SitRS – A Situation Recognition Service based on Modeling and Executing Situation Templates
Motivation – Integrate Internet of Things..

- Increasing interconnection of IT systems and physical objects
  - Smart watches, smart phones, smart production systems, smart home technology, ...

- Huge amounts of sensors generating sensor data
  - Uniform sensor data integration and sensor data processing needed
    - Enable the Internet of Things
Motivation – Situation Recognition

Situation: An occurring event in a SMART* environment (SMART factory, SMART home...)

Examples: production machine damaged, server load critical, room occupied, room temperature increased...

Situations recognized based on sensor data

- But: huge amount of low-level, raw sensor data
  → difficult to process

- We need a means to extract high-level situations based on raw sensor data
Paper Contribution and Agenda

- Concept and architecture for a situation recognition service – SitRS
  - Cloud-based service
  - Automated sensor binding
  - Process raw sensor data to recognize occurring situations

Agenda

1. Situation modeling with Situation Templates
2. Situation recognition method
3. Situation model
4. SitRS architecture
5. Summary and outlook
Situation Detection Pyramid – Levels

1. Register situation recognition (e.g., for server)
2. Adapt to recognized situations

Knowledge Level: Situation

Situation-Model, e.g., state of server “ready” or “critical”

Information Level: Observable Context

Context-Model, e.g., Object.RamState

Data Level: Sensor Data

Low-level data types, e.g., MB of RAM

Smart Environment Level: Observable Objects (Things)

observation

notification
Situation: Detection of a critical state of a Computing node

High-Level Situation

Use-Case: Workflow-based Cloud Infrastructure Management

Logical Operations

Conditions

Context Nodes

Situation Template modeled as Situation Aggregation Tree

Critical State

OR

>90% <100MB !=200

CPU Load Free Memory Status Monitor
SitRS – Situation Recognition Method

1. Register Sensors
2. Model Situation Template
3. Situation Recognition

Technical expert
Step 3 – Detailed View

**Situation Recognition**

- **3.1 Registration**
  - Register on the situation (Situation Template + Object)
- **3.2 Transformation**
  - Map Situation Template to executable model
- **3.3 Deployment**
- **3.4 Execution**
- **3.5 Deregistration**
  - Deregister if the recognition is not needed anymore

**Executable Situation Template**

© Pascal Hirmer
Result of Situation Detection – Situation Object

Machine failure

<table>
<thead>
<tr>
<th>Observed object:</th>
<th>Optimum CNC milling machine F 210 TC-CNC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location:</td>
<td>48.743057, 9.091363</td>
</tr>
<tr>
<td>Timestamp:</td>
<td>1416489737</td>
</tr>
<tr>
<td>Description:</td>
<td>Machine is not available for production and has to be repaired.</td>
</tr>
<tr>
<td>ST-Processing:</td>
<td>ST632 – “power loss“</td>
</tr>
<tr>
<td>Recognition-System:</td>
<td>NodeRed</td>
</tr>
<tr>
<td>Recognition-Quality:</td>
<td>95%</td>
</tr>
</tbody>
</table>
Structure of a Situation Object defined by Situation Model

- **Name**: Name of the situation.
- **Icon**: Figure of the situation as a sketch.
- **Observed Object**: The ID of the object in the context model that is observed and hence described by the situation.
- **Location**: Longitude and Latitude in GPS Coordinates.
- **Observation-Timestamp**: Timestamp in UNIX time when the situation was detected.
- **Description**: Textual description of the situation.
- **ST-Processing**: Link to the situation template that was used.
- **Recognition-System**: System that was used to implement and execute the situation recognition.
- **Quality**: A measurement of the overall quality of the conducted situation recognition process.

**Machine failure**

- **Observed object**: Optimum CNC milling machine F 210 TC-CNC
- **Location**: 48.743057, 9.091363
- **Timestamp**: 1416489737
- **Description**: Machine is not available for production and has to be repaired.
- **ST-Processing**: ST632 – “power loss”
- **Recognition-System**: NodeRed
- **Recognition-Quality**: 95%
SitRS – Architecture

Layer 3: Situation

Situation Model

S1 \ldots Sn

Situation Objects

Layer 2: Situation Recognition (SitRS)

Situation Recognition System

Situation Registration Service

oID+ST

Mapping

Situation Template Repository

Resource Management Platform (RMP)

Create Resource

Sensor Registry

Situation Recognition Service

Sensor Adapter 1 \ldots Sensor Adapter n

Layer 1: Sensing

Physical Objects with Sensors

Tools

Transport

Material

Machines

Production

Situation Objects
Use case – Situational Control Flow Modeling (I)

- Analysts and Planners can model their process as standard workflow
- Domain experts can model their knowledge as Situation Template for situation recognition
  - “Whenever event 5 happens always situation X occurs “
- Together situational exceptions can be modeled
  - Annotate situation to standard workflow
  - Handling different situations with situational workflows
Use case – Situational Control Flow Modeling (II)

- Result: Situation-aware Workflow

Start production → Provide material → Make product → Write invoice → Deliver product

Situational control flow:
- Machine failure
- Material missing
Prototypical Implementation

Situation Registration Service

Physical Sensors

OSLC REST Resources

OSLC Service Layer

Situation Recognition Service

Node-RED

OSLC-based Resource Management Platform

OSLC Service Provider

Data Cache

OSLC Adapter 1

OSLC Adapter n

XML File Store

Java
JAXB
JSON

.mapping

Sensor Registry

Create Resource

Register Sensor

Situation Template Repository

Mapping

Create Resource

Register Sensor

1 2 3 4

push

pull

fID

oID+ST
Load Test of the SitRS Prototype

<table>
<thead>
<tr>
<th>Measurement</th>
<th>ST Transformation</th>
<th>ST Deployment</th>
<th>ST Execution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>219 ms</td>
<td>141 ms</td>
<td>6 ms</td>
</tr>
<tr>
<td>2</td>
<td>219 ms</td>
<td>126 ms</td>
<td>6 ms</td>
</tr>
<tr>
<td>3</td>
<td>234 ms</td>
<td>125 ms</td>
<td>5 ms</td>
</tr>
<tr>
<td>4</td>
<td>203 ms</td>
<td>141 ms</td>
<td>5 ms</td>
</tr>
<tr>
<td>5</td>
<td>204 ms</td>
<td>140 ms</td>
<td>6 ms</td>
</tr>
<tr>
<td>Ø</td>
<td>215,8 ms</td>
<td>134,6 ms</td>
<td>5,6 ms</td>
</tr>
</tbody>
</table>

Use-Case: Workflow-based Cloud Infrastructure Management ➔
Summary and Outlook

SitRS is a general purpose, situation recognition service that can be used in different use-cases

- **Summary**
  - Defined *Situation Templates* to model situations based on the levels of the *Situation Detection Pyramid*
  - Defined a *Situation Model* for defining *Situation Objects* characterizing the state of the environment
  - Showed a way to integrate different processing technologies
    - Data streaming, Complex Event Processing, Internet of Things technologies
  - Initial goal: Recognize situations based on raw sensor data achieved

- **Planned future work**
  - Integrate other event and data stream processing technologies
  - Ontology based sensor registration and integration
  - Automatic situation recognition based on historic data to learn situations
  - Formalization of the SitRS approach
SitRS – A Situation Recognition Service based on Modeling and Executing Situation Templates
SMART factories are an important part within the “Industrie 4.0” movement
- Highly interconnected machines and robots work together to manufacture a product

SitRS for SMART factories
- Goal: recognize error situations, material shortage etc. & react automatically
- Connect (machine) sensors to SitRS
- Monitor the SMART factory using situation recognition
- React on occurring situations accordingly