

A Domain-specific Language and Architecture for Detecting Process Activities from Sensor Streams in IoT

Ronny Seiger, Daniel Locher, Marco Kaufmann, Aaron F. Kurz

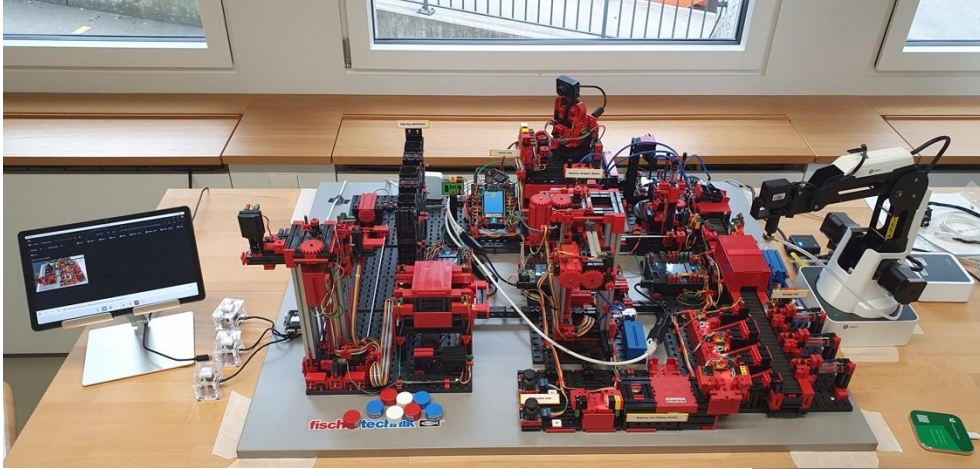
EDOC 2026, Journal First (originally published in Internet of Things, Elsevier)

Enschede, Sep 16, 2026



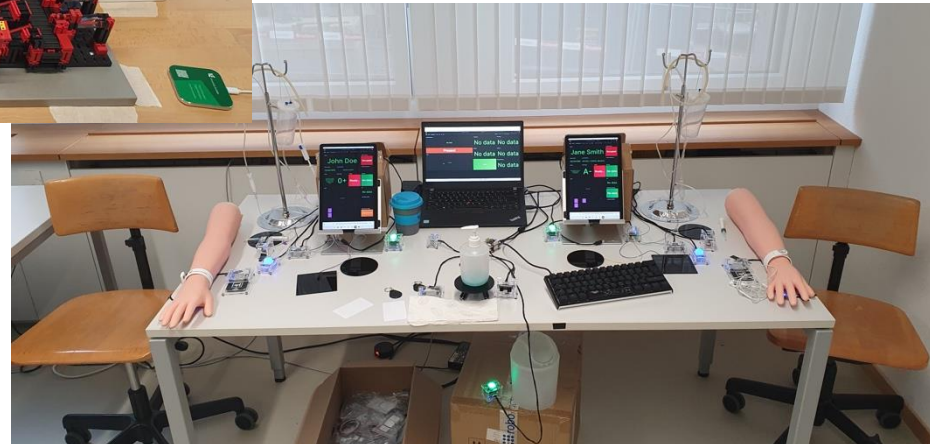
A DSL and Architecture for Detecting Process Activities from Sensor Streams in IoT

Scenarios

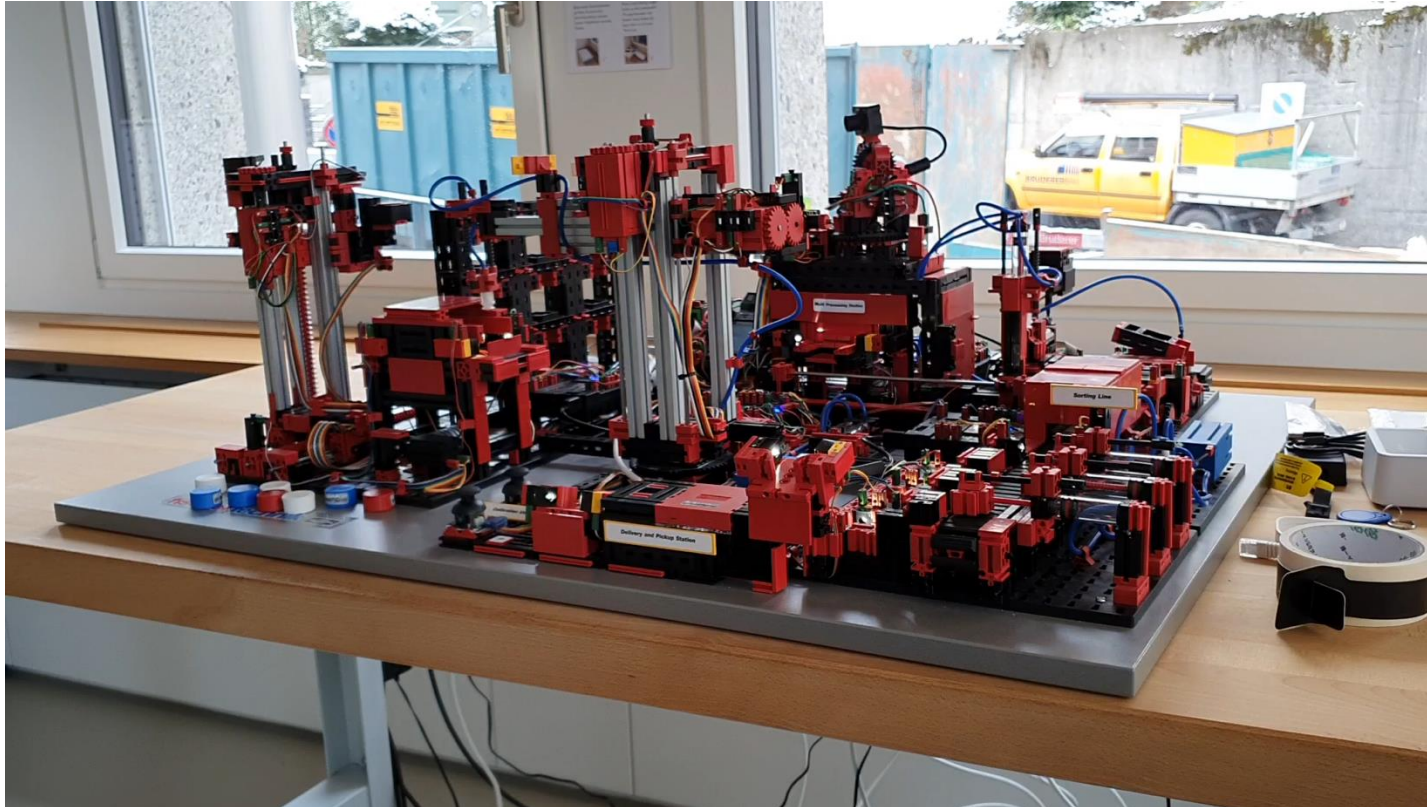


Smart Production simulated with Fischertechnik and robot

Smart Healthcare (Blood donation process) simulated with various IoT sensors



Scenario 1: Smart Production

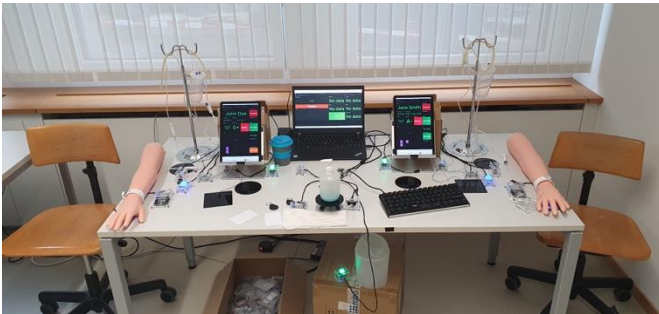


Scenario 2: Smart Healthcare (Blood Donation)



A DSL and Architecture for Detecting Process Activities from Sensor Streams in IoT

Motivation



Sensors:

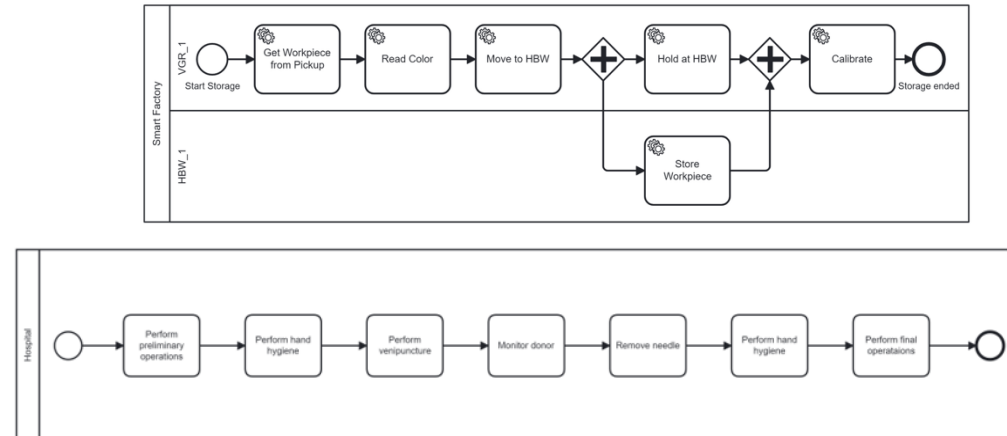
- Switches
- Light barriers
- Color sensors
- Environment
- Camera
- NFC, Touch, Proximity
- Motion, Distance
- Accelerometer

Actuators:

- Motors
- Compressors
- Valves
- Lights

➔ Low-level IoT Data / Events

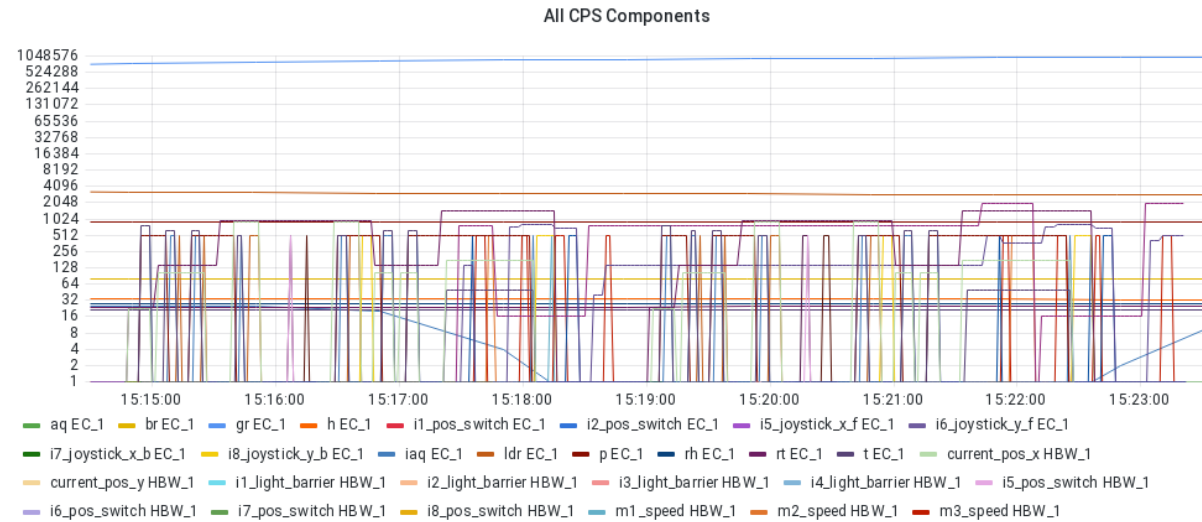
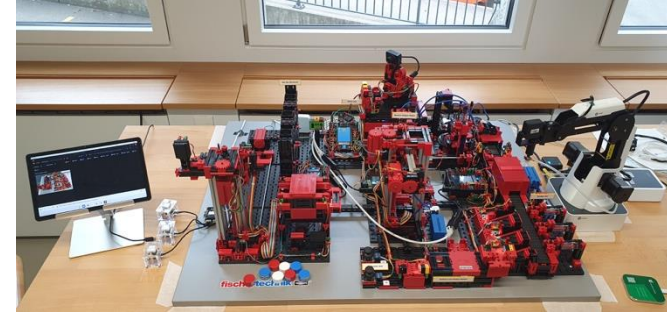
How to link low-level IoT events to process executions?



A DSL and Architecture for Detecting Process Activities from Sensor Streams in IoT

Given

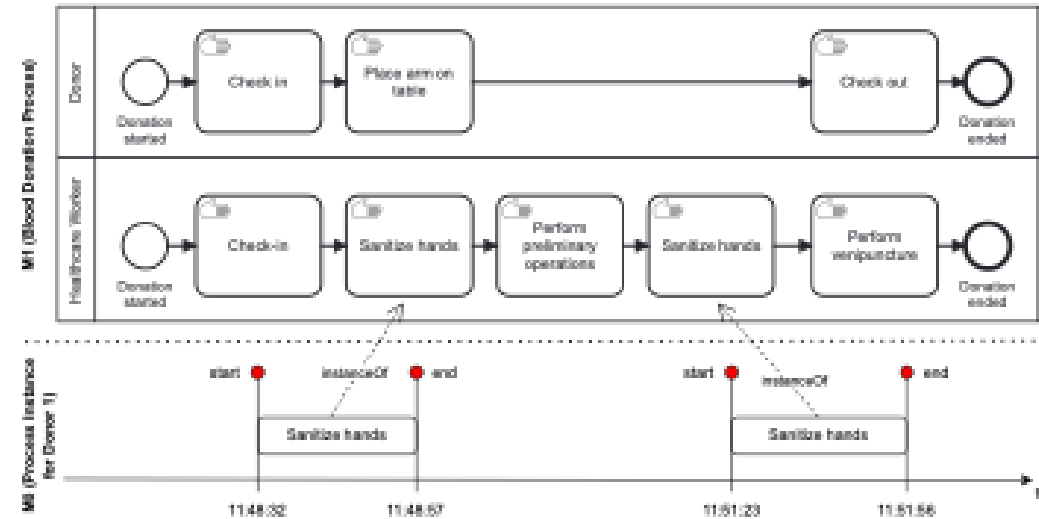
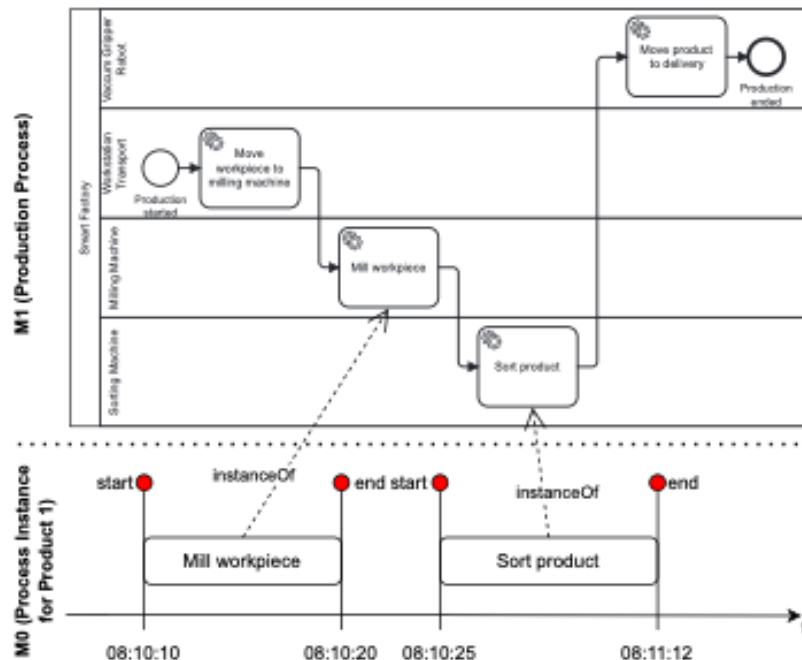
- IoT Data from sensors and actuators (as time series)
- Not segmented, no activity labels
- Assumption: no BPM System is available to orchestrate or monitor processes



A DSL and Architecture for Detecting Process Activities from Sensor Streams in IoT

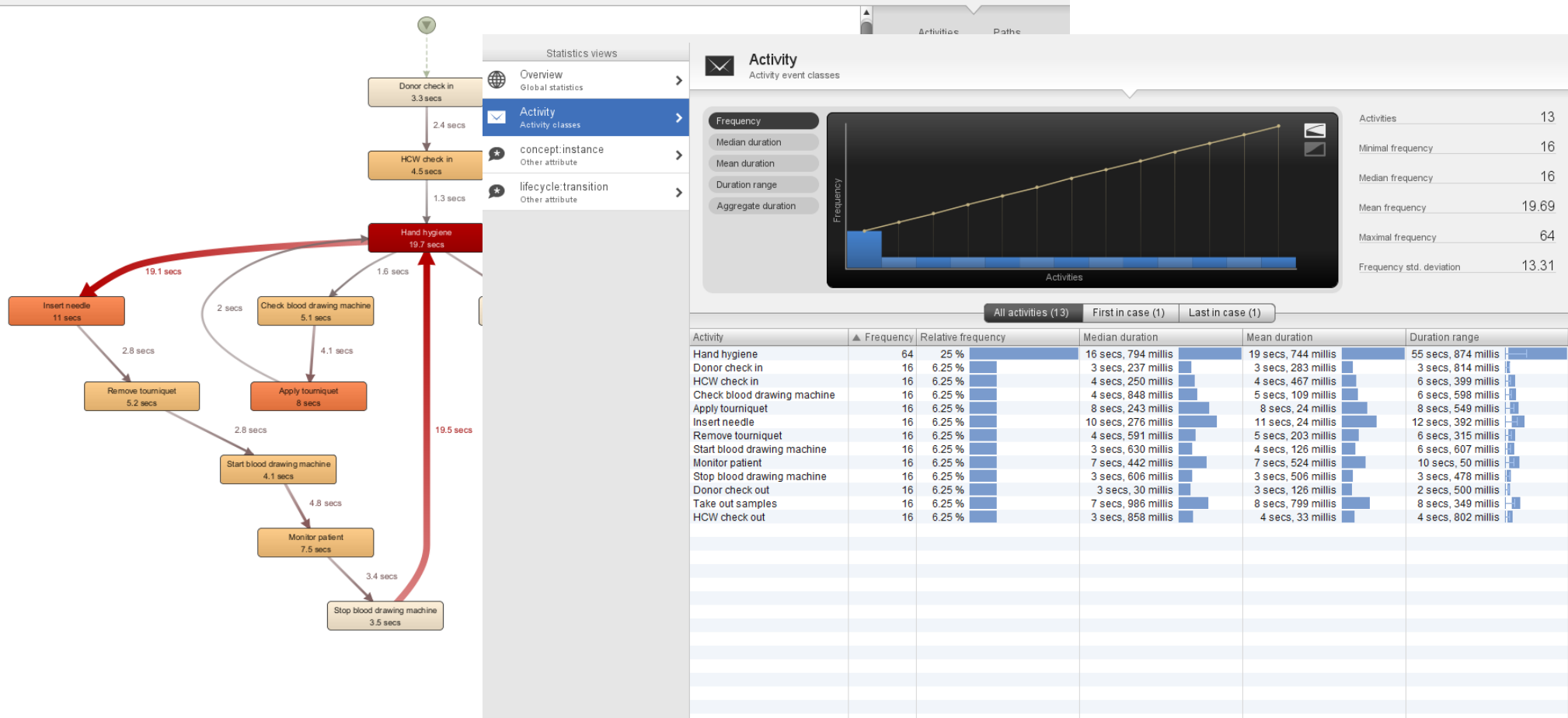
Goal

- Assign activity labels—start & end—to IoT data segments (event abstraction)
- Detect activity executions / start & end events at runtime



A DSL and Architecture for Detecting Process Activities from Sensor Streams in IoT

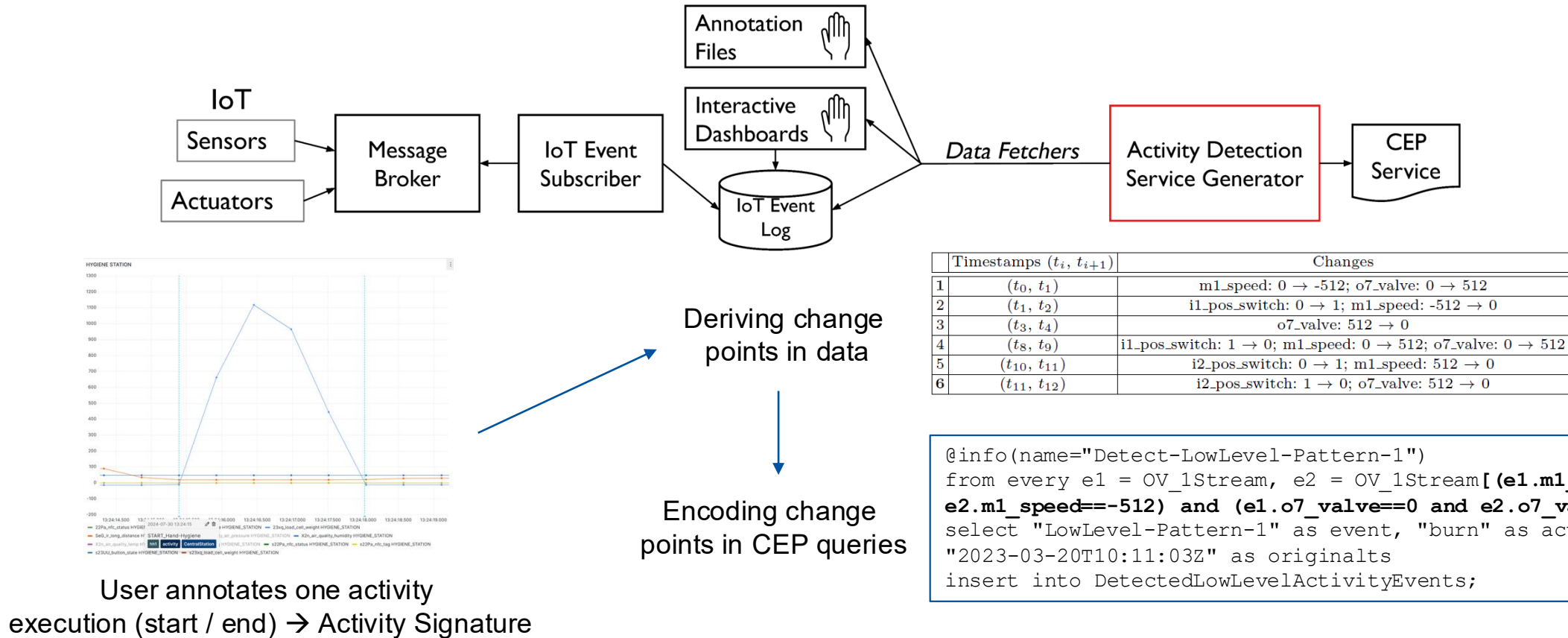
Process Mining on IoT-enhanced Processes



A DSL and Architecture for Detecting Process Activities from Sensor Streams in IoT

Previous Work: Automated Activity Detection Service Generation

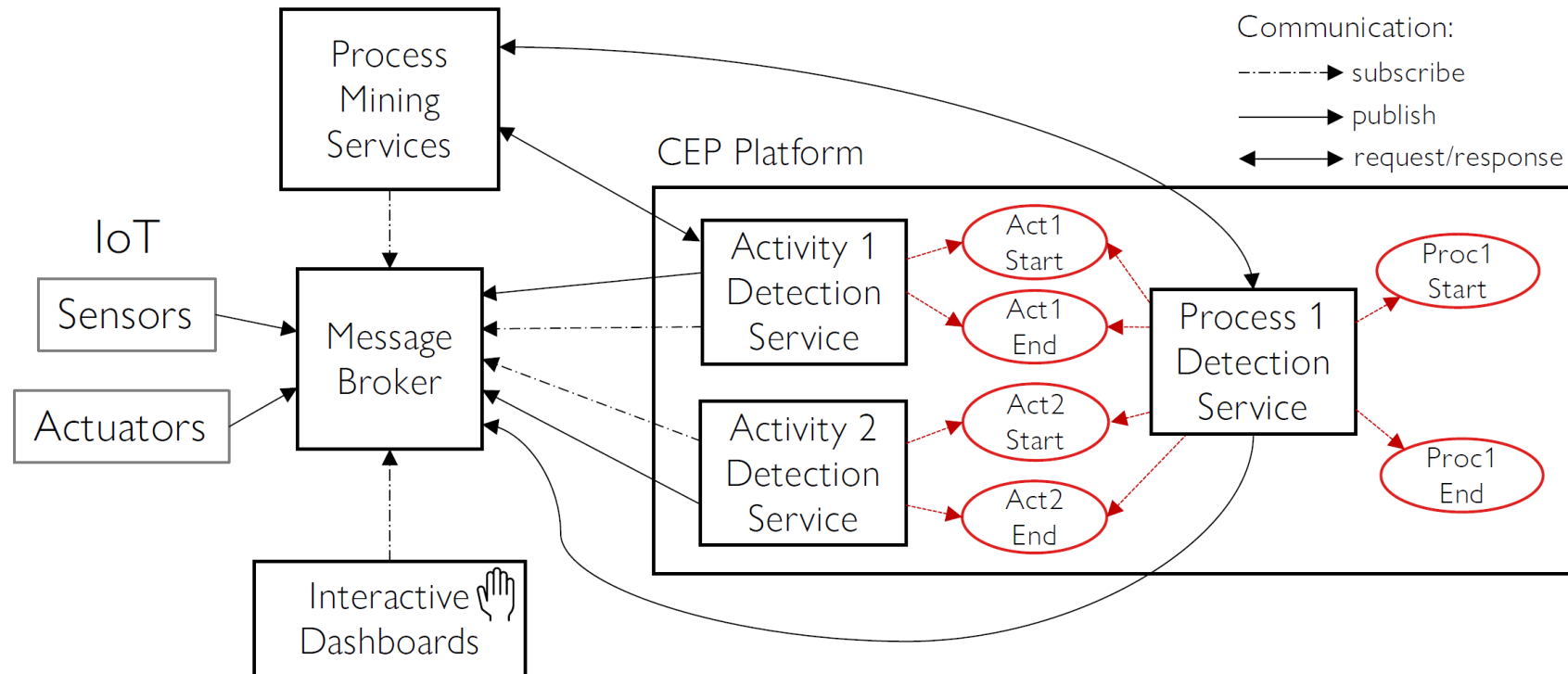
Seiger, R., Kurz, A. F., & Franceschetti, M. (2026).
Online detection of process activity executions from IoT
sensors using generated event processing services.
Future Generation Computer Systems, 174, 107987.



A DSL and Architecture for Detecting Process Activities from Sensor Streams in IoT

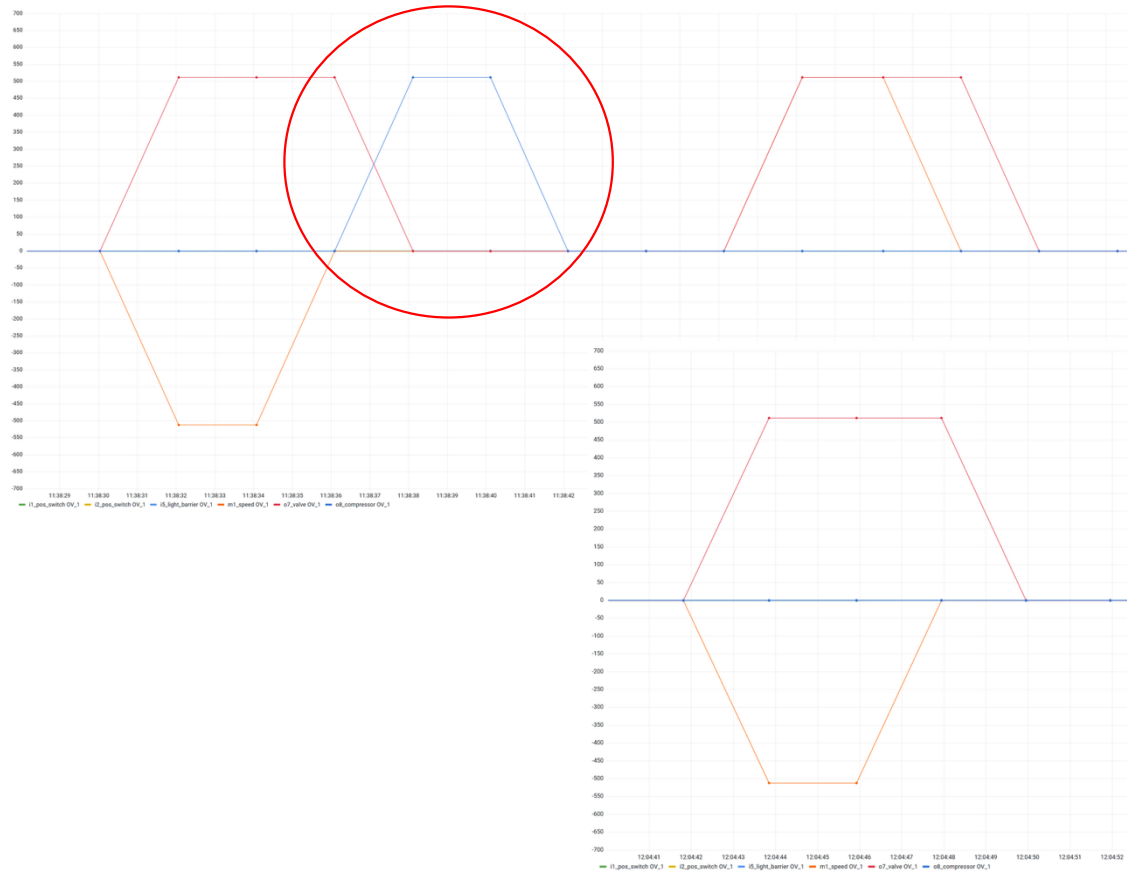
Software Architecture

Seiger, R., Kurz, A. F., & Franceschetti, M. (2026).
Online detection of process activity executions from IoT
sensors using generated event processing services.
Future Generation Computer Systems, 174, 107987.



A DSL and Architecture for Detecting Process Activities from Sensor Streams in IoT

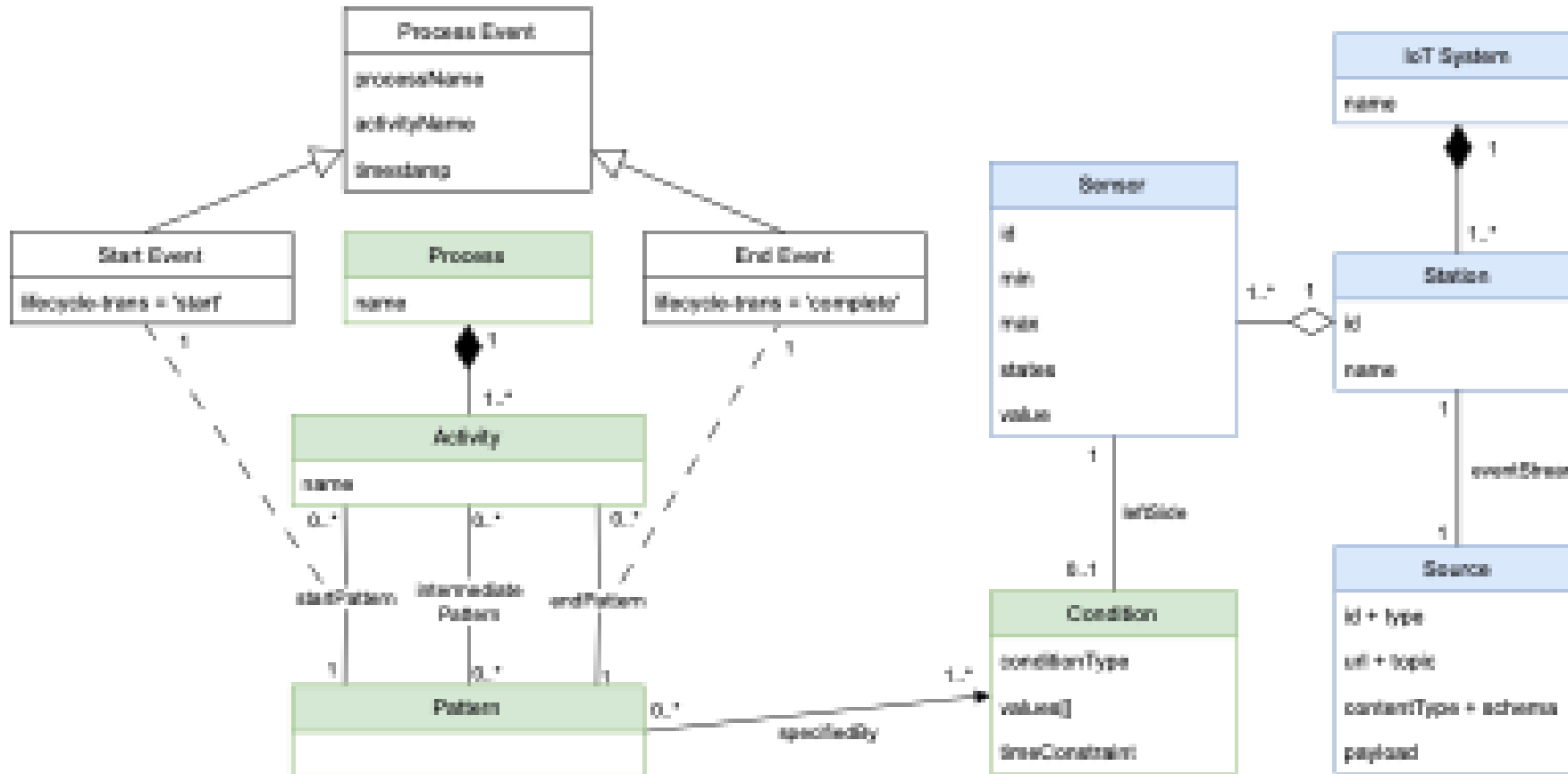
Problem: Variations in Sensor Data



- Low quality in activity detections
- Completely automated approach based on activity signatures not sufficient
 - Incorporate expert knowledge
 - **Domain-specific Language**



Radiant: Meta-model



A DSL and Architecture for Detecting Process Activities from Sensor Streams in IoT

Radiant: Syntax

```
1 'Process' name=ID ':'
2   (activities+=Activity)*;
3
4 Activity:
5   'Activity' name=ID ':'
6     startPattern=Start
7     (intermediates+=Intermediate)*
8     endPattern=End;
```

Process with Activities

```
1 Pattern:
2   Start | Intermediate | End;
3
4 Start:
5   'Start: '
6   (conditions+=Condition+ | cases+=Case+);
7
8 Intermediate:
9   'Intermediate: '
10  (conditions+=Condition | cases+=Case)+;
11
12 End:
13   'End: '
14   (conditions+=Condition | cases+=Case)+;
15
16 Case:
17   'Case: '
18   (conditions+=Condition)+;
```

Types of Patterns with Conditions and Cases

```
1 Condition:
2   'In' station=ID 'sensor' sensor=ID ConditionType
3     (time_constraint+=TimeConstraint)*
4
5 TimeConstraint:
6   'within' amount=INT time_unit=ID;
```

Conditions with optional time constraints

Condition Type	Syntax
ChangeCondition	changes_from <i>Value</i> to <i>Value</i> (TimeConstraint)*;
ChangingCondition	is_changing (TimeConstraint)*;
RangeCondition	in_range <i>Value</i> to <i>Value</i> ;
IsEqualCondition	is_equal <i>Value</i> ;
IsLowerCondition	is_lower <i>Value</i> ;
IsLowerOrEqualCondition	is_lower-or-equal <i>Value</i> ;
IsHigherCondition	is_higher <i>Value</i> ;
IsHigherOrEqualCondition	is_higher-or-equal <i>Value</i> ;
IncreasingCondition	is-increasing (TimeConstraint)*;
DecreasingCondition	is-decreasing (TimeConstraint)*;

Types of Conditions



A DSL and Architecture for Detecting Process Activities from Sensor Streams in IoT

Radiant: Examples

```
1 Process Production:
2   Activity Mill_workpiece:
3     Start:
4       In MM_1 sensor i1_pos_switch is_equal 1;
5       In MM_1 sensor o8_compressor
6         changes_from off to on;
7     End:
8     In MM_1 sensor m1_speed changes_from 512 to 0;
9
10  Activity Sort product:
11    Start:
12      In SM_1 sensor m1_speed
13        changes_from 0 to -512;
14      In SM_1 sensor i1_light_barrier is_equal 1;
15    Intermediate:
16      In SM_1 sensor i2_color_sensor is_changing;
17    End:
18    Case:
19      In SM_1 sensor o5_valve
20        changes_from open to closed;
21    Case:
22      In SM_1 sensor o6_valve
23        changes_from open to closed;
```

```
1 Process Blood_donation:
2   Activity Sanitize_hands:
3     Start:
4       In CHS sensor load_cell
5         changes_from low to high within 30 seconds;
6     End:
7       In CHS sensor load_cell
8         changes_from high to low within 30 seconds;
```

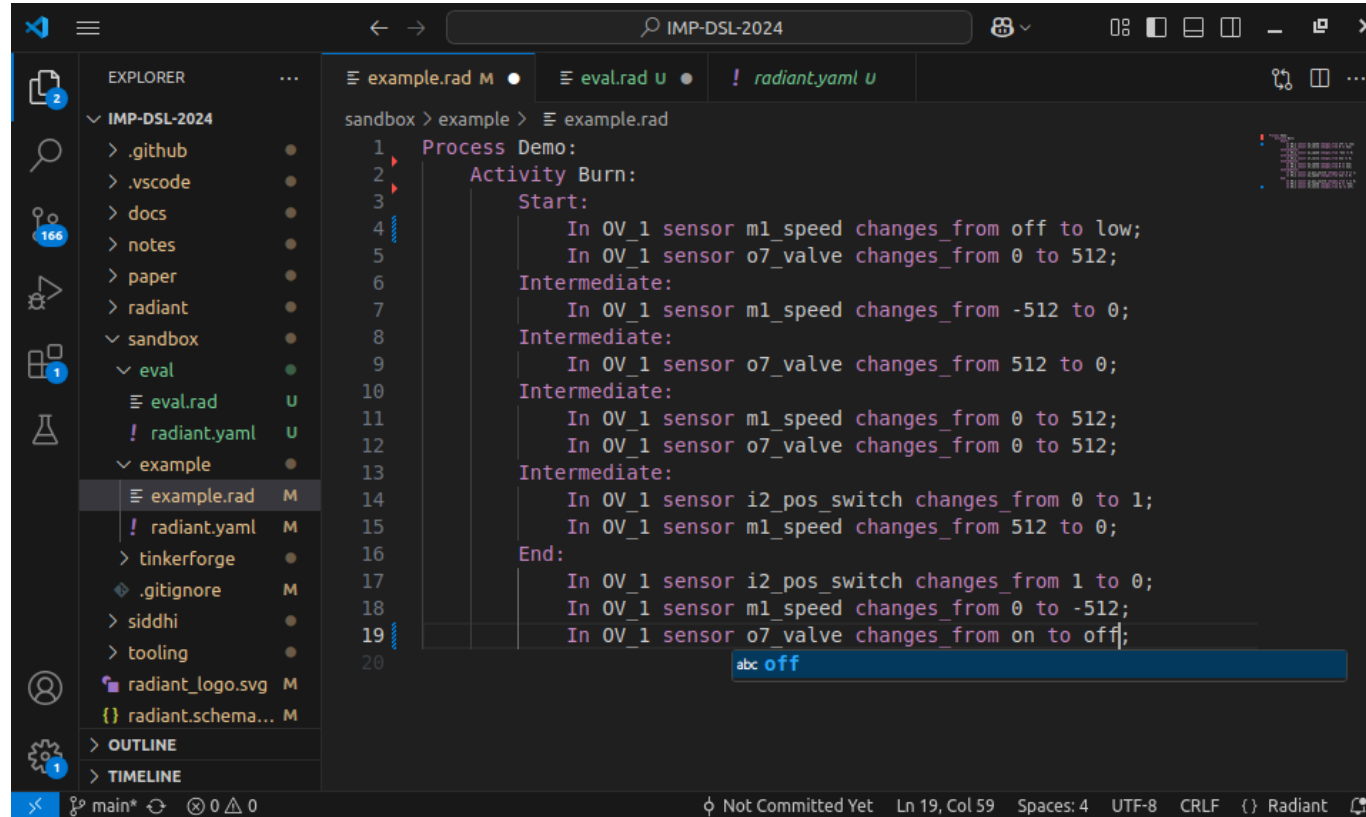
Sensor configurations in external YAML files

```
1 presets:
2   - id: motor_preset
3     min_value: -512
4     max_value: 512
5     states:
6       low: -512
7       off: 0
8       high: 512
9
10 stations:
11   - id: SM_1
12     name: Sorting Machine
13     source: SM_1Stream
14     sensors:
15       - id: i1_light_barrier
16         type: switch
17       - id: i2_color_sensor
18         discretization:
19           lower: [1725, "red"]
20           intermediate: [1725, 1790, "blue"]
21           upper: [1790, "white"]
22       - id: m1_speed
23         type: int
24         preset: motor_preset
25       - id: o5_valve
26         type: int
27         states:
28           open: 75
29           closed: 0
30
31 sources:
32   - id: SM_1Stream
33     type: mqtt
34     url: ${MQTT_URL}
35     client_id: mqtt.SM_1.Sort
36     topic: FTFactory/SM_1
37     content_type: json
38     schema:
39       ts: string
40       i1_light_barrier: int
41       i2_color_sensor: int
42       m1_speed: int
43       o5_valve: int
```



A DSL and Architecture for Detecting Process Activities from Sensor Streams in IoT

Radiant DSL in VS Code (Langium-based)



```
1 Process Demo:
2   Activity Burn:
3     Start:
4       In OV_1 sensor m1_speed changes_from off to low;
5       In OV_1 sensor o7_valve changes_from 0 to 512;
6     Intermediate:
7       In OV_1 sensor m1_speed changes_from -512 to 0;
8     Intermediate:
9       In OV_1 sensor o7_valve changes_from 512 to 0;
10    Intermediate:
11      In OV_1 sensor m1_speed changes_from 0 to 512;
12      In OV_1 sensor o7_valve changes_from 0 to 512;
13    Intermediate:
14      In OV_1 sensor i2_pos_switch changes_from 0 to 1;
15      In OV_1 sensor m1_speed changes_from 512 to 0;
16    End:
17      In OV_1 sensor i2_pos_switch changes_from 1 to 0;
18      In OV_1 sensor m1_speed changes_from 0 to -512;
19      In OV_1 sensor o7_valve changes_from on to off;
20
```

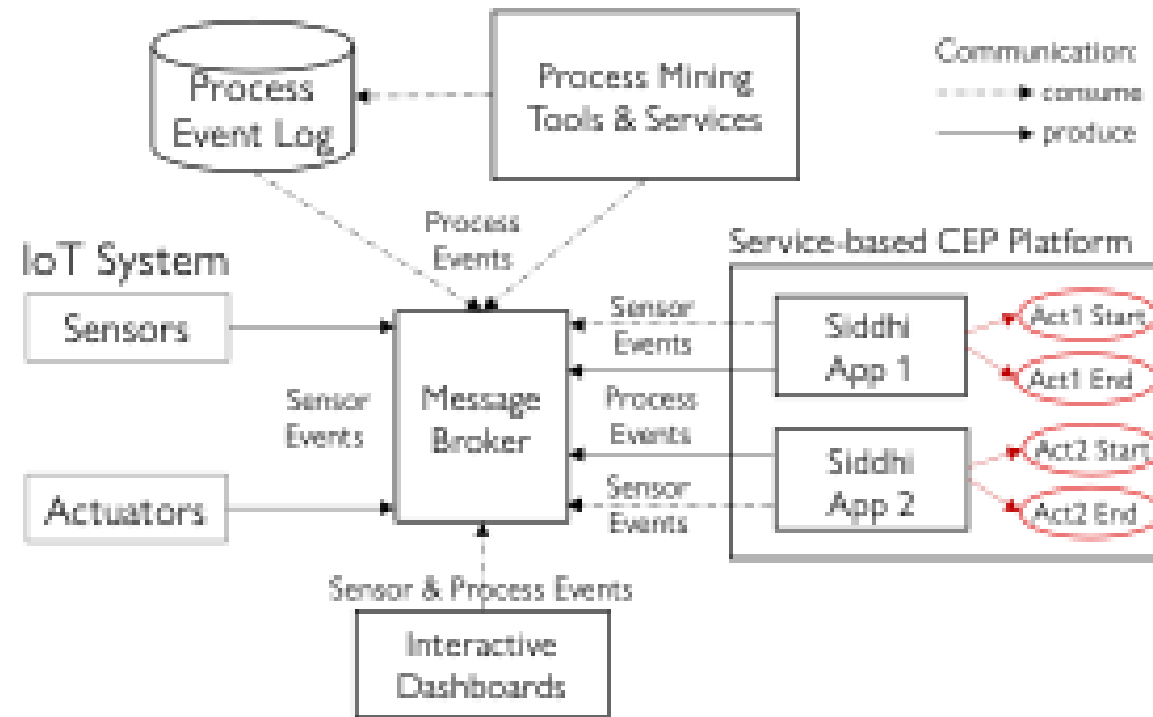


Radiant: Generated CEP Apps Siddhi CEP platform

```
1 @App:name('Production-Sort product')
2
3 @source(type = 'mqtt', url = '${MQTT_URL}', topic = 'PTFactory/SM_1',
4         @map(type = 'json'))
5
6 define stream SM_1Stream(ts string, i1_light_barrier int,
7                          i2_color_sensor int, m1_speed int, o5_valve int, o6_valve int);
8
9 @sink(type = 'log')
10
11 define Stream DetectedActivities(event string, activity string,
12                                 ts_start string, ts_end string);
13
14 @info(name='StartPattern')
15
16 from every e1 = SM_1Stream, e2 = SM_1Stream[(e1.m1_speed==0 and
17        e2.m1_speed==512) and (e2.i1_light_barrier==1)]
18 select 'Start' as event, 'Sort product' as activity, e2.ts as ts
19 insert into DetectedPatterns;
20
21 /* ... */
22
23 @info(name='EndPattern')
24
25 from every e1 = SM_1Stream, e2 = SM_1Stream[((e1.m1_o5_valve==75 and
26        e2.m1_o5_valve==0)) or ((e1.m1_o5_valve==75 and
27        e2.m1_o6_valve==0))]
28 select "EndPattern" as event, "Sort product" as activity, e1.ts as ts
29 insert into DetectedPatterns;
30
31 @info(name='Detect-Activity')
32
33 from every e1 = DetectedPatterns[event == "StartPattern"] -> not
34     DetectedPatterns[event == "StartPattern"] and e2 =
35     DetectedPatterns[event == "IntermediatePattern"] -> not
36     DetectedPatterns[event == "StartPattern"] and e3 =
37     DetectedPatterns[event == "EndPattern"]
38 select "Sort product" as activity, e1.ts as ts_start, e3.ts as ts_end
39 insert into DetectedActivities;
```



Software Architecture



A DSL and Architecture for Detecting Process Activities from Sensor Streams in IoT

Prototype: Activity Detection at Runtime

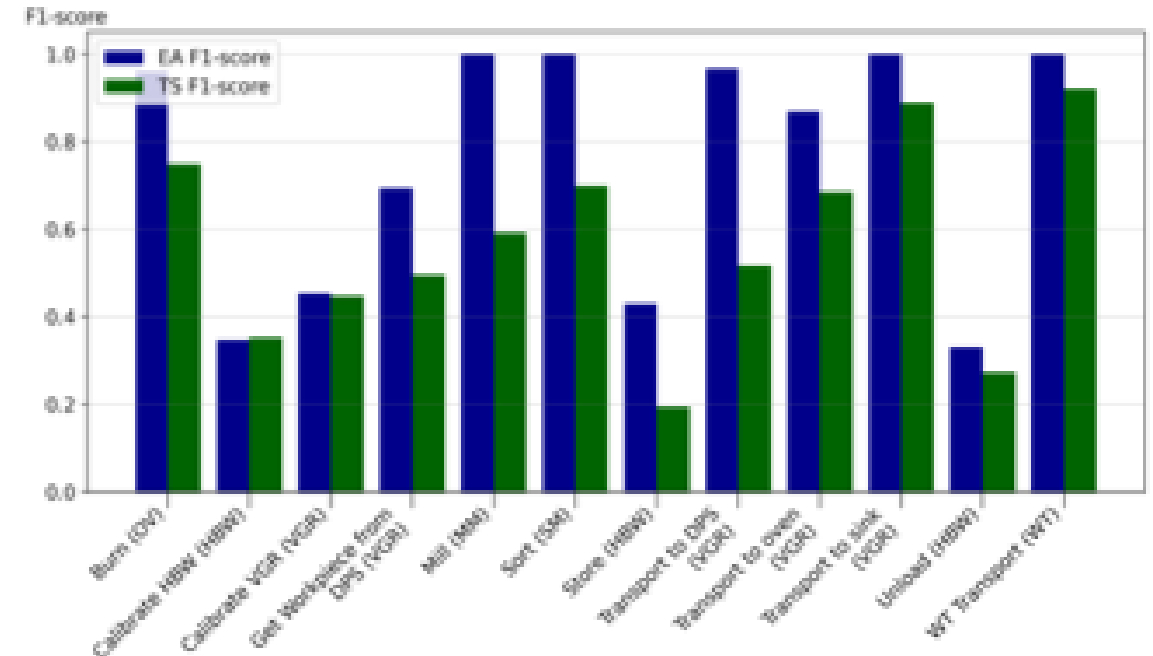
Note: Video does not match data



Evaluation: Smart Manufacturing

- Analysis of processes executed in smart factory
- 181.4 minutes, 424 executed activity instances
- 77 sensors
- Ground truth recorded by workflow engine

Metric Category	Metrics			
Two Set	Precision	Recall	F1	Bal Acc
	0.3482	0.5956	0.4395	0.7513
Event Analysis	Precision	Recall	F1	
	0.6276	0.6675	0.6469	
Other	Damerau-Lev-Norm	Cross-Correlation		
	0.4934	0.7438		



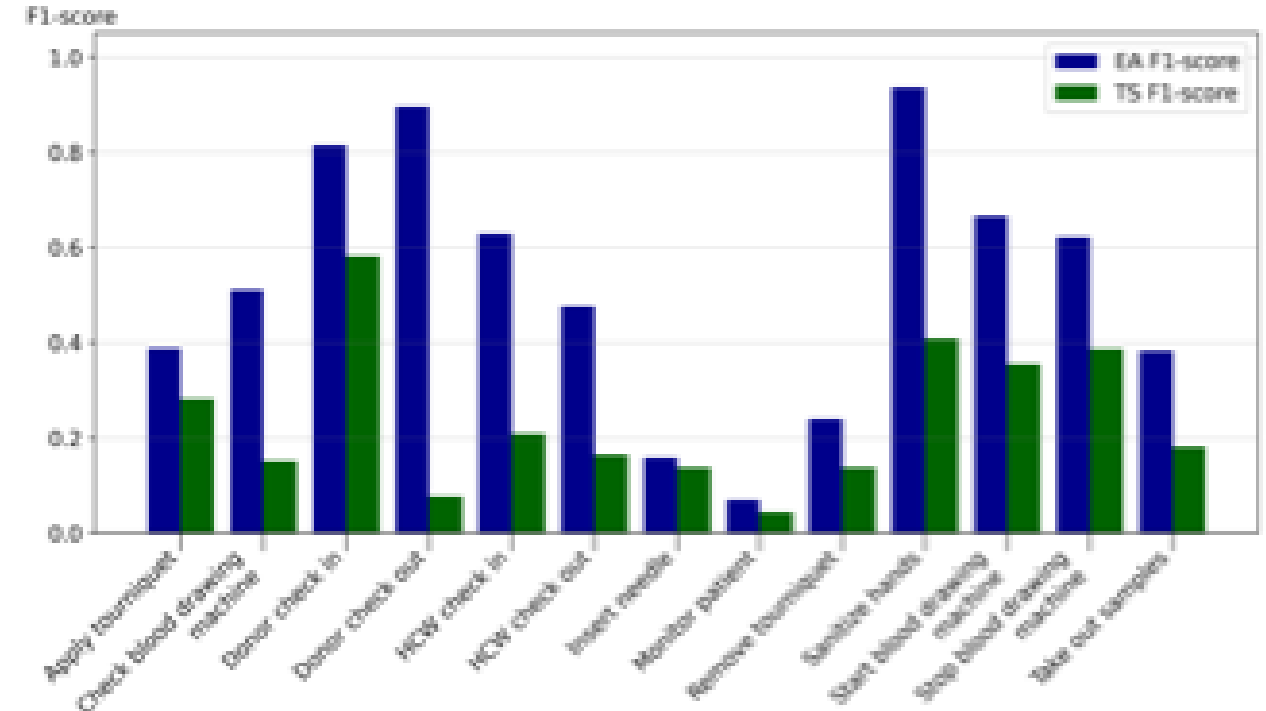
Dataset: <https://zenodo.org/communities/iopt>



Evaluation: Smart Healthcare

- Analysis of processes executed in smart healthcare by medical students
- 65.31 minutes, 240 activity executions
- 31 sensors
- Ground truth recorded manually

Metric Category	Metrics			
Two Set	Precision	Recall	F1	Bal Acc
	0.1358	0.3387	0.1939	0.6143
Event Analysis	Precision	Recall	F1	
	0.2643	0.7578	0.3919	
Other	Damerau-Lev-Norm	Cross-Correlation		
	0.725	0.7109		



Dataset: <https://zenodo.org/communities/iopit>



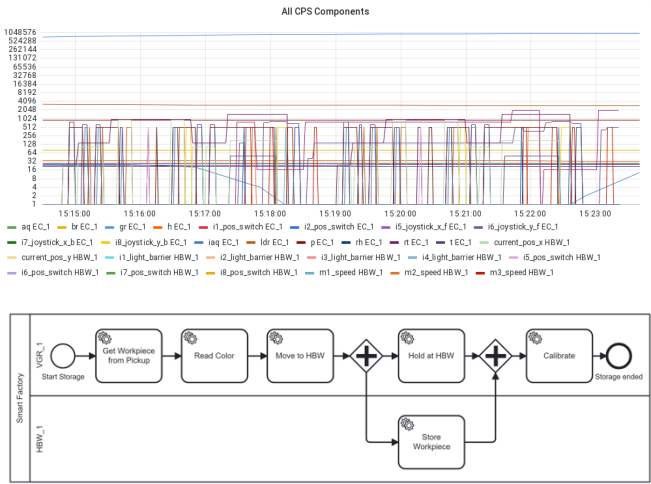
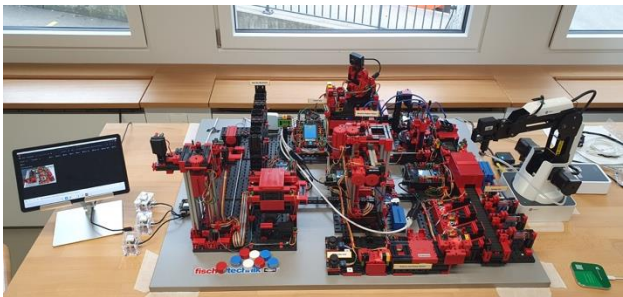
Lessons Learned

- Variations in sensor data heavily influence detection qualities
- Highly automated activities are less prone to variations vs. activities executed by humans
- Domain-specific knowledge can help to improve detections
- Start and end patterns are a must; intermediate patterns help with robustness
- Overfitting vs underfitting (depends on the activity characteristics)
- CEP as efficient and light-weight technology for detection at runtime
- Combine DSL / domain knowledge with other (semi-)automated analysis techniques



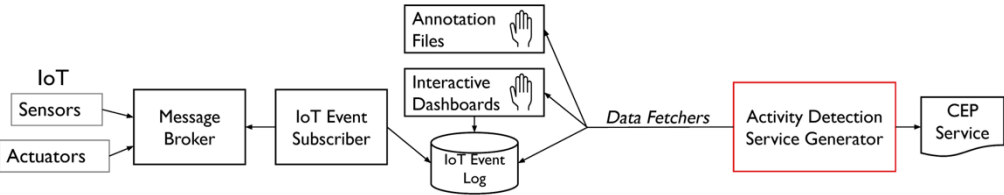
A DSL and Architecture for Detecting Process Activities from Sensor Streams in IoT

Summary



? Event abstraction

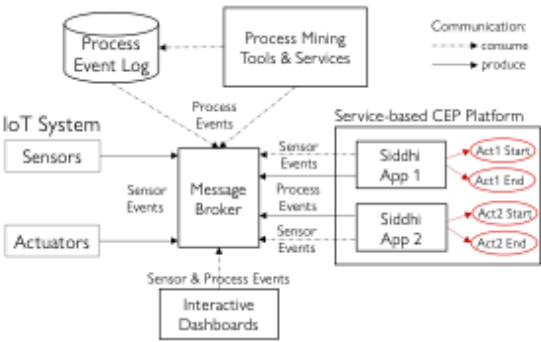
Previous Work:



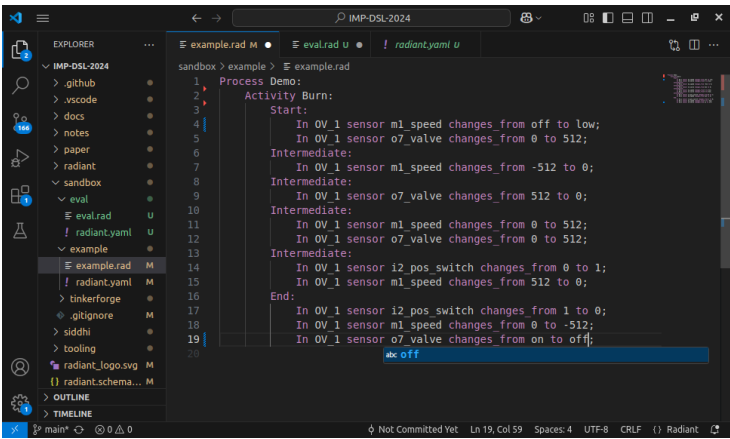
Detection at runtime



Architecture



New: Radiant DSL



Software
Architecture
Teaching and
Research Unit

