TSAMI:

Tango StAte Machine Interpreter

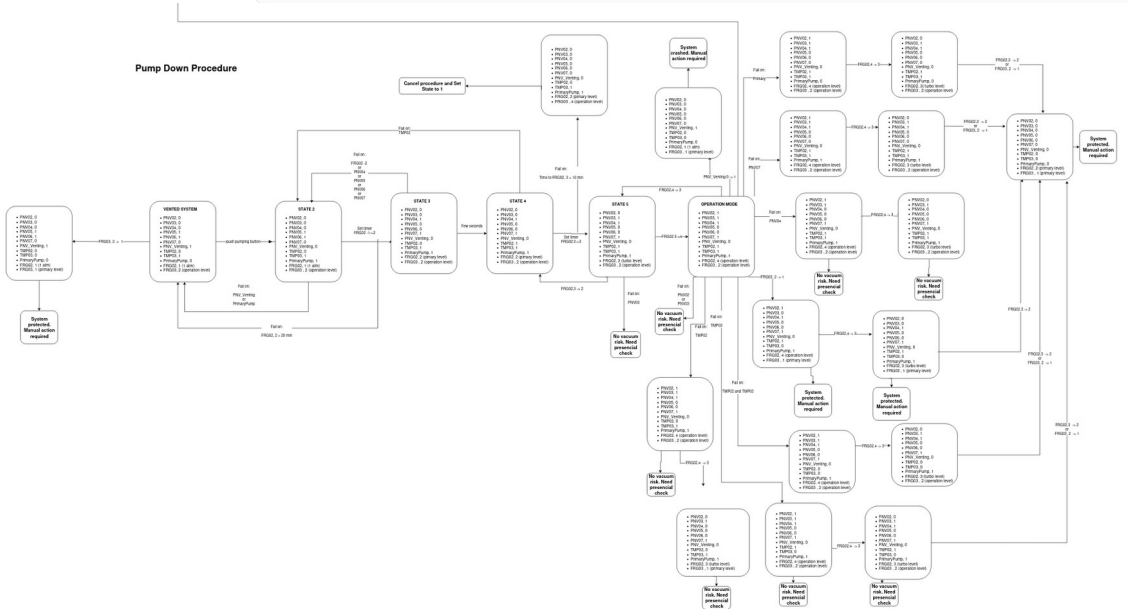
Albert Olle (aolle@cells.es)

30/05/2024

Table of contents

The Problem

- We were given a big state machine to implement a vacuum system.
  - States: ~45
  - Edges: ~110
  - Inputs: ~20



The Problem

- We were given a big state machine to implement a vacuum system.
- It had to constantly be running to listen for physical button inputs.
- Subject to a fair ammount of revisions by BL scientists.

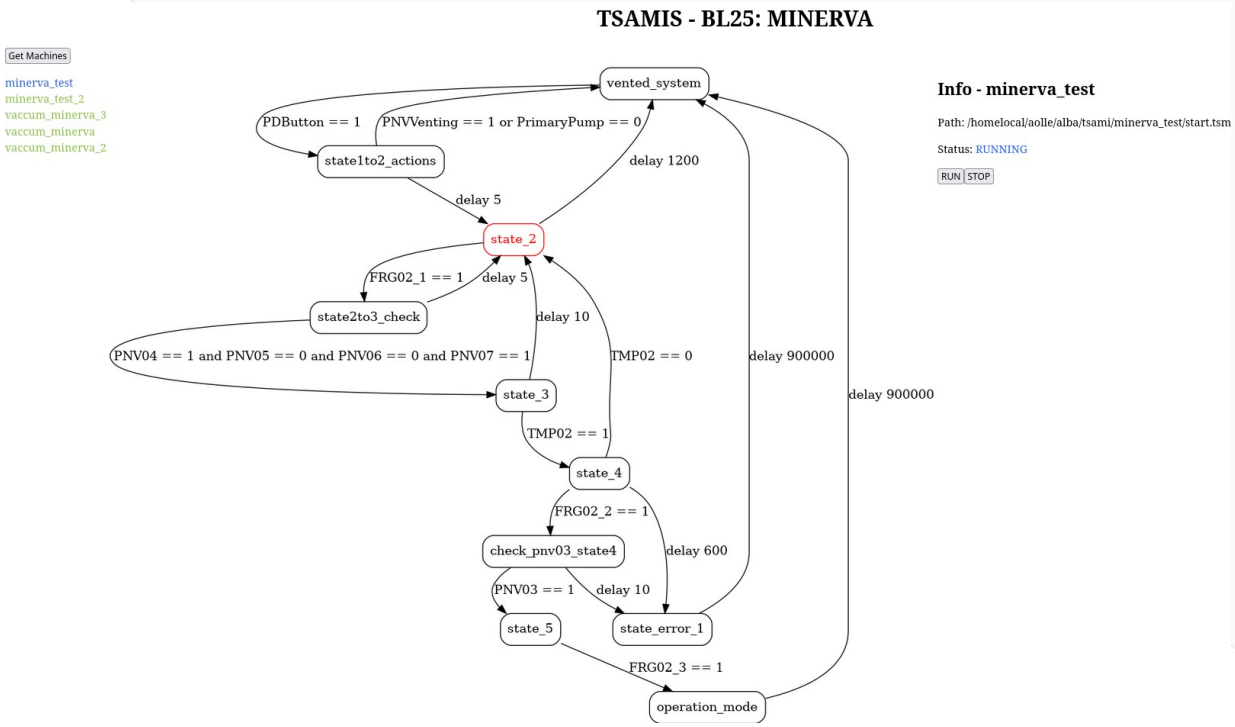
Possible solutions explored

- A bunch of “if” statements:
  - No dependencies, low extrinsic complexity
  - Very little modularity, low maintaniability, not adaptable to other use cases
- Sardana macros:
  - Familiar system, widely used, capable of runing python scripts
  - Would require some development to the core
- Panic:
  - Can act on alarm raised
  - Can only do one action per alarm, can grow very complex
- EPICS’ SNL:
  - Exactly what we want
  - Doesn’t exist for tango and porting it wasn’t the optimal solution

Introducing: TSAMI

```
state vented_system {
  when (PDButton == 1){
  } goto state1to2_actions
  when (SET_POINT1 < 53){
  } goto operation_mode
}

state move_to_state_2 {
  entry {
    set PNVVenting 0;
    set PrimaryPump 1;
  }
  /* If after 5 seconds they are not in the position we
  wanted then go back to vented system state */
  when (PNVVenting == 1 or PrimaryPump == 0) {
    set PNVVenting 1;
    set PrimaryPump 0;
  } goto vented_system
  when (delay 5) {
  } goto state_2
}
```



The Syntax

```
name "vacuum_minerva";
```

Name for the state machine.

```
assign {
  FRG02_1:DevLong64 "testds/eps/1/Value";
  FRG02_2:DevLong64 "testds/eps/2/Value";
  PDButton:DevLong64 "testds/eps/3/Value";
  PNVVenting:DevLong64 "testds/eps/4/Value";
  PrimaryPump:DevLong64 "testds/eps/5/Value";
  TMP02:DevLong64 "testds/eps/6/Value";
  SET_POINT:float 17.546;
}
```

Define variables and assign them to tango attributes. It also supports simple variables.

```
safe_start {
  PNV02 is 0;
  PNV03 is 0;
  TMP02 is 0;
  FRG02_1 is 0;
  FRG02_2 is 1;
  PNVVenting is 1;
  PrimaryPump is 0;
} goto vented_system
```

We can define a state in which we should be to start safely running the state machine and which is the starting state.

The Syntax (II)

```
state state1to2_actions {

  entry {
    set PNVVenting 0;
    set PrimaryPump 1;
  }
  /* I am a comment */
  when (PNVVenting == 1 or PrimaryPump == 0) {
    set PNVVenting 1;
    set PrimaryPump 0;
  } goto vented_system
  when (delay 5) {
  } goto state_2
}
```

Define a new state

Actions to be done upon entering state, optional.

Defining exit edges with conditons and Actions to be taken when condition is met before going to next state.

Option to set a timeout

The DEMO

- Improve functionallity of the WebUI:
  - Override safe start and inital state
  - More visibility into state changes, logs
- Define an interaction model so that state machines can communicate with each other and the webUI.
- Join the web server and the rest API into one.
- Improve deployment
- Contributions welcome Here!
  (https://gitlab.com/tango-controls/device-servers/DeviceClasses/generic/tsami)

Thanks for your attention
Questions?