Architecture

You are to develop software to track the states of machines in a factory consisting of several machines. Each machine is one of several states: running, idle, waiting, down for unscheduled maintenance, down for scheduled maintenance or off. A running machine is currently processing an order. An idle machine is available for processing but is currently without an order to process. Waiting machines are discussed below. A machine that is down for either scheduled or unscheduled maintenance cannot process any orders; however, it has an estimated time at which it will be available for processing again.

Machine A may be dependent upon machine B, meaning that A receives material directly from B and hence cannot process an order unless B is also processing that order. In such a case, if machine B stops running, machine A must enter the waiting state; A must continue waiting either until B is processing again or until A is assigned a di_erent order. Furthermore, waiting is propagated; any machine that depends upon A must also enter the waiting state.

Each machine has a maintenance schedule. Such a schedule has a date of the last scheduled maintenance and the period from that date until the next scheduled maintenance. When the date for the next maintenance arrives, the operator must be informed that the machine should enter the scheduled maintenance phase. However, scheduled maintenance need not occur immediately; the operator determines when scheduled maintenance occurs.

Unscheduled maintenance is conducted when a machine breaks while in operation. Each machine in connected to a failure monitor, which is an external device that senses hardware failure in the machine. When the monitor senses a failure, it informs the machine; then, the machine changes state to unscheduled maintenance state and informs the operator of the state change. A running machine that either enters unscheduled or scheduled maintenance or is turned off also inform any machines that depend upon it that it is not running.

A machine operator issues commands that change the machine's sate. When the machine's state is altered by an internal action, however, the machine must inform the operator that the state change occurred. The cases where the machine's state is altered internally are as follows:

> A machine senses its failure and enters the unscheduled maintenance.
>
> The machine enters the waiting state.
>
> A machine leaves the waiting state because all of the machines that it depends on are running again.

> All changes of machine states must be entered in a log. Each entry in the log includes the date and time of the state change, which machine changed state, and the state from and to which the machine changed. (Note: You do not need to worry about how orders are entered into the machine or about how machine dependencies are established. A separate part of the system will handle that problem.)

Task 1:

Draw a use case diagram of the system and make the following UML models to describe its architecture:

- A model describing what software executes at what node. (HINT: There should be an UML diagram showing the nodes. Nodes are modelled as 3d boxes which you get from the ‘deployment diagram’ in Tau. To model what software executes on each node use a class diagram)
- A sequence diagram showing what you believe happens when there is a ‘failure’

Show how this system will appear in a three layer architecture model (presentation layer, application layer and persistence layer).

Task 2:

Most of you have used - or at least know about Amazon.com, the book seller. Amazon.com gets enormous amounts of requests; therefore there are multiple computers interacting to solve this task.

Discuss - and draw - how this may be achieved. In particular:

* There are multiple web-servers, that take on the first request. How is one chosen?
* A web-server cannot really also handle the books database. On what server/s are the book database stored?
* What happens if one book-server database fails?
* What happens if one web-server fails?
* Sometimes, one will have to replace the software of one of the above servers. Lets say that there are 50 servers. How is this done? How do one ensure that an ongoing transaction is not interrupted?
* Where is billing handled?
* Generation of dynamic webpages is nice, but can be very slow. Discuss how one can provide a faster user experience by generating pages in advance, as soon as a user has identified himself by either sending a cookie, or, by logging in manually.