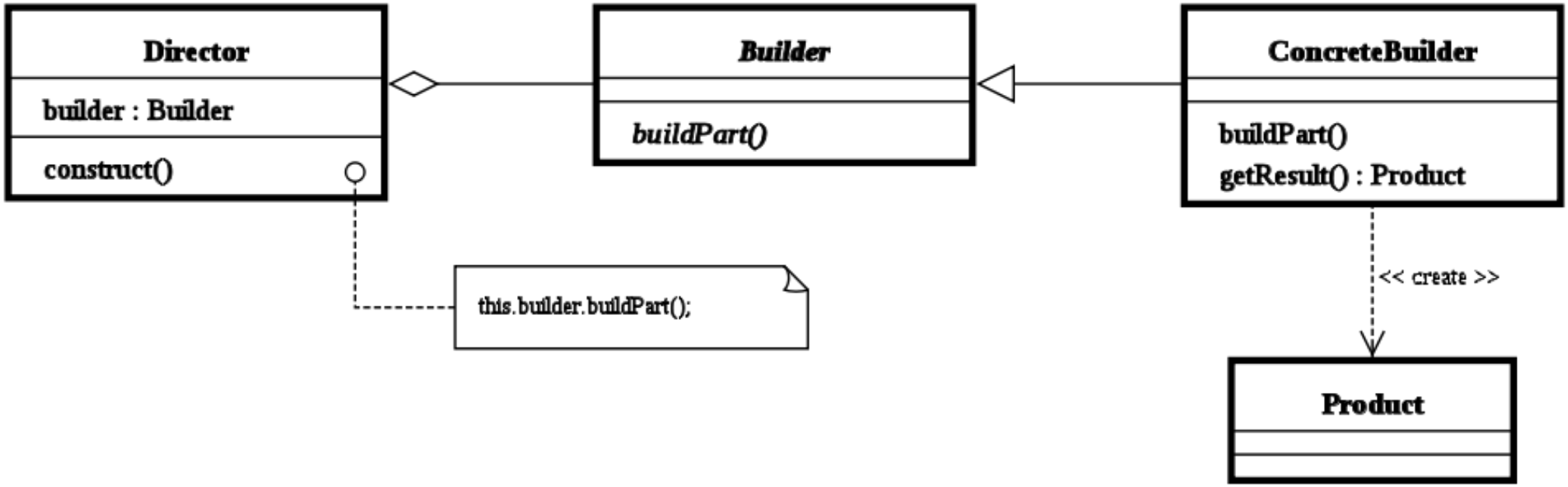
Creation is a bit complex for constructor, and even static factory method; several input parameters and/or several optionals leading to multiple variations of constructors.

Then:

Create a fluent interface Builder that takes each parameter one by one and then returns an instance of a class in a single step.

- http://en.wikipedia.org/wiki/Builder_pattern

Builder



– http://en.wikipedia.org/wiki/Builder_pattern

Data persistence patterns

SRP: Separate logic and persistence!

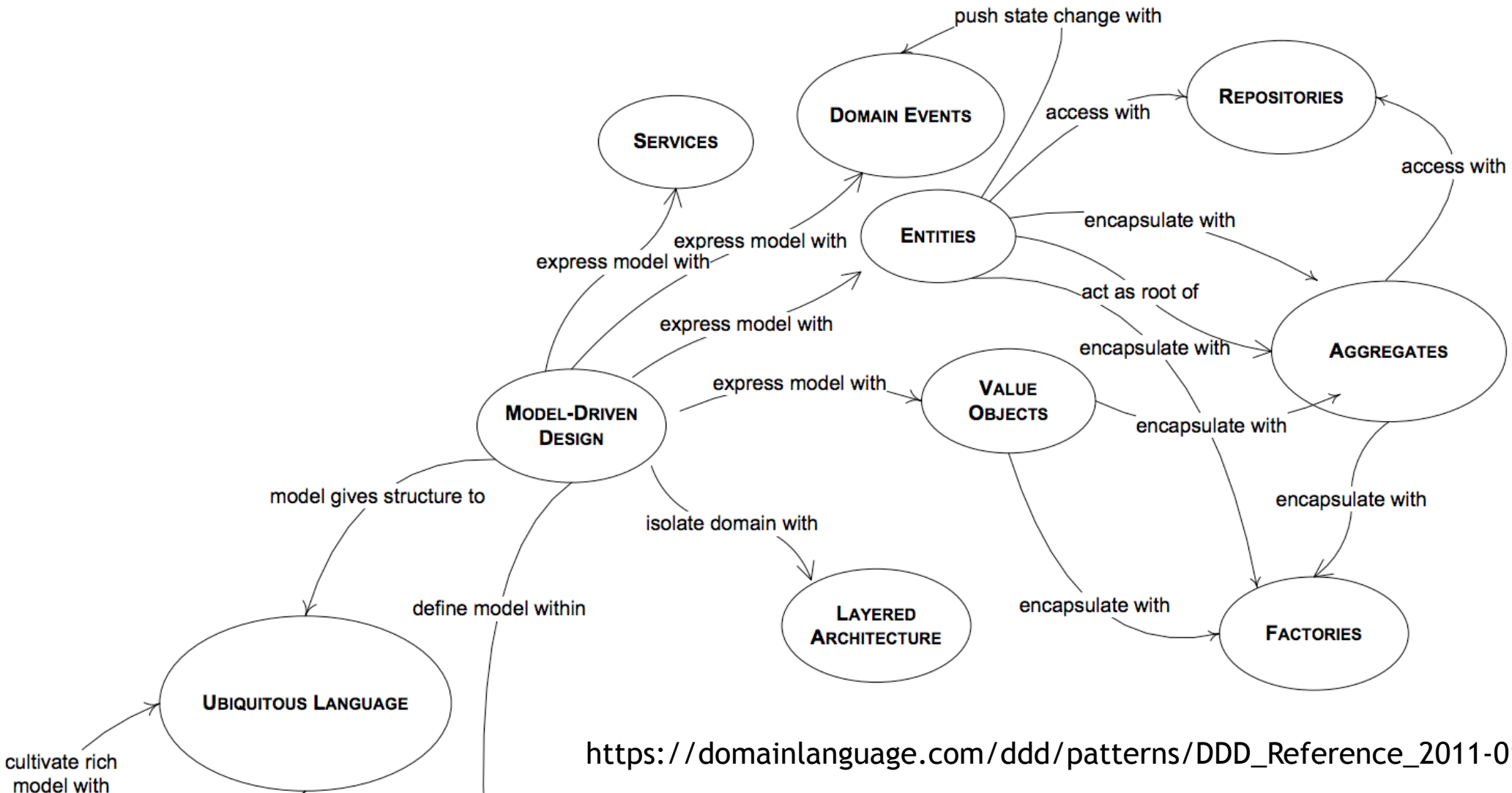
- DAO
  - CRUD for one entity
- Repository
  - CRUD* for aggregate: entity and it’s abstractions
  - *Not actually CRUD, but methods reflecting domain

Pattern Language

“A pattern language is a method of describing good design practices within a field of expertise”

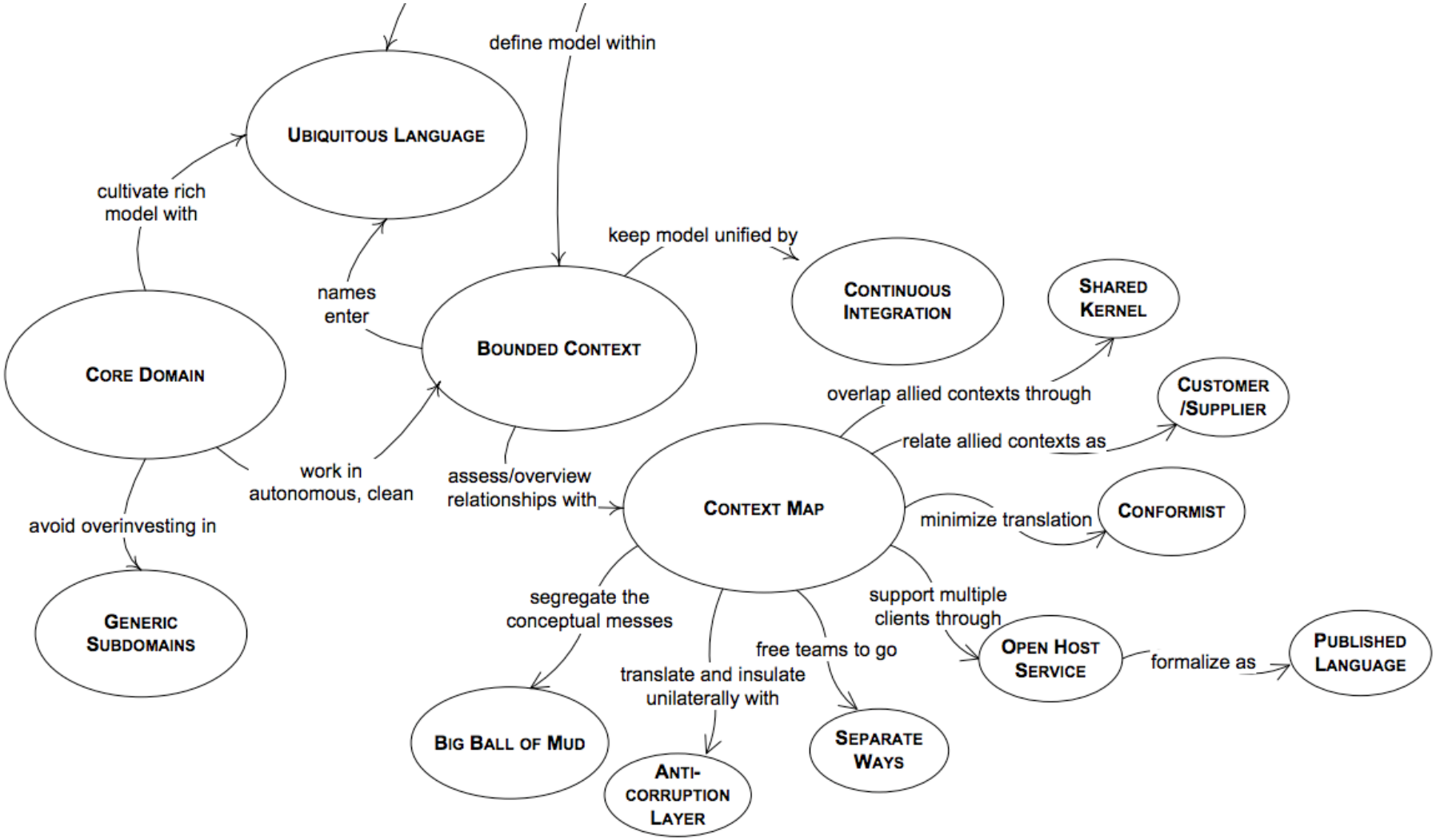
http://en.wikipedia.org/wiki/Pattern_language

Tactical DDDesign patterns



https://domainlanguage.com/ddd/patterns/DDD_Reference_2011-01-

Strategic DDDesign patterns



https://domainlanguage.com/ddd/patterns/DDD_Reference_2011-01-

Exempel e-handel

- REST
- Layers
  - infrastructure (persistence + resources)
  - (application)
  - domain
- Static factory method
- Adapter (Resource)
- Repository (Katalog)
- Singleton