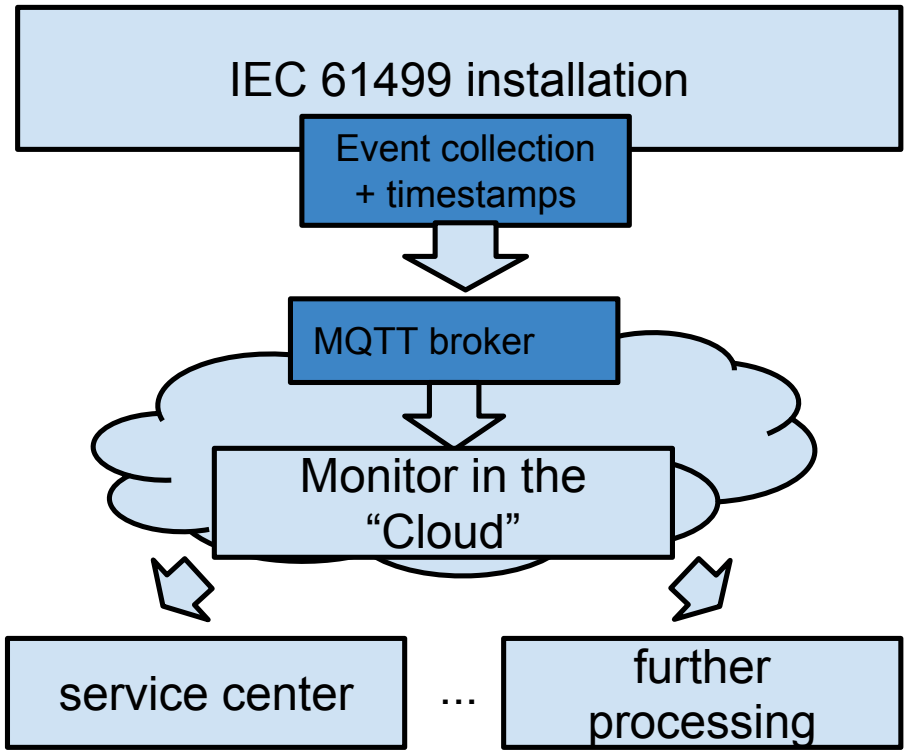
Towards using Formal Behavioral Specifications in IEC 61499 for Remote Software Health Monitoring

Jan Olaf Blech

with figures from Monika Wenger and Alois Zoitl

RMIT UNIVERSITY

Overview



[Monika Wenger, Alois Zoitl, Jan Olaf Blech, Ian Peake, Lasith Fernando. Cloud Based Monitoring of Timed Events for Industrial Automation. Automated Testing of Cyber-Physical Systems in the Cloud, IEEE, 2015.]

Ingredients: behavioral types as monitor specification, IEC 61499, demonstrator setup

Motivation: “Business Case”

1. IEC 61499 code + monitor controls a remote machine
2. IEC 61499 based monitoring detects deviation
3. Information is send to a service center
4. Service center works on new code version / resolving the issue
5. Remote machine is updated
6. Normal operations continue

No technical staff required at remote machine

Behavioral Types

- Behavioral Types are:
  - Formal specification mechanism
  - Here: based on state machine / regular expressions
  - Protocol for component interactions
- Extended Service Interaction → Behavioral types

- Type compatibility
- Type conformance
- Type coercion

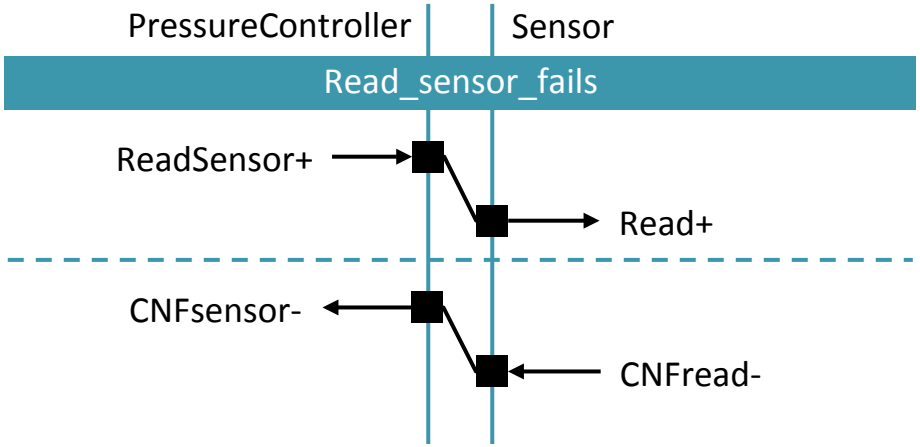
Here: ensuring type conformance at runtime

→ Runtime monitoring

Behavioral Types

- Interface automata [de Alfaro, Henzinger 2001]
- Behavioral types in Ptolemy(2) [Lee & al.]
- Behavior for UML
  - E.g., a Fully Abstract Semantics for UML Components [de Boer & al. 2005]
- Abstract Behavior Types: A Foundation Model for Components and Objects [Arbab 2003]
- Our work: Behavioral Types for OSGi
  - Framework proposition: [Blech, Falcone, Ruess, Schaetz 2012]
  - Theorem prover connection: [Blech, Schaetz 2012]
  - Compatibility checking: [Blech 2013: Towards a Framework for Behavioral Specifications of OSGi Components.]
  - Ensuring behavioral types at runtime: this talk

Behavior Description of IEC 61499

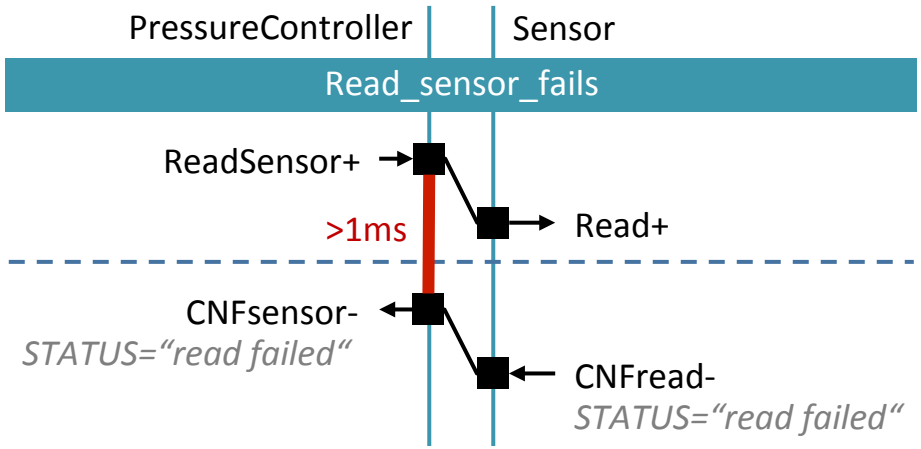



```
SEQUENCE Read_sensor_fails
    PressureController.ReadSensor+ → Sensor.Read+;
    Sensor.CNFread- → PressureController.CNFsensor-;
    Sensor.ERROR- → PressureController.CNFsensor-;
END_SEQUENCE
```

Limits of IEC 61499 service sequence description:

- No statement about completeness of event combination
- No additional properties
- No specific timing constraints
- No compatibility check of two service sequences
- Limited expressibility

Behavior Description of IEC 61499



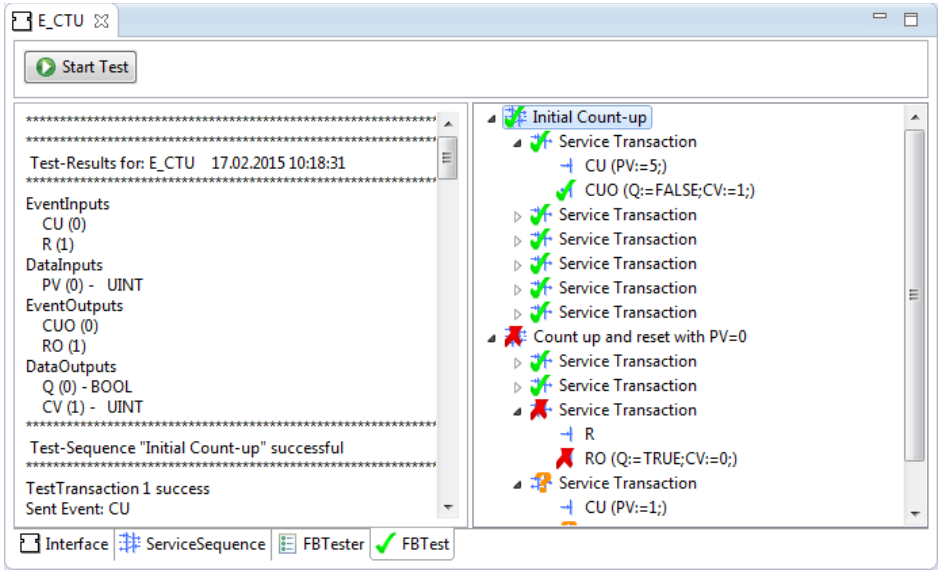

```
SEQUENCE Read_sensor_fails
    PressureController.ReadSensor+
    → Sensor.Read+;

    Sensor.CNFread- (STATUS="read failed")
    → PressureController.CNFsensor- (STATUS="read failed");

    Sensor.ERROR- (STATUS="read failed")
    → PressureController.CNFsensor- (STATUS="read
failed");
END_SEQUENCE
```

- Extensions: expressiveness (alternatives, repetition, values)

Behavioral Type Based Monitoring

Static Checking

- FB type = software component
- unit testing with service sequences
- No compatibility check



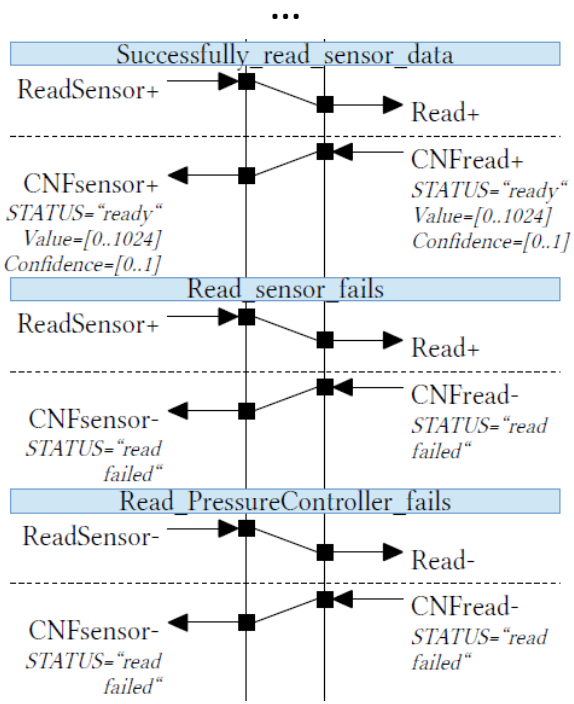
Runtime Monitoring

- Compliance of code and specification during runtime

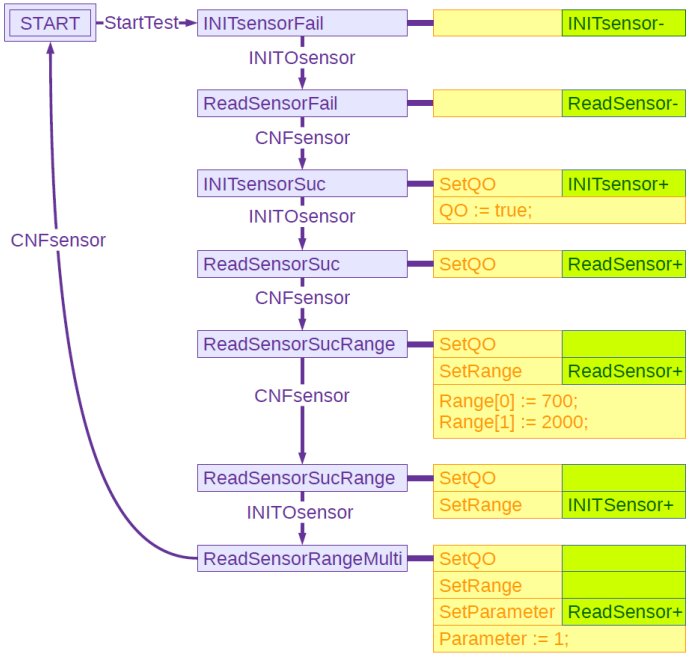
```
int FORTE_FBx::update_monitor_pc(int eiID){
    switch(this_monitor_state->protocol_state){
        case(0) :
            if(INITsensor == eiID && true == QO && val[0] >= 0)
&& val[0]) <= 1024){
                this_monitor_state->protocol_state = 1;
                return true;
            }if(INITsensor == eiID && false == QO && val[0] >= 0
&& val[0] <= 1024){
                this_monitor_state->protocol_state = 2;
                return true;
            }if(ReadSensor == eiID && true == QO){
                this_monitor_state_pc->protocol_state = 3;
                return true;
            }if(ReadSensorID == eiID && false == QO){
                this_monitor_state->protocol_state = 4;
                return true;
            }break; …
    }return 0;}
```

Monitoring Results

Service Sequence Specification of the PressureController



Test Cases Executed by the PressureController



Error Log Produced During Test Case Execution

...
TRACE: T#012484ms: OutputEvent: **FB** PressureController_v1 sending event: ReadSensor
TRACE: T#012484ms: InputEvent: **FB** PressureSensor_v1 got event: Read
ERROR: T#012484ms: **FB** PressureSensor_v1 send wrong output event CNFread in state 0!
TRACE: T#012484ms: OutputEvent: **FB** PressureSensor_v1 sending event: CNFread
TRACE: T#012484ms: InputEvent: **FB** PressureController_v1 got event: CNFsensor
ERROR: T#012484ms: **FB** PressureController_v1 received wrong input event CNFsensor in state 0!

Conclusion

Results

- Extension of service sequences
- Timing information
- Runtime monitoring of FBs
- Monitoring of multiple devices using a cloud-based service
- Several experiments have been conducted

Ongoing Extensions

- Reaction to deviation from an expected behavior
  - Reaction at runtime
  - System halt and user defined reactions
- Remote debugging and maintenance as a reaction to unwanted behavioral derivations
- Detailed evaluation

Thank you very much for your attention

RMIT UNIVERSITY