# Conceptual Model

Entity behaviours and interactions

State Variables - Defining the state $\mathcal{S}$ of the system. These are **NOT** programmatic variables. They represent more of a system lexicon or glossary than variables in equations or programs.

Assumptions about probability distributions or their parameterizations when **uninformed** by empirical data.

We use a lot of "made up" systems in this course and pluck probability distributions and parameters out of thin air, these are correctly mentioned in the Conceptual Model — when we use **real world data** for the same task, the details are in the SPECIFICATION MODEL.

# Specification Model

Mathematical abstract data types for state variables: integer instead of `long int`, $\mathfrak{Real}$ instead of `double`, FIFO, set, sequence, or mapping instead of `std:queue`, *balanced tree*, `std:list`, `std:map`.

Choosing probability distributions or their parameters from emperical data.

Analysis of any historical data for the input or parameterization of a simulation.

Equations and algorithms describing interactions between entities of the conceptual model, or describing the behaviour ("decision making") of conceptual model entities.

Additional details or data **from the real world** that clarifies conceptual model elements **without** micro-managing or dictating implementation details that belong in the COMPUTATIONAL MODEL.

# Computational Model

Choice of language (implicitly part of the computational model)

Choice of actual variable types and data structures (deque vs list as queue, red-black vs AVL balanced trees,)

How will the code be “tooled up” for verification testing?

# Consistency Checks

**In Verification Phase**: contrived inputs, distributions, parameterizations such that the result given the **specification model** can be reasoned out or calculated without using the simulation code base.

**In Validation Phase**: simulation results with plausible input data or parameterization *do not contradict* our real world experience with the system.

**VETO POWER**: A failed consistency check scuttles the works, but consistency checks **alone** are insufficient to claim a correct simulation.