INSTITUT
POLYTECHNIQUE
DE PARIS

AN EVENT-B MODEL OF A MECHANICAL LUNG VENTILATOR

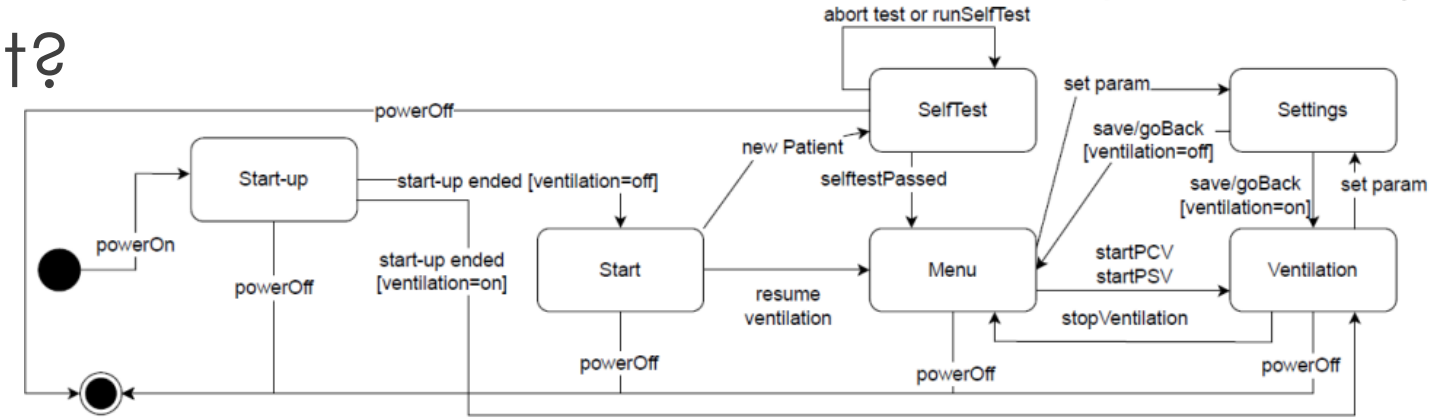
AMEL MAMMAR

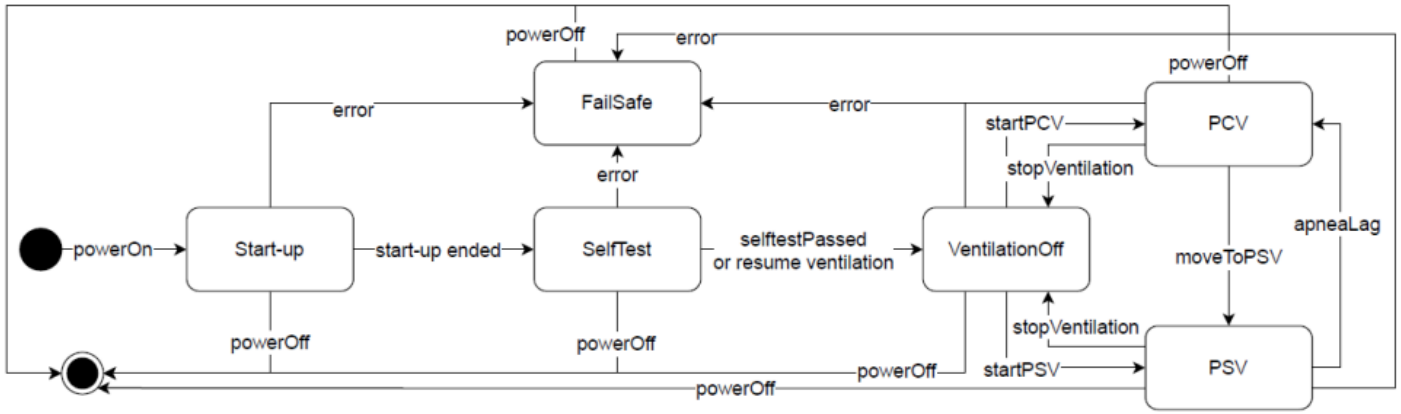
EVENT-B METHOD

- State–based formal method
- A model is made up on:
  - Contexts: sets (user types), constants and axioms
  - Machines/Refinements: variables, invariants, events
- Proof correctness:
  - Machine/Refinement: each event must preserve the invariant of the machine/refinement
  - Refinement:
    - The guard of a concrete event must be stronger than that of the abstract one
    - The effect of a concrete event is included in that of the abstract one

MODELING STRATEGY: 1/4

- Which component, GUI or the controller, should be modelled at first?

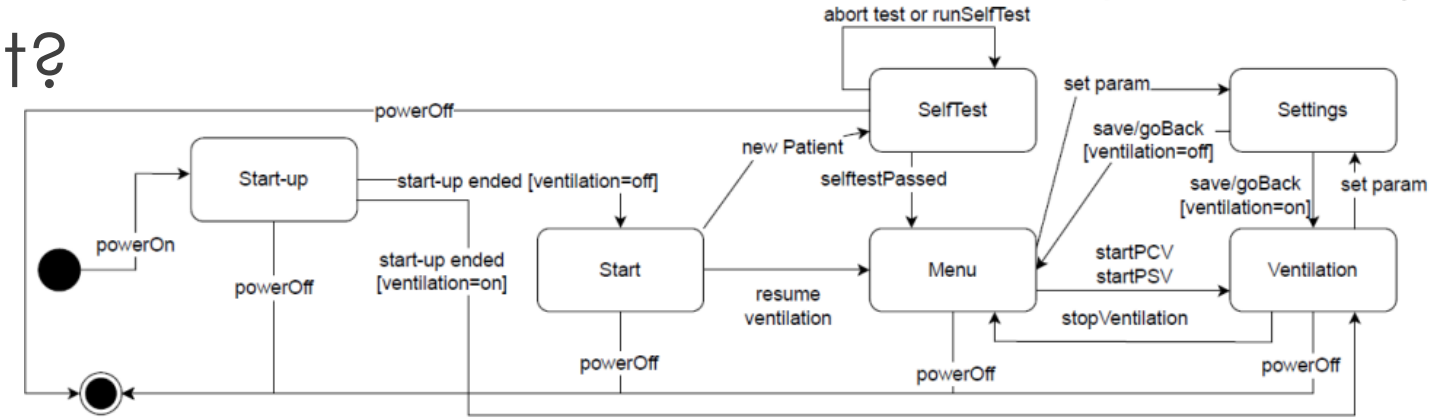


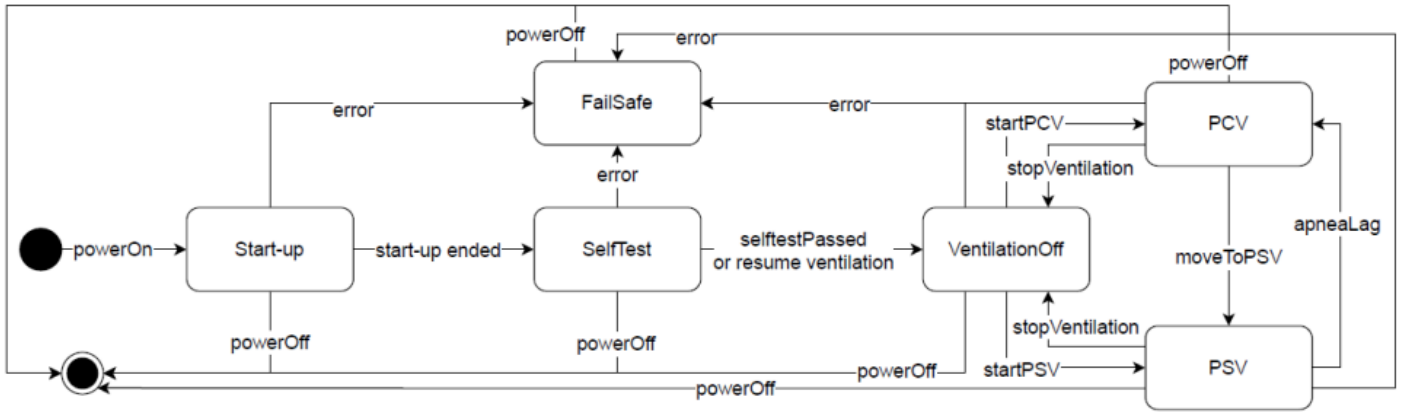


First step of the controller's behavior is independent from that of the GUI

MODELING STRATEGY: 1/4

- Which component, GUI or the controller, should be modelled at first?

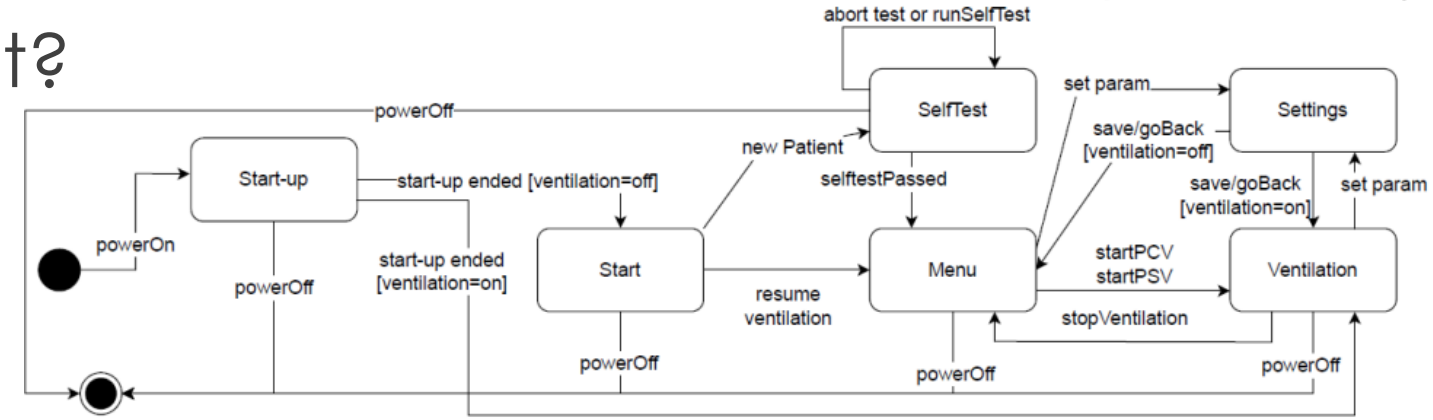


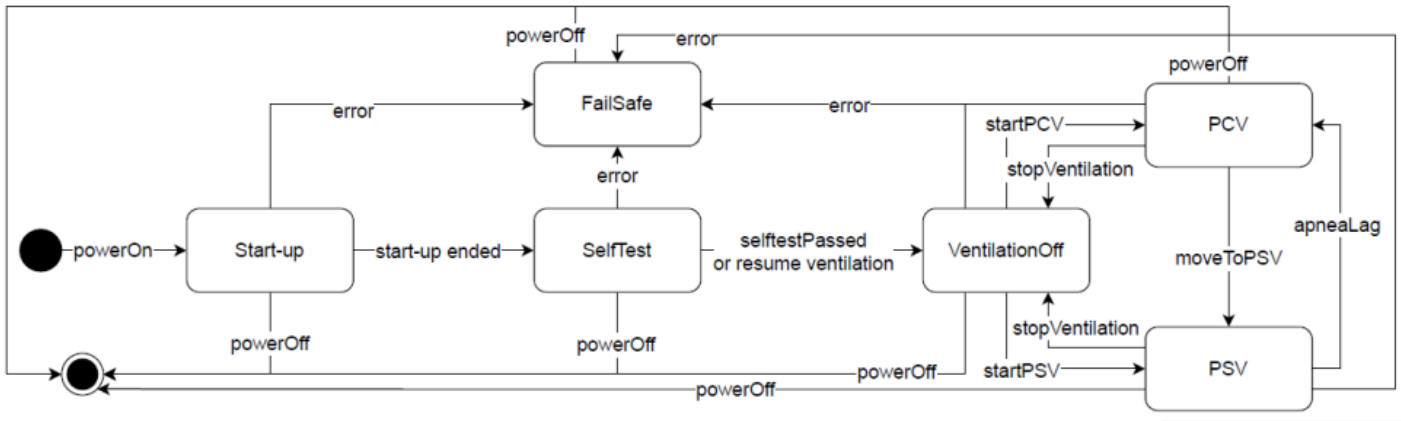


Main functionalities of the controller depend on the orders received from the GUI

MODELING STRATEGY: 1/4

- Which component, GUI or the controller, should be modelled at first?





GUI functionalities is modeled before those of the controller

MODELING STRATEGY: 2/4

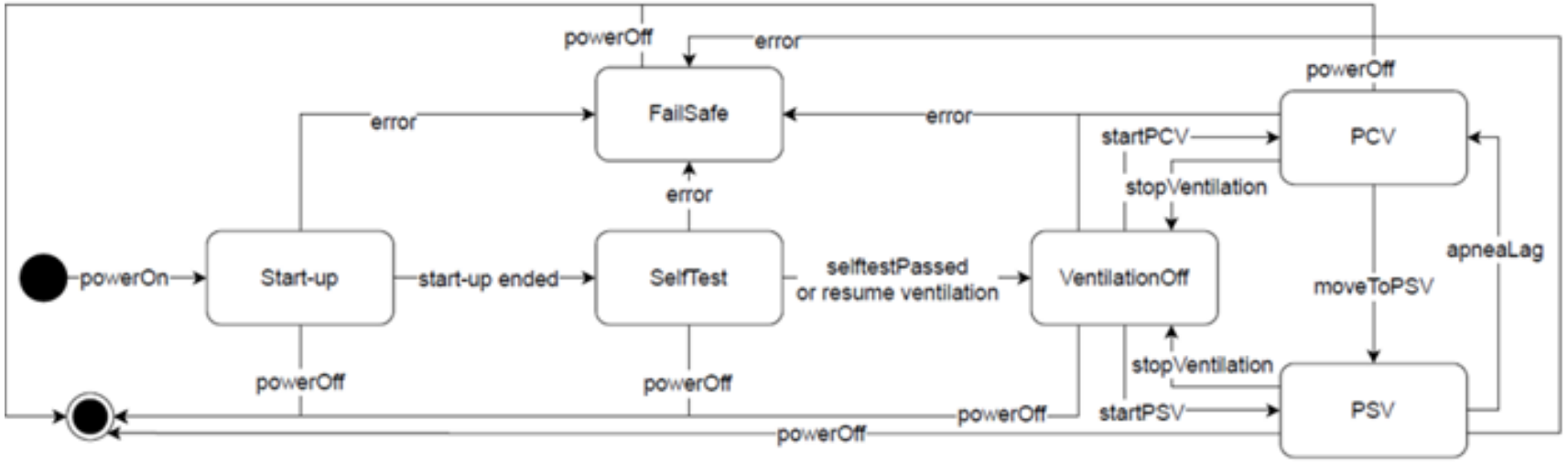
- Which level is the best to introduce time?

The behavior of the system may depend on the internal battery whose level depends on time progression

The level of the baterry is modelled in an early level with the usual time progress event

MODELING STRATEGY: 3/4

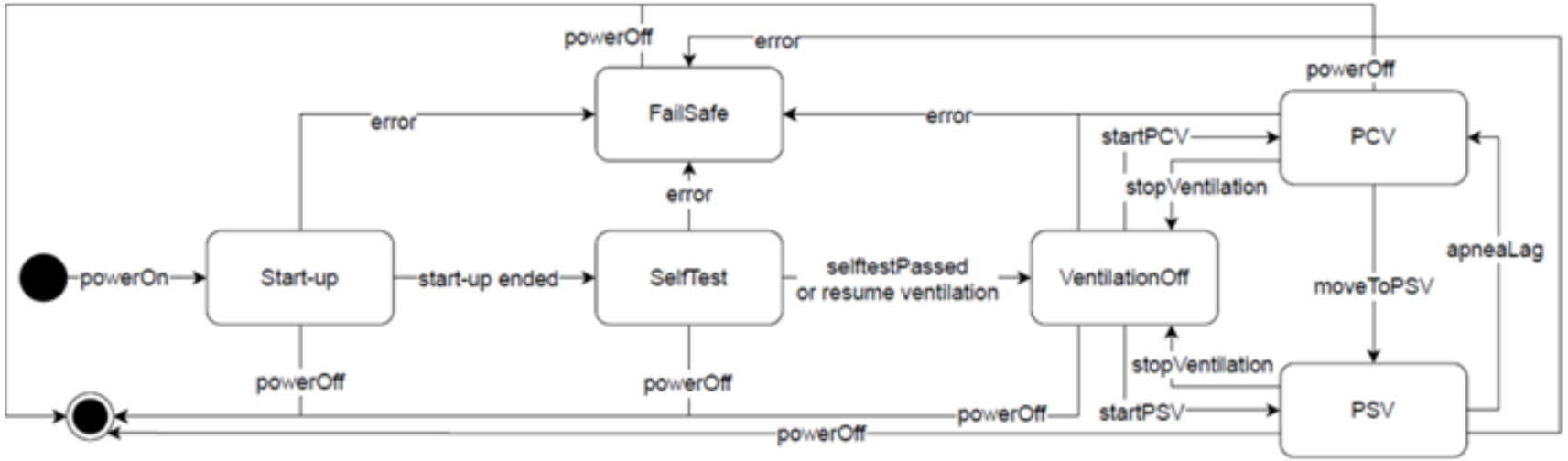
- Can the controller event error be dealt with like other ever



  - an error may be due to several causes (valve failure, backup battery failure, etc.)
  - an error may be raise at any moment

MODELING STRATEGY: 3/4

- Can the controller event error be dealt with like other ever



Event error is modeled as a refinement of the event progress

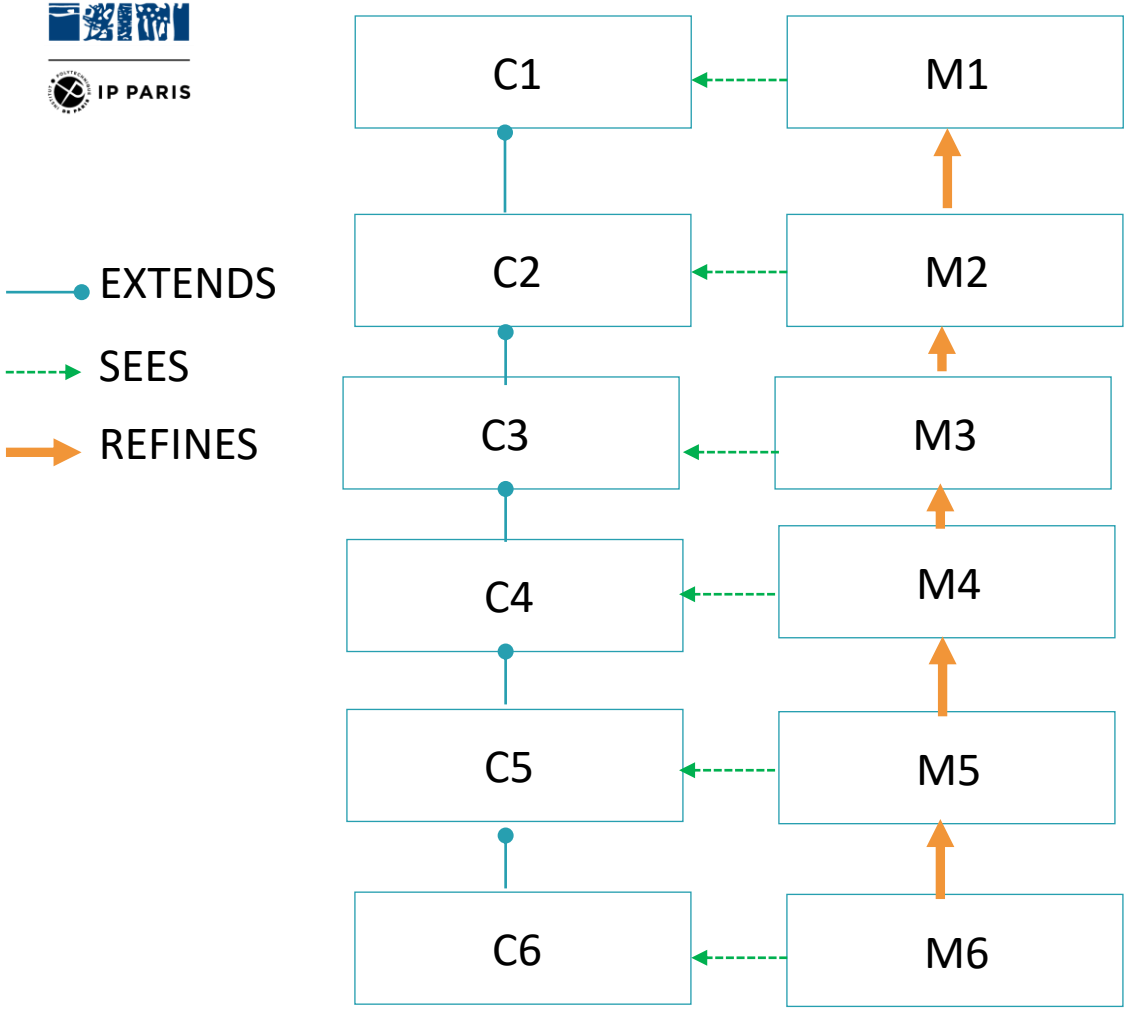
MODELING STRATEGY: 4/4

- Can the transitions common to the GUI and the controller (powerOn, powerOff, start-up ended, etc.) be represented by a single event?

Transitions with the same effect: a single Event-B event (powerOn, powerOff)

Transitions with different effects: distinct Event-B events (start-up ended)

EVENT-B MODEL STRUCTURE



GUI functionalities

Controller functionalities

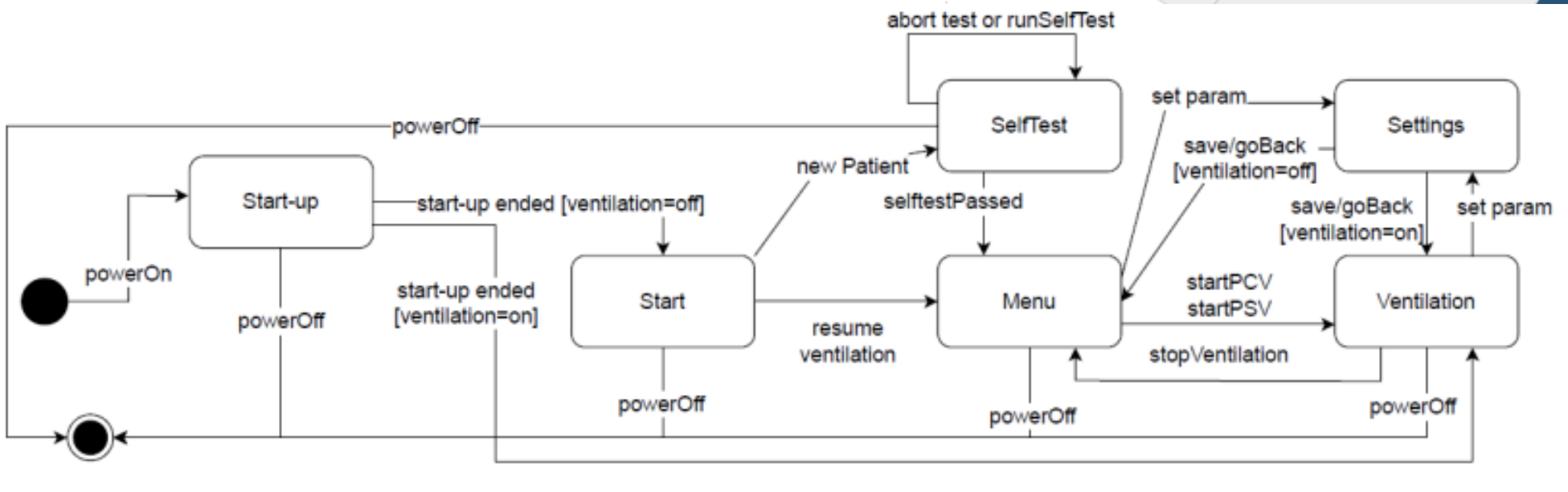
Ventilation modes: PCV, PSV

Ventilation phases: inspiration, expiration, etc.

Valves

Alarms

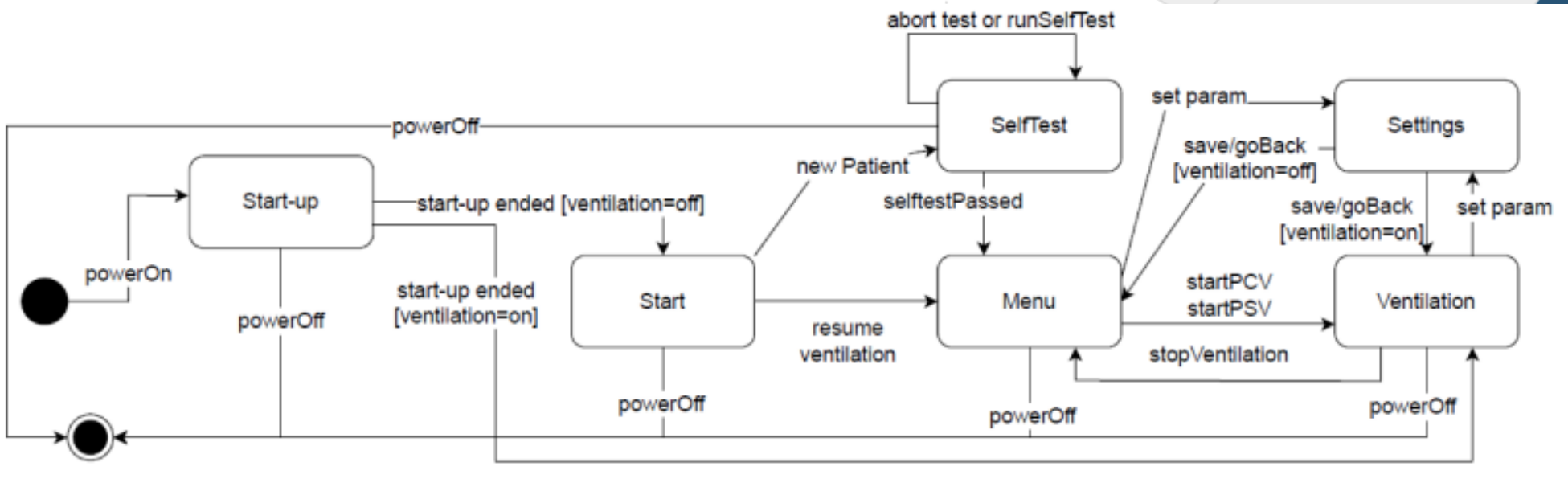
1ST LEVEL: GUI FUNCTIONALITIES (M1+C1)



In the contexte C1:

$$\text{axm1: } \mathbf{partition}(Ventilation, \{PCV\}, \{PSV\})$$
$$\text{axm2: } \mathbf{partition}(ModesG, \{\texttt{StartUp}\}, \{Start\}, \{\texttt{Menu}\}, \{\texttt{SelfTest}\}, \{\texttt{Settings}\}, \texttt{Ventilation}, \{\texttt{Off}\})$$

1ST LEVEL: GUI FUNCTIONALITIES (M1+C1)



In the machine M1:

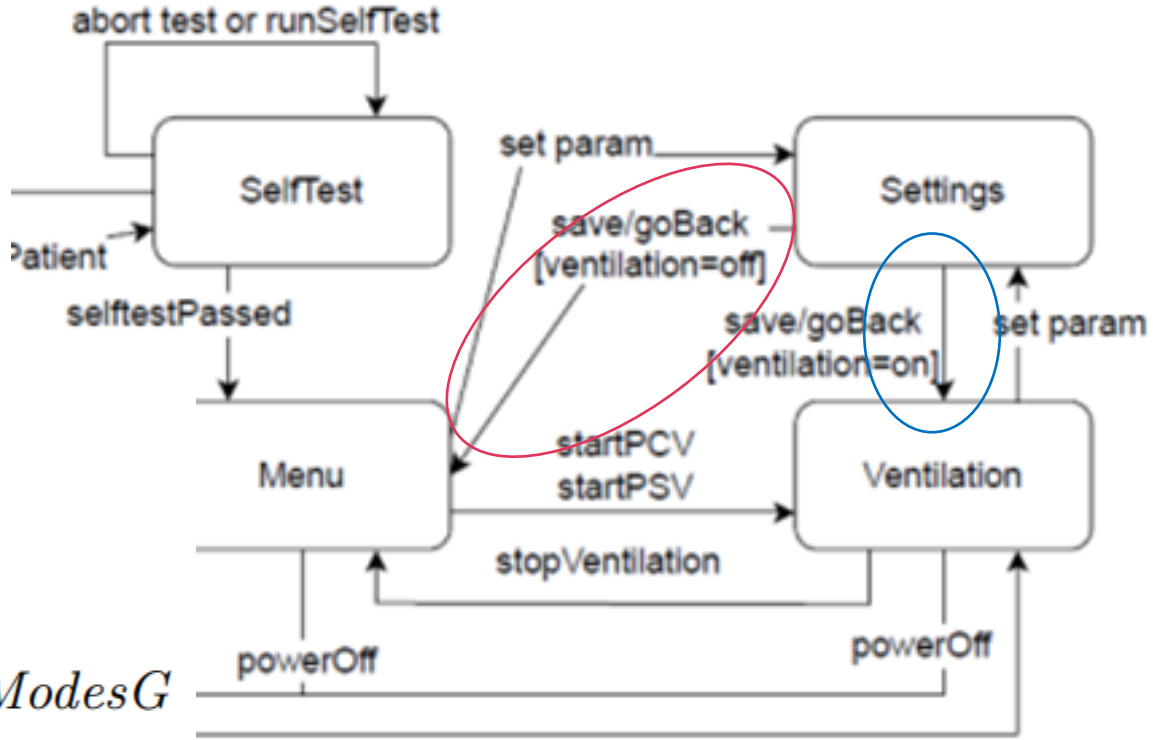
$$
\begin{aligned}
&modeGP \neq modeG \Rightarrow \\
&\quad modeGP \mapsto modeG \in \\
&\qquad possTransG(\mathbf{bool}(power{=}\mathbf{TRUE} \wedge crashed = \mathbf{FALSE} \wedge \\
&(onAC = \mathbf{TRUE} \vee (switchover = \mathbf{TRUE} \wedge batLev > 0 \wedge batFail{=}\mathbf{FALSE})))
\end{aligned}
$$

1ST LEVEL: GUI FUNCTIONALITIES (M1+C1)

In the machine M1:

- Transitions with a same label give an Event-B event

Event saveBackAbort $\widehat{=}$
  any
    $modeg$
  where
    grd1: $modeG = \texttt{Settings} \wedge modeg \in ModesG$
    grd2: $modeg \in Ventilation \cup \{\texttt{Menu}\}$
  then
    act1: $modeG := modeg$
    act2: $modeGP := modeG$
  end



***modeGP*: previous state of the GUI**
***modeG*: current state of the GUI**
***modeg*: next state of the GUI**

2ND LEVEL: CONTROLLER FUNCTIONALITIES (M2+C2)

In the machine M2:

Event saveBackAbort $\widehat{=}$

any

$modeg$

where

grd1: $modeG = \texttt{Settings} \wedge modeg \in ModesG$

grd2: $modeg \in Ventilation \cup \{\texttt{Menu}\}$

$modeC \neq FailSafe$

$modeC \in Ventilation \Rightarrow modec \in Ventilation \wedge modeg = modec$

$modeC \notin Ventilation \Rightarrow modec = modeC \wedge modeg = Menu$

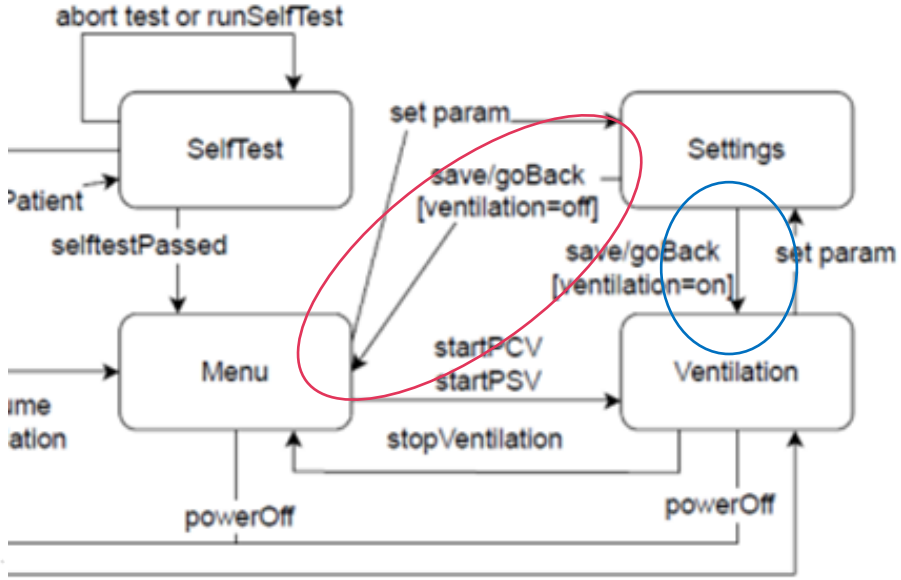
then

act1: $modeG := modeg$

act2: $modeGP := modeG$

...

end



***modeC*: current state of the controller**

***modec*: next state of the controller**

3RD LEVEL: VENTILATION MODES (M3+C3)

$$psvParamsValC \in 0..curTime \pfun (psvParams \pfun \mathbb{N}_1)$$

$$modeC = \mathrm{PSV} \Rightarrow \mathbf{dom}(psvParamsValC(\mathbf{max}(\mathbf{dom}(psvParamsValC)))) = psvParams$$

- ***Mode change request***

$$PCV2PSV = \mathbf{TRUE} \Rightarrow modeG = \mathtt{Settings} \land modeC \in \{\mathtt{PCV}, \mathtt{FailSafe}\}$$

Event saveBackAbort $\widehat{=}$ saveBackAbort

any

$modeg$, $modec$, sv, $psvC$, $psvG$,...

where

grd1: ...

...

grd6: $PCV2PSV = \mathbf{TRUE} \land modeC = \mathrm{PCV} \Rightarrow modec = \mathrm{PSV}$

grd7: $PCV2PSV = \mathbf{FALSE} \lor modeC \neq \mathrm{PCV} \Rightarrow modec = modeC$

grd8: $psvG = \{\mathbf{TRUE} \mapsto psvParamsValG(\mathbf{max}(\mathbf{dom}(psvParamsValG))),$
$\mathbf{FALSE} \mapsto psvParamsValC(\mathbf{max}(\mathbf{dom}(psvParamsValC)))\}(sv)$

grd9: $psvC = \{\mathbf{TRUE} \mapsto psvParamsValG(\mathbf{max}(\mathbf{dom}(psvParamsValG))),$
$\mathbf{FALSE} \mapsto psvParamsValC(\mathbf{max}(\mathbf{dom}(psvParamsValC)))\}(sv)$

grd10: $modec = \mathrm{PSV} \Rightarrow \mathbf{dom}(psvC) = psvParams$

then

act1: ...

...

act5: $psvParamsValG(curTime) := psvG$

act6: $psvParamsValC(curTime) := psvC$

act7: $PCV2PSV := \mathbf{FALSE}$

end

4TH LEVEL: VENTILATION PHASES (M4+C4)

In C4: axm1: **partition**($ventSates$,{inspBeg},{inspEnd},{expBeg},{expEnd},...)

In M4:

inv1: $cycleMode \in cycles \rightarrow Ventilation$

inv2: $ventilPhase \in cycles \rightarrow ventSates$

inv3: $inspBegT \in cycles \rightarrow \mathbb{N}$ inv4: $inspEndT \in cycles \rightarrow \mathbb{N}$

Each breathing phase

➔

2 events: *start* and *end*

Event inspStart $\widehat{=}$

any

cy, $inspT$

where

grd1: $modeC \in$ **Ventilation** $cy \in Cycles \setminus cycles$

...

then

act1: $cycles := cycles \cup \{cy\}$

act2: $ventilPhase(cy) :=$ **inspBeg**

act4: $inspBegT(cy) := curTime$

act5: $inspEndT(cy) := inspT$

end

5TH LEVEL: VALVES (M5+C5)

inv1: $(\exists\, c.\ c \in cycles \wedge ventilPhase(c) \in \{\texttt{inspBeg}, \texttt{inspEnd}\}) \wedge inValveF = \mathbf{FALSE} \wedge modeC = \texttt{Ventilation} \Rightarrow inValve = \mathbf{TRUE}$

Event inspStart $\widehat{=}$

any

$cy,\ inspT$

where

grd1: $modeC \in \texttt{Ventilation}\ cy \in Cycles \setminus cycles$

...

then

act1: $cycles := cycles \cup \{cy\}$

act2: $ventilPhase(cy) := \texttt{inspBeg}$

act4: $inspBegT(cy) := curTime$

act5: $inspEndT(cy) := inspT$

$inValve := \{\mathbf{FALSE} \mapsto \mathbf{TRUE}, \mathbf{TRUE} \mapsto inValve\}(inValveF)$

end

6TH LEVEL: ALARMS (M6+C6)

In C6: $\mathrm{axm1}: \mathbf{partition}(Alarms, \{\mathtt{guiFailure}\}, \{\mathtt{contFailure}\}, \{\mathtt{inValveFailure}\}, \{\mathtt{patConnected}\}, \ldots)$

In M6: $alarmRaised \in Alarms \rightarrow \mathbf{BOOL}$

6TH LEVEL: ALARMS (M6+C6)

$$\text{axm1: } \mathbf{partition}(Alarms, \{\texttt{guiFailure}\}, \{\texttt{contFailure}\}, \{\texttt{inValveFailure}\}, \{\texttt{patConnected}\}, \ldots)$$

$$\begin{aligned}
&alarmRaised(\texttt{inValveFailure}) = \mathbf{bool}(\exists\, c.\ (c \in cycles \wedge \\
&\quad ((((ventilPhase(c) = \texttt{inspBeg} \wedge curTime > inspBegT(c)) \vee \\
&\qquad (ventilPhase(c) = \texttt{rmBeg} \wedge curTime > rmBegT(c))) \wedge \\
&\qquad\qquad inValveP = \mathbf{FALSE} \\
&\quad)) \vee \\
&\quad (((ventilPhase(c) = \texttt{expPauseBeg} \wedge curTime > expPauseBegT(c)) \vee \\
&\quad (ventilPhase(c) = \texttt{expBeg} \wedge curTime > expBegT(c)) \vee \\
&\quad (ventilPhase(c) = \texttt{inspPauseBeg} \wedge curTime > inspPauseBegT(c))) \wedge \\
&\qquad\qquad inValveP = \mathbf{TRUE})))
\end{aligned}$$

6TH LEVEL: ALARMS (M6+C6)

Event progress $\widehat{=}$
refines progress
any
$step, \ldots$
where
grd1: $step \in \mathbb{N}1 \land \cdots$
grd2: $modec \in \{\texttt{FailSafe}, modeC, \texttt{Off}, \texttt{StartUp}\}$
$\cdots$
then
act1: $curTime := curTime + step$
act4: $modeC := modec$
$\cdots$
end

Event progress $\widehat{=}$
refines progress

any

$step, \ldots$

where

grd1: $step \in \mathbb{N}1 \land \cdots$

grd2: $modec \in \{\texttt{FailSafe}, modeC, \texttt{Off}, \texttt{StartUp}\}$

$\cdots$

$alarmInV = \textbf{bool}(\exists\, c.\ (c \in cycles \land ($
$(ventilPhase(c) \in \{inspBeg, inspEnd, rmBeg, rmEnd\} \land inValve = \textbf{FALSE})$
$\lor$
$(ventilPhase(c) \in \{expPauseBeg, inspPauseBeg, inspPauseEnd\} \land$
$inValve = \textbf{TRUE}))))$

$alarmInV = \textbf{TRUE} \Rightarrow modec = \texttt{FailSafe}$

then

act1: $curTime := curTime + step$

act4: $modeC := modec$

$alarmRaised := alarmRaised \lhd\!\!+ \{\ldots, inValveFailure \mapsto alarmInV\}$

$inValveP := inValve$

$\cdots$

end

Event progress $\widehat{=}$
refines progress

any

$step, \ldots$

where

grd1: $step \in \mathbb{N}1 \land \cdots$

grd2: $modec \in \{\texttt{FailSafe}, modeC, \texttt{Off}, \texttt{StartUp}\}$

$\cdots$

$alarmInV = \textbf{bool}(\exists\, c.\ (c \in cycles \land ($
$(ventilPhase(c) \in \{inspBeg, inspEnd, rmBeg, rmEnd\} \land inValve = \textbf{FALSE})$
$\lor$
$(ventilPhase(c) \in \{expPauseBeg, inspPauseBeg, inspPauseEnd\} \land$
$inValve = \textbf{TRUE}))))$

$alarmInV = \textbf{TRUE} \Rightarrow modec = \texttt{FailSafe}$

then

act1: $curTime := curTime + step$

act4: $modeC := modec$

$alarmRaised := alarmRaised \lhd\!\!+ \{\ldots, inValveFailure \mapsto alarmInV\}$

$inValveP := inValve$

$\cdots$

end

WHAT HAS BEEN DONE …

- Different states/transitions of the controller and the GUI
- Ventilation modes (PCV and PSV) and the switching from one to the other
- Ventilation parameters and their update before and during the ventilation
- Position of the valves (in and out) during the ventilation and their failures
- Alarms: the valves (in and out), patient connection while the system is in the state StartUp, the backup battery, the switchover, the FI1/FI12/oxygen sensors.

SOME STATISTICS

- Three months development/proof
  - 6 refinement levels
  - 44 variables + 90 invariants +35 events
- Models are proved and verified
  - Model checking: ProB for detecting obvious invariant violations
  - Validation: ProB on our own scenarios
  - Proof obligations correctness
    - 1322 proof obligations
      - 312 automatic (23%)
      - 225 interactive

FEEDBACKS ON THE SPECIFICATION DOCUMENT

- Under which conditions the controller must move to the state *FailSafe*?
- Does the controller continuously check the presence of undesirable events or not?
- Does the controller continuously check the communication with the GUI or only in the state *StatUp*?
- What is the impact of the alarms' levels on the system?