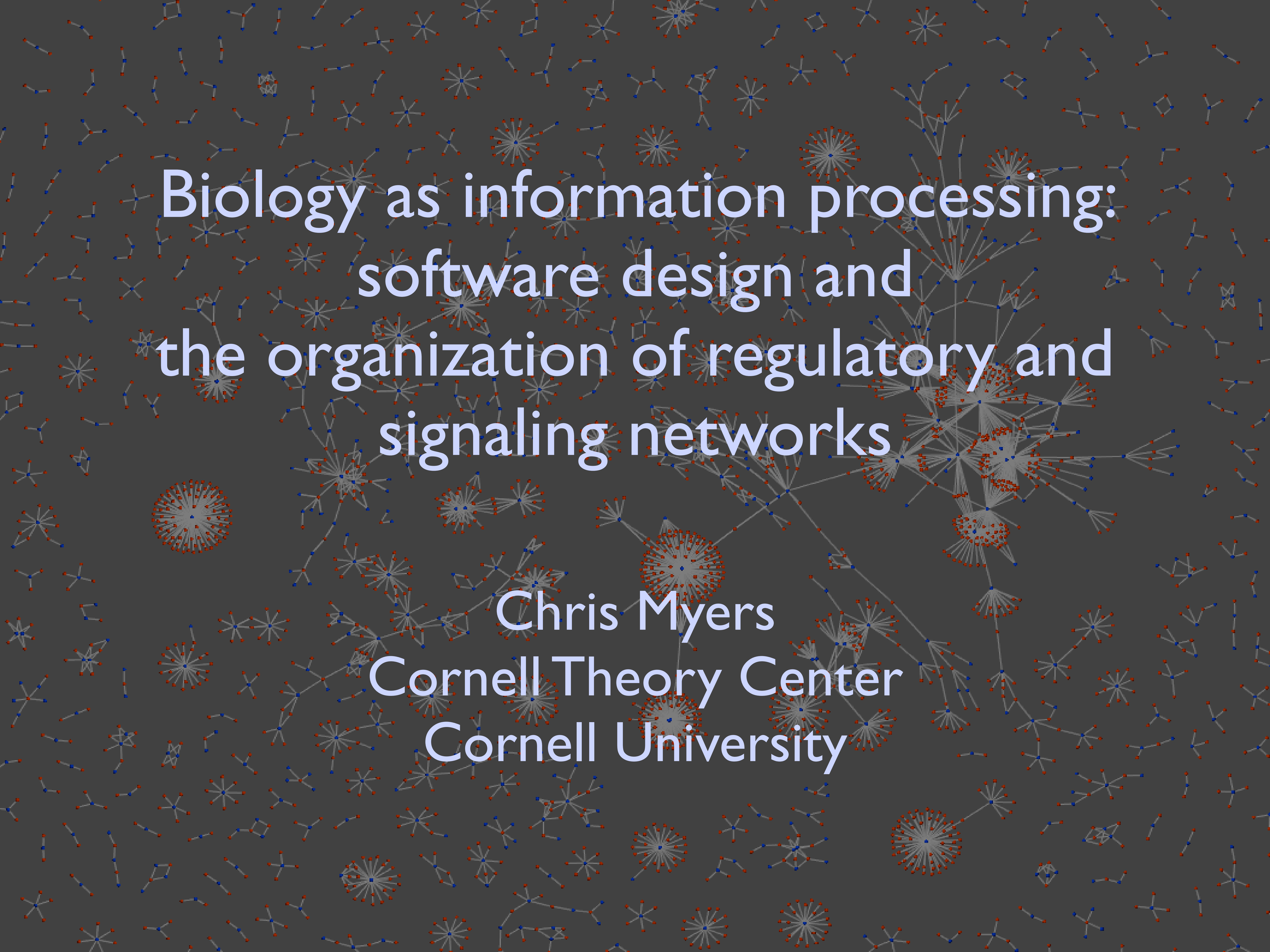
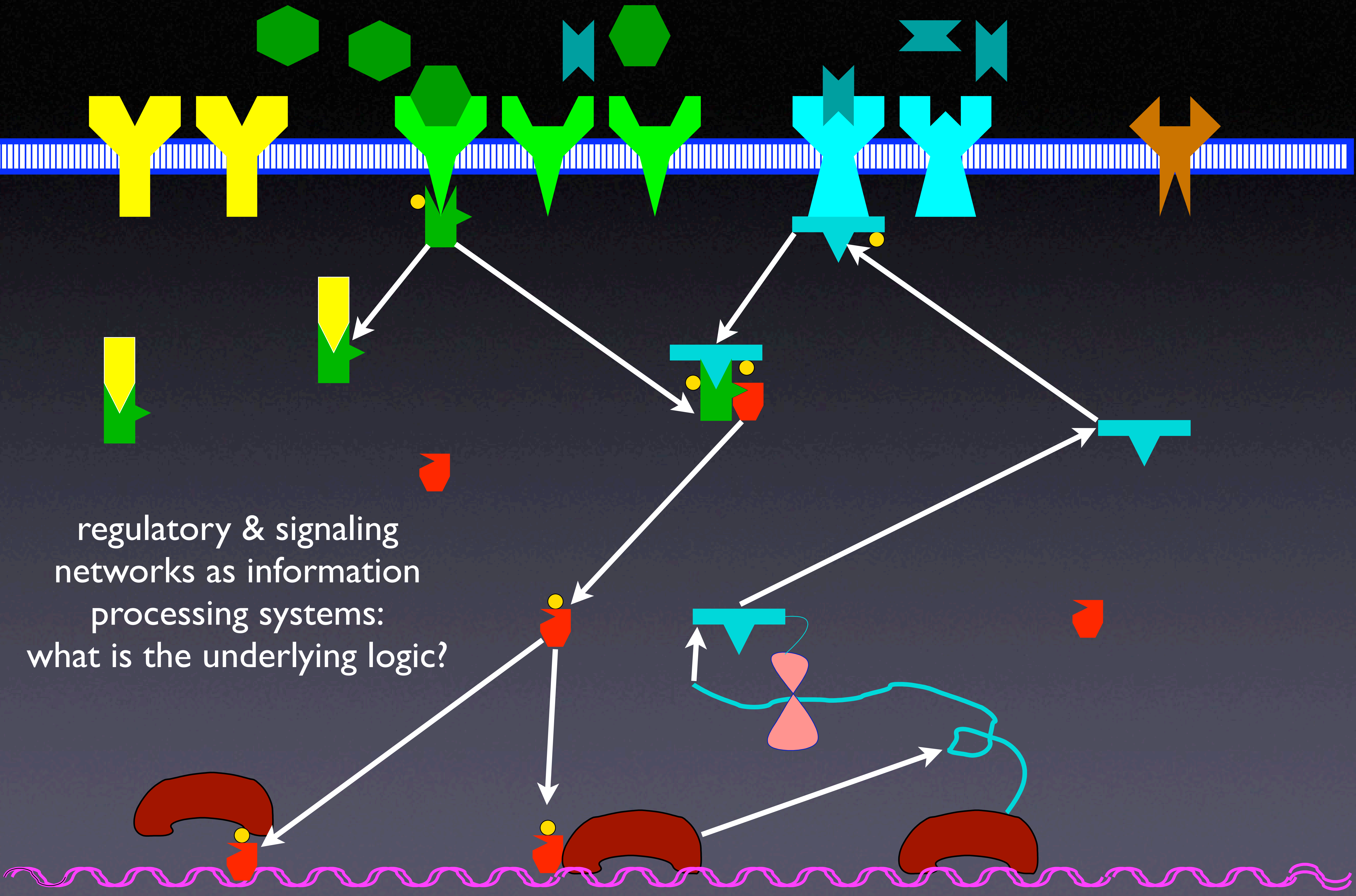


Biology as information processing: software design and the organization of regulatory and signaling networks

Chris Myers
Cornell Theory Center
Cornell University

Cellular regulation and signaling



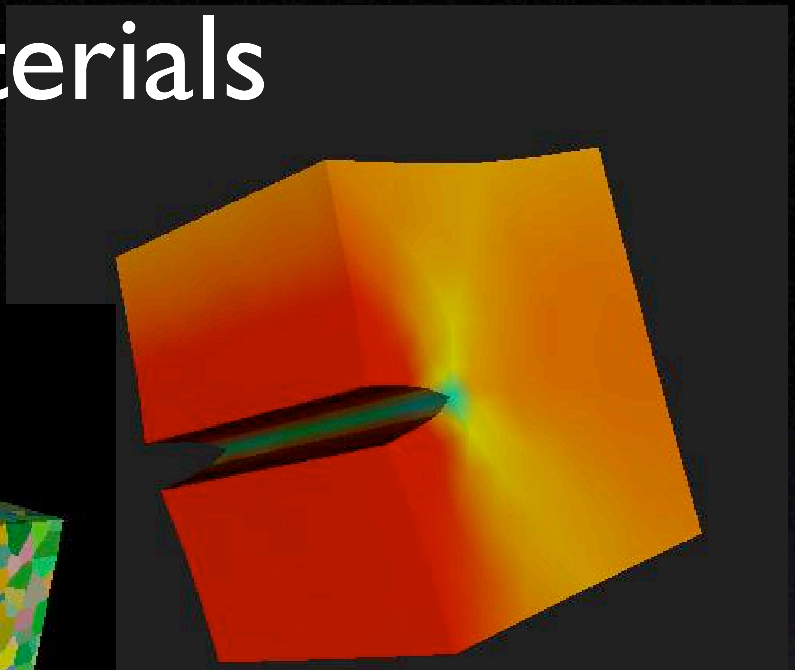
regulatory & signaling
networks as information
processing systems:
what is the underlying logic?

Software design as a window into the logic of regulatory and signaling networks:

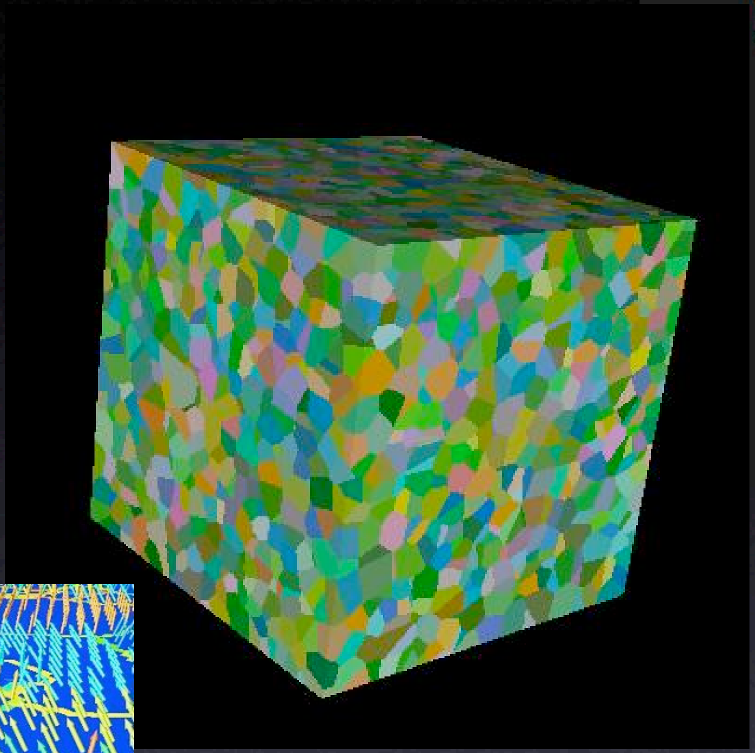
How can systems be organized to provide sufficient specificity for regulation while allowing sufficient flexibility for evolution?

- Acknowledges inherent tradeoff between specificity and control
  - overly specific solutions are brittle and unevolvable
  - generic solutions require composition and control
- Software design highlights:
  - the role of interaction specificity as an important control parameter
  - the role of network structures to organize specificity
- How do biological information processing networks organize the distribution of specificity to navigate this landscape of constraints?

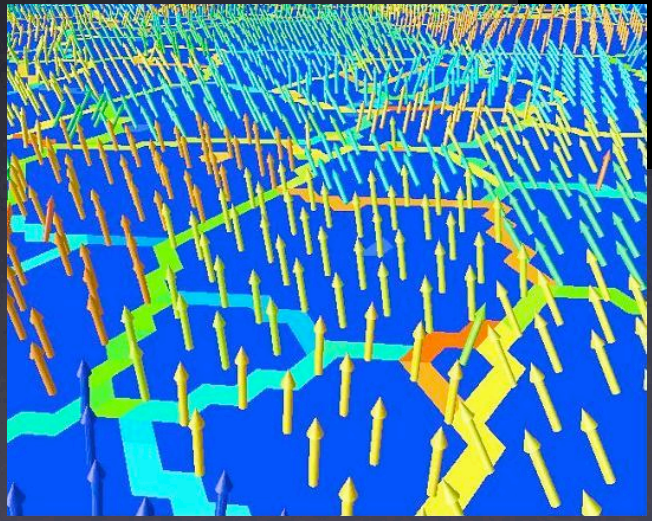
Multiscale modeling of materials



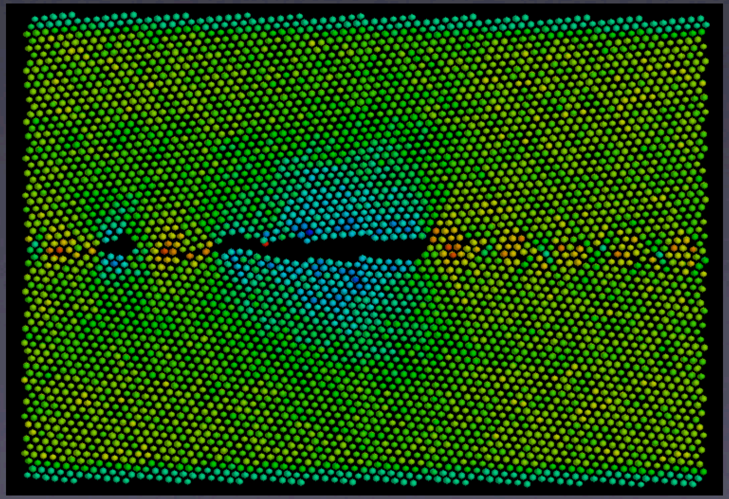
macro-material



polycrystals



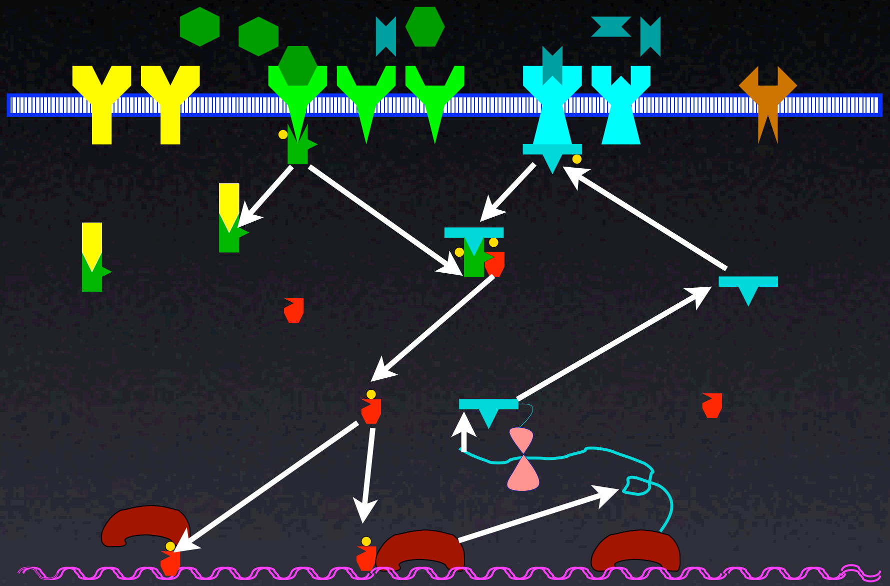
grain boundaries



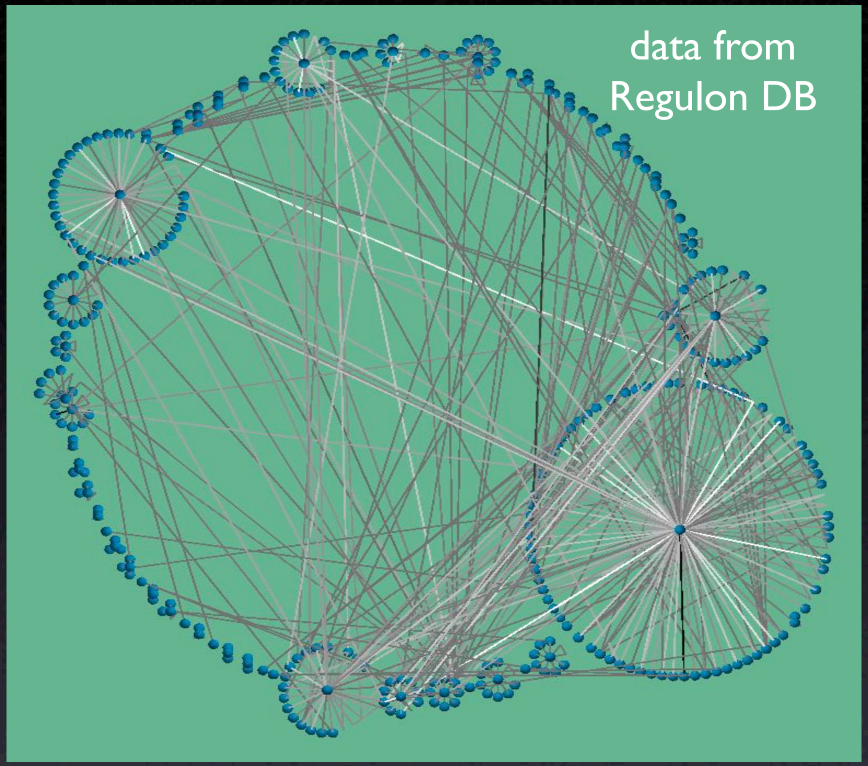
atoms

geometric structure as a *lingua franca* for multiscale coupling in materials

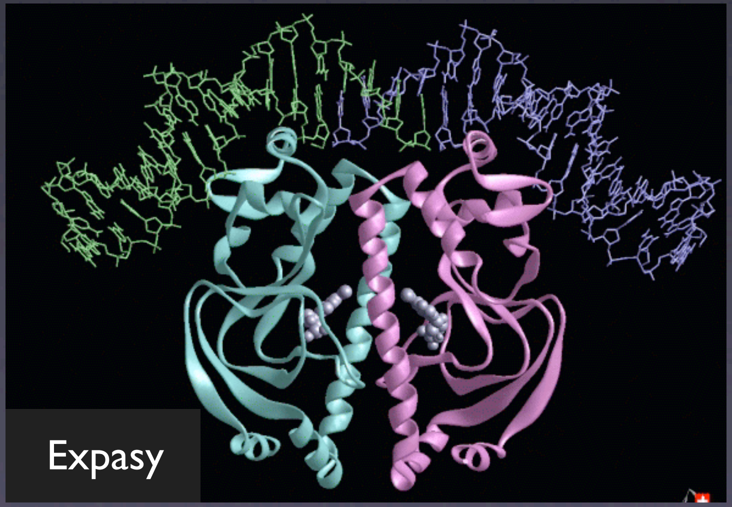
Multiscale modeling of biological networks?



pathways & subsystems



global networks



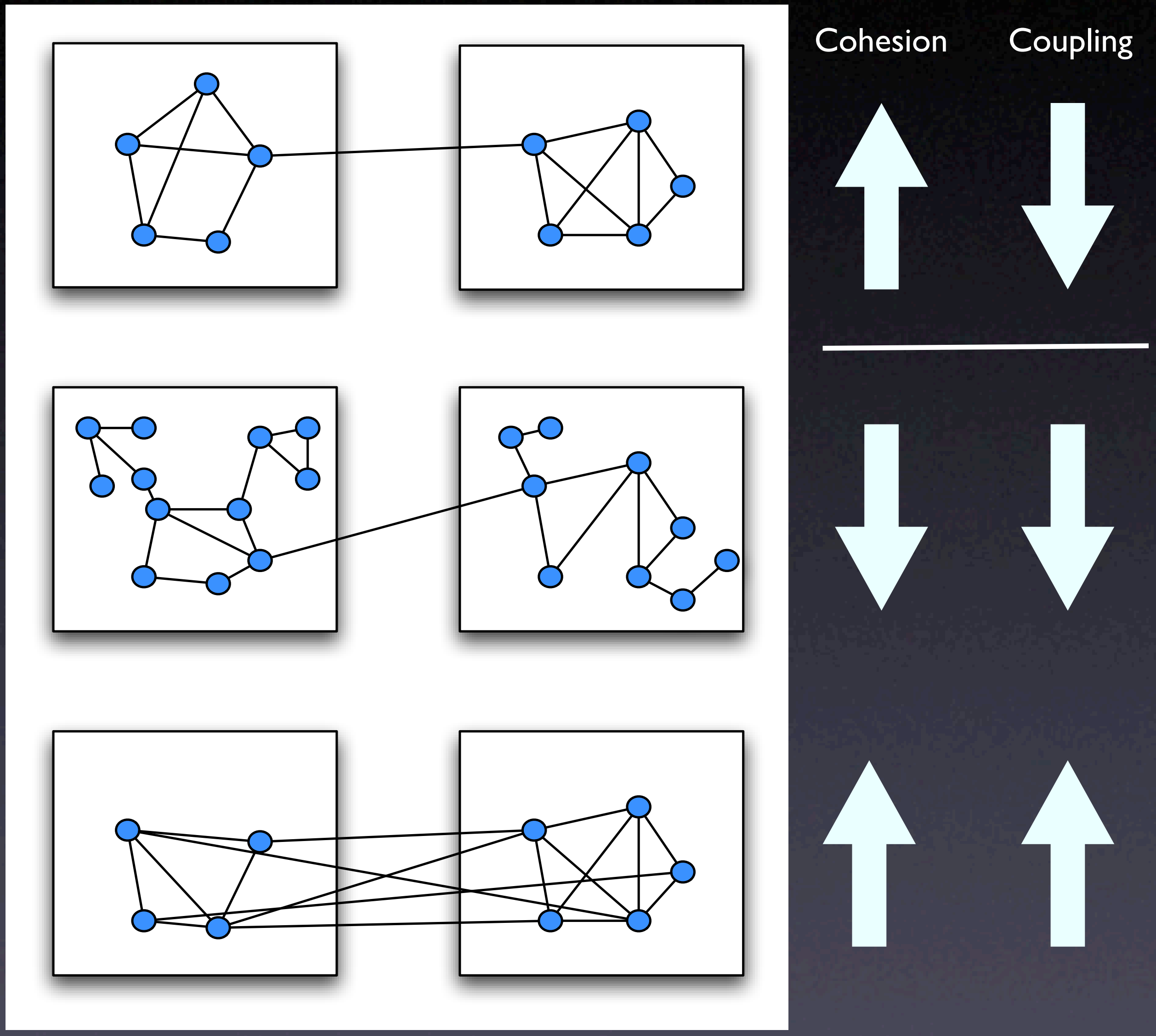
molecular binding

multiscale descriptions in the functional domain...
what is the *lingua franca*?

- modular decoupling of subsystems
- control of specificity ⇒ evolvability

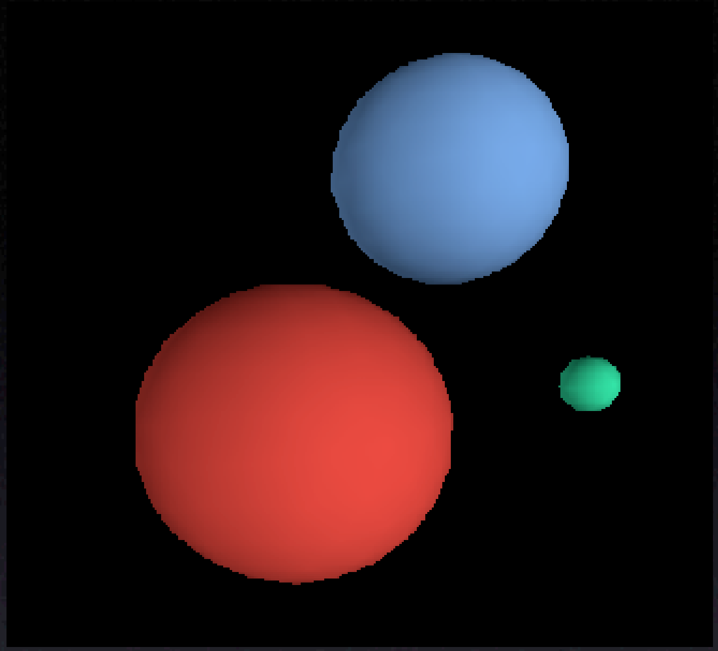
Modularity in software

- High cohesion / low coupling
  - focus of purpose within a module
  - weak linkage & low dependency across modules
- Strength of coupling arises from specificity of interaction
- Many scales of granularity: hierarchy
- Variability & evolvability: encapsulation & reuse



Binding specificity in biology

- specificity: (inverse) measure of the fraction of configuration space bound
  - e.g., fraction of 4^L DNA sequences of length L

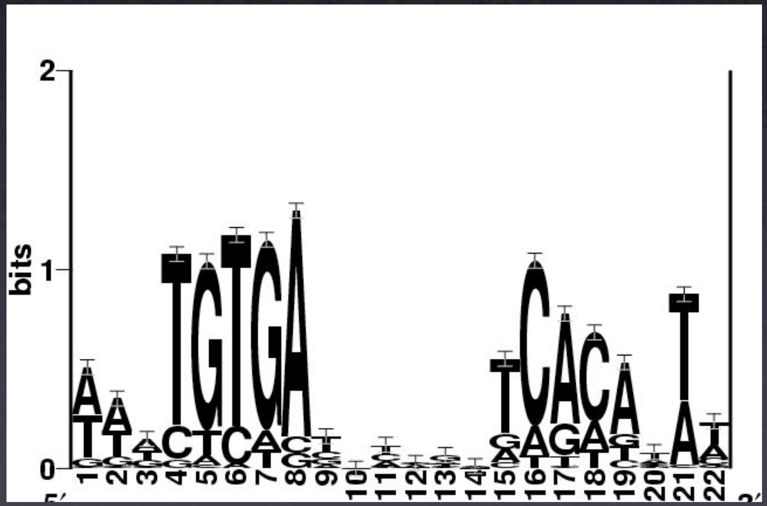


- information theoretic interpretation

  uncertainty in binding sequences =

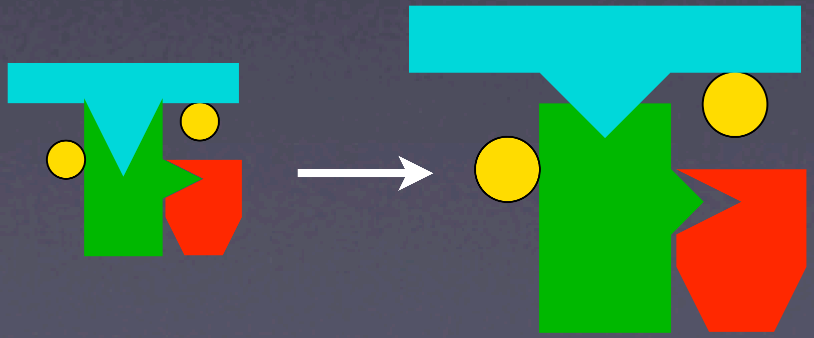
  $$-\sum_i p_i \log p_i \qquad \text{(i=A,C,G,T for DNA)}$$

  - sequence logos: information content $\longrightarrow$



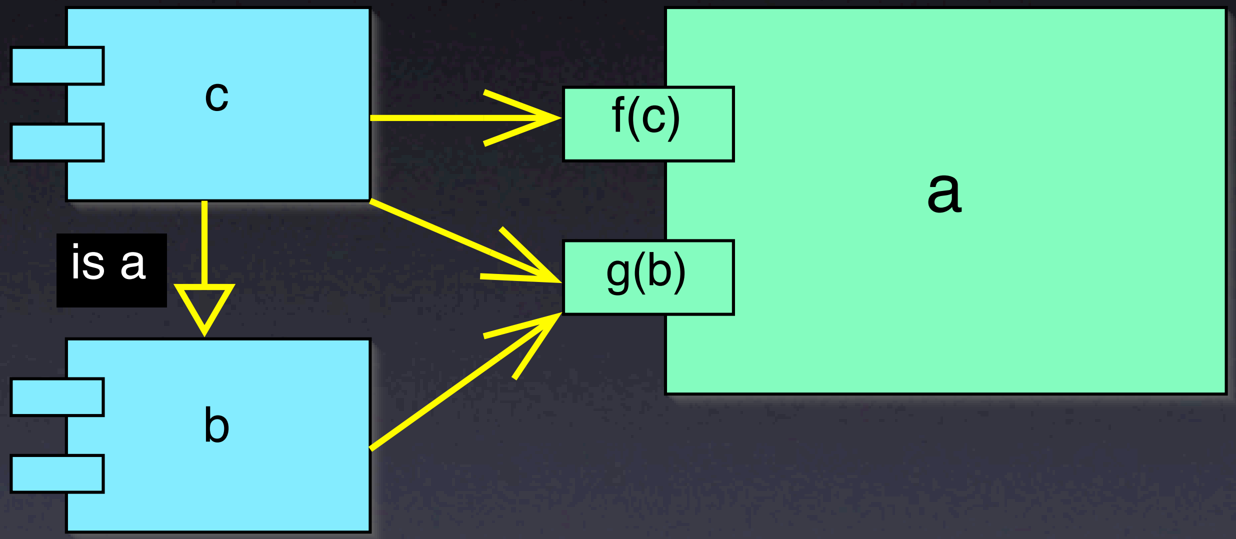
CRP in *E. coli*, from DPInteract

- interactions in regulation and signaling: more promiscuous and fuzzy than simple "lock & key"
  - combinatorial control (distributed specificity)
  - greater evolvability & robustness to mutation

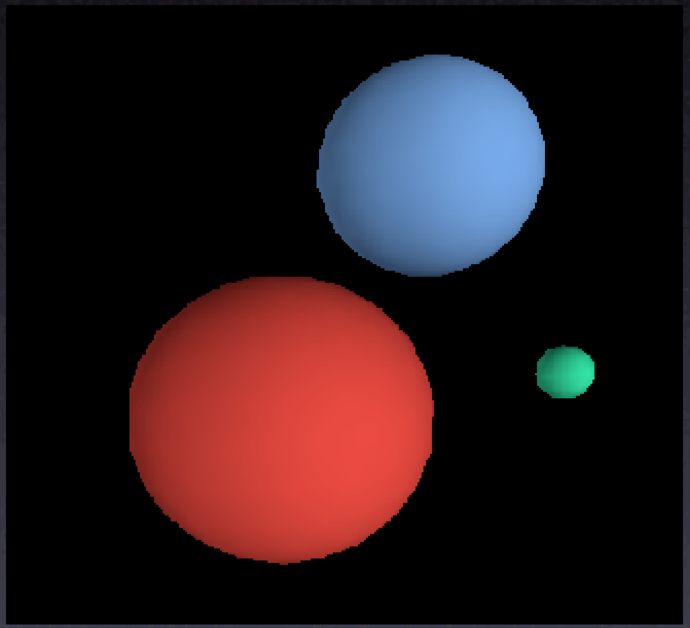


Binding specificity & polymorphic design in software

- The molecular nature of software
  - object's public interface ⇔ external binding sites
  - type information ⇔ binding specificity

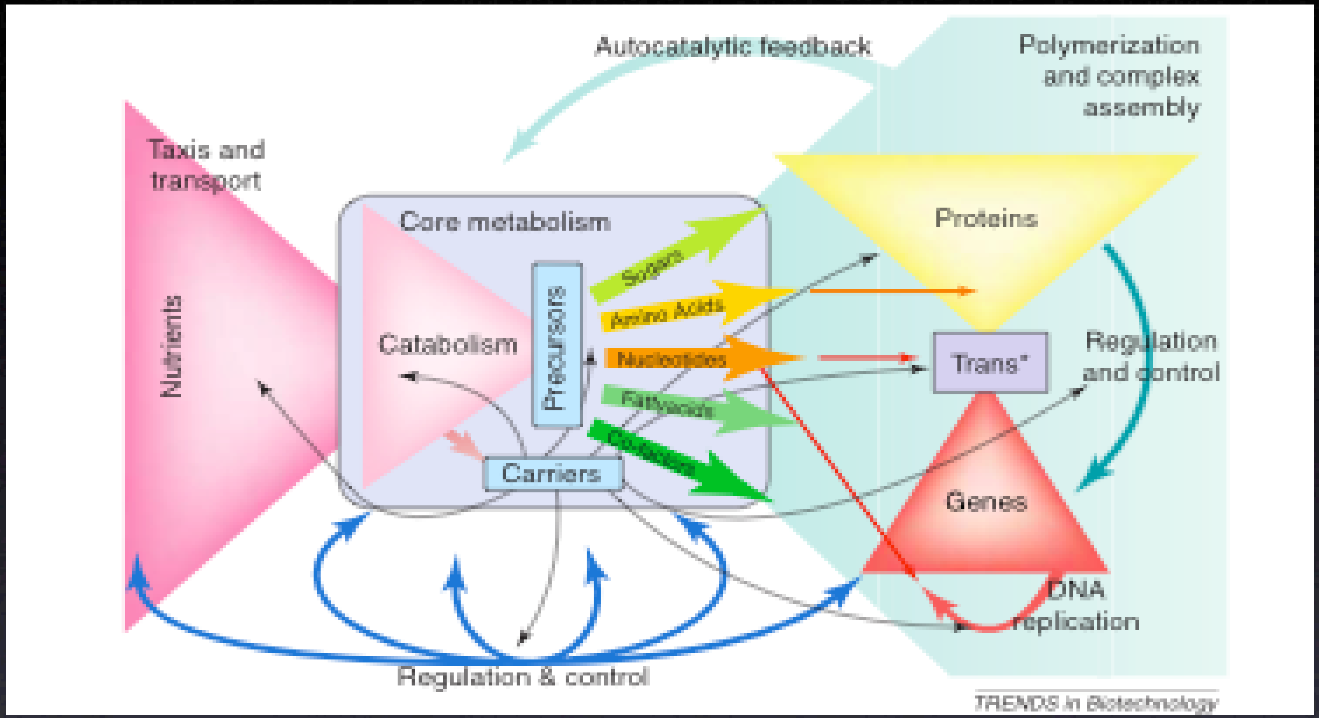


more generic interfaces can bind broader classes of objects

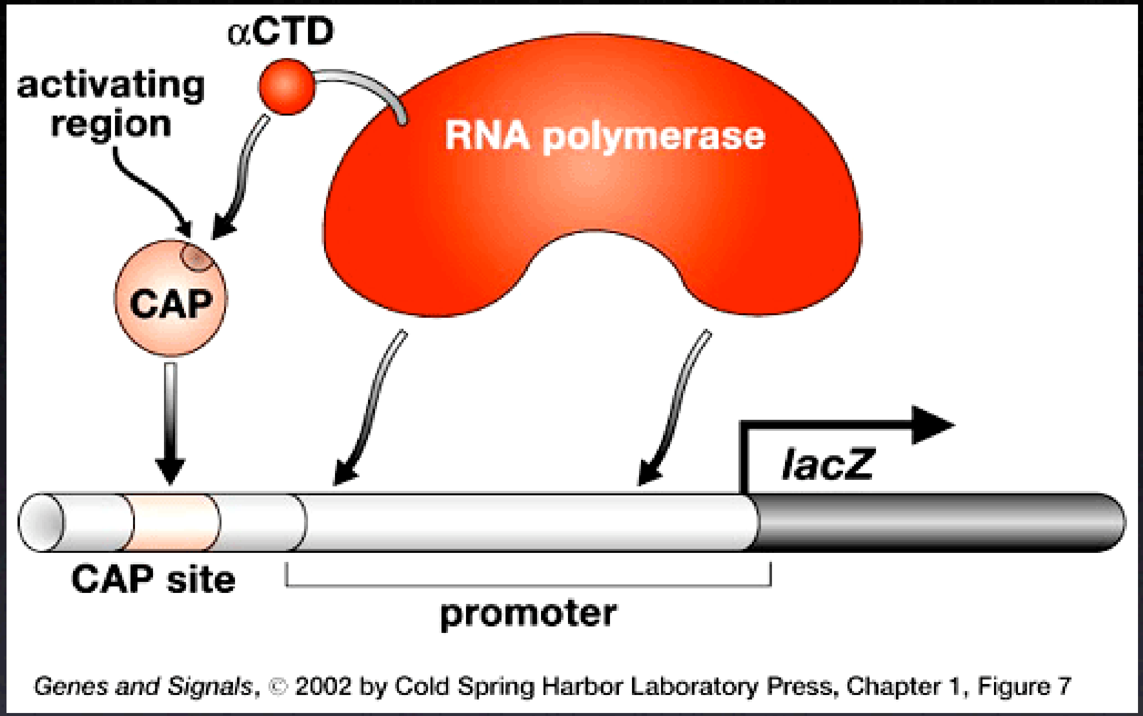


- Polymorphism: the ability to substitute objects with matching interfaces for one another at runtime
- Evolvable polymorphic design encourages minimally specific interfaces
  - make interactions as generic as possible consistent with function
- Binding specificity is a key control parameter in network organization

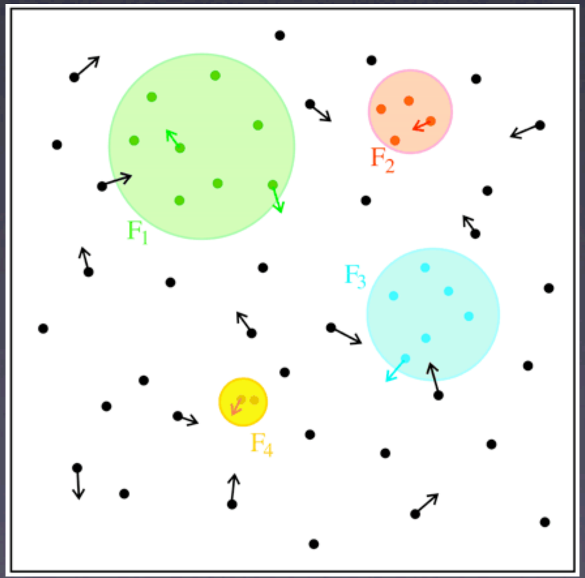
Specificity & evolvability in biomolecular networks



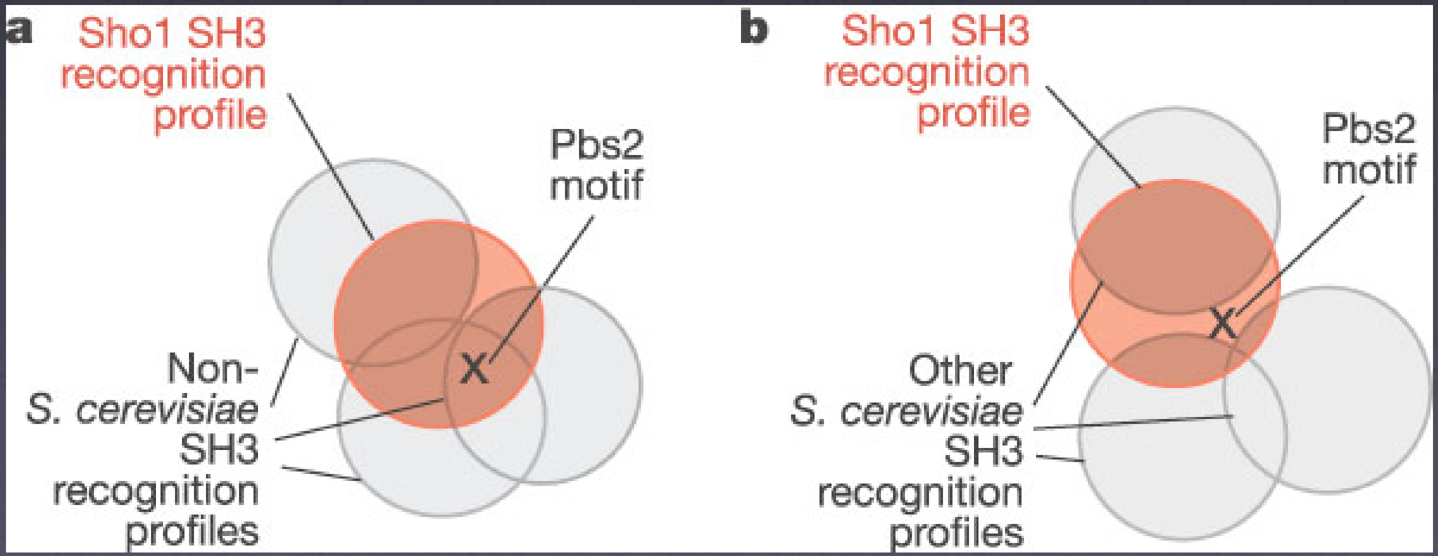
Regulation on the network periphery
(Csete & Doyle, 2004)


αCTD
activating region
RNA polymerase
CAP
lacZ
CAP site
promoter
Genes and Signals, © 2002 by Cold Spring Harbor Laboratory Press, Chapter 1, Figure 7

Regulated recruitment
(Ptashne & Gann, 2002)



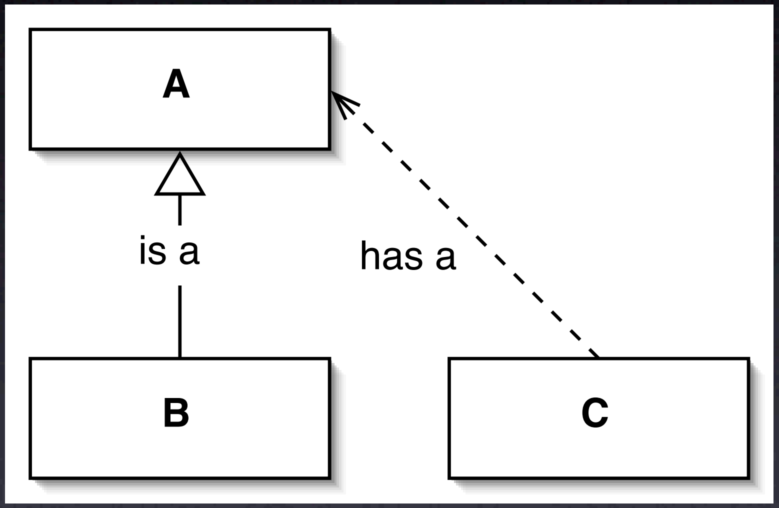
Fuzziness from specificity vs evolvability
(Sengupta et al., 2002)


a
Sho1 SH3 recognition profile
Pbs2 motif
X
Non-*S. cerevisiae* SH3 recognition profiles
b
Sho1 SH3 recognition profile
Pbs2 motif
X
Other *S. cerevisiae* SH3 recognition profiles

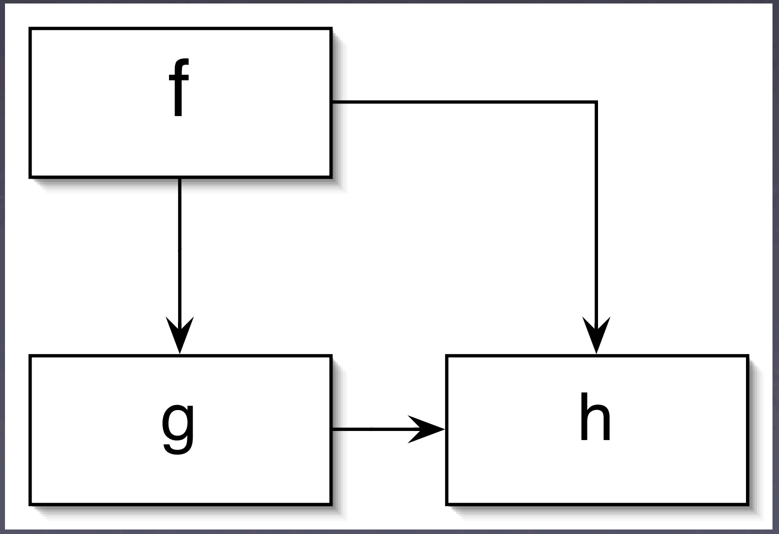
Specificity in sequence niches
(Zarrinpar et al., 2003)

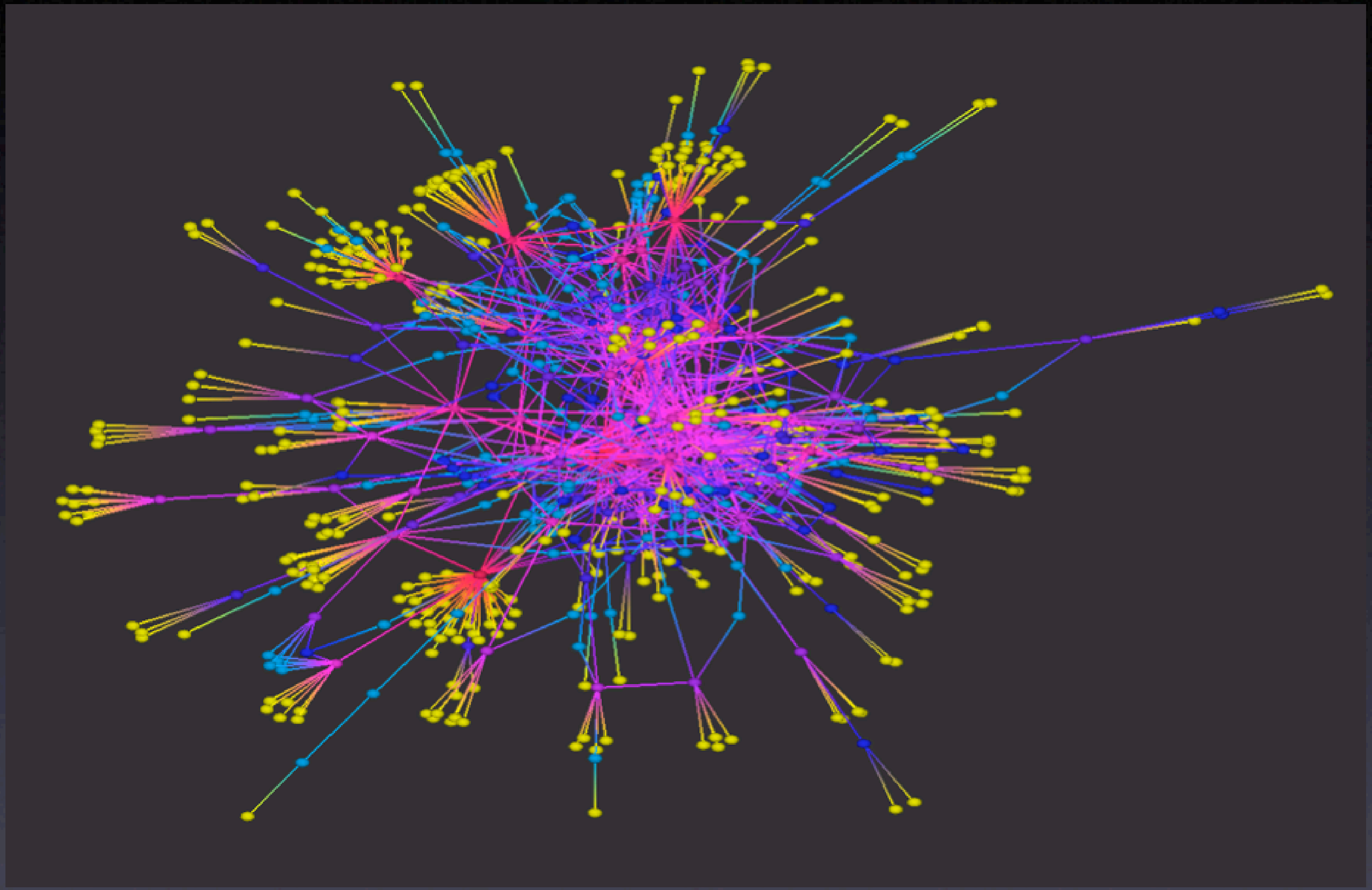
Networks in software systems

Class relationships in object-oriented programs



Static function call graphs in procedural programs





Class collaboration graph in VTK (Visualization Toolkit)

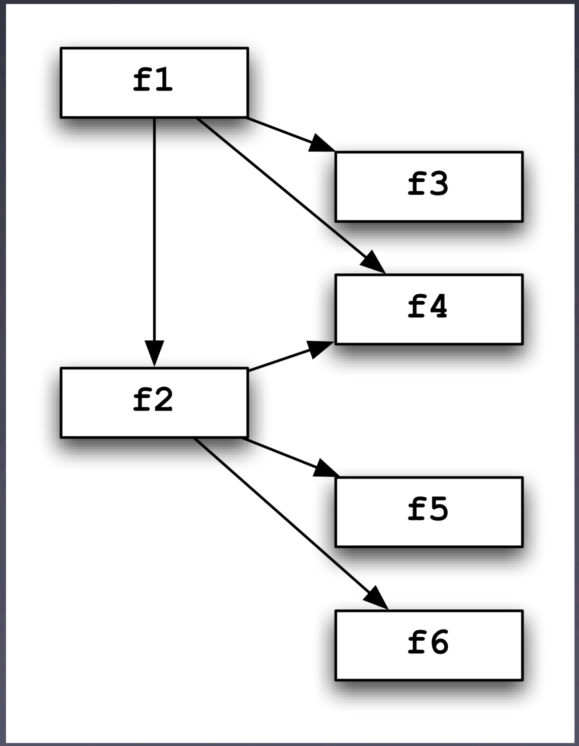
complexity in software structure to support function & evolvability

CRM, Phys Rev E 68, 046116 (2003)

Software collaboration networks

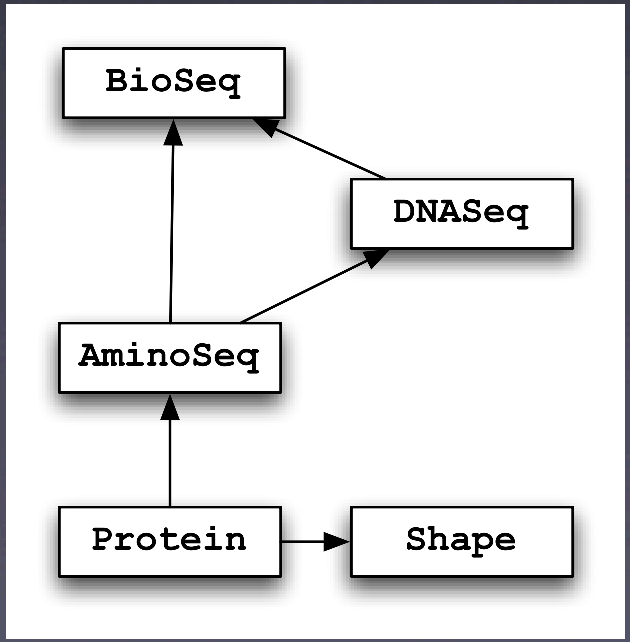
Subroutine call graphs

```
void f1()
{
   f2();
   f3();
   f4();
}

void f2()
{
   f4();
   f5();
   f6();
}
```



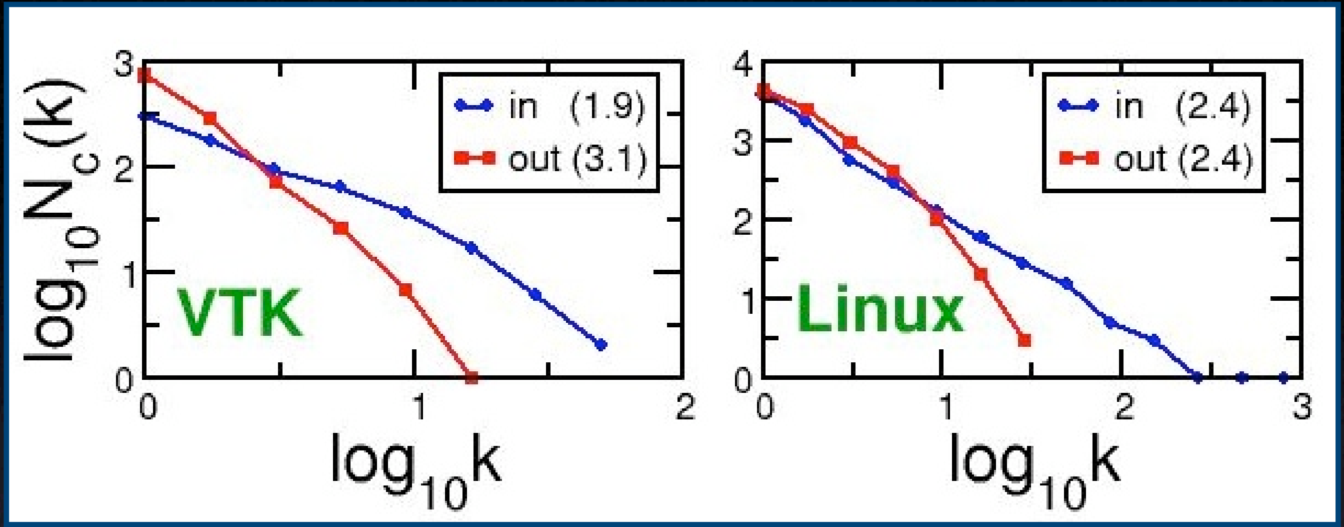
Class collaboration graphs

```
class BioSeq {
};

class DNASeq: public BioSeq {
};

class AminoSeq: public BioSeq {
   DNASeq *dna;
};

class Protein {
   AminoSeq *amino;
   Shape *shape;
};
```



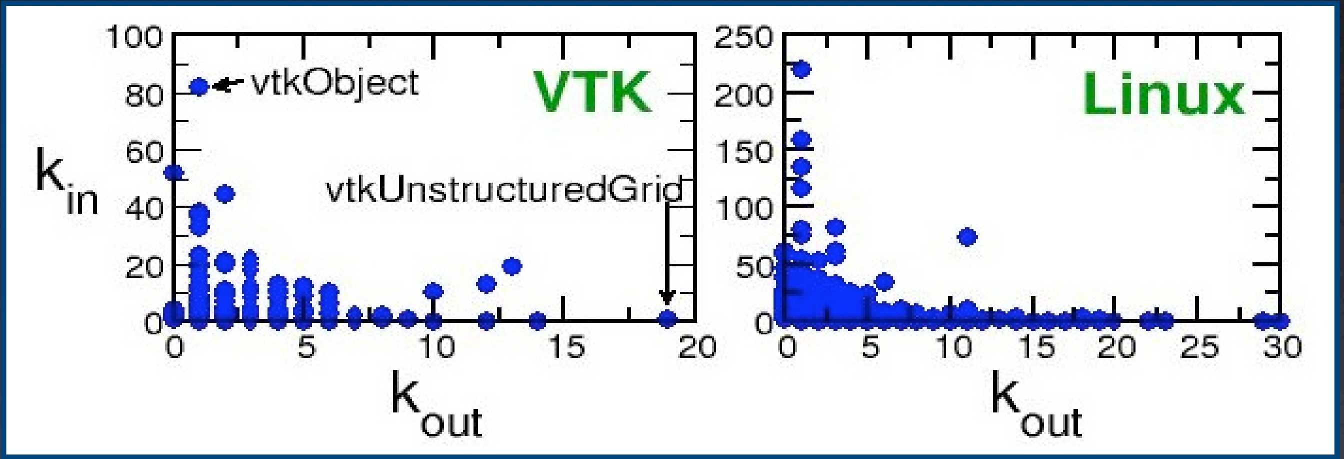
Software systems as complex networks

heavy-tailed ("scale-free") degree distributions



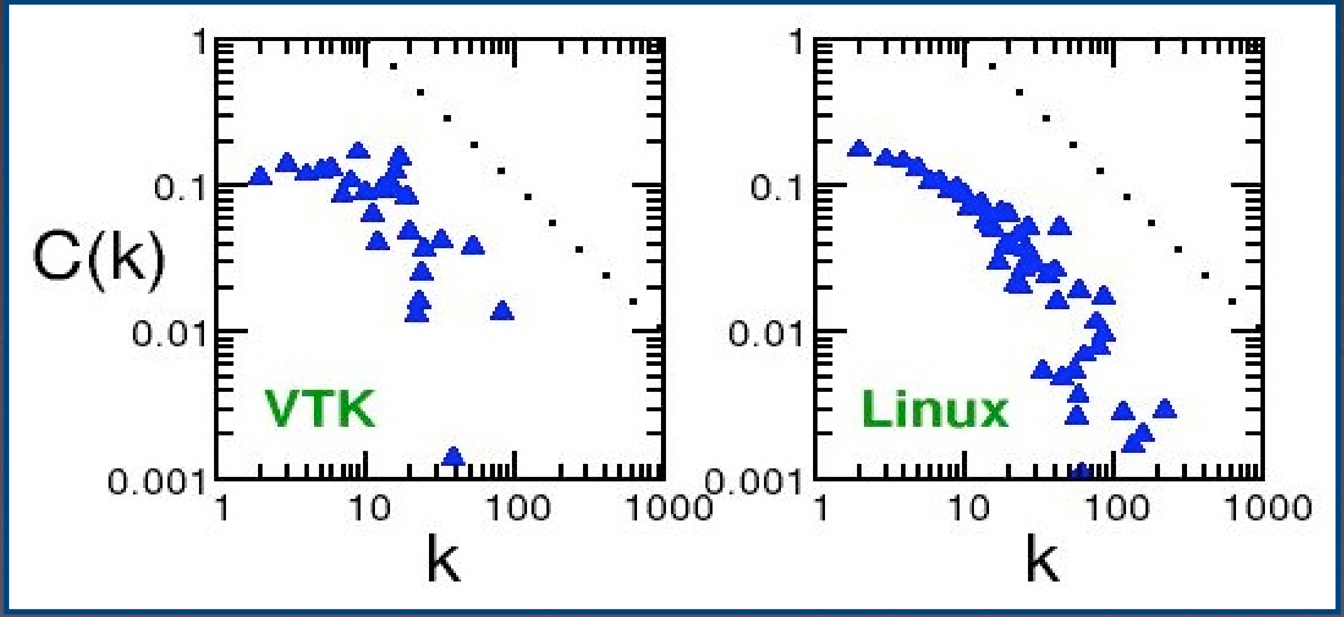
→ broad spectrum of reuse (in) and complexity (out)

in-out degree anti-correlation

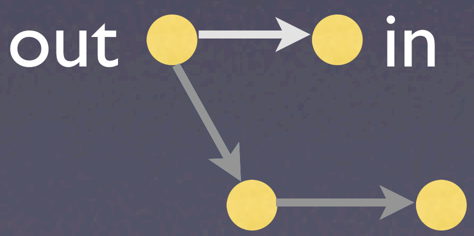


→ producers (in) & consumers (out) well separated

degree-dependent clustering



→ suggestive of hierarchical organization*: $C \sim k^{-1}$

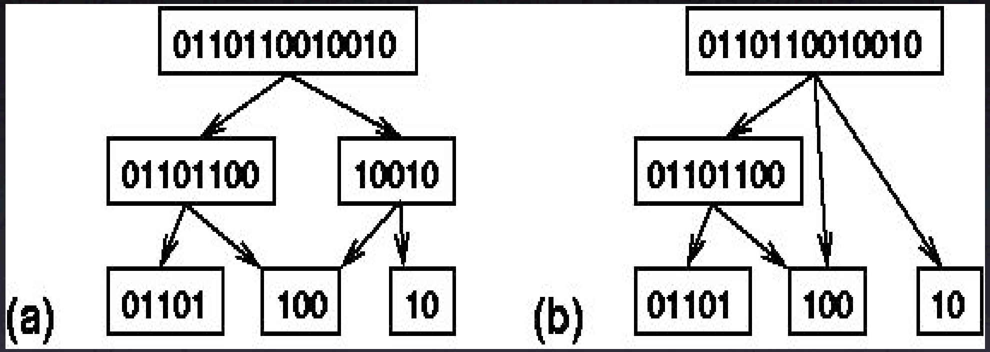


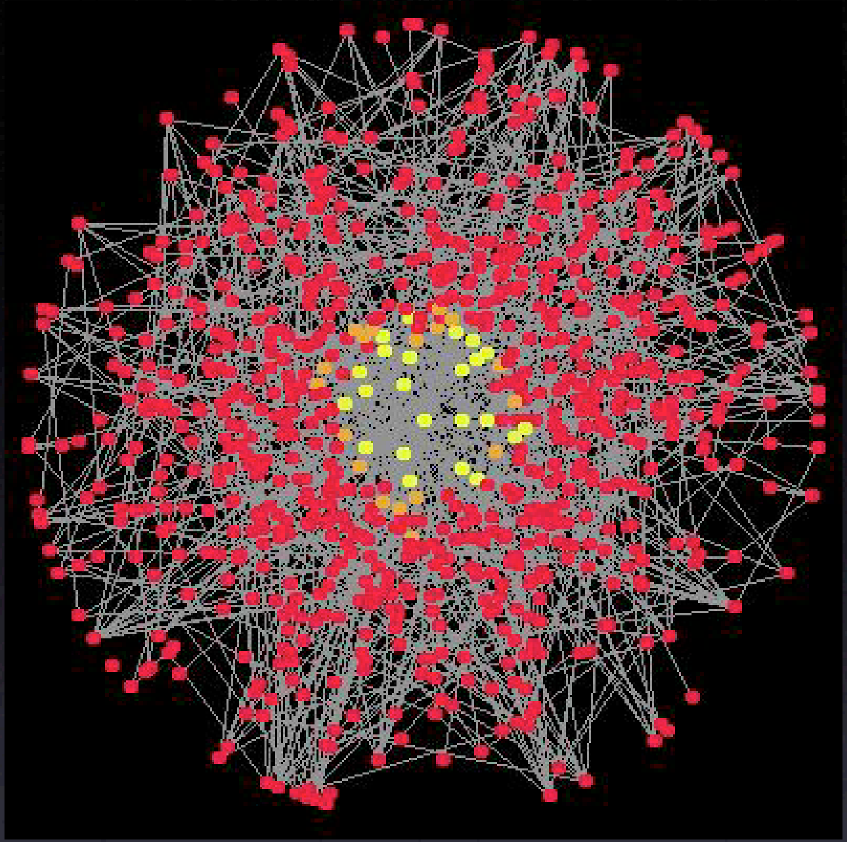
(VTK, DigitalMaterial, Abiword, Linux, MySQL, XMMS)

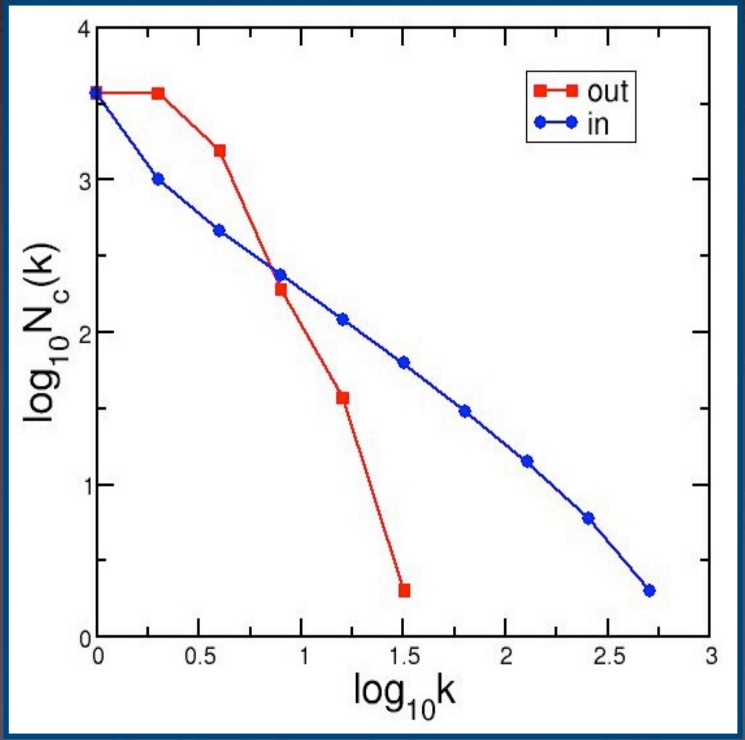
*Ravasz & Barabasi (2003)

Software refactoring in a model system

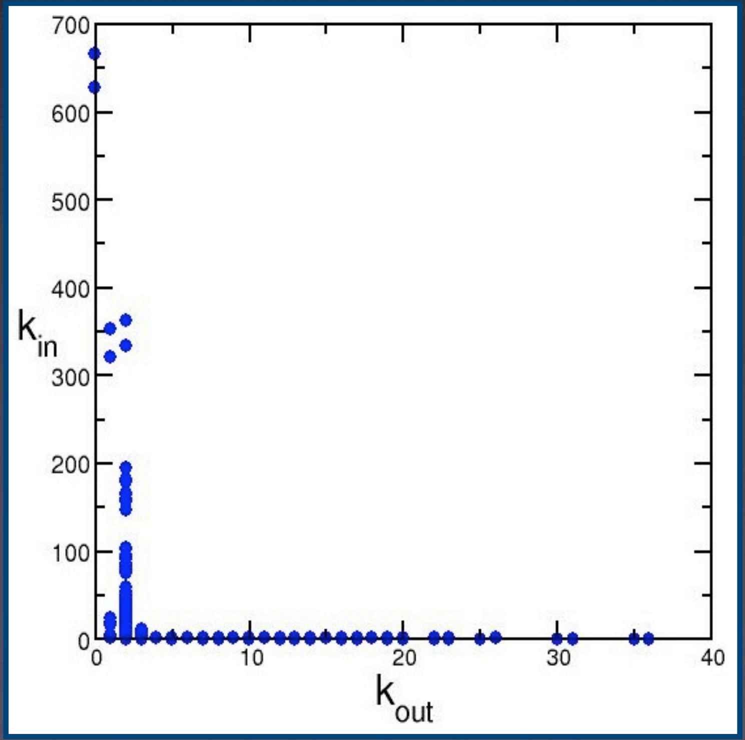
- Refactoring aims to reorganize software systems to make them more evolvable
- Simple model of refactored software system shows similar network topology



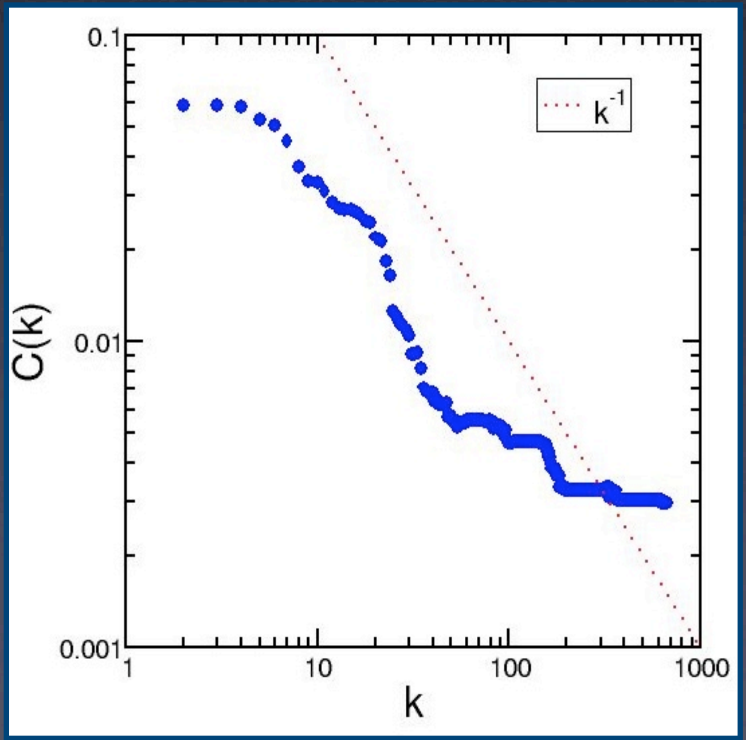




degree distributions

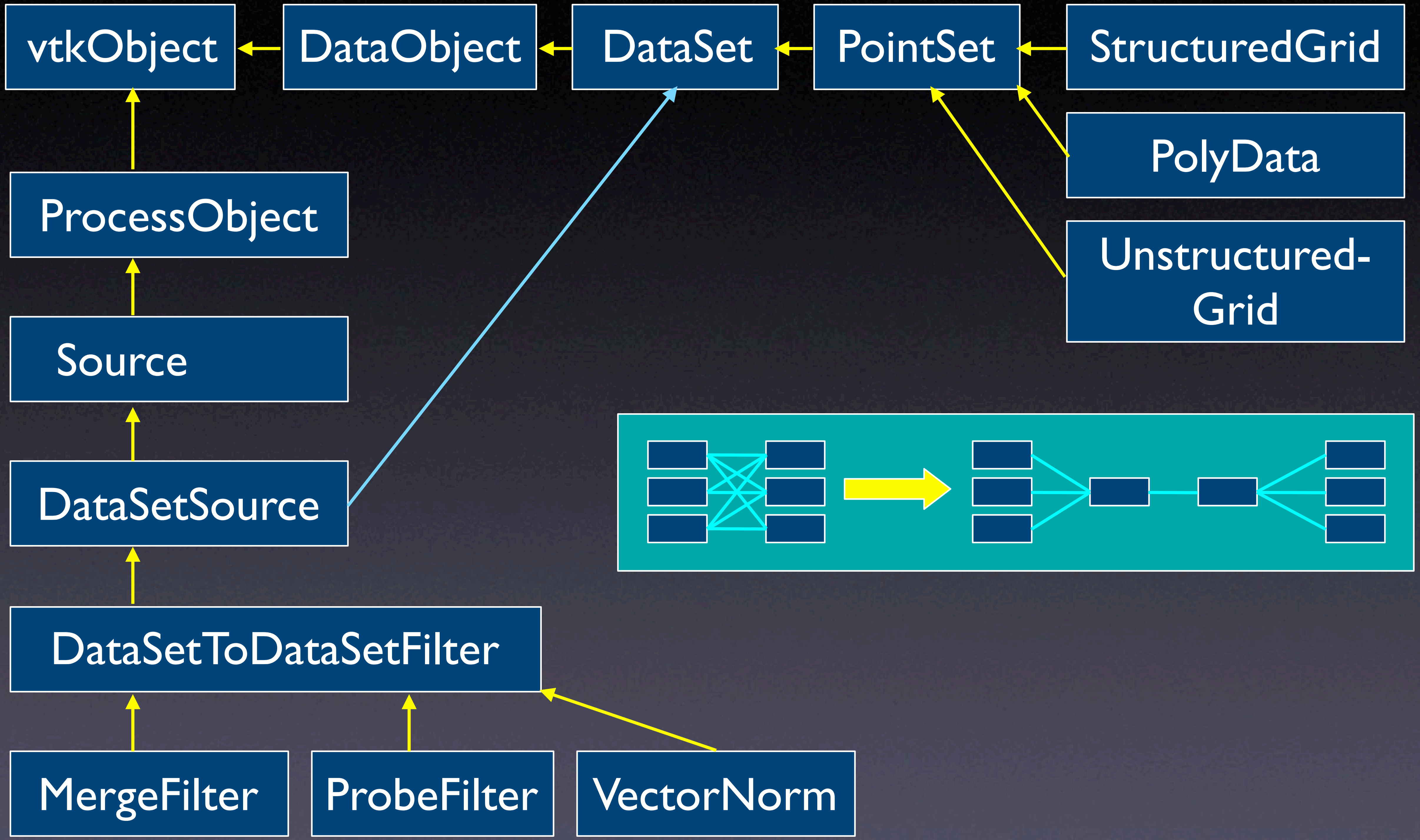


degree correlation



degree-dependent clustering

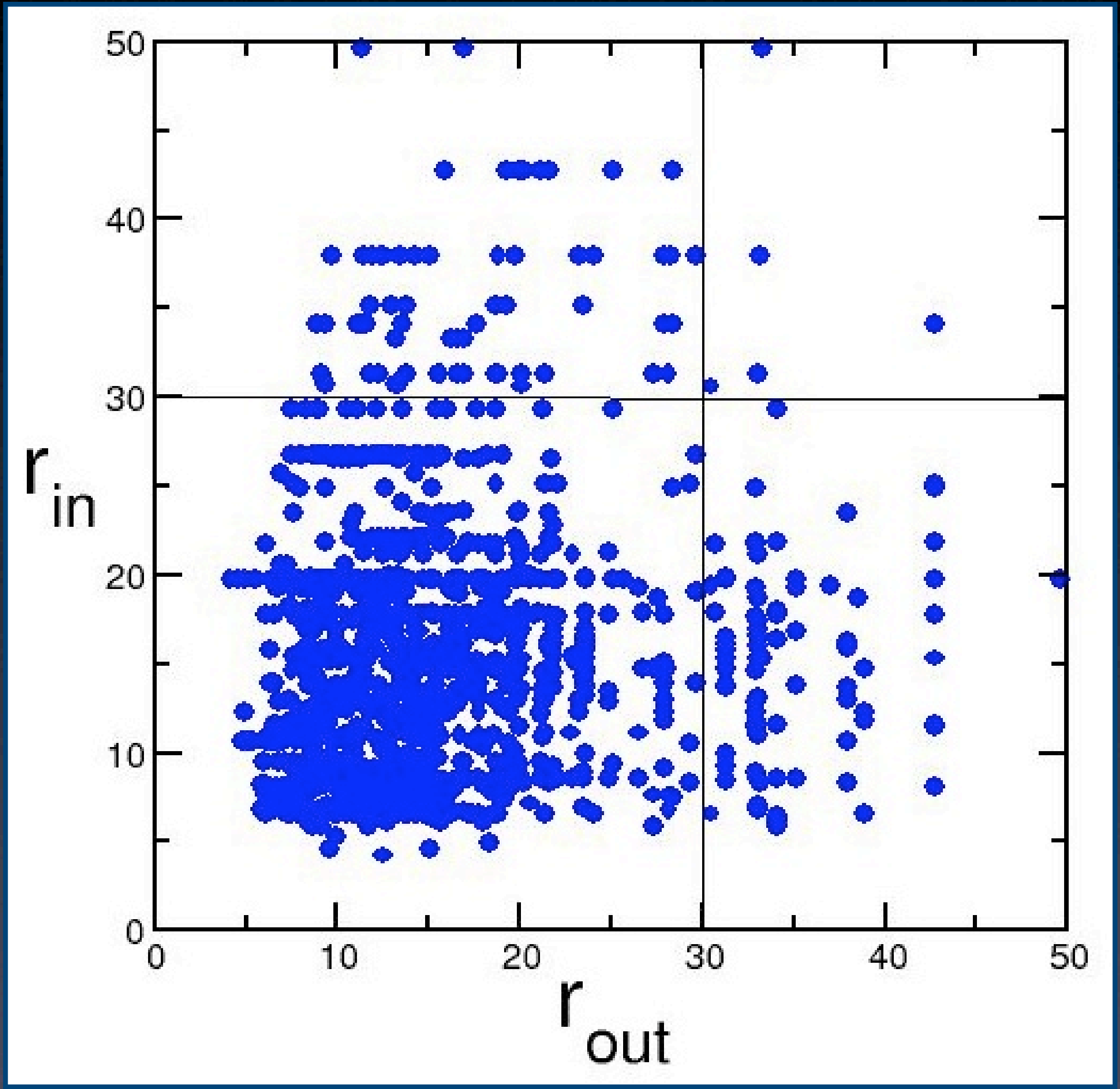
Polymorphic design: the example of VTK



www.vtk.org

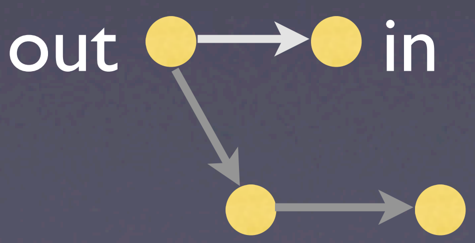
Component evolution & network topology

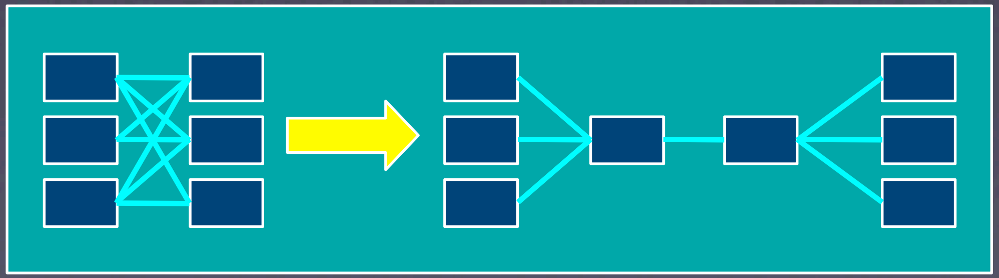
rate of evolution of classes in VTK from CVS repository



- most rapidly evolving classes are complex aggregates on the network periphery
- might expect rapidly evolving classes to need to co-evolve
- rapidly evolving classes do not directly interact: modular separation of specialized subsystems

r_{in}, r_{out} = average rate of evolutionary change of classes at incoming, outgoing nodes on each edge





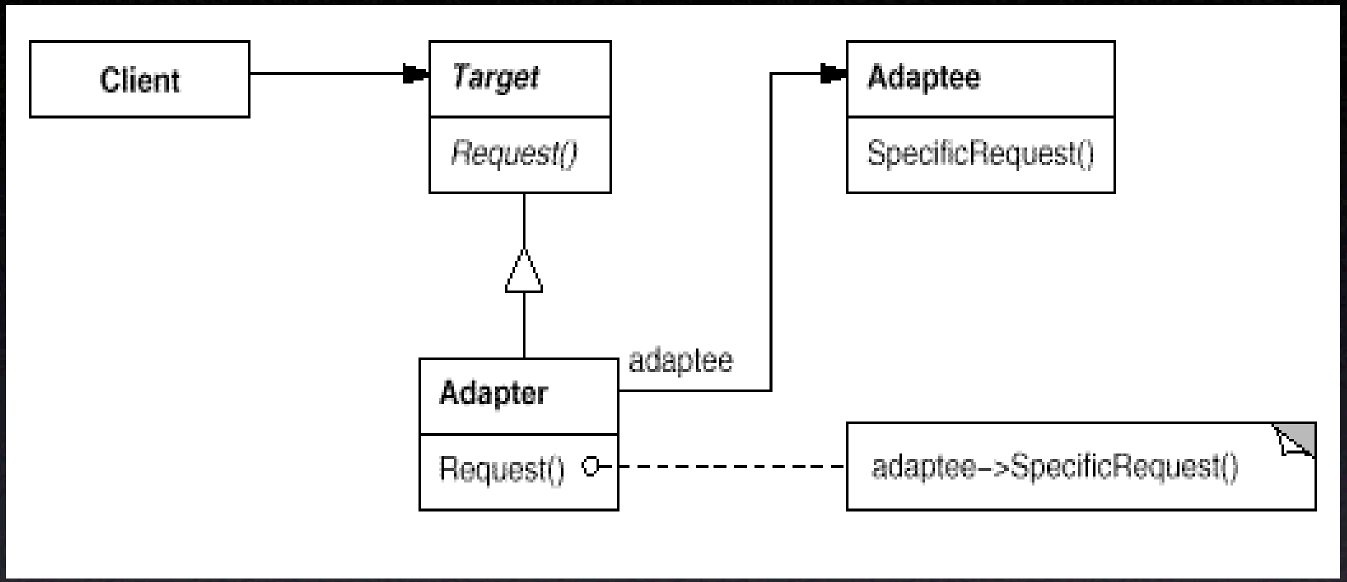
Design patterns in software

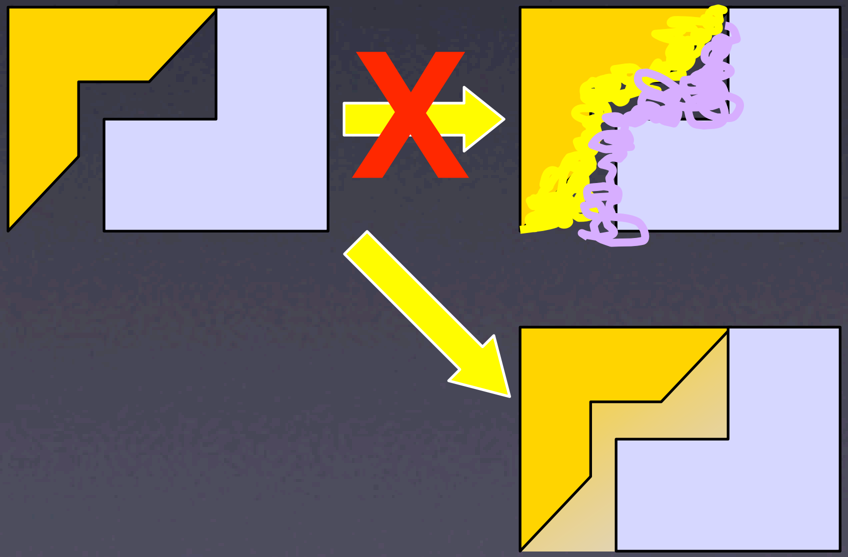
- Object collaborations that use polymorphism to encapsulate variability & support reuse
  - "network motifs" at the mesoscale
- Different patterns for different aspects of variability
  - what objects are created
  - how objects are created
  - how algorithms are implemented
  - which objects act on requests
- Rely on composition, delegation, indirection & shared function, rather than on increasingly specialized objects

see Gamma et al., Design Patterns: Elements of Reusable Object-Oriented Software

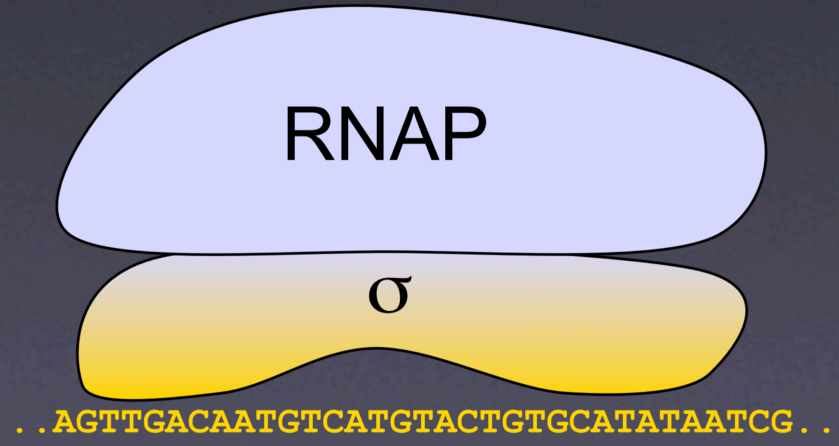
Design patterns: Adapter

- convert an interface to make two interfaces compatible



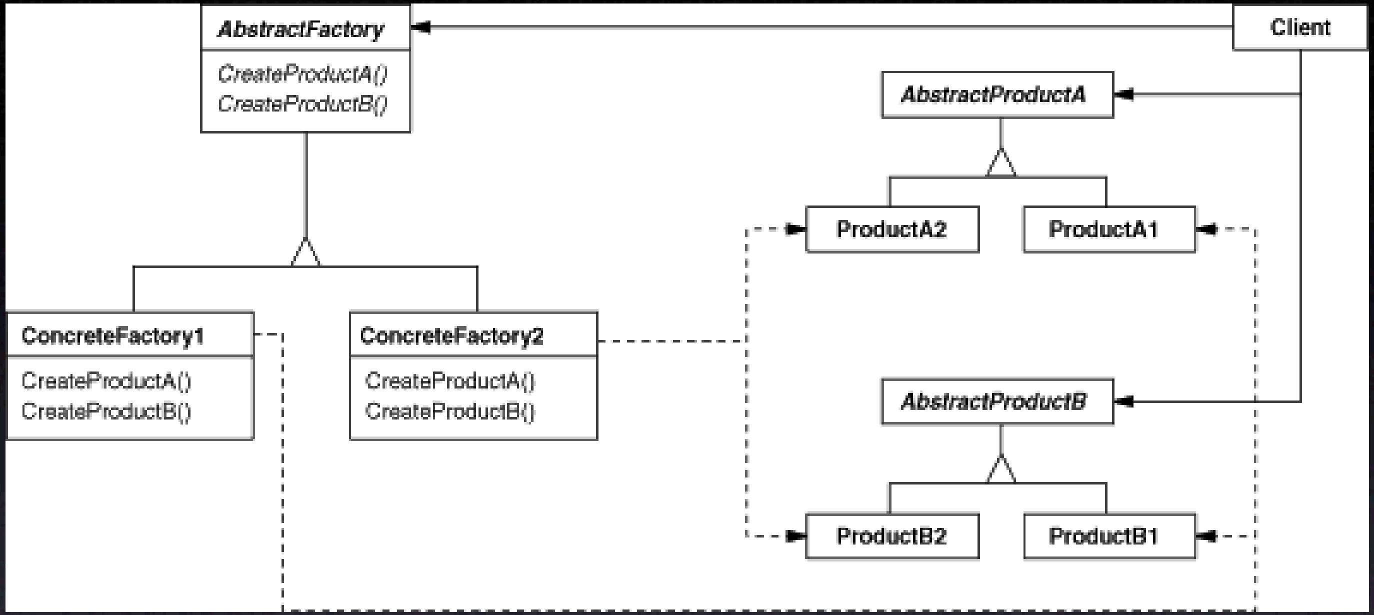


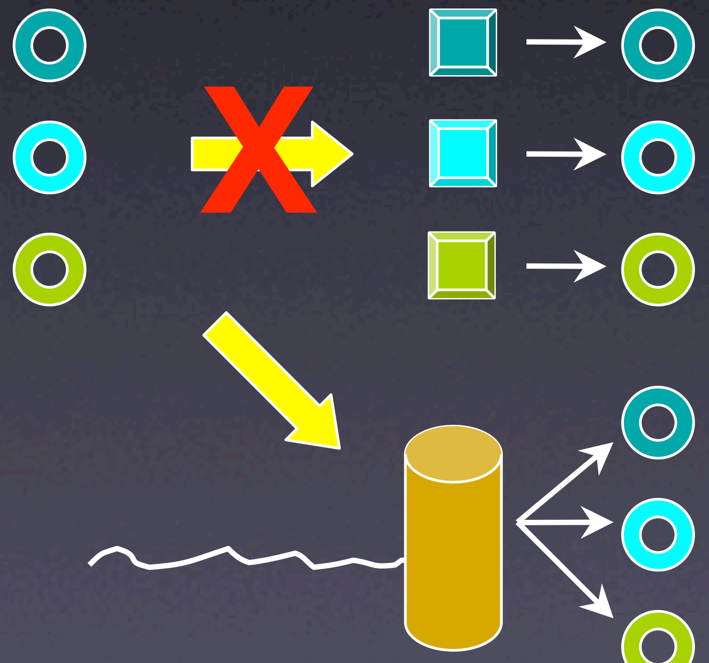
σ factor as a pluggable adapter



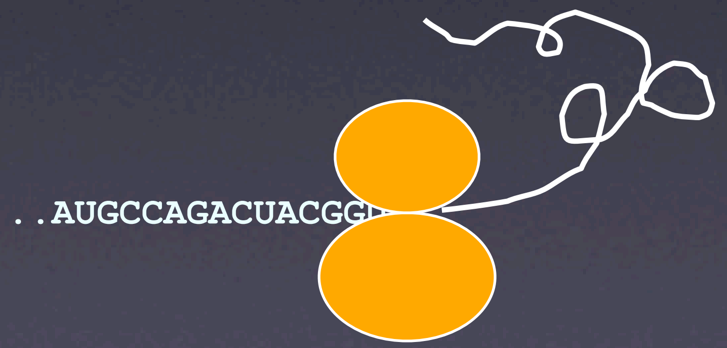
Design patterns: Abstract Factory

- create families of related objects without specifying their concrete classes





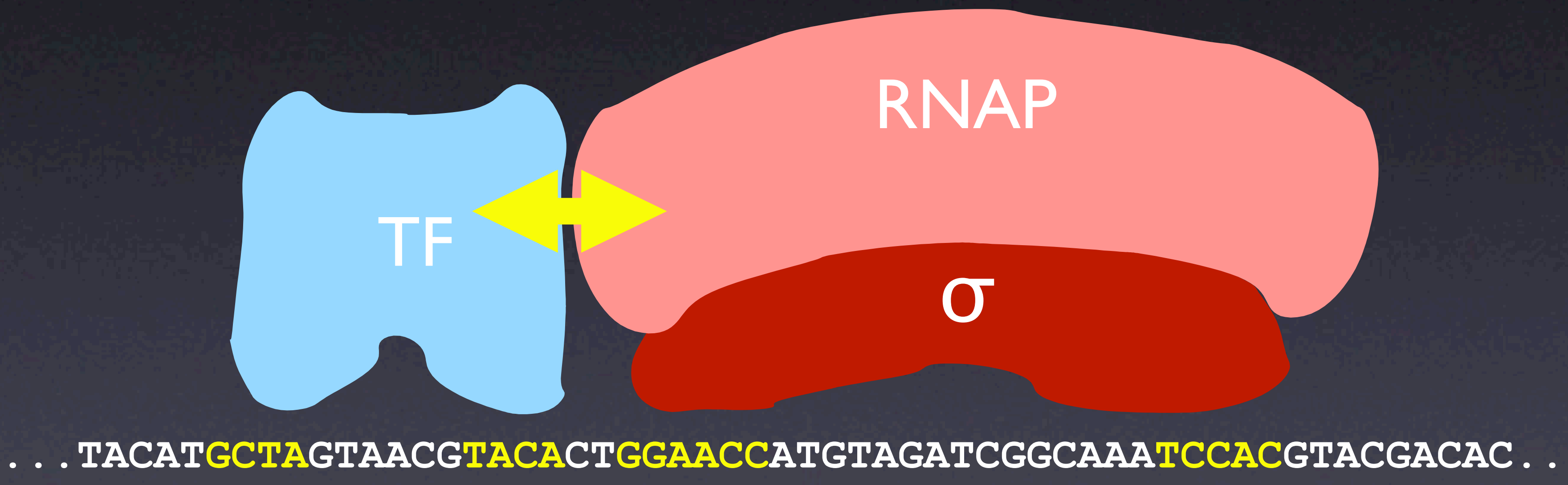
ribosome as an abstract factory



(and tRNAs as visitors)

Regulated recruitment in biomolecular networks

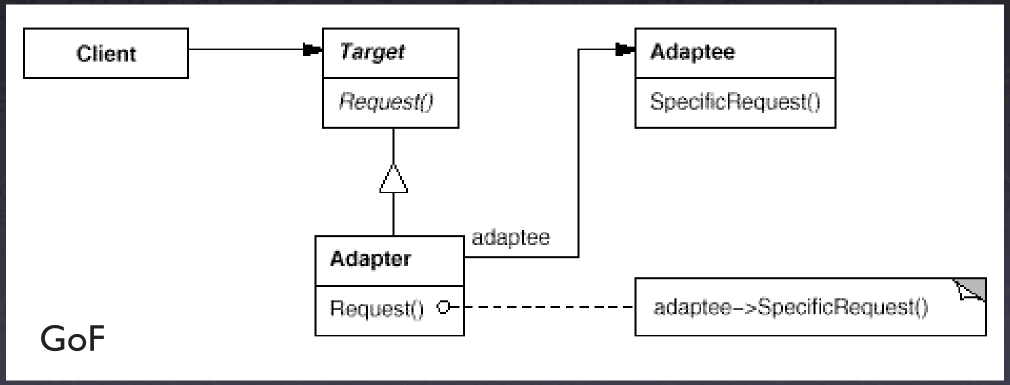
- regulation via weak, generic, "glue-like" interactions rather than highly specific "lock-and-key" mechanisms
  - specificity distributed throughout system and imposed through cooperative binding

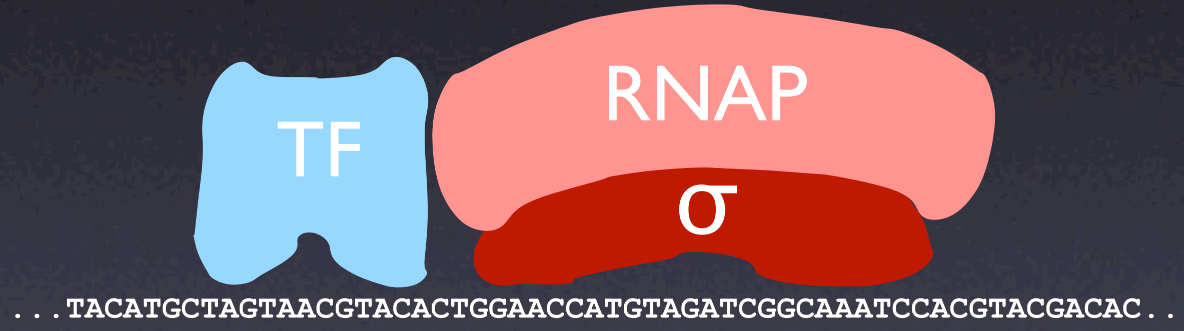

RNAP
TF
σ
. . . TACATGCTAGTAACGTACACTGGAACCATGTAGATCGGCAAATCCACGTACGACAC . .

- greater evolvability via recruitment
  - piecewise & modular accretion on top of partial solutions, rather than re-engineering of intricate allosteric conformations

Specificity, control, patterns, and evolvability

- Modular network organizations allow for control of specificity via composition and combinatorial regulation
  - software: polymorphism, design patterns, etc.
  - nature: regulated recruitment, localization, weak linkage, etc.

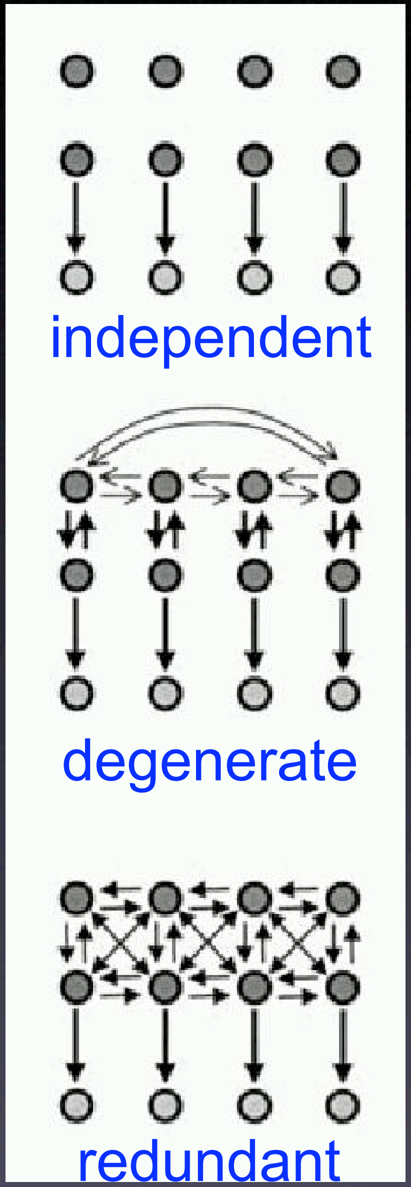



TF
RNAP
σ
. . . TACATGCTAGTAACGTACACTGGAACCATGTAGATCGGCAAATCCACGTACGACAC . .

- A similar story
  - tinkering with partial solutions, rather than re-engineering complex structures
  - the pattern is the message: basic motifs reused with different components

Degeneracy & polymorphism

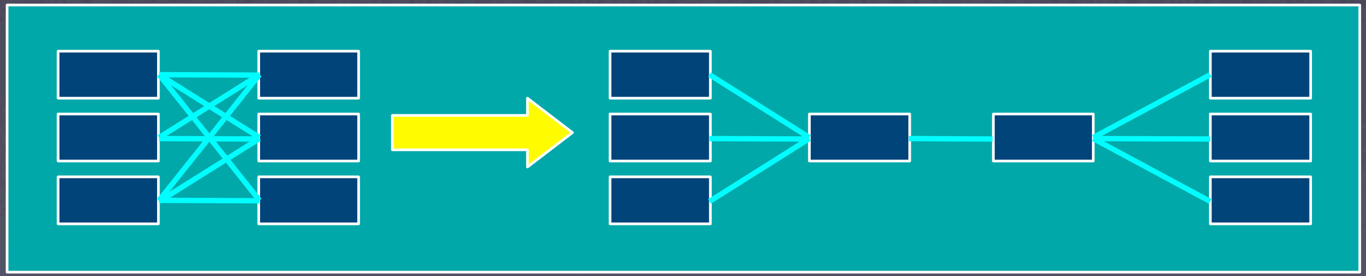
- Degeneracy: "the ability of elements that are structurally different to perform the same function or yield the same output" (Edelman & Gally, 2001)
  - contrasted with redundancy among identical elements
  - differentiation at smaller scales, integration at larger scales
- software systems have little redundancy, but much degeneracy in the form of polymorphism
- polymorphism & neutral spaces



Tononi *et al.* (1999)

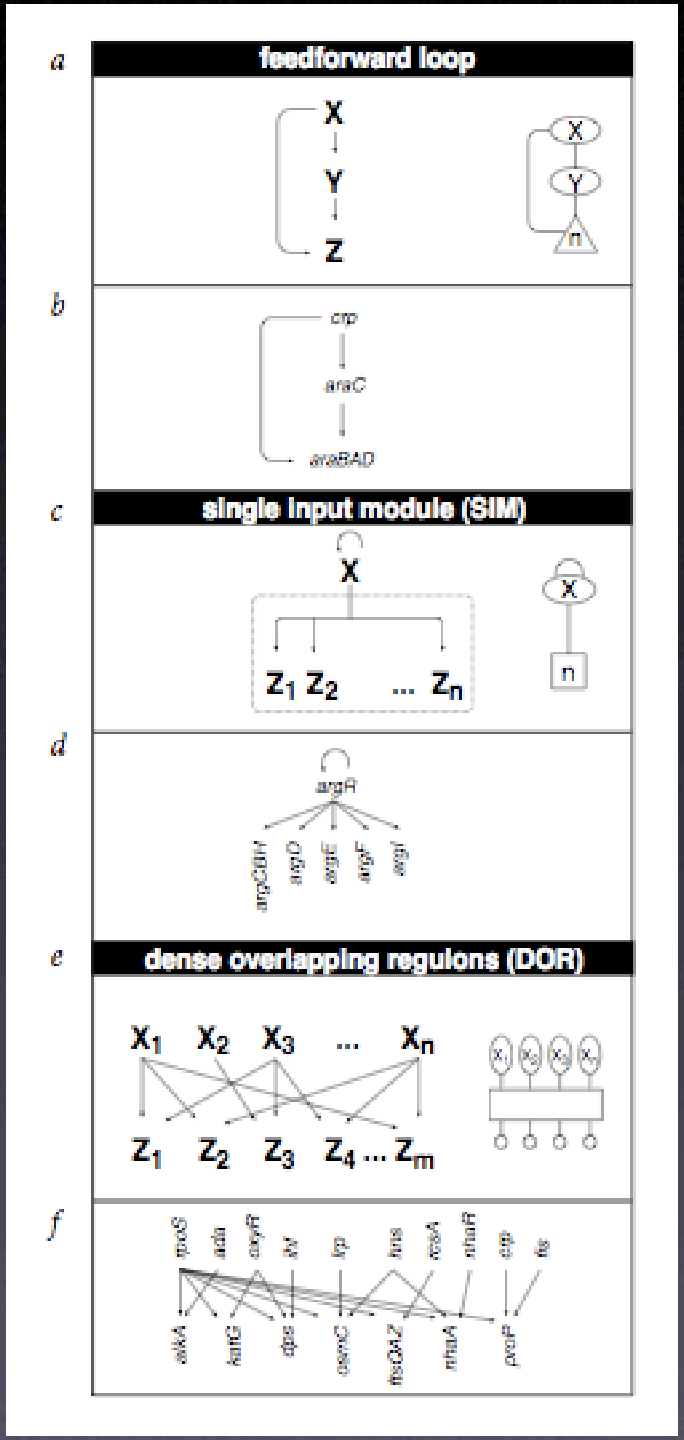
Functional pattern formation?

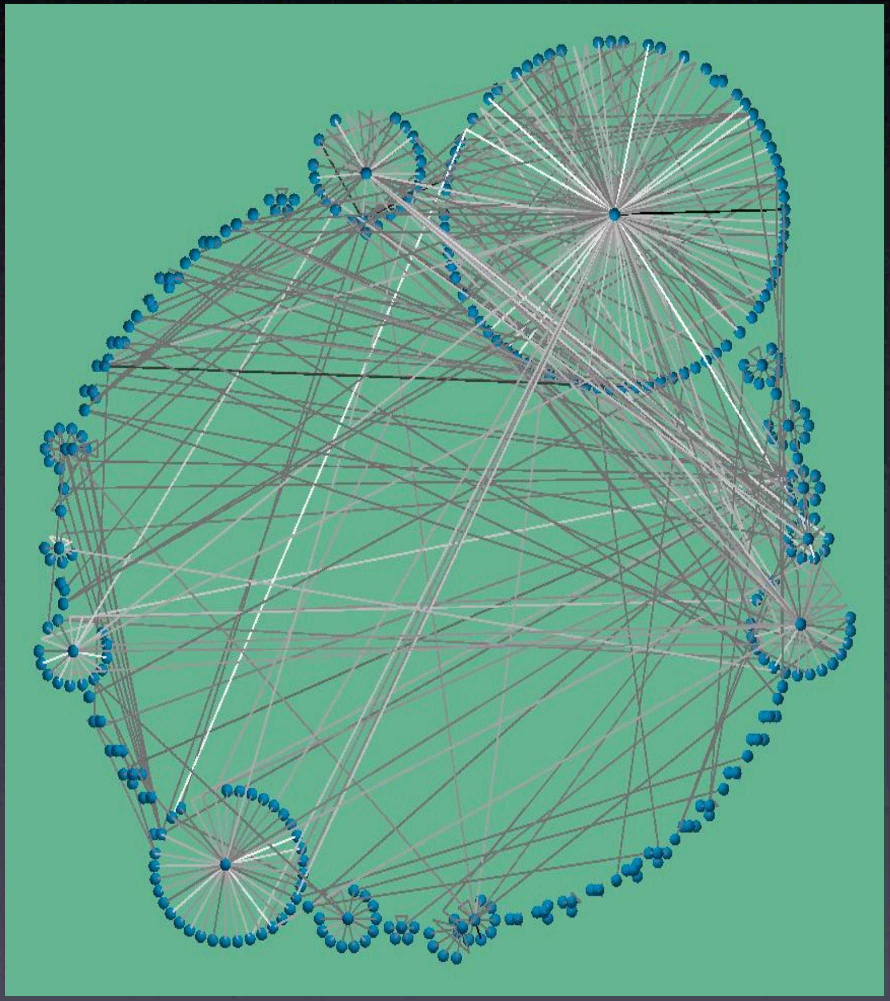
- complex systems typically exhibit nontrivial spatiotemporal responses ("patterns") when driven out of equilibrium
- localized structures often serve to encapsulate strains that would otherwise extend throughout a system
  - e.g., dislocations, vortices, fronts, pulses, solitons
- are there functional analogs to these sorts of spatiotemporal patterns, driven by the variability of the external environment?
  - software design patterns arise to encapsulate variability
  - can patterns spontaneously emerge in nonequilibrium computational systems?
  - modular network structure arises in variable environment (Lipson, Alon)



Beyond network topology

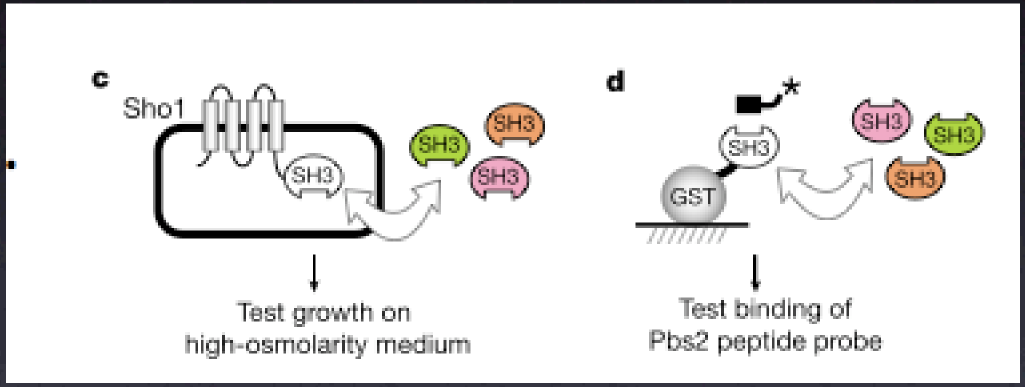
- motifs: recurring patterns in networks
- network topology only part of the story
  - quantifying interaction specificity and correlating that with network structure

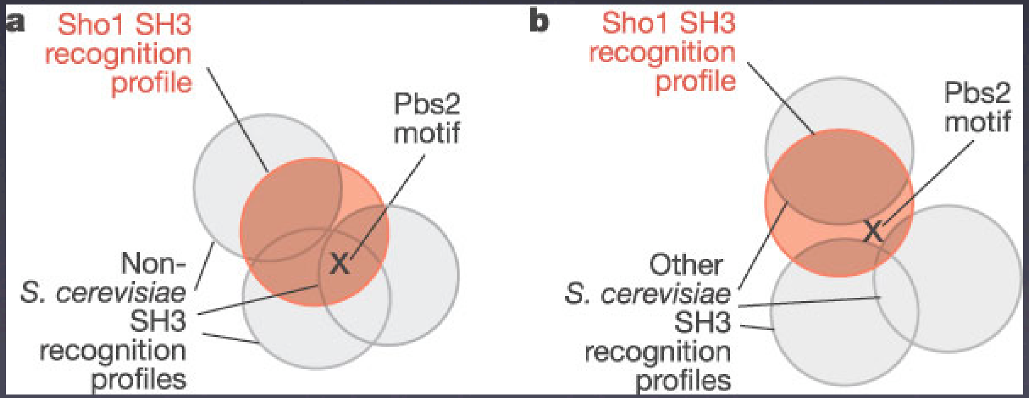


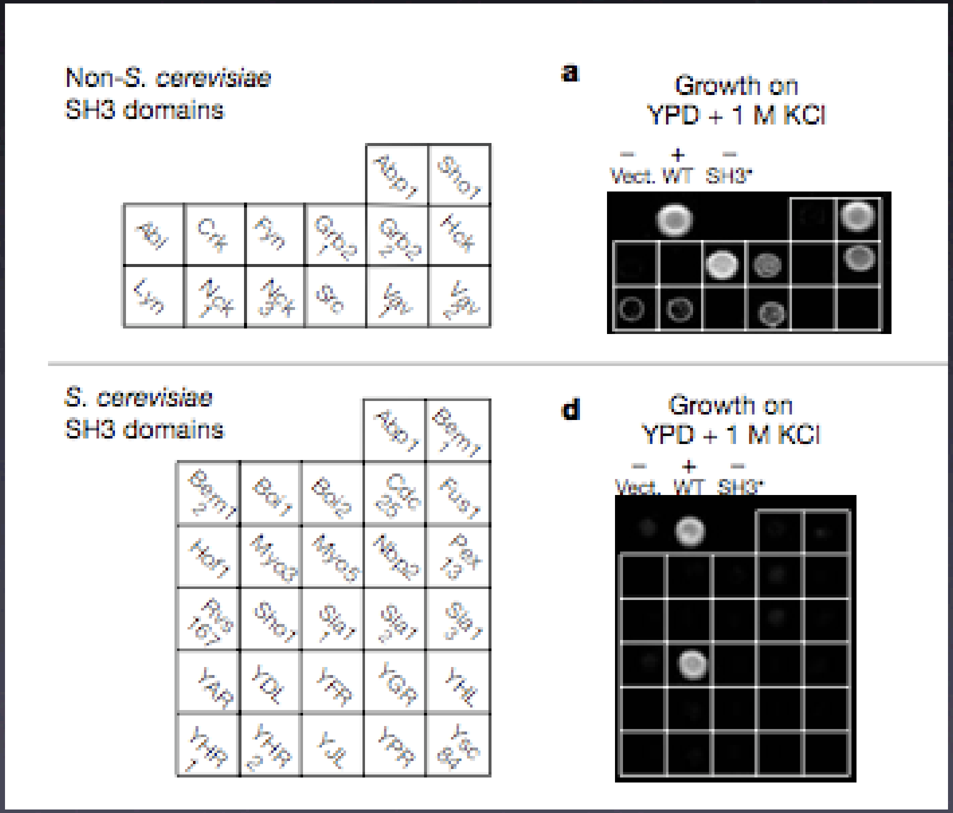


Niches in high-dimensional sequence space

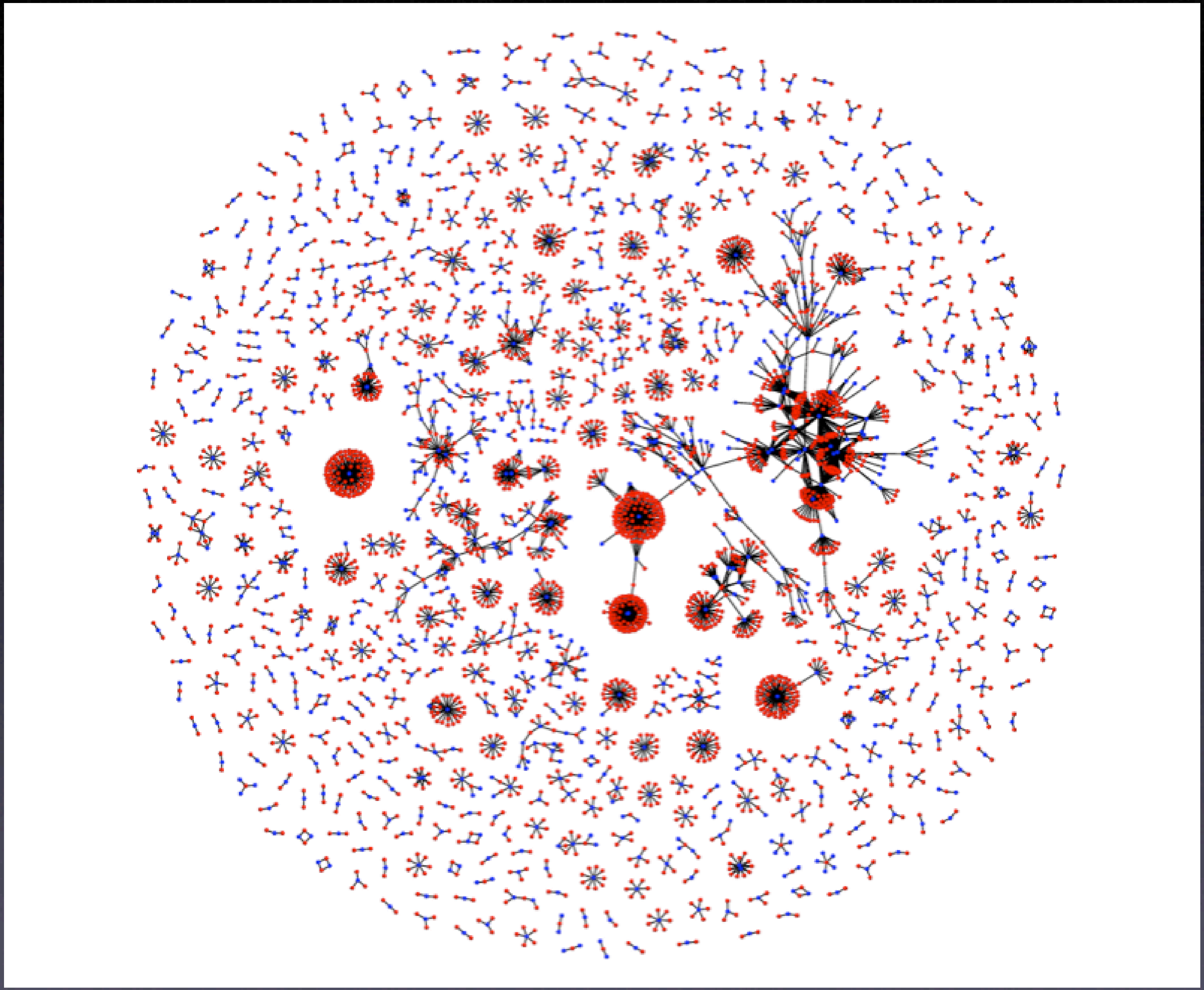
- avoidance of crosstalk in signaling pathways
  - SH3 domains in yeast (Zarrinpar et al. 2003)
  - evolution finds niches in sequence space where proline-rich ligands are uniquely recognized by partner SH3 domains


c
Sho1
SH3
Test growth on high-osmolarity medium
d
GST
Test binding of Pbs2 peptide probe


a
Sho1 SH3 recognition profile
Pbs2 motif
Non-S. cerevisiae SH3 recognition profiles
b
Sho1 SH3 recognition profile
Pbs2 motif
Other S. cerevisiae SH3 recognition profiles



A biological analogue to software collaboration


A “library” of conserved protein domains put to use in *Pseudomonas syringae*
(blue = conserved PFAM domains, red = P. syringae proteins containing domain)

Conclusions

- Software design highlights different sorts of analogies with biological systems than arise in other engineering disciplines
  - network complexity to support evolvability
  - organization of specificity
- Need for quantifying how biological systems organize and process specificity
  - need for methods that estimate intrinsic binding affinities rather than sequence similarity
- Models for studying the emergence of functional patterns?