


ASDP - FESTO Project



A.A. 2022-2023

Simone Cenerini
Giulia Cutini
Marco Sartoni
Daniele Simonazzi

Program Structure (1)

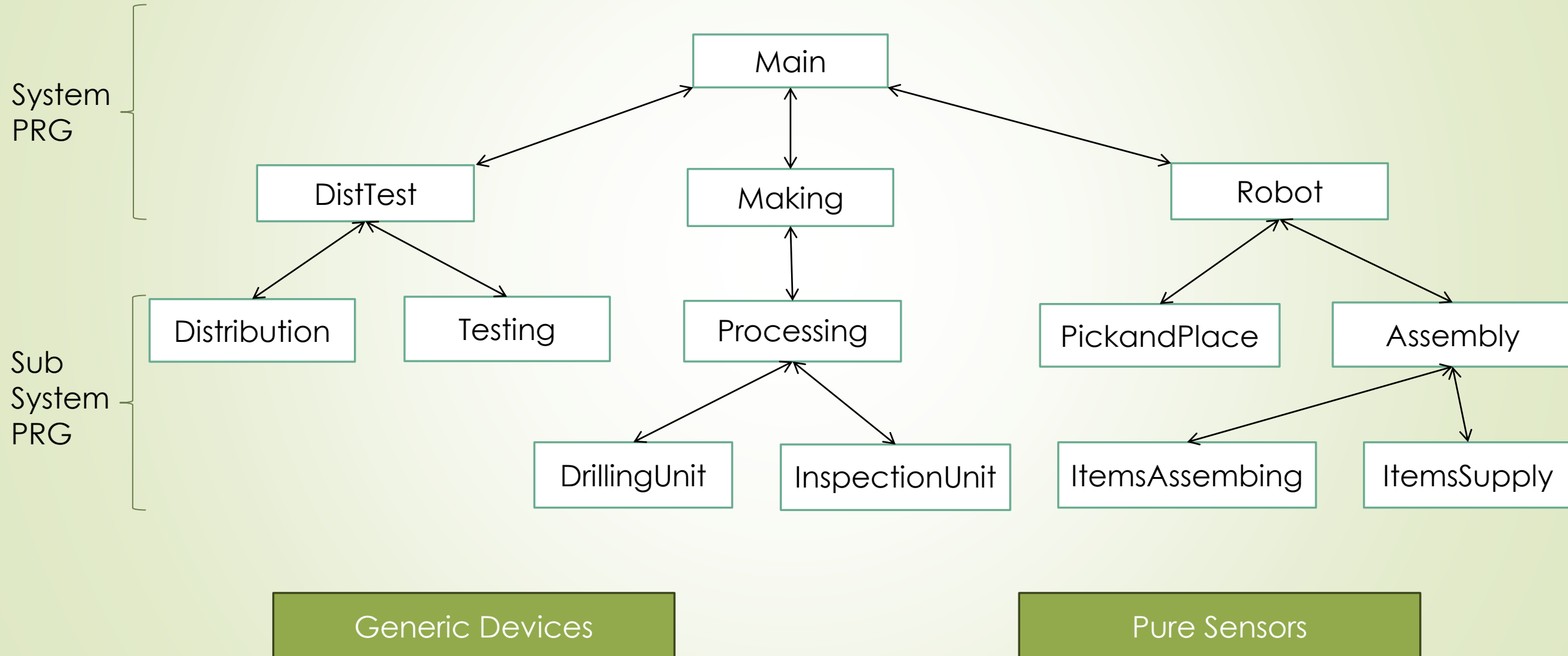
The machine splitted into 4 physical stations: Distribution, Testing, Processing and Assembly.

The machine program is splitted in 3 control programs (Distribution&Testing, Making and Robot), which are controlled by a Main which handles machine initialization, running and stop.

The controls programs act on subsystem programs, which manage the physical machine stations. The assembly station is splitted into 2 subsystems (Assembly and PickandPlace). Moreover some Processing and Assembly actions are splitted into further subsystems programs (DrillingUnit, InspectionUnit, ItemsAssembling and ItemSupply).

All subsystem programs manages directly their generic devices (by enable and disable requests) and have access to their sensors.

Program Structure (2)



Handlers

- **System_Handler:**

- Initialize
- Run
- ImmediateStop

Used to initialize, start and stop (immediately) the system.

Command typology: Start & Stop

- **Subsystem_Handler:**

- Initialize
- Enable
- Disable
- ImmediateStop

Used by the System_PRGs to initialize, enabling and disabling subsystems

Command typology: Do & Done (acknowledgement)

- **Data_Handler:**

- ID
- Colour
- Height
- Orientation
- Discard

Used as memory structure.

The global system memory is an array[1..8] of Data_Handler (one element for each possible piece position in the machine).

Each subsystem has a local Data_Handler for managing data

Memory

As already seen, the memory structure is made by Data_Handler.

Each piece handled in the machine has the following information stored into the memory array:

- ID (INT) : piece unique identifier
- Colour (BOOL): TRUE if the piece is grey or red, FALSE if the piece is black
- Height (BOOL): TRUE if the piece is tall, FALSE if the piece is short
- Orientation (BOOL): TRUE if the piece is correctly oriented, FALSE otherwise
- Discard (BOOL): set to TRUE if the piece has to be discarded, otherwise FALSE

Machine Operation

The machine can be in the following operation states:

- **Ready_to_initialize** : after being powered on, it is waiting for the INIT button to be pressed
- **Initializing**: it is performing initialization
- **Ready_to_Run**: initialization completed, it is waiting for the START button to be pressed
- **Running**
- **ImmediateStopping**: it is performing or has completed an Immediate Stop
- **OnPhaseStopping**: it is performing or has completed an On Phase Stop

We have implemented two kinds of stops:

- **Immediate Stop**: stop as soon as possible. Interruption of all subsystems FSMs
- **On Phase Stop**: stop each subsystem after it has completed its current operation (thus its FSM has reached the 'Ready_to_enable' state, while machine, while system PRGs will be in 'Ready_to_run' state)

Implemented Libraries

- **Generic_Device (FB):** generic device Finite State Machine management, diagnosis information generation (faults)
- **Signal_Filter (FB):** implementing debouncing for buttons and filtering sensor signals
- **Memory Libraries (Functions):**
 - **Save_Data:** saving data from the local subsystem memory into the main system memory
 - **Shift_Data:** shifting to the next position an element of the main system memory
 - **Testing_Colour:** test if the colour and height are coherent and thus correct
 - **Testing_Orientation:** test if the orientation is correct

Signal Management

In order to manage faults and signal generation we have implemented:

- **SignalManagement (FB)**: signal management library (device independent)
- **SignalControl_PRG**: signal control (configuration and generation, device dependent)

We have configured and used the following signal categories:

- **Alarm** : for actuator and sensor faults, which will cause an Immediate Stop
- **Warnings**: for empty warehouses, with AUTO_RESET, which won't stop the machine.

For alarms we have implemented AUTO_CONDITIONED_RESET, which automatically reset the signal if the cause has been repaired and there still are other alarms active. Thus, you will have to press the RESET button only once, when the last alarm cause will be solved.

A light on the RESET button states that you can reset the system to the nominal condition. To start again the system you have to press the START button

Bridges

We have implemented a bridge program for each subsystem:

- Bridge_Distribution_GDs
- Bridge_Testing_GDs
- Bridge_Drilling_GDs
- Bridge_Inspection_GSs
- Bridge_Processing_GDs
- Bridge_Assembly_GDs
- Bridge_ItemsAssembling_GDs
- Bridge_ItemsSupply_GDs
- Bridge_PickandPlace_GDs

Each of these programs perform a bridge from the physical sensors and actuators to the logical ones and connect them to the proper generic device.

Each sensor is filtered.

We have also implemented:

- Buttons
- Lights

Which are programs that bridges physical buttons and LEDs to the logical ones.