1. PROJECT

ROOT_OBJECTS

--|examples_AADL/Common\Data_Model|--,

--|examples_AADL/Common\Base_Types|--,

--|examples_AADL/Common\Math|--,

--|examples_AADL\RedundantSystem|--

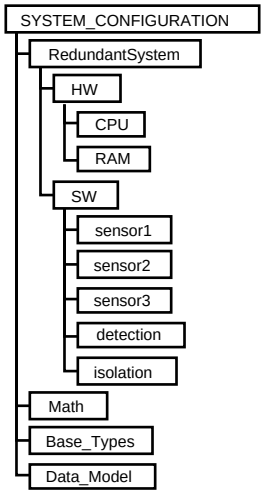
END

1.1. Project Description

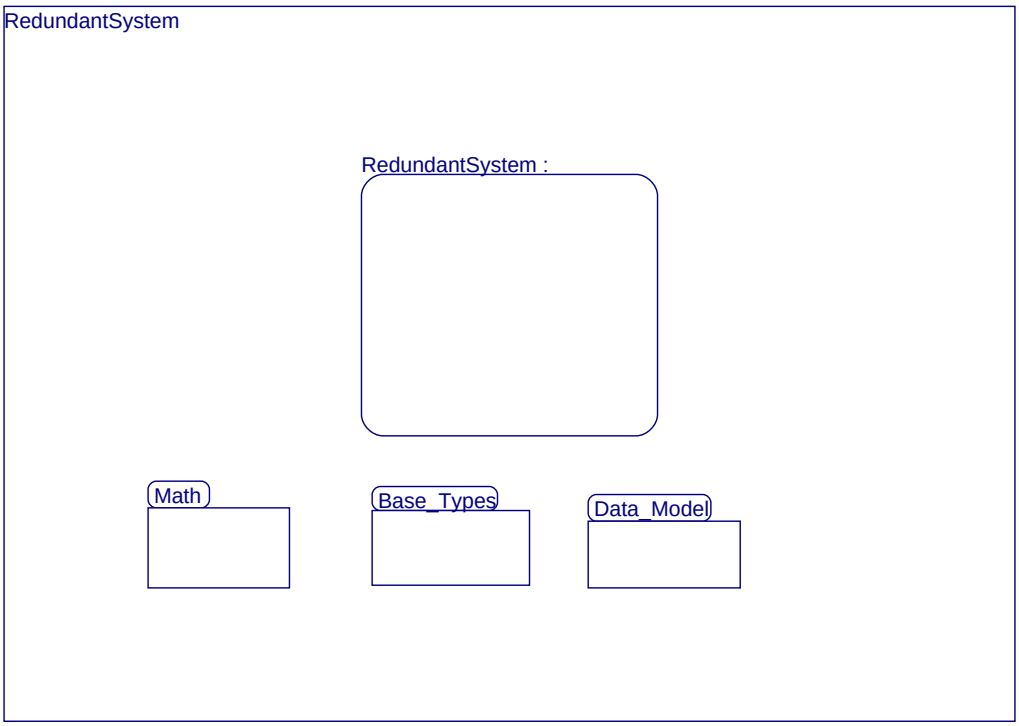
REDUNDANT SYSTEM

This model describes a redundant acquisition system composed of three identical sensors and a fault detection logic. The sensors take the same input value and a different random error value is added to each of them. A voting logic is applied in the detection function.
The sensor isolation function is not fully implemented.

1.2. Design Tree



1.3. AADL Diagram



2. SYSTEM RedundantSystem IS

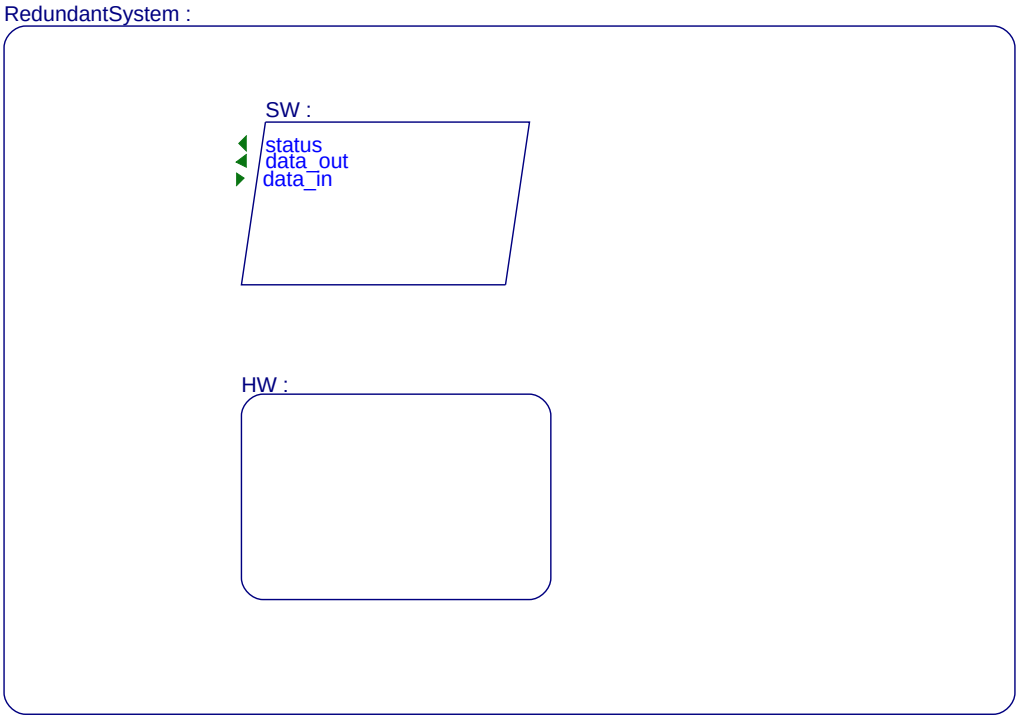
2.1. DESCRIPTION

2.1.1. PROBLEM

2.1.1.1. Statement of the Problem (text)

The redundant system is composed of a simplistic hardware platform onto which the acquisition software runs.

2.1.1.2. AADL Diagram



2.2. IMPLEMENTATION

2.2.1. BEHAVIOR

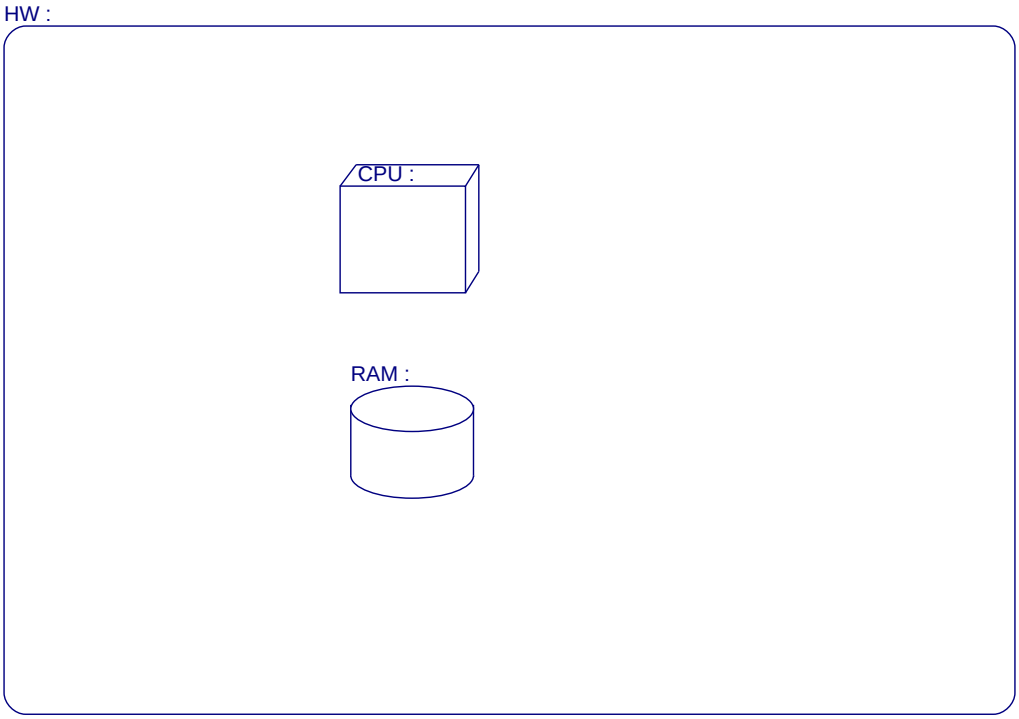
2.2.1.1. BEHAVIOR ANNEX

3. SYSTEM HW IS

3.1. DESCRIPTION

3.1.1. PROBLEM

3.1.1.1. AADL Diagram



3.2. IMPLEMENTATION

3.2.1. BEHAVIOR

3.2.1.1. BEHAVIOR ANNEX

4. PROCESS SW IS

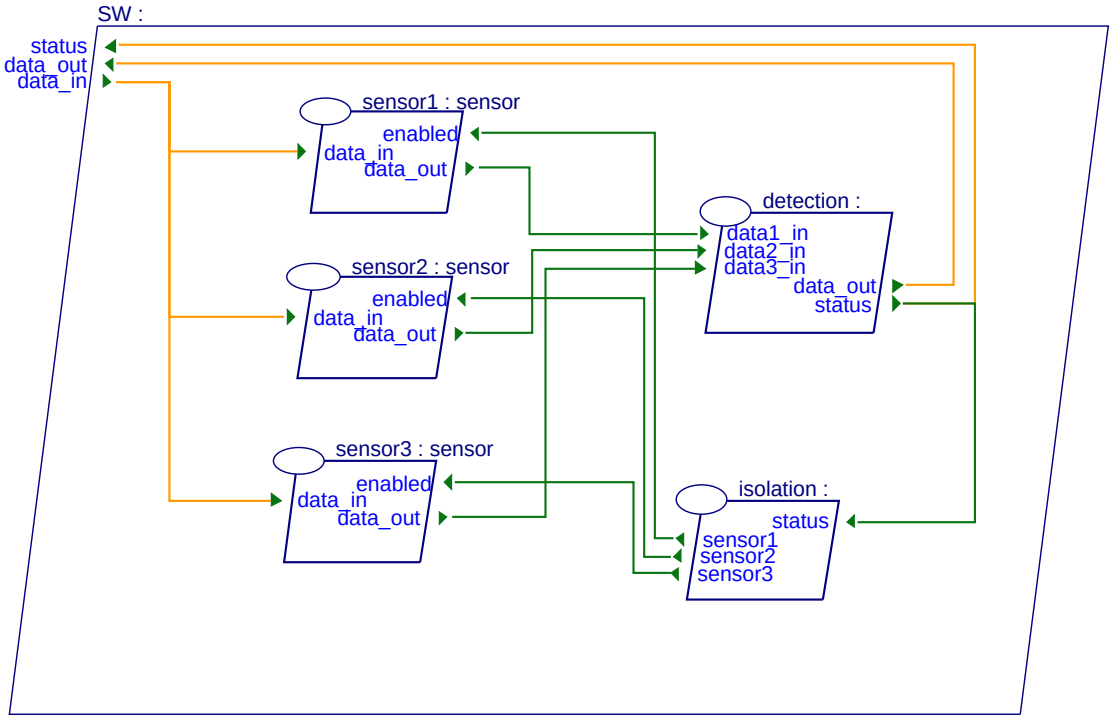
4.1. DESCRIPTION

4.1.1. PROBLEM

4.1.1.1. Statement of the Problem (text)

The acquisition software is composed of three identical sensors representing software device drivers, an error detection thread and a sensor isolation thread.
This AADL process takes the theoritical sensor measurement as input (data_in), and produces the best actual value (data_out) and an indicator of the reliability of this result (status).
- status = 0: no error, the three sensors give the same value
- status = 1: sensor 1 fails and sensors 2 and 3 give the same value
- status = 2: sensor 2 fails and sensors 1 and 3 give the same value
- status = 3: sensor 3 fails and sensors 1 and 2 give the same value
- status = -1: error, the three sensors give different values

4.1.1.2. AADL Diagram



4.2. IMPLEMENTATION

4.2.1. BEHAVIOR

4.2.1.1. BEHAVIOR ANNEX

5. THREAD sensor1 IS

5.1. DESCRIPTION

5.1.1. PROBLEM

5.1.1.1. Statement of the Problem (text)

Each sensor adds a random error to the input data to produce its output value.

5.2. IMPLEMENTATION

5.2.1. BEHAVIOR

5.2.1.1. BEHAVIOR ANNEX

5.2.1.1.1. Behavior Specification (aadl)

```
VARIABLES
  e : int;
STATES
  s : INITIAL COMPLETE FINAL STATE;
TRANSITIONS
  t : s -[ ON DISPATCH ]-> s {
  err!(2,e);
  IF ( enabled = 1 ) data_out := data_in + e END IF;
  IF ( enabled = 0 ) data_out := 0 END IF };
```

6. THREAD detection IS

6.1. DESCRIPTION

6.1.1. PROBLEM

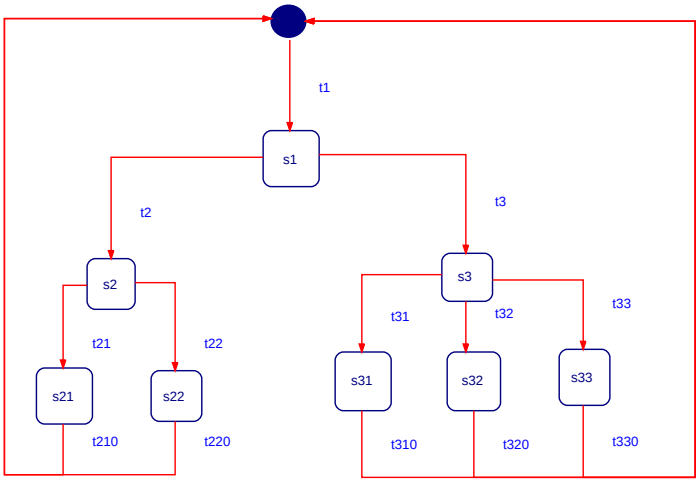
6.1.1.1. Statement of the Problem (text)

The error detection logic is described by a State Transition Diagram from which the AADL Behavior Annex structure is generated. Transition conditions and actions are added in the design sections to ensure a full automatic code generation.

6.2. IMPLEMENTATION

6.2.1. BEHAVIOR

6.2.1.1. State Transition Diagram



6.2.1.2. BEHAVIOR ANNEX

7. THREAD isolation IS

7.1. DESCRIPTION

7.1.1. PROBLEM

7.1.1.1. Statement of the Problem (text)

The isolation thread should disable the failing sensor.

This has not been implemented yet.

7.2. IMPLEMENTATION

7.2.1. BEHAVIOR

7.2.1.1. BEHAVIOR ANNEX

7.2.1.1.1. Behavior Specification (aadl)

STATES

```
  s : INITIAL COMPLETE FINAL STATE;
TRANSITIONS
  t : s -[ ON DISPATCH ]-> s {
  sensor1 := 1;
  sensor2 := 1;
  sensor3 := 1 };
```