CLEARSY
Safety Solutions Designer

Hackathon
JUL2025

AIX
LYON
PARIS
STRASBOURG

WWW.CLEARSY.COM

Airlock
Specifying and implementing a Voter

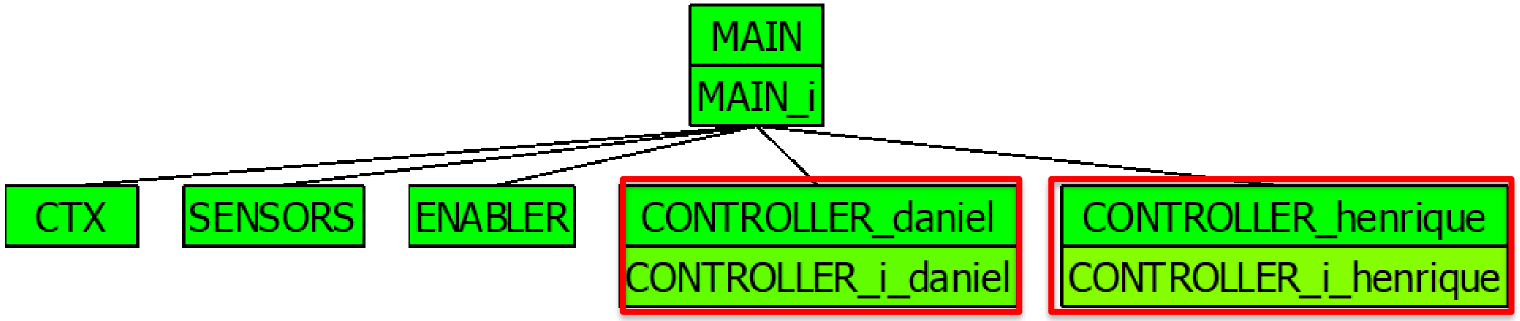
PART V

Thierry Lecomte
R&D Director

THIERRY.LECOMTE@CLEARSY.COM

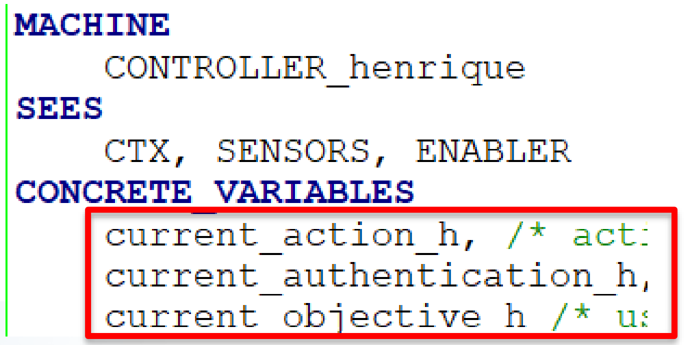
Your turn

- *Implement VOTER in MAIN_I [10 pt]*
  - *CONTROLLER duplicated (thank you Daniel and Henrique !)*

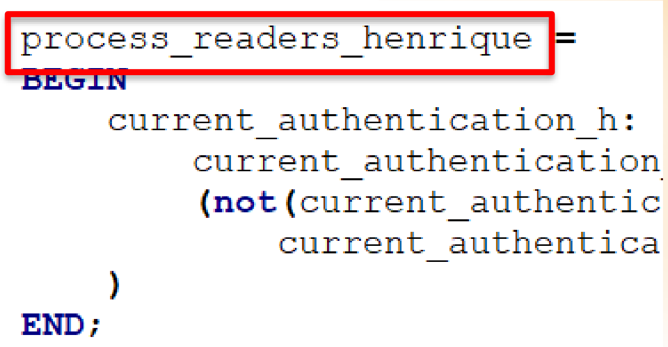


Your turn

- *Implement VOTER in MAIN_I*
  - *CONTROLLER duplicated (thank you Daniel and Henrique !)*
  - *Variables and operations renamed to avoid collisions*



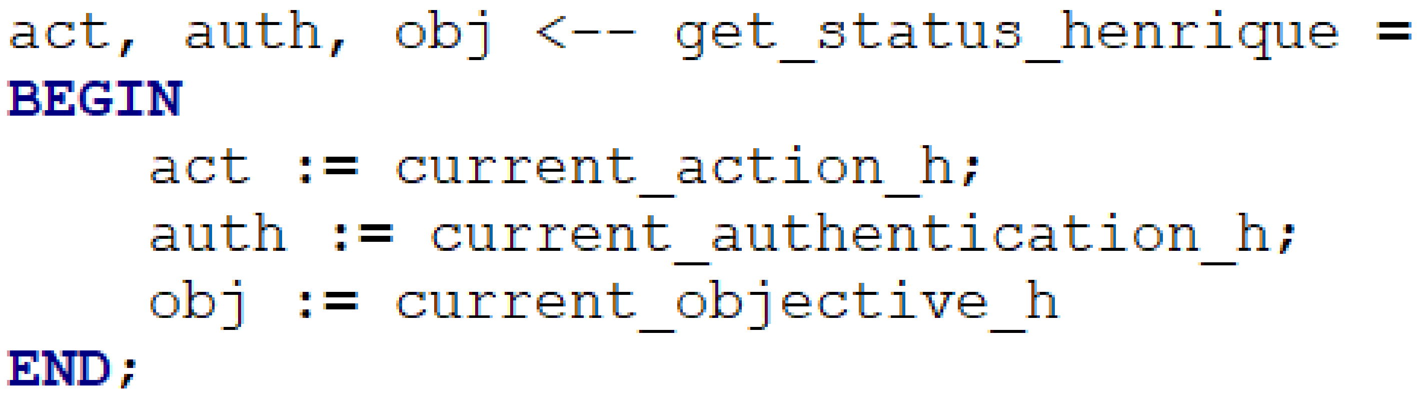
```
MACHINE
    CONTROLLER_henrique
SEES
    CTX, SENSORS, ENABLER
CONCRETE_VARIABLES
    current_action_h, /* act:
    current_authentication_h,
    current_objective_h /* u:
```



```
process_readers_henrique =
BEGIN
    current_authentication_h:
        current_authentication
        (not(current_authentic
            current_authentica
    )
END;
```

Your turn

- *Implement VOTER in MAIN_I*
  - *Creation of accessor operation*



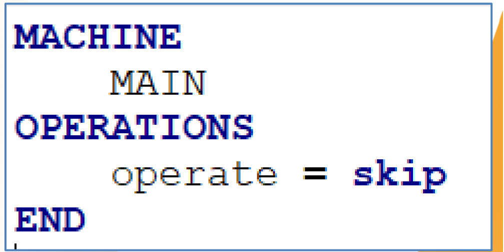
```
act, auth, obj <-- get_status_henrique =
BEGIN
    act := current_action_h;
    auth := current_authentication_h;
    obj := current_objective_h
END;
```

Introducing a Voter

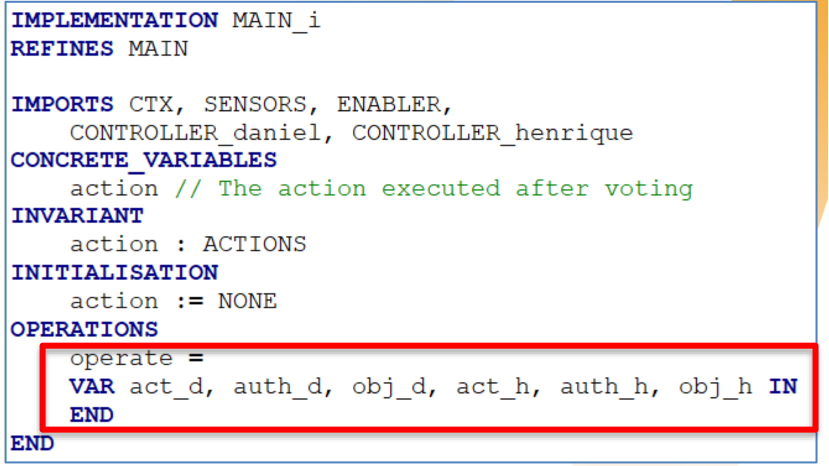
- *Developping a functional component*
  - *that matches exactly the specification of an enabler component is tricky*
  - *You need to give lot of details in the controller specification*

- *Using a voter simplifies the process*
  - *Two independent functions have to provide simultanenously the same permissive output to enable that permissive output (open door A and B)*
  - *The drawback is that we cannot assess the functionality of the controller (blackbox)*

Implementing a voter



```
MACHINE
    MAIN
OPERATIONS
    operate = skip
END
```



```
IMPLEMENTATION MAIN_i
REFINES MAIN

IMPORTS CTX, SENSORS, ENABLER,
    CONTROLLER_daniel, CONTROLLER_henrique
CONCRETE_VARIABLES
    action // The action executed after voting
INVARIANT
    action : ACTIONS
INITIALISATION
    action := NONE
OPERATIONS
    operate =
    VAR act_d, auth_d, obj_d, act_h, auth_h, obj_h IN
    END
END
```

- *Similar to previous testing component*
- *Variable* action *that is the action to be really executed*
- *operate* *update the values of the sensors, process readers on both controllers, control on both controllers, get the 3 variables from both controllers*
- *If 3 variables equal on both controllers, then confirms the action.*
- *If not, select restrictive action*

Your turn

- ***Optional**: What happens to the whole system if the controllers are functionally different? [1 pt]*
- *Optional: is it possible to adapt the voting principle to make it a bit more useful? [1 pt]*