Discrete-Event Simulation (DES): A (discrete-event) simulation is a program that generates a random **sample path** through a state transition system, where time delays are associated with each state.

There is a single global "clock" – a virtual time. Not to be confused with elapsed real time.

State transitions are triggered by **events** which are ordered in time on a virtual timeline, an event "diary".

DES involves invoking events in time order; if an event is invoked ("occurs", "fires", "is triggered", ...) at virtual time t the clock is updated to t and the code for the event:

- Updates the model state.
- Schedules zero or more new future events on the time line.

Note that the state is unchanged between events.

- In practice, DES is based on a few core design principles:
- The virtual time is a floating-point number (call it *now*)
- The state is defined by a set of program variables, which are typically discrete (e.g. booleans, integers, ...)
- The timeline is a priority queue of *(Event, time)* pairs, ordered by virtual time – essentially an event *diary*.
- Events are implemented as objects, functions, procedures, methods, etc.
- A **scheduler** adds new *(Event, time)* pairs to the diary
- A **descheduler** similarly removes them from the diary.
- Additional measurement variables and code need to be added in order to accumulate performance measures (otherwise there will be no output!)