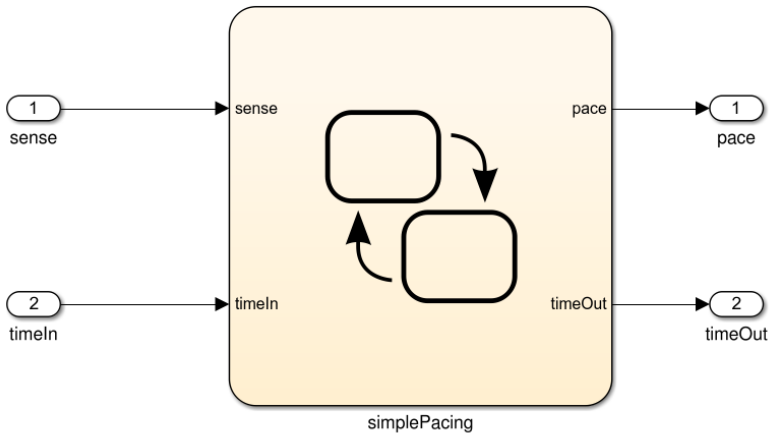
# DIT636/DAT560 - Model-Based Testing Activity

1. **Dracula wants to keep his valuables in a safe that's hard to find. So, to reveal the lock to the safe, Dracula must remove a strategic candle from its holder. This will reveal the lock only if the door is closed. Once Dracula can see the lock, he can insert his key to open the safe. For extra safety, the safe can only be opened if he replaces the candle first. If someone attempts to open the safe without replacing the candle, a monster is unleashed.**

   **Design a finite state machine for the controller software in the secret panel in Dracula's castle. This software controls the system described above. When designing your state machine, model the state from the perspective of the software - that is, what the software is aware of, and what it can control.**

   **For example, states such as "lock revealed" or "candle removed" would make sense - they represent the current operational mode of the system (the lock has been revealed or the candle has been removed and the system is waiting to react to further input) "Dracula can see the lock" would not be a valid state, as it does not reflect something the software is currently doing.**

2. **Consider the following finite state machine representing a simple pacemaker:**




**Input:**

**- `sense` (boolean, represents a sensed heartbeat)**

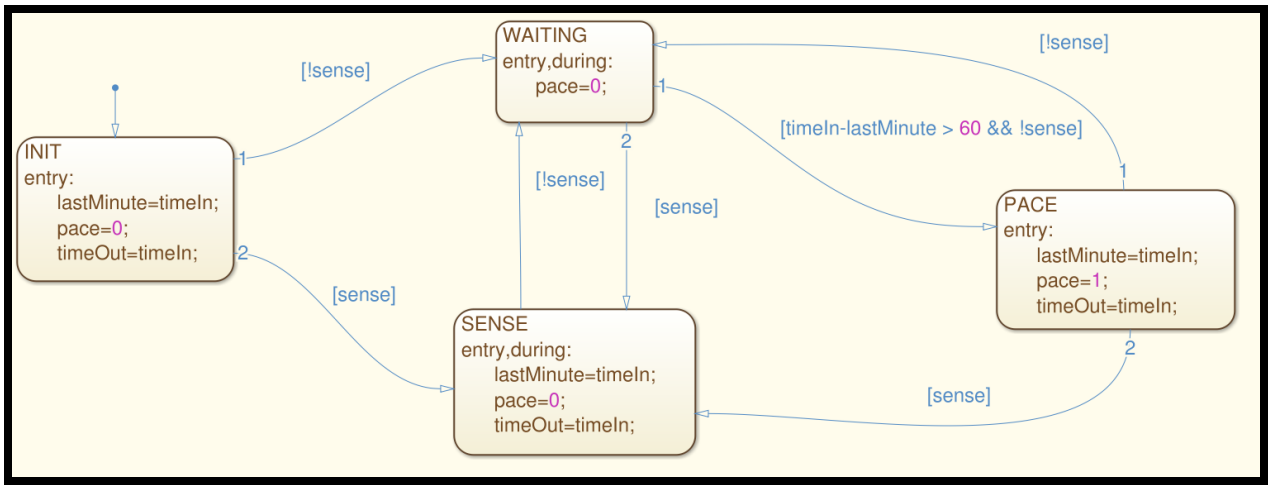
**- `timeIn` (integer, represents the time that the heartbeat was sensed).**

**Output:**

**- `pace` (boolean, represents a command to shock the heart)**

**- `timeOut` (integer, represents the time when the command was made)**

**Model:**



A) **Derive test input (as a series of input values for `sense` and `timeIn`) that achieves state coverage of the model.**

B) **For the same model, derive test input that achieves transition coverage (if your previous tests did not).**