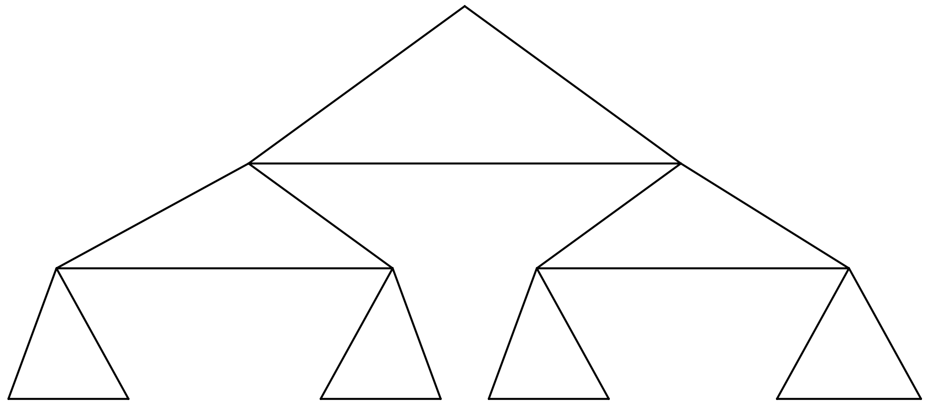
Mixed Trees and Layers

- Layered human organizations are locally relatively horizontal and globally tree structured. Locally they form a team, and rely on leaders to form interconnections resulting in tree structures of clans (see Watts, Chapter 9, also in Dodd, Watts and Sabel’s paper, although they come at it from a very different perspective)



Characeristics of Human Layered Organizations

- Members of a given layer can have multiple parents or can relatively easily switch parents at the layer above them
- Cooperation and trust are important attributes
- Members of a given layer can interact readily with other members at the same layer
- I believe that middle managers in such organizations recognize that a significant part of their job is increasing trust between their team members and members of other teams with whom they will need to work at some point

Overlays

- Large partnerships (e.g., consulting firms) are often layered, and use project teams that may be best modeled as tree structures which are overlayed on the base structure
- Matrix organizations (two bosses) may be viewed as overlays as well. At MIT we have departments as well as centers, and most faculty members have a department head and a center director as ‘bosses.’

Complexity and Flexibility

Complexity and Flexibility: Structural complexity

- Kolmogorov defined the (structural) complexity of a function as the <u>length of its shortest description</u>. This gives a lower bound to the complexity of implementations of the function
- The shortest description is, however, hard to determine
- Abstractions in an implementation permit you to reduce the length of a description (recall the matrix example in Herb Simon's chapter)
- There is a trade-off between a long description in an implementation and one that has a high number of <u>layers of abstractions</u>, each of which is relatively short
- The <u>intricacy (related to messiness)</u> of the interconnection pattern in a system clearly adds to the complexity
- The total <u>number of nodes</u> and <u>interconnections</u> also clearly adds to the complexity of an implementation

Structural complexity of an implementation

- “Spaghetti stack” – a messy pattern of interconnections of components – people sometimes use graph theory to measure the messiness of the interconnection pattern – systems or software engineering design methodology tends to reduce the messiness of the interconnection pattern –this is one of these methodologies’ biggest advantages
- Number of components -clearly related to complexity
- Number of interconnections -also clearly related
- In a hierarchy, the number of levels or layers is related to complexity (in a system using layers of abstractions which reduce the number of components and their interconnections, the depth of the hierarchy, if large, clearly adds to the complexity)

My approach to structural complexity in the analysis below

- I assume that there is regularity in the interconnections structure in a generic structure, such as a tree structure . Thus messiness is not an issue
- I assume that the number of layers is not large, so accounting for the increased complexity for a large number of layers is not much of an issue
- I assume that the number of interconnections is larger than the number of nodes
- Thus I simply count the <u>number of edges or interconnections</u> in a system, and call that the structural complexity of the generic system or organization

Examples of Flexibility

(D-design, R-redesign, O-operation, F-function, P-performance)

- Using a tuner in a radio to switch stations – O, F
- Changing gears in a car or bicycle – O, P
- Adding rules in spreadsheets – O, R, F
- Creating a new layer of software on top of or in between existing layers – D, R, F
- Switching roads to avoid congestion – O, P
- Adding connections in infrastructures to increase flexibility (and robustness) – R, P
- Switching roles in NE Patriots' defensive positions – O, F, P

What is Flexibility?

- A flexible system presents alternatives, usually many alternatives, to its function, performance or other ilities
- Some of these alternatives are obvious in the system's interface, but most are not
- A flexible system also makes it easy to make certain classes of modifications in the system during design, redesign or operation of the system
- Not all modifications by an external designer are easily made in a flexible system (largely because these modifications may not rely on flexible parts of the system)

Continuous Notions of Flexibility

- Flexibility is assumed here to be a property of discrete systems in the process of making alternative choices
- English also uses the term flexibility in continuous systems, such as a flexible bow
- One may have a flexible body permitting one to make a large variety of moves and take on a large variety of positions or states
- One could model or simulate these large number of relatively continuous positions with a discrete system, but we shall choose to ignore this possibility, and continue to emphasize flexibility in relatively clear-cut discrete situations

What is Flexibility -2

- Flexibility is related to the relative ease of implementing classes of changes in a system’s function. It does not mean that all changes must be easy to implement. Big changes will usually not be easy. Some apparently simple changes will not be anticipated in the original design, and may not be easy to introduce (Social Security example).
- I define the <u>flexibility</u> of a system to be

  log(paths/nodes)

  where paths is the number of paths in the system, starting with a root node and ending in leaf nodes, but counting cycles just once

Flexibility in relation to other goals and characteristics of systems

- Increased use of flexibility to modify systems usually increases **complexity**
- The **architecture** of a system will play a key role in determining the relationship between flexibility and complexity
- Flexibility may not be free – there is usually some loss in **performance**.
- Flexibility may be used to get around failing nodes or connections in a (networked) system, thus obtaining some **robustness** or **resilience**
- Due to such relationships, flexibility is, for me, the queen of the ilities

Implementing Flexibility

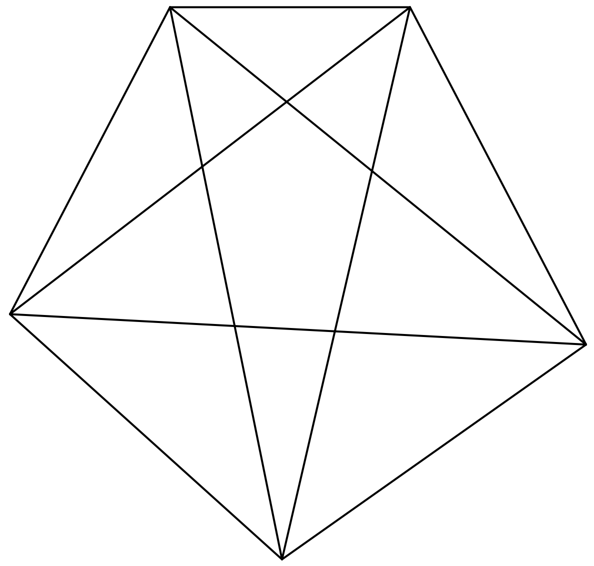
- Flexibility is, for me, a relatively implementation-oriented ility
- Flexibility is related to having discrete alternatives in the system or adding such alternatives in a redesign
- Adaptability is, to me, largely having continuous alternatives or changes, as in a feedback system (e.g., use of a thermostat is related to adaptability, not flexibility)
- Switches (and routers) are ways of building-in alternatives
- IF statements in software are switches
- Interpreters, which may accept an infinite number of different inputs (they have a loop in addition to routers), provide great flexibility
- Uncertainty is less of an issue in software if one can rely on interpreters to handle a huge number of future states

Why are paths important in the definition of flexibility?

- Surely, one can add or modify function by adding new nodes and connecting them to the existing system. In business one does this when merging firms. We, unfortunately, tend to think this is easier than it often turns out to be.
- Often it is possible to make changes by connecting nodes that were previously not connected, and make some small changes within the pair of nodes. This is not possible in a pure tree structure, although it is often done, but it then usually adds a lot to the structural complexity of the resulting (increasingly messy) system.

Team structures

- Team with five members or nodes– a fully connected graph – ten interconnections or edges

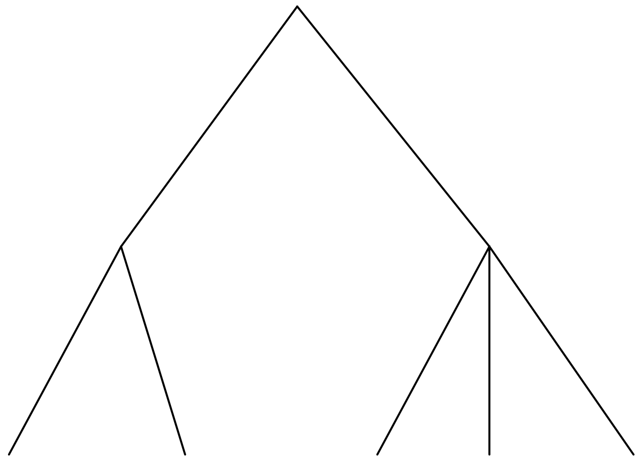


Complexity and Flexibility of Teams/Families

- The number of paths in a team of n nodes is O(n!) which is huge, but the complexity is $O(n^2)$. Thus n needs to be small for human teams/families, due to George-Miller-type restrictions. Clans can be larger, but they too are limited in size

Tree Structures

- A tree with 8 nodes and 7 edges, 5 paths from root node to bottom nodes, 3 levels



Complexity and Flexibility of Tree Structures

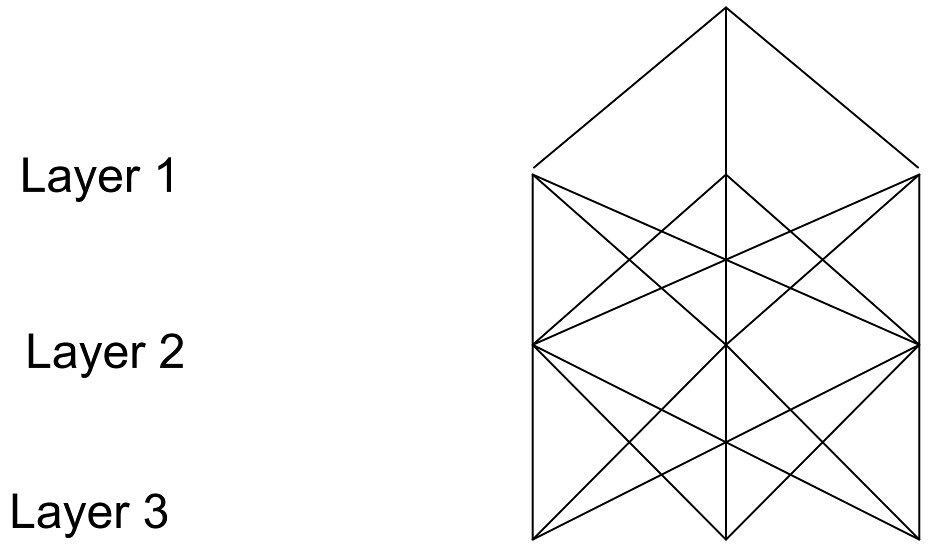
- Complexity of a tree structure is O(n), but the number of paths is also O(n), since the number of paths is equal to the number of bottom (leaf) nodes. The flexibility measure is negative. Essentially pure tree structures are inflexible. Increasing the flexibility of a pure tree by adding edges or interconnections will increase flexibility, but at a potentially significant increase in complexity.

Flexibility and Performance – Tree structured systems

- A balanced binary tree has 2^d nodes at depth d (in a balanced tree nearly all leaf or bottom nodes are at the same depth).
- The number of paths from top to bottom is also 2^d
- The number of switches that are triggered in each node in a path is log (base 2) of the total number of paths – that is, d
- The loss in performance due to the switches is usually bounded by a constant multiple of the number of switches triggered, or log (base 2) of the number of paths in a balanced tree
- An alternative to switching is to have 2^d direct connections between the top and bottom, which is usually difficult to achieve due to fan-out problems, but this alternative eliminates much of the performance loss due to switching. Direct connections may also lose the potential explanatory power of a hierarchical architecture
- A key weakness of a hierarchical tree-structure architecture is that redesigns may add greatly to the complexity of the system and may prevent additional changes after some point. In other words, pure tree structured systems (often the key intermediate result of systems engineering or software engineering) are often overly complex and inflexible after some redesign

Layered Hierarchies

- Layered structure with three layers – no purely horizontal interconnections – may connect to any or even all nodes in layer immediately above or below



Complexity and Flexibility of Layered Structures

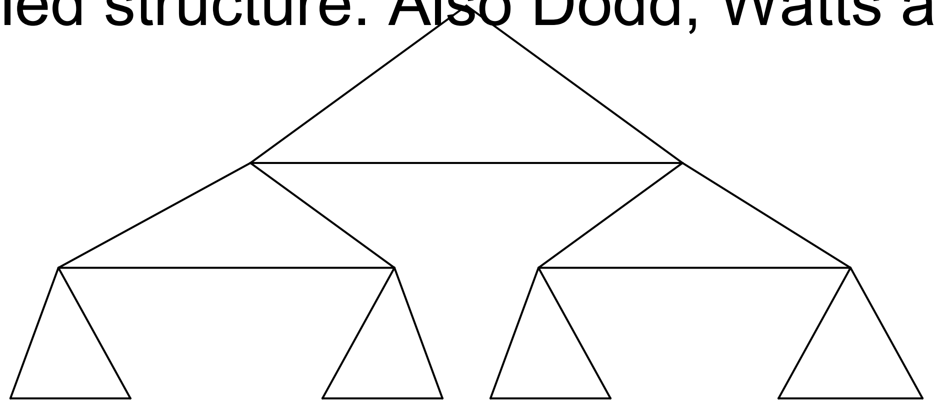
- Assume a layered structure with no horizontal interconnections, with k layers, each of n/k nodes. The number of paths will be $O((n/k)^k)$, far higher than a tree structure with similar number of nodes, except for k=1. The complexity will be quadratic in n, thus too high for human organizations (use hybrid organization instead) and for some technical systems (use routers in some cases). This level of complexity presents no problem in pure mathematics

Flexibility and Performance – Layered Systems

- A hierarchy of routers or interpreters is one set of examples of a layered system
- Layered systems are related to levels of abstractions, and have relatively easy explanations in comparison to the overall system
- Performance suffers somewhat due to the cost of getting through each layer – people who don't like a layered approach tend to emphasize this point
- If one is careful, one may permit the violation of an abstraction by allowing direct paths through one or more layers in order to increase performance in certain cases

Hybrid Hierarchy or Mixed Trees and Layers

- Layered human organizations are locally relatively horizontal and globally tree structured. Locally they form a team, and rely on leaders to form interconnections resulting in tree structures of clans (see Watts, Chapter 9 for a similar diagram, but different analysis and different detailed structure. Also Dodd, Watts and Sabel)



Analysis of a hybrid or mixed trees and layers

- What is the flexibility of a mixed tree and layer system?
  - Assume teams with t members each, depth d. Can you determine what the number of paths is for a balanced mixed tree and layer? It will be equal to or higher than $O(n^2)$, but usually not as high as $O(n^3)$
- What is its structural complexity?
  - Answer: O(t n), more complex than a tree, but bearable when t is relatively small
- Thus a mixed tree and layer reduces complexity to something akin to a tree, albeit higher, but flexibility is quite a bit higher than that of a tree
- In contrast to some network models, this hybrid architecture is designed by humans and is relatively controlled, rather than ad hoc or random

Summary of Complexity and Flexibility Analysis

Architecture	Complexity	Number of paths
Family/team	$O(n^2)$	$O(n!)$
Tree structure	$O(n)$	$O(n)$
Layered structure	$O(n^2)$	$O(n^d)$
Mixed/hybrid Tree and Layer	$O(n)$, but higher than tree structure	$O(n^{2+})$

Analysis of layered systems that use routers

- Complexity is $O(n \log(n))$ with routers
- The number of paths is still $O((n/k)^k)$ for k layers (with no horizontal connections)
- Efficiency is lower since one has to go through a router, and routers may get congested with requests. Use multiple routers for robustness and some reduction in congestion
- This makes computer/communication hardware architectures a potentially challenging problem

Complexity and Flexibility of Networks

- General network structures take many forms. However, a grid type of network (nearest neighbors are connected) will have the number of paths that is exponential in the number of nodes. Consider adding just one new node and connect it to its nearest neighbors, and see that the number of paths is at least doubled. The complexity is O(n), which is good. However, such networks lend themselves to highly distributed control as well as cycles (and thus feedback), and this makes it often relatively difficult to analyze or control the behavior of the system

Laterality

- We can distinguish parent/child vertical connections from ones that are more horizontal or lateral
- We define laterality to be the ratio of the number of lateral connections to the number of vertical connections
- The higher the laterality the more paths we are likely to have and thus the greater the flexibility of the system (and thus the greater likelihood of robustness)

Flexibility of Linearly Structured Systems

- Consider the number of ways of assigning offices on a given floor, where the number of offices to be assigned varies from 0 to k, $k \leq n$, and different groups require m contiguous spaces, $m \leq k$
- Claim: the number of alternate assignments is 2^n, the number of subsets of n
- Hence doubling n yields a squaring of the number of alternatives

Staging Alternatives

- The office assignment problem is related to the number of alternative assignments of groupings in a staged system (e.g., the number of satellites to be placed in orbit in stages, the number of floors in a garage to be added in stages)
- In the garage example, one may need to add performance up-front (to the beams) so that one has the option to add floors at a later point
- The B52 can be viewed as an unexpected example of staging. Its performance was higher than was actually needed. Thus in later stages of its life new equipment could be added to the plane that lowered the maximum height performance, but the height performance was still above the minimum needed to avoid enemy fire, and the new functions thus obtained were extremely useful

Flexibility and Routers

- Routers rely on tags or addresses to make the switch. As a result, the number of alternatives tends to be higher than in a (binary) switch.
- Routers play key roles in infrastructures as well as in software systems
- Language interpreters are often implemented as routers, but include a loop. This is about as close as software normally gets to having feedback
- Layered software/hardware systems can be considered as (layered) hierarchies of routers/interpreters
- New layered abstractions greatly increase the system's flexibility and expressive power
- In biology, cells, bilateral architectures, limbs, and the neo-cortex all provide such abstractions (viz. Kirschner and Gerhart - they don't use this terminology, but should)

Flexibility, networks and Robustness

- Networks tend to have very large number of possible paths from a starting node to end nodes, thus their flexibility is very high
- The number of paths is often so high that it is relatively easy to circumvent a failing node or connection, thus obtaining a measure of robustness
- Interestingly, the ease of changing internally (flexibility) allows one to successfully resist internal or environmental challenge or change (robustness)
- Moreover, analyses of network flows may lead to the identification of the few additional connections that would result in an even more flexible and robust network design

Flexibility and Rates of Change

- Very slowly changing systems do not need to be very flexible – one may change these systems slowly and pay a penalty in complexity
- Systems that are the first implementation of their type need flexibility – second implementations may pay more attention to performance

Measuring Flexibility

- Our emphasis on paths makes the number of paths a natural component of the measure of flexibility
- By itself such a measure is not very useful. It needs to be related to other system measures and characteristics, such as (structural) complexity and (generic) architectures
- It’s about the relationships, stupid

Universality

- Interpreters (e.g., instruction execution in a microprocessor, language interpreters) can execute (or simulate) any well-defined procedure (Church-Turing thesis)
- All it takes is a router, a looping capability, a read/write capability and unbounded time and storage
- This property of information systems may be a reason why uncertainty is less of an issue in software systems – one expects to handle any future state (ignoring efficiency considerations)
- Universality is the ultimate in flexibility

Layering as an Algebraic Concept

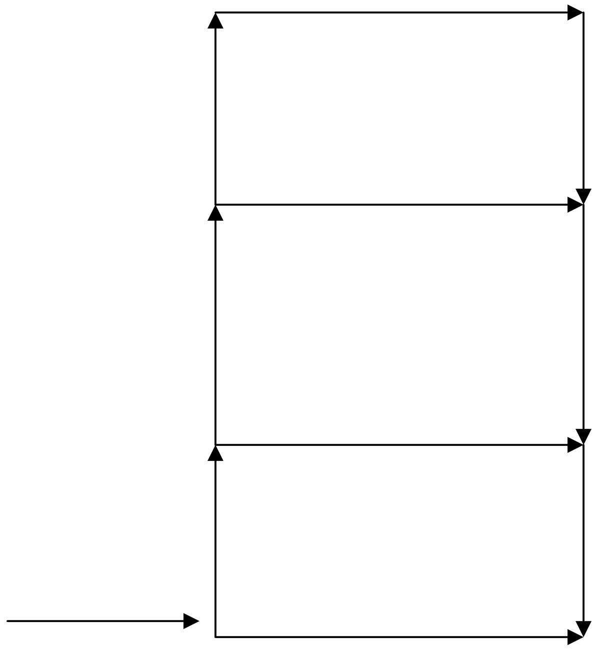
- Trees and layered systems are describable using combinatorics, especially graph theory
- Layered systems are, however, probably best described using abstract algebra
- Each layer is an abstraction of the layer immediately below it, and a specialization of the layer immediately above it

The Telephone Network as Example of Layered Architecture

- The AT&T system in the US (certainly prior to 1983) was basically composed of three layers (each using a different architecture)
  - Local switching loops (routers) at the bottom of the hierarchy
  - Regional interconnections in the middle
  - National network at the top layer
- The capacity of the US system was 10 billion phones given 10 digit phone numbers
- The system was scalable, flexible and robust
- Unfortunately, the rate of change of technology was kept low, partly due to AT&T’s long term investment in copper

The Landline Telephone Network as a Commutative Diagram

Three layered model of land-line telephone system



Landline Telephone Architecture

- Lowest layer (10,000 lines) – uses a large switch or router
- Middle or regional layer (1,000) – might use a team structure, that is a nearly fully connected graph of a few nodes, each a switch of the architecture above
- Upper layer (1,000 area codes) – uses a national network between regions with central offices that switch potentially thousands of ‘calls’ as if they were a liquid, for robustness and efficiency reasons

Abstractions in Algebra: Simple Algebraic Examples

- Consider the (infinite) set of integers, Z
- The rational numbers, R, are an abstraction of Z
- Each integer has an infinite number of rationals for which it acts as numerator or denominator – no problem having so many implicit connections
- The integers modulo 3 (i.e., 0, 1, 2) are a specialization of Z
- Polynomials in x with integer coefficients, $P_z(x)$ are an abstraction of Z
- Polynomials in y whose coefficients are in $P_z(x)$, $P_z(x,y)$ is an abstraction of $P_z(x)$

Integration using algebraic abstractions

- Consider $\int f(x)\,dx$, such as $\int x\, e^x\, dx = x\, e^x - e^x$
- Suppose f(x) is in an abstraction (extension) of the rational functions in x, say F(x)
- (Liouville, Ritt, Risch Theorem) The integral of f(x), if it is expressible in terms of the usual functions in the calculus (exponentials, logs, roots of such functions), then

  $\int f(x)\,dx = A(x) + \sum c_i \log(B_i(x))$

  , where c_i are constants, and A and the B_i are also in F(x)

Simple Examples

- $\int e^{x^2} dx$
- The integrand is in $R(x, e^{x^2})$. In this special case we know that there are no log terms, and that the form of the integral is
- $\int e^{x^2} dx = A(x)\, e^{x^2}$, where $A(x)$ is in $R(x)$, that is A is in the immediately lower field
- Differentiate both sides

  $e^{x^2} = A'(x)\, e^{x^2} + 2\,x\,A(x)\, e^{x^2}$, divide both sides by e^{x^2}

  $1 = A' + 2x\,A$

  If A is a rational function in x, could it have a nontrivial denominator? No, since A' would have a denominator of higher order and the other terms would not cancel it.

  Thus A must be a polynomial in x of degree n, say

  What could n be?

  0 (n-1) (n+1) the degrees of the three terms on the two sides of the equation

  Thus, 0 = n+1, or n= -1, but that is a contradiction since A would then have a non-trivial denominator

  Hence the integral is not expressible in closed form

  This proof used to be dozens of pages long prior to 1970

Simple integration examples

- $\int x\, e^{x^2} dx = A(x)\, e^{x^2}$, where A(x) is in R(x)
- Differentiate both sides and divide by e^{x^2}
- $X = A'(x) + 2\, x\, A(x)$
- As before, A has no nontrivial denominator
- Say A is a polynomial of degree n in x
- 1 (n-1) (n+1) the degrees of the three terms above
- So n=0, A(x) = a, a constant
- $X = 0 + 2\, x\, a$
- Solve for a
- a= 1/2
- $\int x\, e^{x^2} dx = ½\, e^{x^2}$

Abstraction and Problem Solving

- A key value of abstractions is that they simplify a problem by hiding layers (not just modules) of detail in its solution
- A possible approach to problem solving (in contrast to hierarchical decomposition) is “repeated abstraction”
- Also instead of starting at the bottom (or top) one may want to start in the middle (“middle-out”) and develop abstractions (going up) and specializations (going down) from there
- Such an approach may be best when one is exploring a new problem domain. Top-down, in order to be successful, usually assumes you already know what the specs are or ought to be
- Just as there is no unique decomposition of a given problem, there is no unique abstraction

Arguments against abstractions and layering

- It is not clear how to create a “good” abstraction. The mathematician who invented matrices circa 1860 thought that there would be hardly any use for them – it is thus even hard to tell when you have a good abstraction
- Similarly, it is not clear how to create a good decomposition, but it is relatively easier to create some decomposition
- Abstractions leading to higher layers will likely lead to some, possibly minimal, loss of performance
- In CS, this loss of performance has been used against most new abstractions, such as high level languages (e.g., FORTRAN) and VLSI design languages
- New computer architectures and improved speeds have often vitiated such arguments

Data Abstraction

- See Chapter 2 of Abelson and Sussman’s 6.001 text
- Rational number arithmetic procedures (+, *) are based on lower layer procedures for integer arithmetic
- Polynomial arithmetic in one variable with rational number coefficients is based on rational number arithmetic procedures
- Etc.