DEPARTMENT OF COMPUTER & INFORMATION SCIENCES

University of Strathclyde Science

# Software Engineering

## Lecture 9: Models & Methods

### Plus Big O Notation & Dev Ops

Billy Wallace
w.wallace@strath.ac.uk

CS993

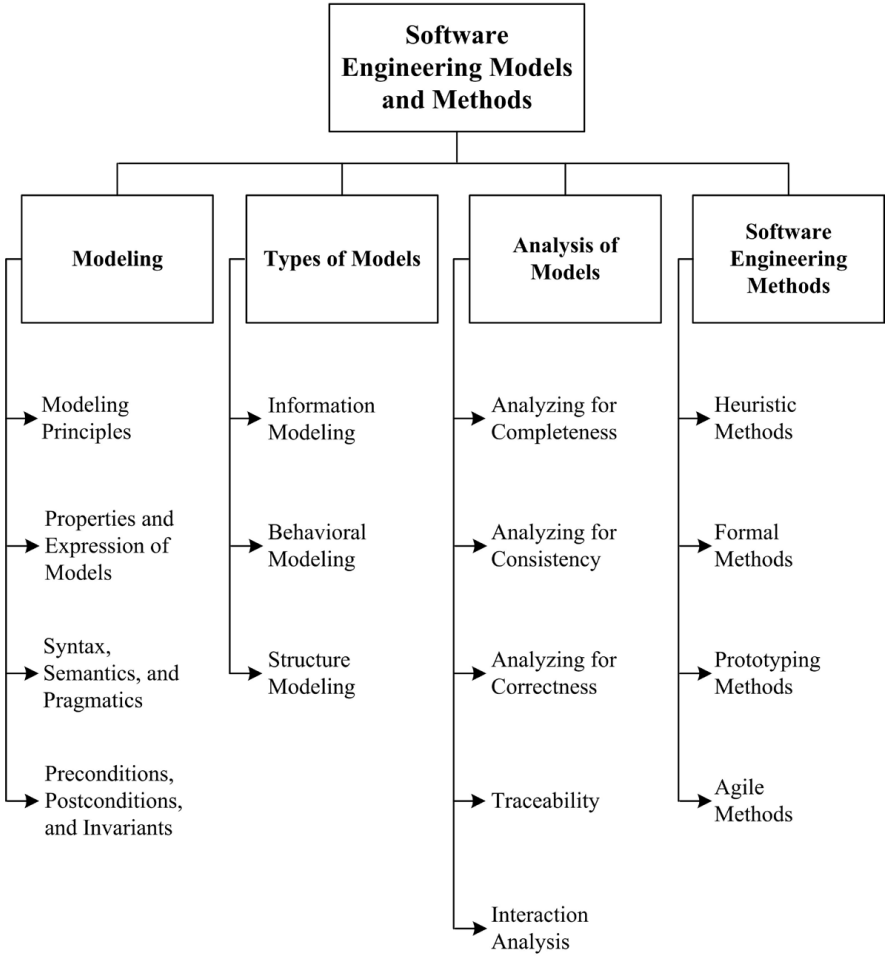
University of Strathclyde Science

# Lecture Outline

- A (very) brief overview of models & methods.
- Some theory.
- Regular Expressions.
- Parsers.
- Big O notation.

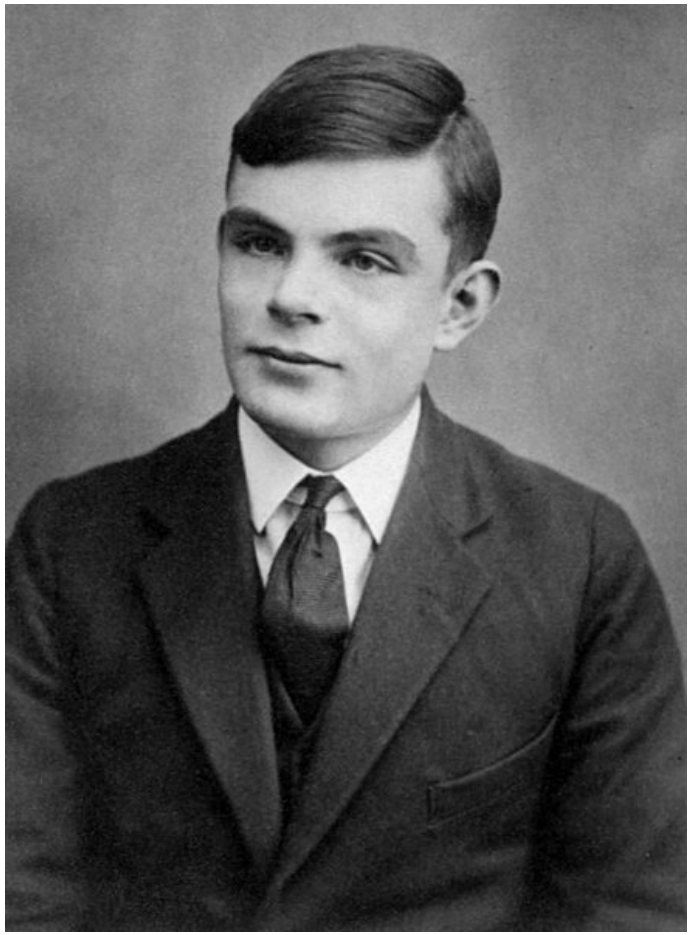
- Dev Ops
  - Virtual Machines and Containers.
  - Continuous Integration.

# Models & Methods



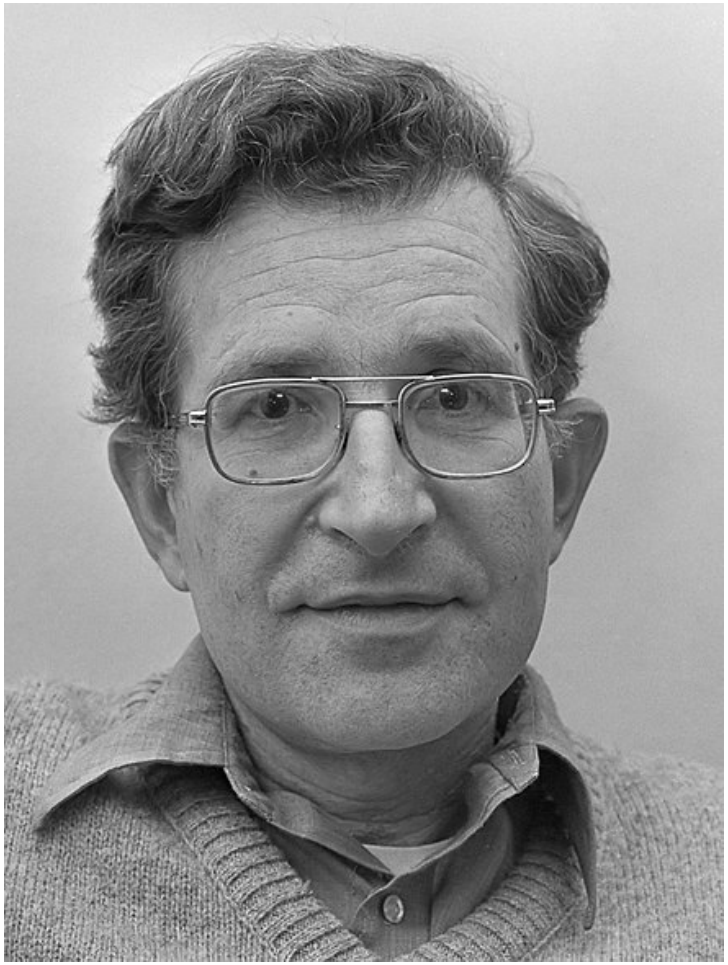
University of Strathclyde Science

# Who's this?

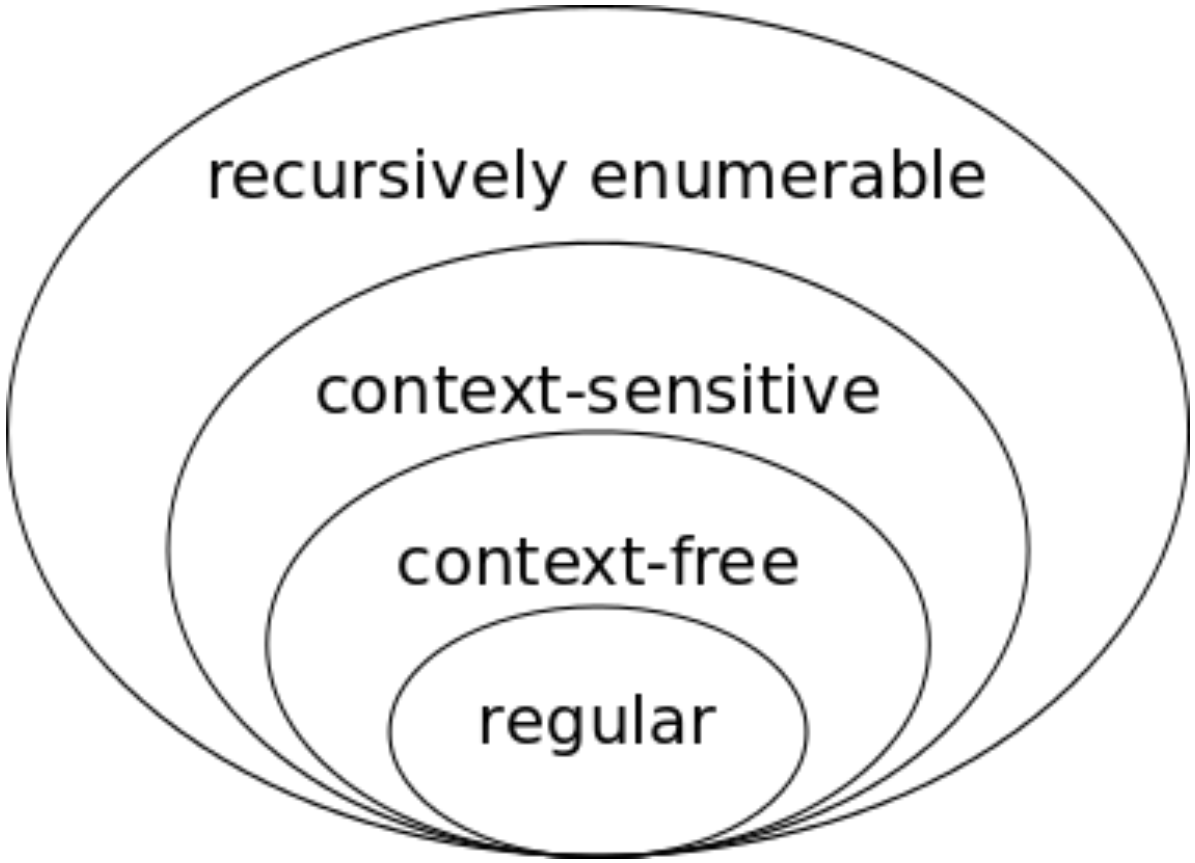


University of Strathclyde Science

# Who's this?



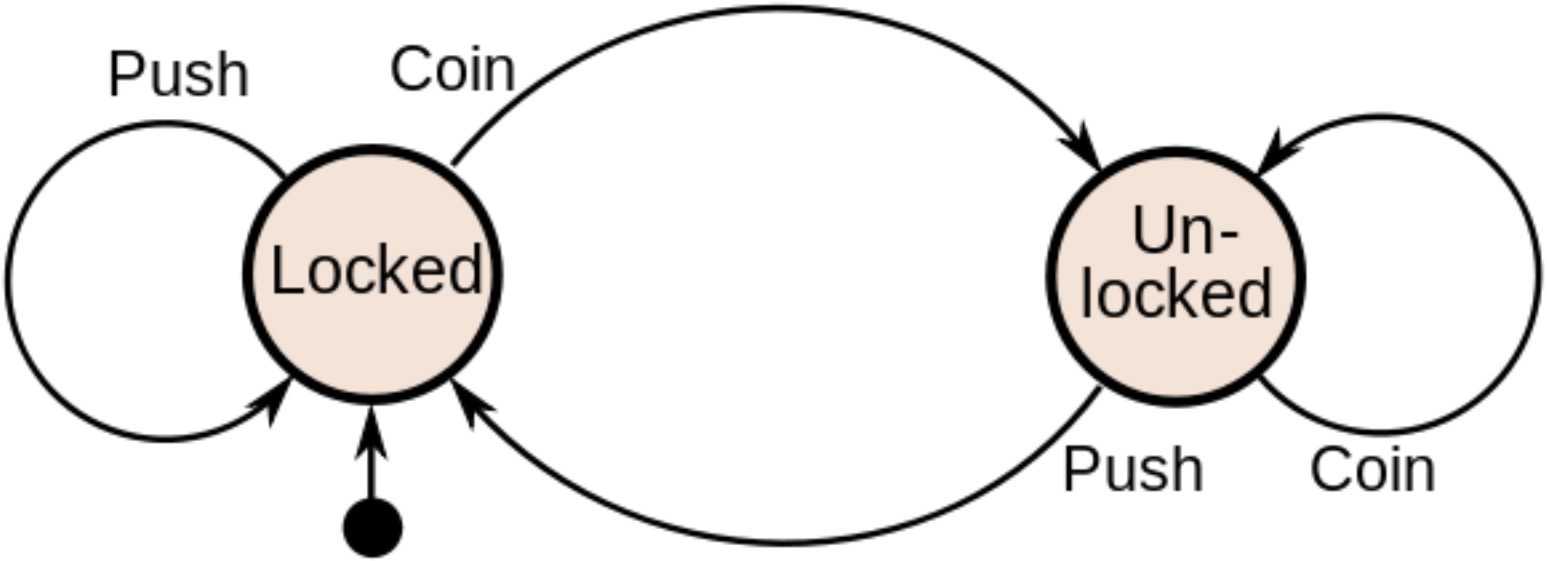
# Chomsky Hierarchy



# Regular Expressions

- Java Strings have this built-in. This can be a great way to do things like check the format of data being entered.
- Note that the wildcard formats for filenames are different.
- You can play with regular expressions using the Unix grep command. What do you think the “re” in the middle of grep stands for?

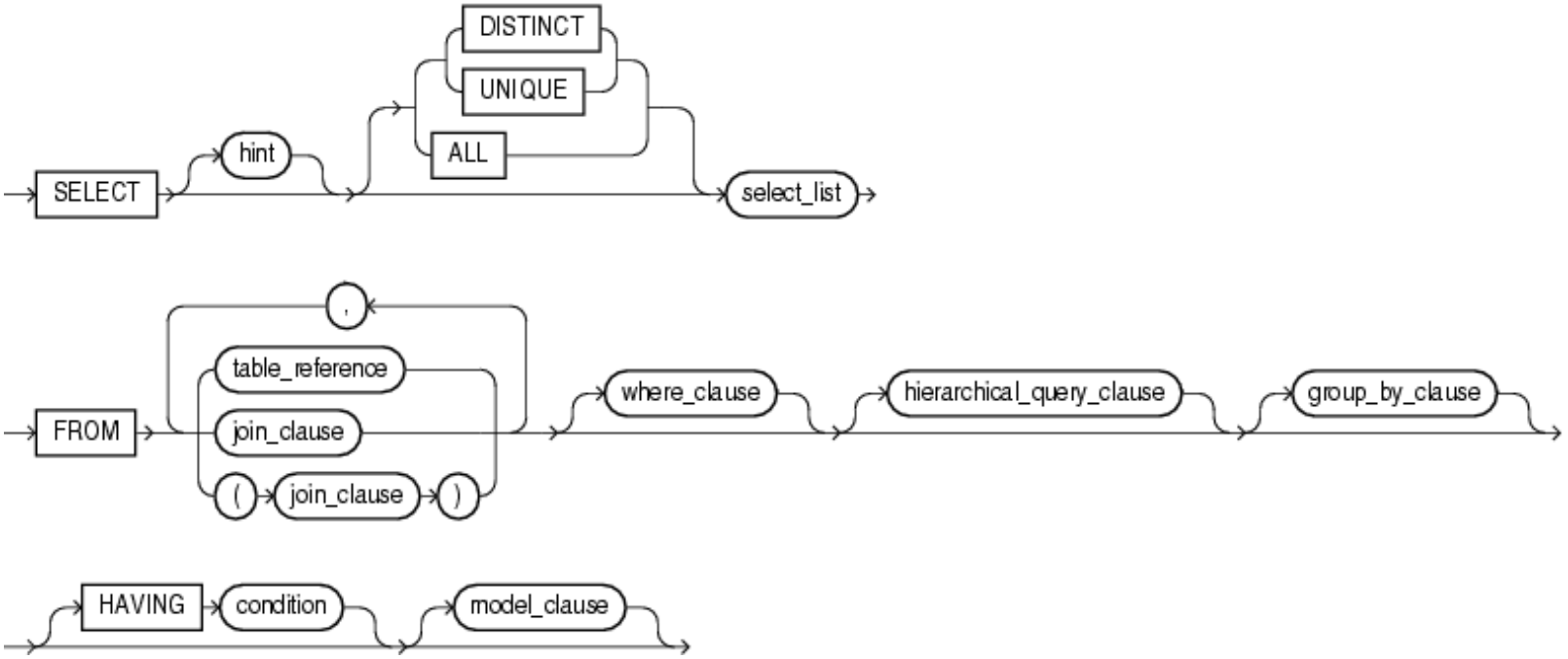
# Finite State Machines (FSMs)



# Backus-Naur form

- BNF is a commonly-used text format that can describe languages.
- The Wikipedia page has good examples:
- https://en.wikipedia.org/wiki/Backus–Naur_form

# Oracle SQL Syntax



# Big O Notation

- This allows us to show how running time or space requirements grow for algorithms or data structures.
- We are interested here in how things scales. We look at asymptotic behaviour and are only interested in the dominant factor. Scale and efficiency are two separate and different concepts.
- This often features in software interviews.

# Linear Search and O(n)

- If we walk through an unordered array looking for a particular value, we need to go through all n elements, so this is O(n).
- If we sort the array, we can stop when we find the item or something bigger than the item, so on average we only search through half of the items - this is still O(n).

## Comparing everything to everything else is O(n^2)

- If we wanted to find the two items that were closest in a multi-dimensional space, we might need to compare everything against everything else.
- Consider a naive collaborative filtering scheme implemented with 2D arrays. We can't scale such a data structure as it quickly becomes too big.
- For n^2 time, we look for clever data structures (e.g. kd trees) or brute-force it using GPUs (redefine what we mean by big).
- For n^s space, we look for smarter data structures, e.g. above we could use sparse arrays.

# Binary searches are O(log n)

- Going back to our sorted array, we can search using a “binary chop”. Each step we split the data in half, so for 1 million elements we only need to look at 20 to search.
- The same goes for binary tree structures.
- Technically we are doing base 2 log here. Data structures like b-trees can use a different base.

# Sorting is O(n log n)

- Good sorting algorithms like QuickSort are O(n log n).

# Complexity

- Sometimes people will say “complexity” when they mean big O notation, but sometimes they mean something different.
- There are classes of problems that are horrifically complex, e.g the exact solution to the Steiner Problem in Graphs.
- You may hear people talk about things being NP Complete or NP Hard. This just means that it is so difficult that you aren’t going to find an exact solution unless the problem size is very small.
- Since algorithms for exact solutions don’t exist, we typically attack these problems by using heuristics to get approximate solutions.
- Think of games. In noughts and crosses, we can produce an exact algorithm that will always win or draw, but for chess or go, we can only look for approximations that might win (we can now beat humans most times).

# Dev Ops

- This has been generating a lot of buzz, but not everyone even agrees on what it is.
- Personally I think that the benefits are far from proven and that you'd be better being supported by a proper systems administrator.
- However, there are some useful technologies spinning around this.

# Virtualization & Containerization

- Tools like VMWare, Oracle VirtualBox and Xen on Linux are popular.
- Being able to create virtual machines to host specific services has made it easier to carry out dev ops.
- Containers are lighter in weight and allow this to be accelerated.
- Docker is the main platform for containers.

# Automation Tools

- Automation of software development processes is in my view the big benefit of dev ops.
- Ansible, Chef and Puppet are the main tools here.
- These aren't just shell scripts, they allow things to be automated and orchestrated across many machines.

# Continuous Integration (CI)

- Building development pipelines for CI can be a big help.
- Jenkins is a good tool in this area.
- Continuous Deployment/Delivery (CD) supports the larger lifecycle.

# Summary

- Understanding how to interpret grammars for FSMs and similar is an essential tool.
- There's some underlying theory that we don't need to be too worried about.
- This kind of formality is the kind of modelling we can do to express things in ways that are more systematic.
- Big O notation was described.
- We also looked at Dev Ops and the tools to support this.

University of Strathclyde Science

# Moving on from here …

- The practical shows a couple of tools for building regular expressions and parsers.
- Regular expressions are built in to Java.
- Antlr is an excellent parser generator.

University of Strathclyde Glasgow