RedundantSensors

Davide Marcantonio, Stefano Tonetta

March 17, 2014

Abstract

The proposed model represents a component that senses an input variable and handles the failure of the sensing function by using two redundant sensors. The modeling language used is OCRA [CDT13] and the model itself is inspired by an industrial case study developed within the ESA project FoReVer [FOR]. Beside the functional aspect it is considered an error model concerning the possibility of sensor failures in terms of their outcomes. The scope of the system is thus to guarantee a reliable output even in presence of such failures, under some assumptions. These assumptions concern the failure occurrences and the variance of the environmental input. Both sensor outcomes are monitored in order to detect and possibly isolate failures in order to determine the most proper action to maintain the output value reliable.

Contents

1 Modeling paradigm

The architecture is modeled in a compositional way: each component is defined in terms of its interface with input and output ports specification. The connections within components are defined linking the ports of the components pairing an input port with an output one. The model is then enriched with contracts that define an assume/guarantee reasoning on the I/O properties of components, i.e. some assumptions on the inputs, guaranteeing certain properties on output. Since each component has one or more contracts it is then possible to formalize and refine top level requirements along the architecture levels. A state machine can be associated to each leaf component (i.e. a component not further refined) as behavioral model. The behavior of a composite component (including the top-level component) is given by the composition of such state machines.

OCRA supports the verification of the contract refinement and the verification of the component state machine against the component contracts.

Within this document and case study, we consider a synchronous discrete-time semantics of the components. We use LTL [Pnu77] as specification language for the contracts.

The system model has two versions, with or without the failures as part of the model. We refer to the latter version as the nominal model, and to former as the extended model.

2 Model structure

2.1 Overview

In this case the system component (i.e. the top level one) contains six subcomponents: two sensors, three monitors and a selector. All components ports have a domain or type of data associated. The only types used in the model are boolean ($\{TRUE, FALSE\}$) and bounded integers, which represent the readings from the environment and the sensors. The model is specified in two versions, one nominal and the other with the explicit failure model. The only differences are the system input that includes the two failure flags for the sensors and the sensors components. The system is at discrete time. All components have an instantaneous reaction of propagation whereas the Selector has a reaction that takes one tick of the system. This is necessary to have the last value to perform the variance check in the monitors. The overall reaction of the system is thus one tick: the corresponding output for a given input is the one generated in the next state.

2.2 Parameters

The domain of the data variables is defined by an integer interval between two fixed bounds ($\{lower\_bound..upper\_bound\}$). The system is then at finite precision values, i.e. the domain of the variables involved in the system is finite. It is important to notice that in the model are used parameters to specify properties and contracts, in particular: `max_sensor_error`, `max_variance`, `lower_bound` and `upper_bound`. These parameters are contained in the *parameter.h* header file. Note that these parameters are arbitrary chosen but fixed. The system is scalable on all these parameters.

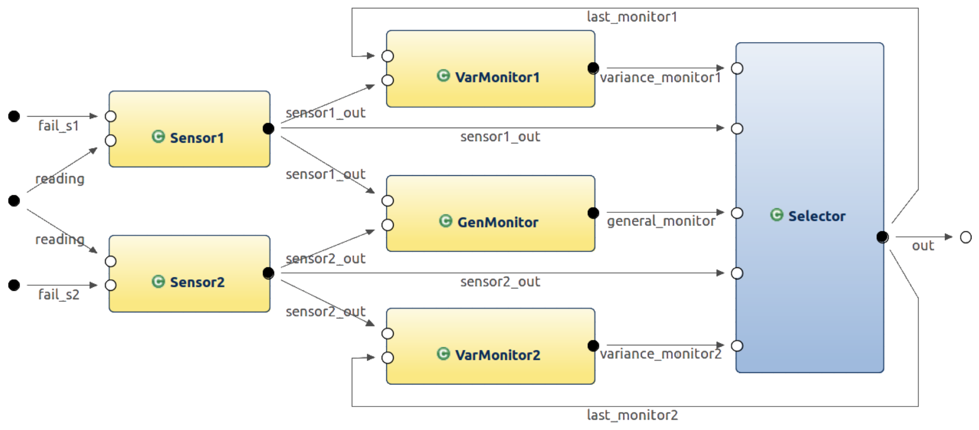

Figure 1: Architecture layout of extended version

2.3 Sensors

The two sensor components are identical. They receive as input a value from the environment that represents the physical quantity that they should read and then give a correlated outcome.

2.3.1 Nominal version

The nominal version of the sensor represents the version without considering the error model, it thus have one port for the input and one for the output. The nominal behavior is that at each value in input the sensor adds a non-deterministic but bounded value that simulates the intrinsic error of the sensor. This error is bounded in absolute value at `max_sensor_error`. The sensor adds this error cutting the resulting sum at the domain range and thus the sensor output is always inside the data domain. In the nominal case the output value of the sensor is function only of the input value.

2.3.2 Extended version

To model the error of a possible failure on a sensor, it has been added a failure flag input for each sensor. It is a boolean that determines wether a sensor is failed ($flag{=}TRUE$) or non failed ($flag{=}FALSE$). If the sensor is not failed its behavior is the nominal one, whereas when failed the sensor simply gives as output a non-deterministic value inside the data domain, completely unrelated with the input value.

2.4 Monitors

There are three monitors of two types: two variance monitors and a general one.

General monitor The general monitor has two input ports that are linked to the two output ports of the sensors. It implements the failure detection feature in the system comparing the two sensor outcomes. The check is based on the maximum sensor error assumption: i.e. if the sensors are both correctly working then their reciprocal output difference in the worst case is twice the maximum sensor error, in particular when one sensor error is at `-max_sensor_error` and the other has its error at the other extreme of the error interval `+max_sensor_error`. Basing on this observation we can say that if the difference of the two sensor outputs is more, in absolute value, than twice the maximum sensor error, then a failure is surely present in one of the two sensors. The output of this monitor is a boolean `Valid` that is equal to true if no failure is detected and false otherwise:

$$Valid := |In1 - In2| \leq 2 * max\_sensor\_error$$

The condition for a failure is only sufficient, so when a failure is detected it is surely present in one of the sensors, but when a failure is not detected not necessarily there is absence of failures.

Variance monitor The variance monitors are two: one is relative to the first sensor whilst the other to the second one. These monitors implement the feature of failure isolation in the system, therefore when a failure is detected by the general monitor then the system tries to isolate which sensor is causing the failure (it is assumed at top level that the sensors are not both failed at the same time). This monitor is based on the assumption regarding the input variance. The variance monitor component is thought to compare the current outcome of a sensor with the last output of the system. Since the system takes one tick to evaluate an input the out comes out in the next step. The variance monitor checks the consistency of a sensor output on the base of its the variance in each tick considering that the system variance is assumed bounded and that the maximum sensor error is fixed. The reasoning is that if the last value is considered reliable then:

$$|real\_last - last| \leq max\_sensor\_error$$

Maximum system error reasoning Thus in the worst case the error in module is equal to the maximum sensor error. If then the input changes as much as possible in a state, i.e. it changes of the value of the maximum variance and in the current state the sensor introduces another worst error, then the overall error is finally given in terms of:

$$error = |last - current| \leq max\_sys\_error$$

where `max_sys_error` is the maximum between:

$$Max\{2 * max\_sensor\_error, max\_sensor\_error + max\_variance\}$$

This is due to the general monitor guarantee, because the last value given as output in a failure case with no isolation has the maximum possible error equal to twice the sensor error. If the maximum variance is less than the maximum sensor error:

$$max\_sensor\_error < max\_variance$$

then the worst case the error is not given by the sum: *max_sensor_error + max_variance*, but by *2*max_sensor_error* since:

$$max\_sensor\_error + max\_variance < 2 * max\_sensor\_error$$

2.5 Selector

This component receives as input the two sensors outputs and the three monitor outcomes. Basing onto these data the selector gives as output either the average of the sensors or the value of one sensor or the last value given in output. The average is necessary to maintain the performances because in the worst case the general monitor does not detects a failure and one sensor is far from the real input by its maximum error, then the other sensor is failed but far, in absolute value from the first sensor by at most twice the max sensor error. In this case the overall error is three times the max sensor error, doing the average of the two sensor is reducing the error at twice the maximum sensor error. This component as the name suggests selects the value to be given as output basing on the information from the monitors, that are the failure detection and isolation components. After the hierarchy of the FDI system, the selector primarily controls the general monitor $Valid$ flag, if true then no failure is detected and the output is the average of the two sensors, if false it checks the variance monitor flags trying to determine which sensor is failed and isolate the problem. If one variance monitor valid flag is at false the failure had been isolated on the relative sensor and the output in this case is the non failed sensor value. If none of the two variance monitors isolate the failure then the output is the last value given as output. This last value, when a failure is detected, is reliable because of the assumptions: there are no possible double failures so if a failure is happening in the current state then in the previous state there were no failures at all. Indeed the first state is assumed correct and this reasoning can propagate by induction. Considering that the maximum error of the system in the previous step is (see 2.4):

$$|last - current| \leq max\_sys\_error$$

Then the worst error of the system is bounded at this constant `max_sys_error` that for definition is equal to the maximum between `max_variance + max_sensor_error` and `2*max_sensor_error`.

3 Requirements refinement

The top level requirement states that the output has always to be consistent with the relative input of the system under some assumptions. To formalize this we say that the difference in absolute value of an output with its respective input is bounded by a constant:

$$|input - ouput| <= \texttt{max\_sys\_error}$$

Under the following assumptions:

- **consistent first state**, in the first state both sensors work correctly,
- **bounded variance** of the input,
- **singular and instantaneous failures**, the two sensors cannot be failed at the same time and when a failure occurs, in the next state there are no ones.

This requirement is then refined in the various subcomponents requirements that are described component per component in the following section. The refinement is such that the composition of the requirements of the subcomponents matches the top level one.

4 Formal architecture

The architecture of the model, as briefly described, is composed of a system component RedundantSensor refined in: two redundant sensors, three monitors and a selector. The system has only one layer of refinement:

- RedundantSensor
  - Sensor1
  - Sensor2
  - VarMonitor1
  - VarMonitor2
  - GenMonitor
  - Selector

4.1 RedundantSensors component

At top level the nominal system has two ports: one for the input and another one for the output, both within the data range specified. In the extended model was added a pair of failure flags as input ports from the environment. These flags induct the sensors in behaving correctly or simulating a failure.

4.1.1 Input ports

The input interface in the nominal version has only the *reading* port and in the extended version the two flags for failures also.

`reading: value_domain;` physical quantity then measured by the sensors

`fail_s1: boolean;` failure flag of Sensor1

`fail_s2: boolean;` failure flag of Sensor2

When a failure flag is false the respective sensor behaves nominally, whereas when this flag is true the sensor behaves in failure mode and give as its output a random value not consistent neither based on the reading input.

4.1.2 Output ports

`out: value_domain;`

4.1.3 Contract system_error

Nominal system top level contract.

```
assume: always
         (abs_diff(reading,next(reading)) <= max_variance);

guarantee: always
            (abs_diff(reading, next(out)) <=
              max_sys_error);
```

4.1.4 Contract single_tmp_failure

Extended version version top level contract.

```
assume:
       (not fail_s1 and not fail_s2) and
        always (
         (abs_diff(reading,next(reading)) <= max_variance) and
           not (fail_s1 and  fail_s2) and
            ((fail_s1 or fail_s2) implies then
             (not fail_s1 and not fail_s2)) );

guarantee: always
            (abs_diff(reading, next(out)) <=
              max_sys_error);
```

4.2 Sensor

4.2.1 Input Ports

The nominal sensor has only the *In* port, the *Fail* one is only present in the extended model of the sensor.

`In:` `value_domain;`

`Fail:` `boolean;`

4.2.2 Ouput ports

`Out:` `value_domain;`

4.2.3 Contract nominal

This is the nominal version contract and specifies that without assumptions the output error is bounded with respect to the input at the maximum sensor error.

```
assume: TRUE;
guarantee: always (abs_diff(In, Out) <= max_sensor_error);
```

4.2.4 Contract failure

The failure version contract states that if the failure flag is false the output error is bounded as in the nominal case, otherwise nothing is guaranteed on the output value.

```
assume: TRUE;
guarantee: always
            (not Fail implies
             (abs_diff(In, out) <= max_sensor_error) );
```

4.3 VarMonitor

4.3.1 Input Ports

`In:` `value_domain;` sensor outcome

`Last:` `value_domain;` last output of the system

4.3.2 Ouput ports

`Valid: boolean;`

- Valid = true: no failure detected
- Valid = false: failure detected, the sensor output is not reliable

4.3.3 Contract isolate

```
  assume: TRUE;
  guarantee: always
              (Valid iff
               (abs_diff(In, Last) <=
                ((2*max_sensor_error) + max_variance)) );
```

4.4 GenMonitor

4.4.1 Input Ports

`In1: value_domain;` Sensor1 outcome

`In2: value_domain;` Sensor2 outcome

4.4.2 Ouput ports

`Valid: boolean;`

- Valid = true: no failure detected, not ensures absence of failures
- Valid = false: failure detected

Of course a failure can happen and not even being detected if casually the non-deterministic value is near the real reading.

4.4.3 Contract detect

```
 assume: TRUE;
 guarantee: always
             (Valid iff
              (abs_diff(In1,In2) <= 2*max_sensor_error) );
```

4.5 Selector

4.5.1 Input Ports

`sensor1_out: value_domain;`

`sensor2_out : value_domain;`

`variance_monitor1 : boolean;`

`variance_monitor2 : boolean;`

`general_monitor : boolean;`

4.5.2 Output ports

```
out : value_domain;
```

4.5.3 Contract Selector

The contract means:

- if no failure is detected, i.e. `general_monitor` is true then the outcome is chosen to be the average of the sensors
- if the general monitor detects a failure and the sensor failed is detected by having a `variance_monitor` false, then the outcome is the functioning sensor
- in case of no isolation the last value is used as output

```
assume: TRUE;
guarantee: always
            ((general_monitor implies
               next(out) = (sensor1_out+sensor2_out)/2) and
            ((not general_monitor and
               not variance_monitor1 and variance_monitor2) implies
                next(out) = sensor2_out) and
            ((not general_monitor and
               not variance_monitor2 and variance_monitor1) implies
                next(out) = sensor1_out) and
            ((not general_monitor and variance_monitor1 and
               variance_monitor2) or
             (not general_monitor and not variance_monitor1 and
                not variance_monitor2) implies
                   next(out) = out));
```

References

[CDT13] Alessandro Cimatti, Michele Dorigatti, and Stefano Tonetta. Ocra: A tool for checking the refinement of temporal contracts. In *ASE* [DBL13], pages 702–705.

[DBL13] *2013 28th IEEE/ACM International Conference on Automated Software Engineering, ASE 2013, Silicon Valley, CA, USA, November 11-15, 2013.* IEEE, 2013.

[FOR] FoReVer project. `https://es.fbk.eu/projects/forever/`.

[Pnu77] Amir Pnueli. The Temporal Logic of Programs. In *FOCS*, pages 46–57. IEEE Computer Society, 1977.