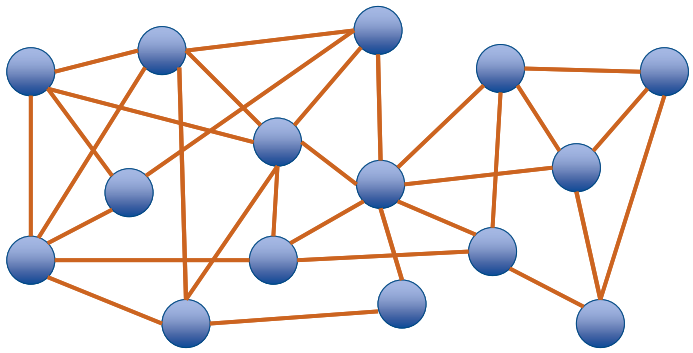
Complexity

In this video we will examine the relationship between software size, measured in lines of code (LOC) and the complexity of the software. We will see that as software size increases, complexity increases exponentially.

Complexity vs. LOC

- Complexity is related to the number of **interactions** between lines of code.



Software is one of the most complex things invented by humans. This is because software is not a monolith, but is made up of thousands, or even millions of lines of code. Each line can theoretically affect all of the other lines of code.

So the complexity of a program is not directly related to the number of lines of code, but, in fact, it is related to the number of interactions between the lines of code.

Now not every line of code will be connected to every other line of code, but the complexity of a program is some fraction of the theoretical maximum interactions between lines of code. What is this theoretical maximum?

We can apply a theorem from graph theory to help us figure this out.

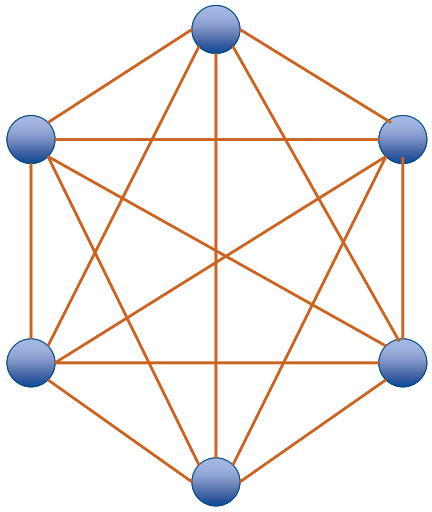
A graph is a collection of nodes and edges. Here the nodes are the circles and the edges are the lines.

So lets represent the lines of code as nodes, and the dependencies between them as

edges between the nodes.

Number of Edges in a Complete Graph

n	e
1	0
2	1
3	3
4	6
5	10
6	15
n	n(n-1)/2



Let's count the number of edges in a complete graph. A complete graph is one in which each node is connected to each other node.

If there is only one node, there are no edges.

If we have two nodes, there would be one edge.

If we add a third node, the number of possible edges is three.

Adding a fourth node means that the maximum number of edges would be six.

And so forth.

There is a theorem from graph theory that says that the number of edges between

nodes in a complete graph is given by n(n-1)/2, where n is the number of nodes in the graph. There is a nice proof by induction for this theorem, but we won't go into that here.

Moore's Law

- The density of transistors on a chip will double every two or so years
- This law has held up pretty well over the years
- Corollaries:
  - RAM size in a computer
  - Hard drive space in a computer
  - Internet bandwidth

Gordon Moore was one of the founders of Intel, the chip maker. Moore postulated that the density of transistors per square inch was doubling roughly every two years. There are multiple corollaries to Moore's law. One says that the cost of RAM decreases every couple of years. This has enabled computer designers to put more and more memory into computer as time went along. Anyone who has bought computers over the course of five or ten years can attest to this. Program size (in lines of code, LOC) is influenced by the amount of RAM memory in a computer. More RAM, more lines of code can be squeezed in.

Complexity Trend

- Maximum number of lines of code, n, grows geometrically, $n = 2^t$
- Maximum number of interconnections between lines of code grows exponentially n(n-1)/2

The bottom line is this: as the typical size of programs in lines of code grows geometrically: 2 to the power of time, t.

The complexity of those programs grows exponentially: n(n-1)/2 or just n squared, which is much steeper growth curve.

So, what are we to do about this increasing complexity?

Answer: figure out a way to reduce the complexity. We will look at this in the next couple of videos.

Next

- Coming up:
  - Goals of Software Engineering
  - Principles of Software Engineering

In the next video we will discuss the goals of software engineering.

We will then talk about some of the important principles that we have found to be useful in achieving those goals.