CAMUNDA CON LIVE

Controlling a Smart Factory with Processes in Camunda

Ronny Seiger, University of St. Gallen

University of St.Gallen

Universität Trier

German Research Center for Artificial Intelligence

Controlling a Smart Factory with Processes in Camunda

Ronny Seiger, Barbara Weber
University of St.Gallen, Switzerland

Lukas Malburg, Ralph Bergmann
University of Trier, Germany
German Research Center for AI (DFKI)

CamundaCon Live 2021
22 September 2021

"From insight to impact"

University of St.Gallen

Universität Trier

German Research Center for Artificial Intelligence

Ronny Seiger

- Diploma (2011) & PhD (2018) in Computer Science at TU Dresden
- Post-Doc at University of St.Gallen since 2019
- Research
  - Cyber-physical Systems, Internet of Things
  - Software Engineering, Software Architecture
  - Business Process Management
- Teaching
  - BPM, Software Engineering

ronny.seiger@unisg.ch

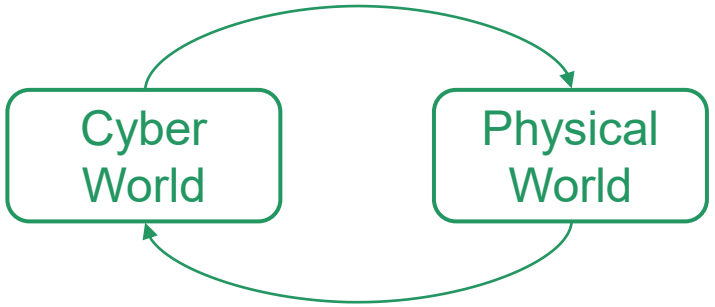
University of St.Gallen

Universität Trier

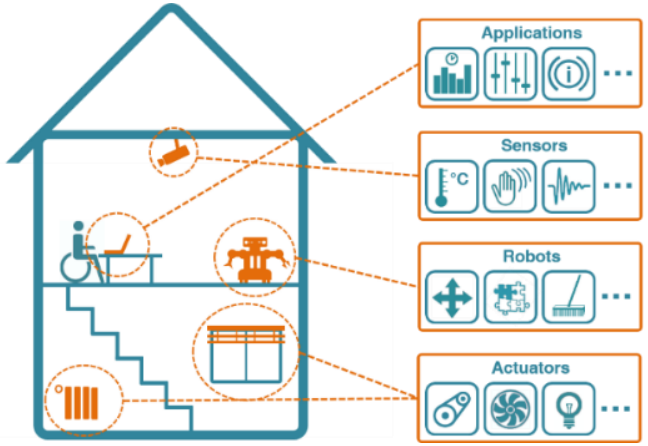
German Research Center for Artificial Intelligence

Emergence of Smart Spaces

Cyber-physical Convergence



Cyber-physical System

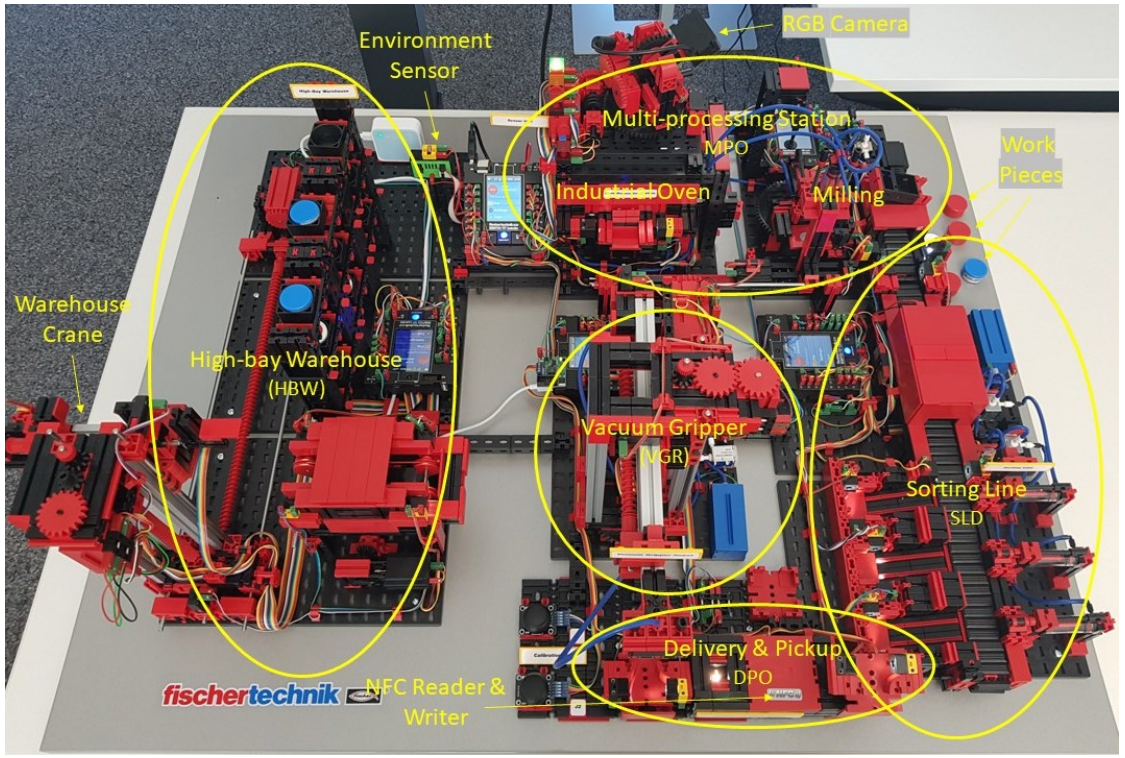


University of St.Gallen

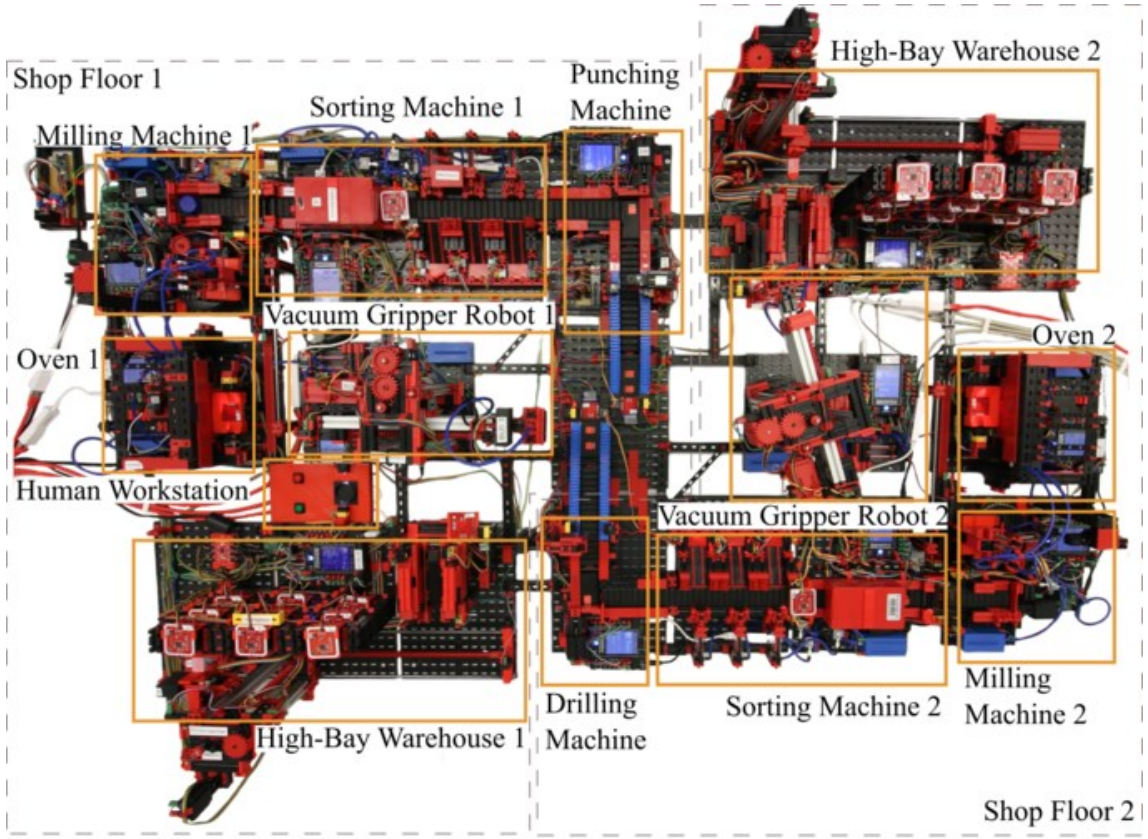
Universität Trier

German Research Center for Artificial Intelligence

Smart Factory Simulation Models



Uni St.Gallen



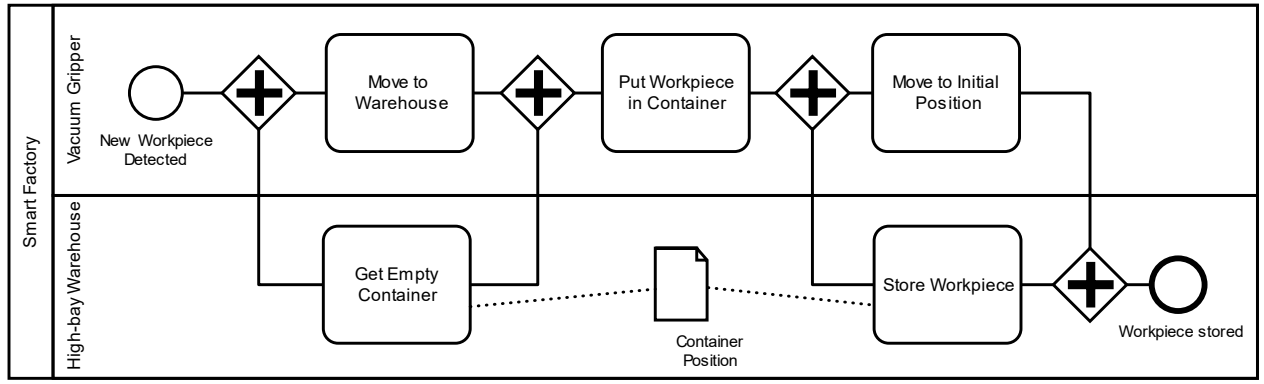
Uni Trier

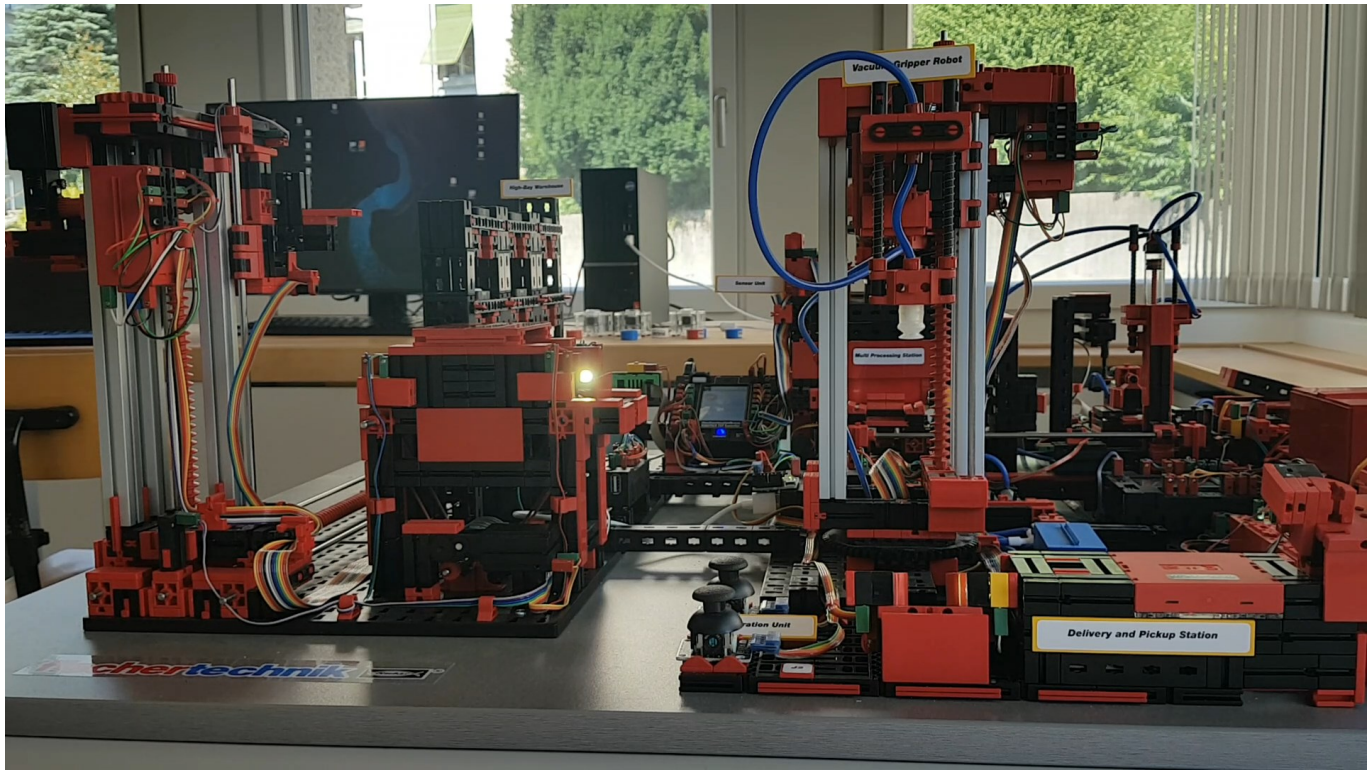
University of St.Gallen

Universität Trier

German Research Center for Artificial Intelligence

Smart Factory Process: Storage



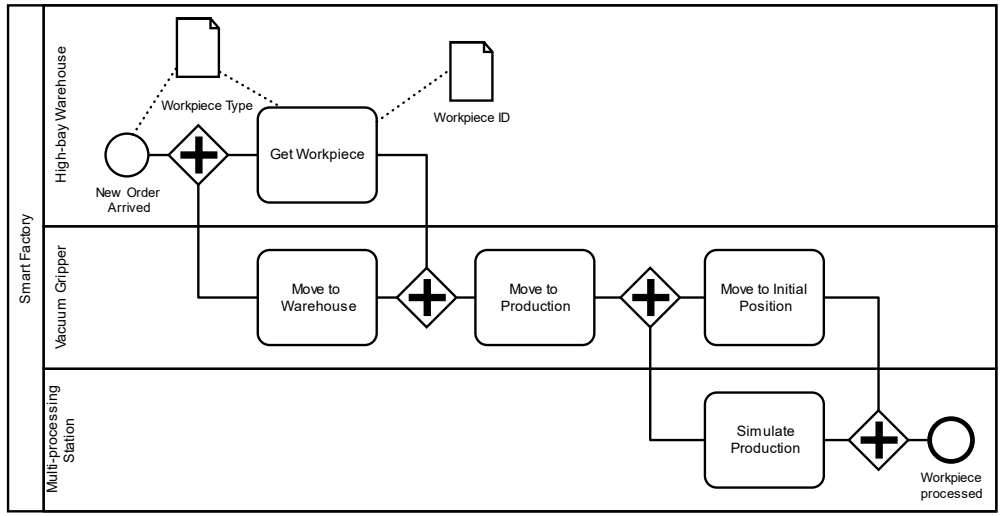


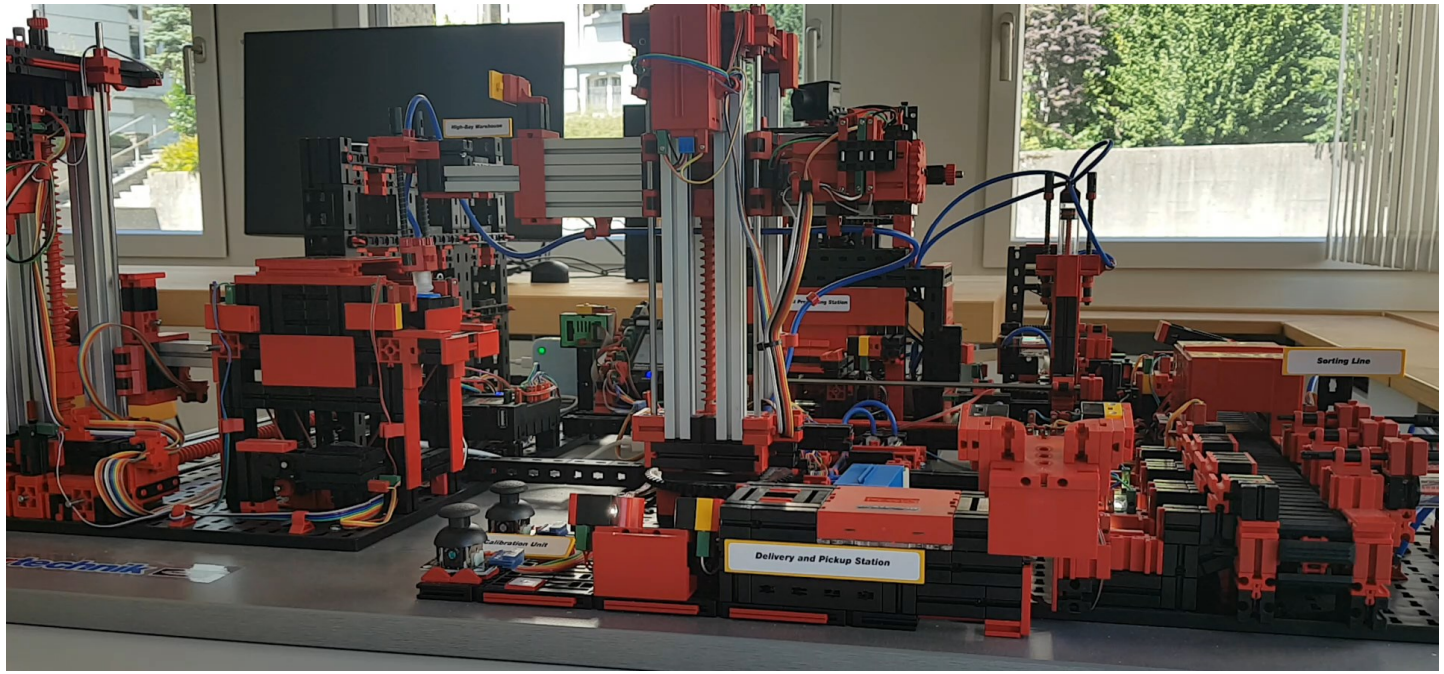
University of St.Gallen

Universität Trier

German Research Center for Artificial Intelligence

Smart Factory Process: Production





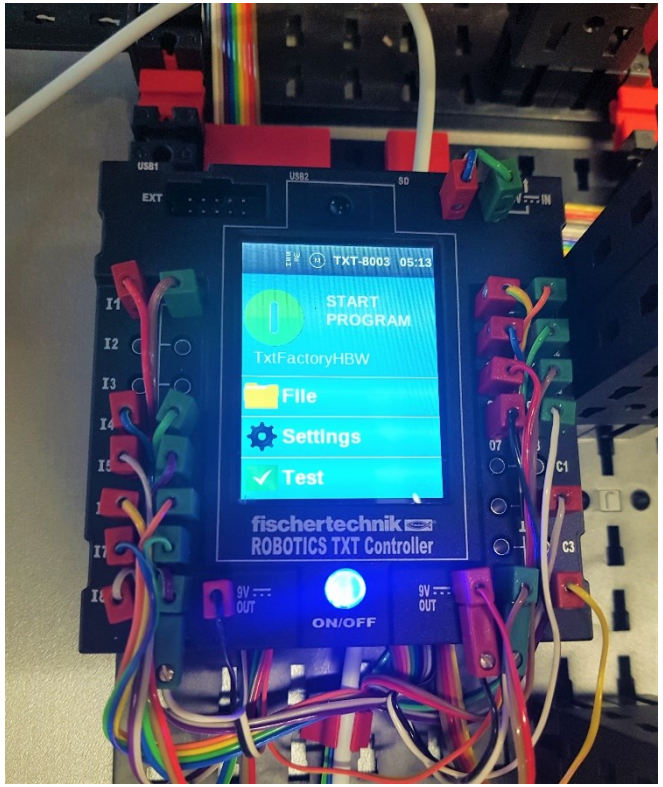
University of St.Gallen

Universität Trier

German Research Center for Artificial Intelligence

Programming the Factories: As-is State

```
#elif CLIENT_HBW
		if (msg->get_topic() == TOPIC_OUTPUT_STATE_ACK) {
			SPDLOG_LOGGER_DEBUG(spdlog::get("console"), "DETECTED state ack:{}", msg->get_topic());
			std::stringstream ssin(msg->to_string());
			Json::Value root;
			try {
				ssin >> root;
				std::string sts = root["ts"].asString();
				SPDLOG_LOGGER_DEBUG(spdlog::get("console"), "  ts:{}", sts);
				if (ft::trycheckTimestampTTL(sts))
				{
					hbw_.requestQuit();
				}
			} catch (const Json::RuntimeError& exc) {
				std::cout << "Error: " << exc.what() << std::endl;
			}
			SPDLOG_LOGGER_DEBUG(spdlog::get("console"), "OK.", 0);
		} else if (msg->get_topic() == TOPIC_LOCAL_SSC_JOY) {
			SPDLOG_LOGGER_DEBUG(spdlog::get("console"), "DETECTED joy:{}", msg->get_topic());
			std::stringstream ssin(msg->to_string());
			Json::Value root;
			try {
				ssin >> root;
				std::string sts = root["ts"].asString();
				SPDLOG_LOGGER_DEBUG(spdlog::get("console"), "  ts:{}", sts);
				if (ft::trycheckTimestampTTL(sts))
				{
					ft::TxtJoysticksData jd;
					jd.aX1 = root["aX1"].asInt();
					jd.aY1 = root["aY1"].asInt();
					jd.b1 = root["b1"].asBool();
					jd.aX2 = root["aX2"].asInt();
					jd.aY2 = root["aY2"].asInt();
					jd.b2 = root["b2"].asBool();
					hbw_.requestJoyBut(jd);
					SPDLOG_LOGGER_DEBUG(spdlog::get("console"), "  1:{} {} {} 2:{} {} {}", jd.aX1, jd.aY1, jd.b1, jd.aX2,
				}
			} catch (const Json::RuntimeError& exc) {
				std::cout << "Error: " << exc.what() << std::endl;
			}
			SPDLOG_LOGGER_DEBUG(spdlog::get("console"), "OK.", 0);
		} else if (msg->get_topic() == TOPIC_LOCAL_VGR_DO) {
			SPDLOG_LOGGER_DEBUG(spdlog::get("console"), "DETECTED vgr do:{}", msg->get_topic());
```



6–12 Embedded TXT Controllers

- 1 C++ application / controller
- 2 hard-wired processes

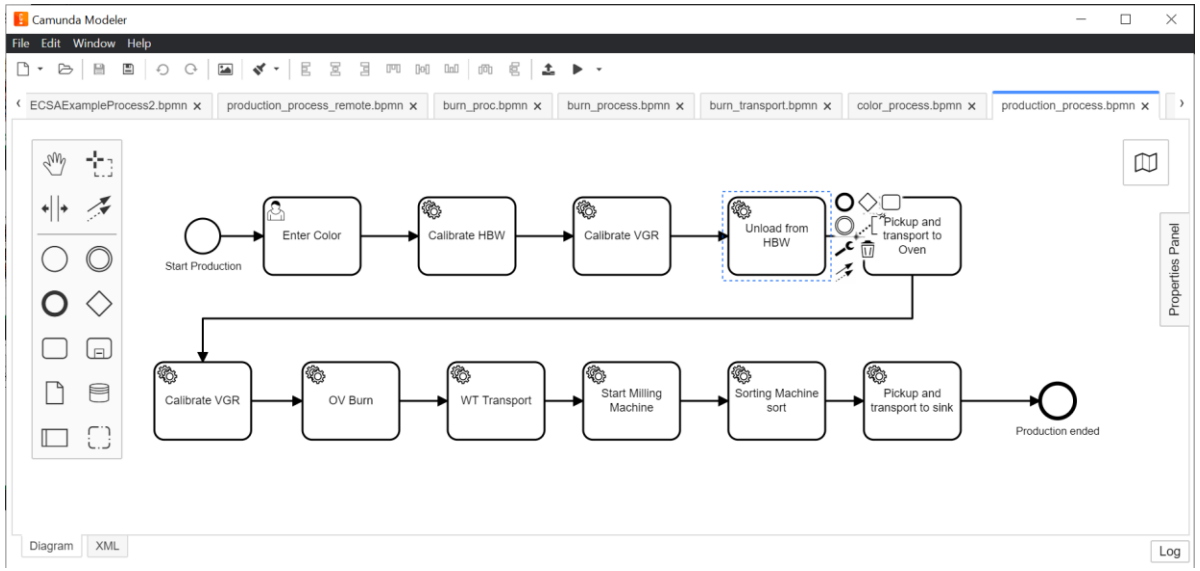
https://github.com/fischertechnik/txt_training_factory

University of St.Gallen

Universität Trier

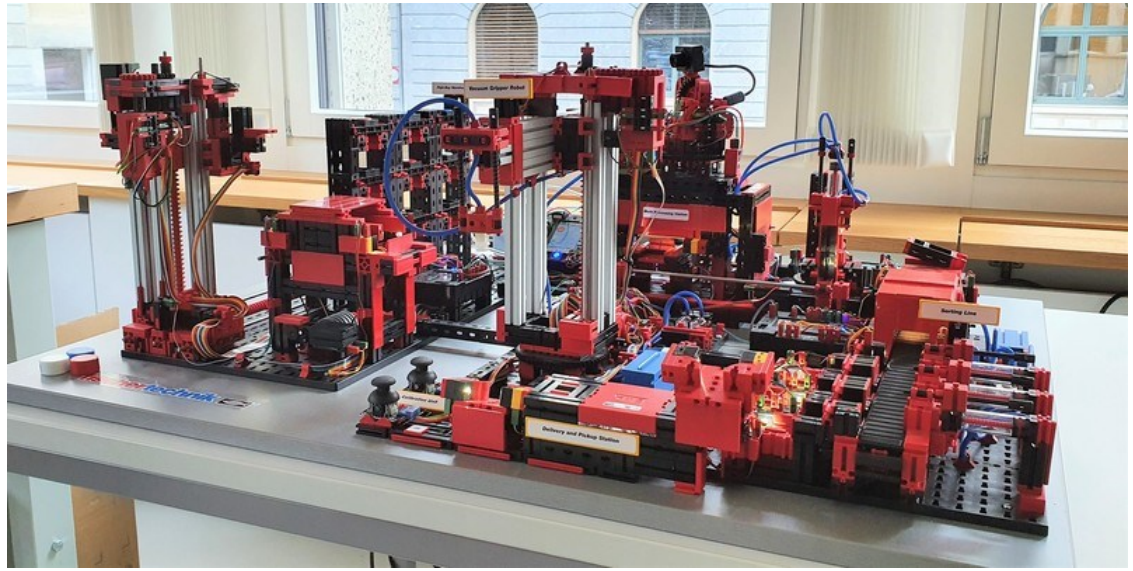
German Research Center for Artificial Intelligence

«Programming» the Factories: To-be State



Model Process in BPMN in Camunda Modeler (*low-code*)

Execute via Camunda Platform



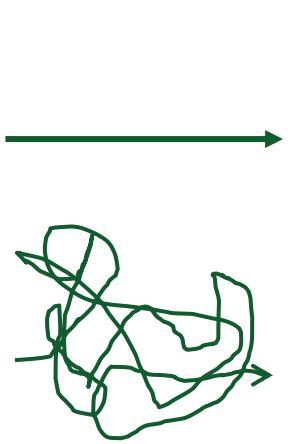
University of St.Gallen

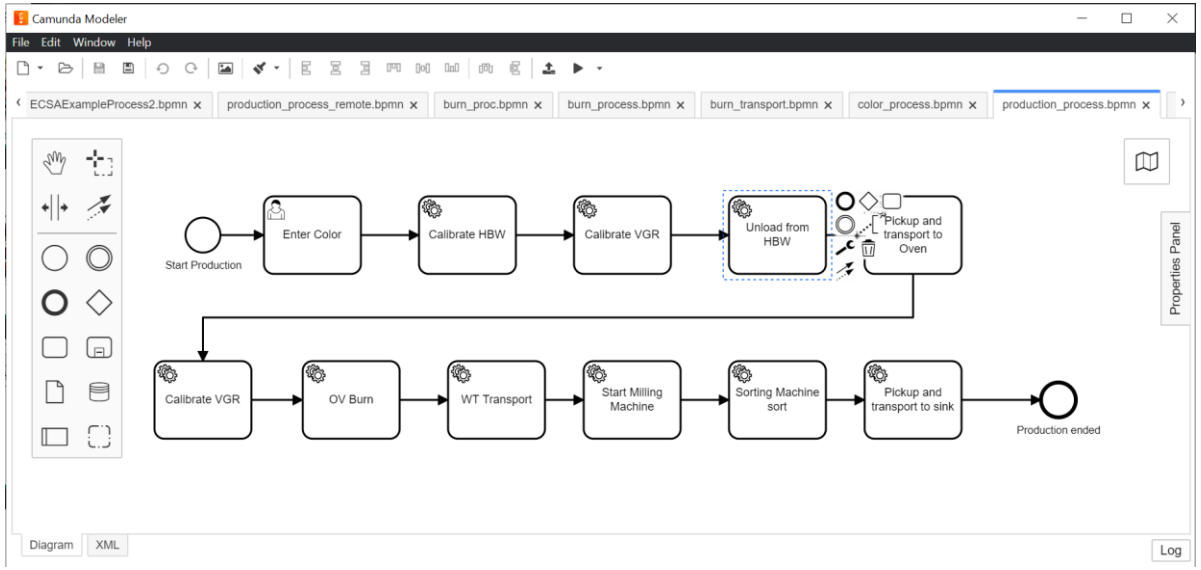
Universität Trier

German Research Center for Artificial Intelligence

The Journey

```
elif CLIENT_HBW
        if (msg->get_topic() == TOPIC_OUTPUT_STATE_ACK) {
                SPDLOG_LOGGER_DEBUG(spdlog::get("console"), "DETECTED state ack:{}", msg->get_topic());
                std::stringstream ssin(msg->to_string());
                Json::Value root;
                try {
                        ssin >> root;
                        std::string sts = root["ts"].asString();
                        SPDLOG_LOGGER_DEBUG(spdlog::get("console"), "  ts:{}", sts);
                        if (ft::trycheckTimestampTTL(sts))
                        {
                                hbw_.requestQuit();
                        }
                } catch (const Json::RuntimeError& exc) {
                        std::cout << "Error: " << exc.what() << std::endl;
                }
                SPDLOG_LOGGER_DEBUG(spdlog::get("console"), "OK.", 0);
        } else if (msg->get_topic() == TOPIC_LOCAL_SSC_JOY) {
                SPDLOG_LOGGER_DEBUG(spdlog::get("console"), "DETECTED joy:{}", msg->get_topic());
                std::stringstream ssin(msg->to_string());
                Json::Value root;
                try {
                        ssin >> root;
                        std::string sts = root["ts"].asString();
                        SPDLOG_LOGGER_DEBUG(spdlog::get("console"), "  ts:{}", sts);
                        if (ft::trycheckTimestampTTL(sts))
                        {
                                ft::TxtJoysticksData jd;
                                jd.aX1 = root["aX1"].asInt();
                                jd.aY1 = root["aY1"].asInt();
                                jd.b1 = root["b1"].asBool();
                                jd.aX2 = root["aX2"].asInt();
                                jd.aY2 = root["aY2"].asInt();
                                jd.b2 = root["b2"].asBool();
                                hbw_.requestJoyBut(jd);
                                SPDLOG_LOGGER_DEBUG(spdlog::get("console"), "  1:{} {} {} 2:{} {} {}", jd.aX1, jd.aY1, jd.b1, jd.aX2
                        }
                } catch (const Json::RuntimeError& exc) {
                        std::cout << "Error: " << exc.what() << std::endl;
                }
                SPDLOG_LOGGER_DEBUG(spdlog::get("console"), "OK.", 0);
        } else if (msg->get_topic() == TOPIC_LOCAL_VGR_DO) {
```



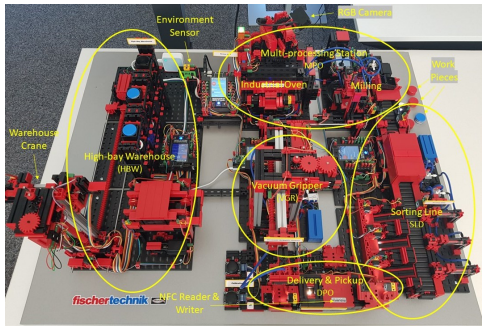


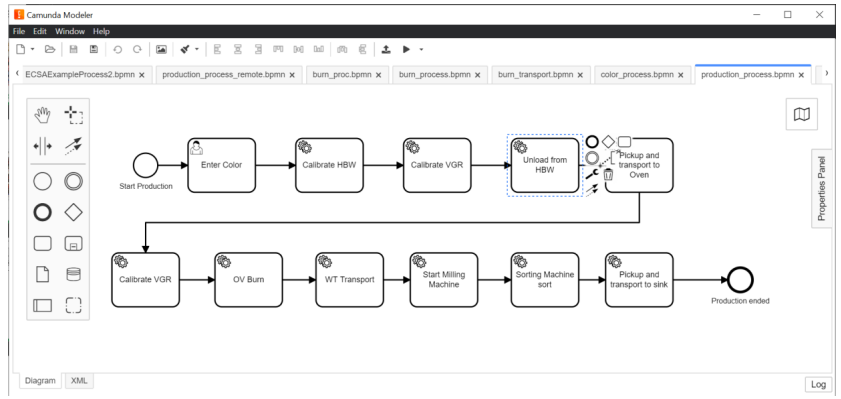
University of St.Gallen

Universität Trier

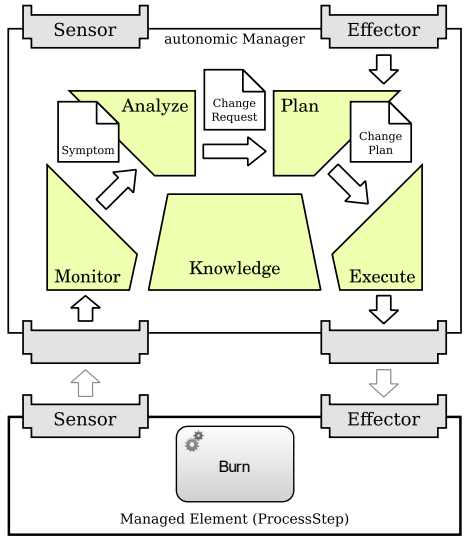
German Research Center for Artificial Intelligence

Agenda





- Hardware
- Programming
  - Low-level & High-level
- Web Services & Business Processes
- Sensors
  - Stream Processing & Machine Learning
- Research: Autonomous Processes

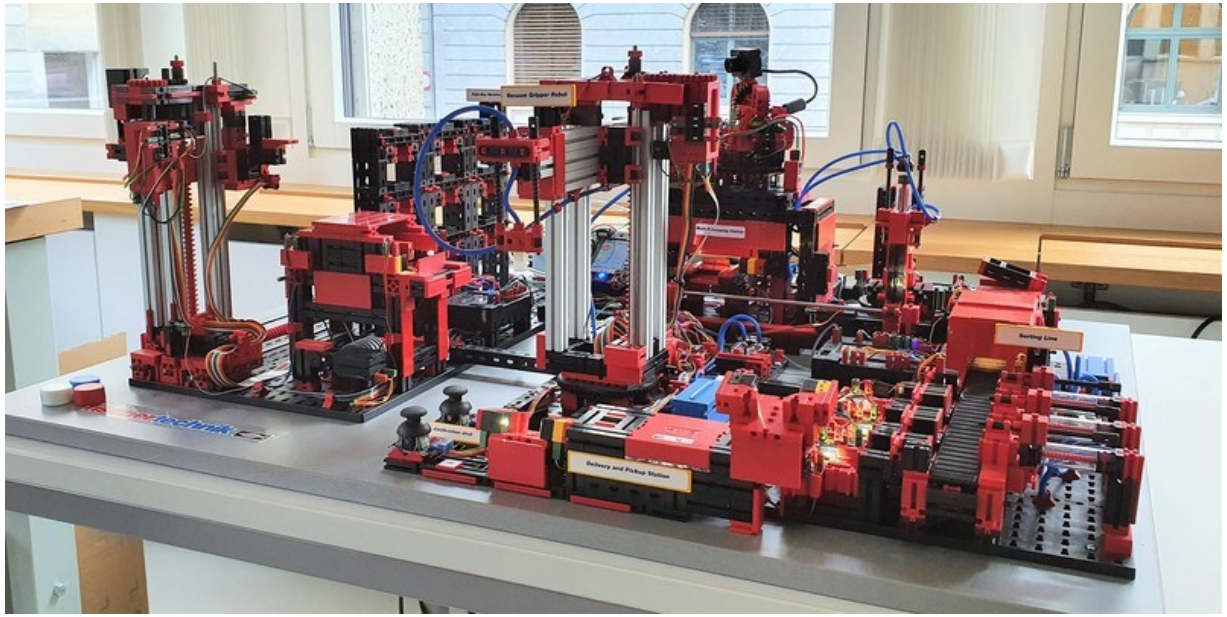


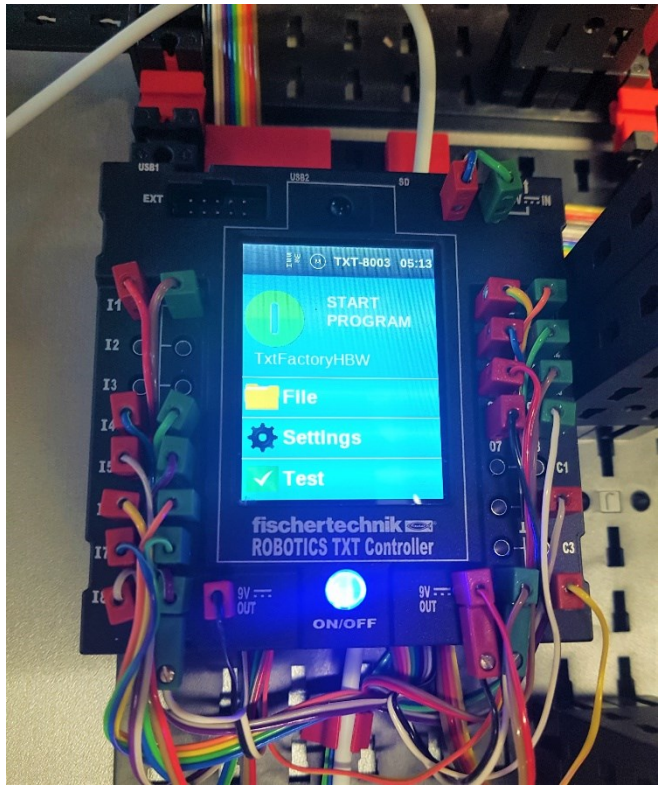
University of St.Gallen

Universität Trier

German Research Center for Artificial Intelligence

The Embedded TXT Controllers: Hardware





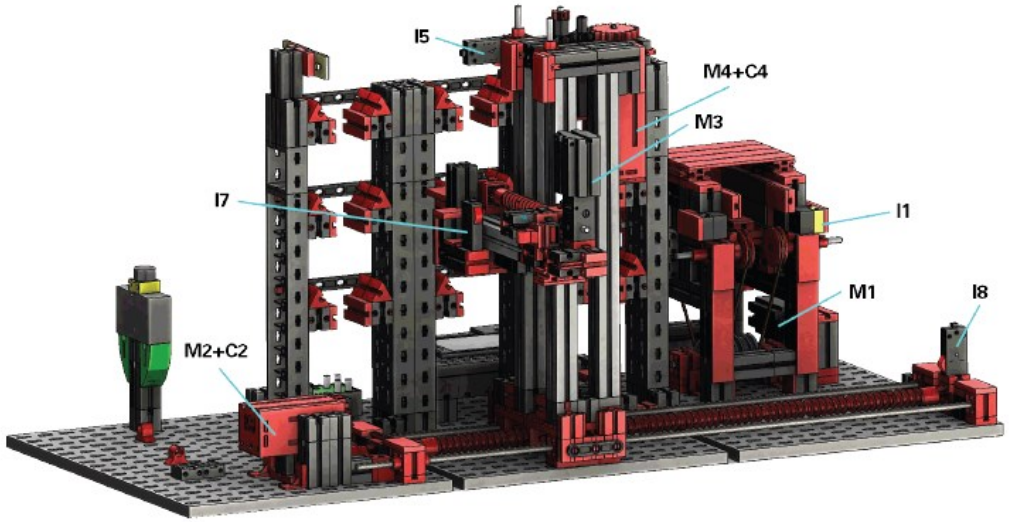
University of St.Gallen

Universität Trier

German Research Center for Artificial Intelligence

Know Your Wiring!

Number	Function	Input / output
1	Photo-transistor external	I1
2	Photo-transistor internal	I4
3	Reference pushbutton horizontal	I5
4	Reference pushbutton probe arm rear	I6
5	Reference pushbutton probe arm front	I7
6	Reference pushbutton vertical	I8
7	Encoder horizontal	C2
8	Encoder vertical	C4
9	Motor conveyor belt	M1
10	Horizontal motor	M2
11	Motor probe arm	M3
12	Vertical motor	M4



from https://www.fischertechnik.de/-/media/fischertechnik/fite/service/elearning/lehren/lernfabrik/fabrik_2019_englisch_neu.ashx

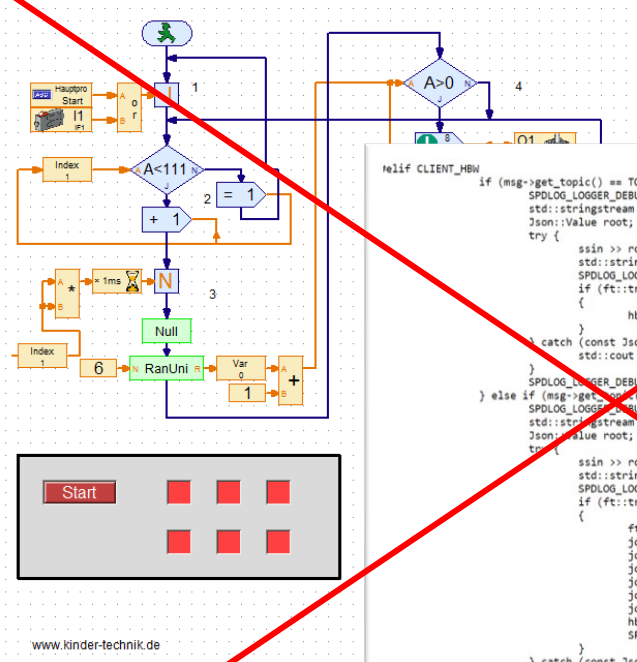
Hardware ✓

University of St.Gallen

Universität Trier

German Research Center for Artificial Intelligence

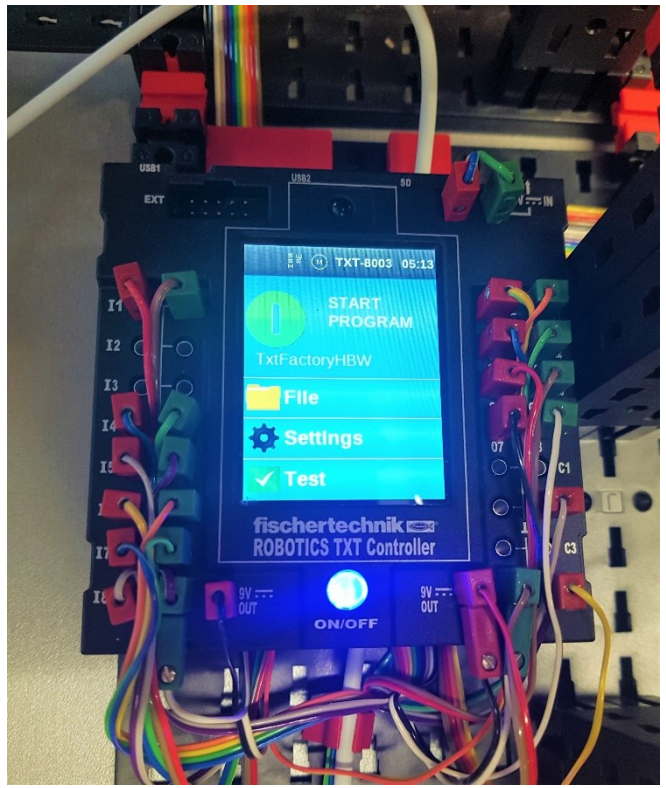
The Embedded TXT Controllers: Software



Fischertechnik RoboPro

```
#elif CLIENT_HBW
        if (msg->get_topic() == TOPIC_OUTPUT_STATE_ACK) {
                SPDLOG_LOGGER_DEBUG(spdlog::get("console"), "DETECTED state ack:{}", msg->get_topic());
                std::stringstream ssin(msg->to_string());
                Json::Value root;
                try {
                        ssin >> root;
                        std::string sts = root["ts"].asString();
                        SPDLOG_LOGGER_DEBUG(spdlog::get("console"), "  ts:{}", sts);
                        if (ft::trycheckTimestampTTL(sts))
                        {
                                hbw_.requestQuit();
                        }
                } catch (const Json::RuntimeError& exc) {
                        std::cout << "Error: " << exc.what() << std::endl;
                }
                SPDLOG_LOGGER_DEBUG(spdlog::get("console"), "OK.", 0);
        } else if (msg->get_topic() == TOPIC_LOCAL_SSC_JOY) {
                SPDLOG_LOGGER_DEBUG(spdlog::get("console"), "DETECTED joy:{}", msg->get_topic());
                std::stringstream ssin(msg->to_string());
                Json::Value root;
                try {
                        ssin >> root;
                        std::string sts = root["ts"].asString();
                        SPDLOG_LOGGER_DEBUG(spdlog::get("console"), "  ts:{}", sts);
                        if (ft::trycheckTimestampTTL(sts))
                        {
                                ft::TxtJoysticksData jd;
                                jd.aX1 = root["aX1"].asInt();
                                jd.aY1 = root["aY1"].asInt();
                                jd.b1 = root["b1"].asBool();
                                jd.aX2 = root["aX2"].asInt();
                                jd.aY2 = root["aY2"].asInt();
                                jd.b2 = root["b2"].asBool();
                                hbw_.requestJoyBut(jd);
                                SPDLOG_LOGGER_DEBUG(spdlog::get("console"), "  1:{} 2:{} {} {}", jd.aX1, jd.aY1, jd.b1, jd.aX2
                        }
                } catch (const Json::RuntimeError& exc) {
                        std::cout << "Error: " << exc.what() << std::endl;
                }
                SPDLOG_LOGGER_DEBUG(spdlog::get("console"), "OK.", 0);
        } else if (msg->get_topic() == TOPIC_LOCAL_VGR_DO) {
```

Fischertechnik C++ Project



fischertechnik TXT community firmware

ftrobopy

https://github.com/ftrobopy/ftrobopy

University of St.Gallen

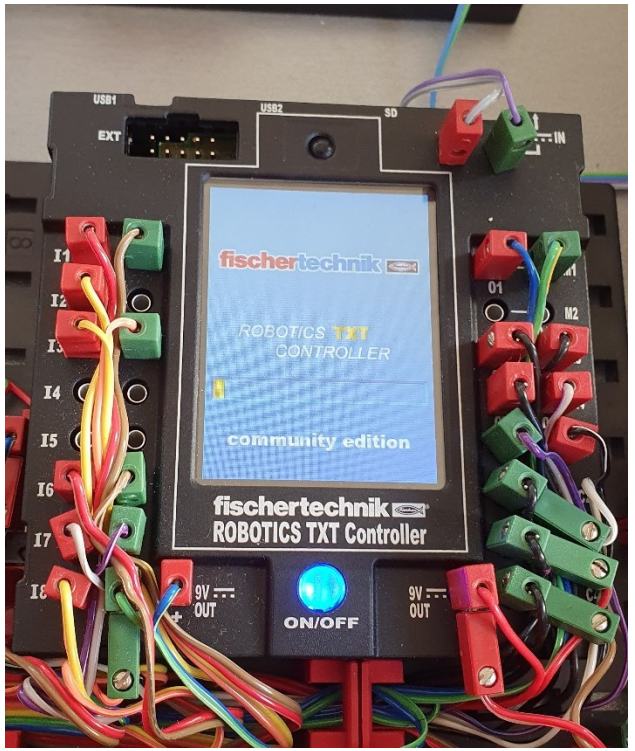
Universität Trier

German Research Center for Artificial Intelligence

The Embedded TXT Controllers: Software

ftrobopy

- Unix Process on controller
  - Provides TCP Sockets
- Python library
  - Deploy locally or on remote computer




```
1  import ftrobopy
2
3  my_txt1 = ftrobopy.ftrobopy(host=f"192.168.0.12", port=65000, update_interval=0.01)
4
```

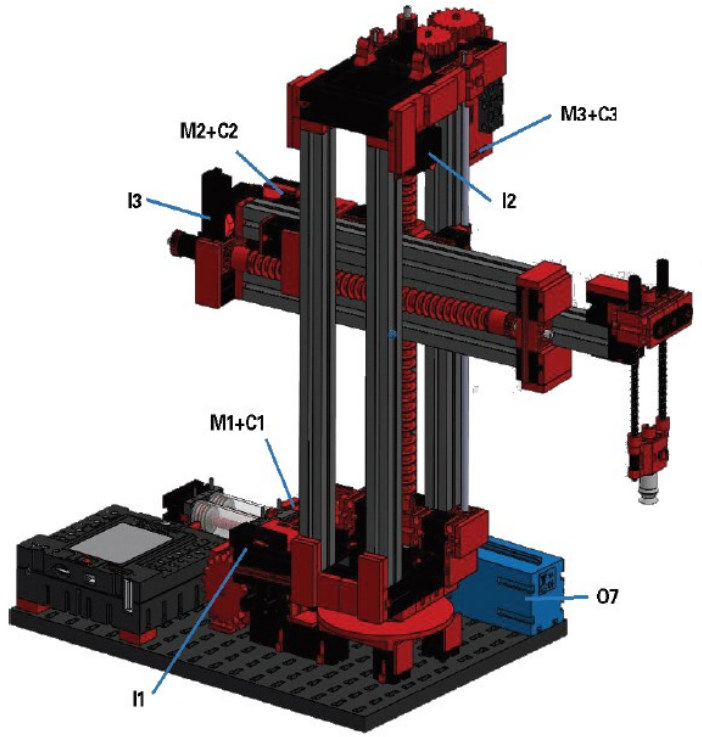
University of St.Gallen

Universität Trier

German Research Center for Artificial Intelligence

Bringing Hardware & Software Together

```
5   m1 = my_txt1.motor(1)
6   m1.setSpeed(512)
7
8   m2 = my_txt1.motor(2)
9   m2.setSpeed(-256)
10
11  o7 = my_txt1.output(7)
12  o7.setLevel(512)
13
14  i3 = my_txt1.input(3)
15  if i3.state() == 1:
16      print("Switch i3 pressed")
```

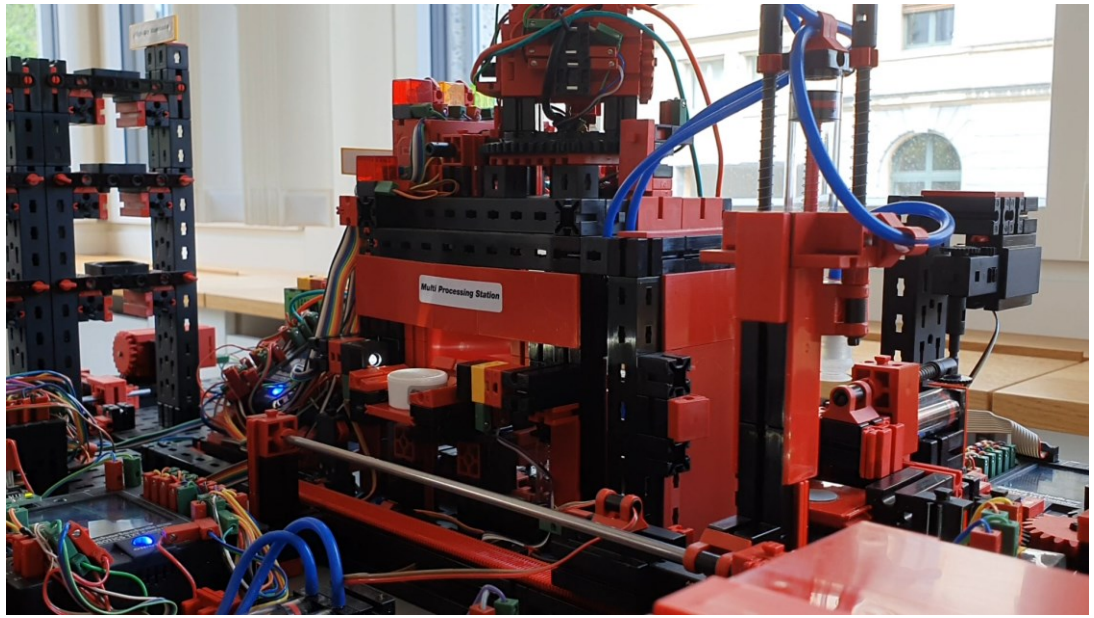


University of St.Gallen

Universität Trier

German Research Center for Artificial Intelligence

Small Example: Oven



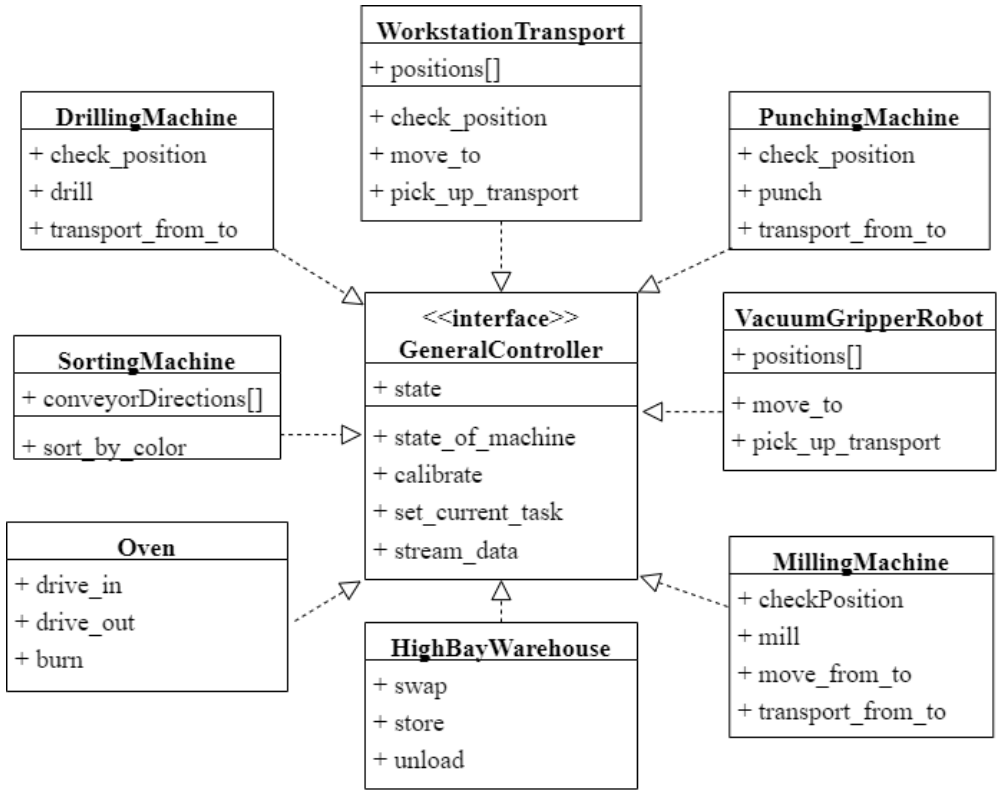
```
import ftrobopy

oven_txt = ftrobopy.ftrobopy(host=f"192.168.0.14",
                             port=65000, update_interval=0.01)
o7 = oven_txt.output(7)
m1 = oven_txt.motor(1)
i1 = oven_txt.input(1)

# Open Oven door
o7.setLevel(512)

# Drive in slide
m1.setSpeed(-512)
while i1.state() == 0:
    pass
m1.stop()

# Close Oven door
o7.setLevel(0)
```

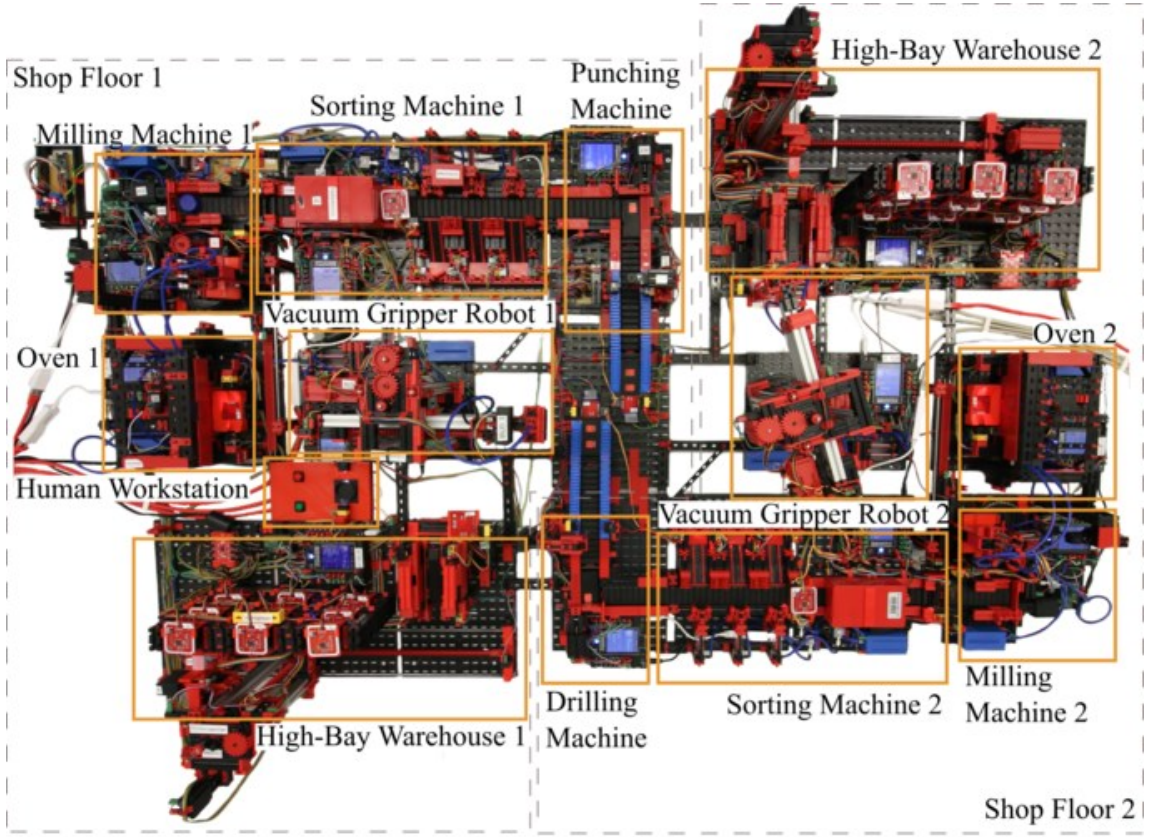
University of St.Gallen

Universität Trier

German Research Center for Artificial Intelligence

Going Object-oriented



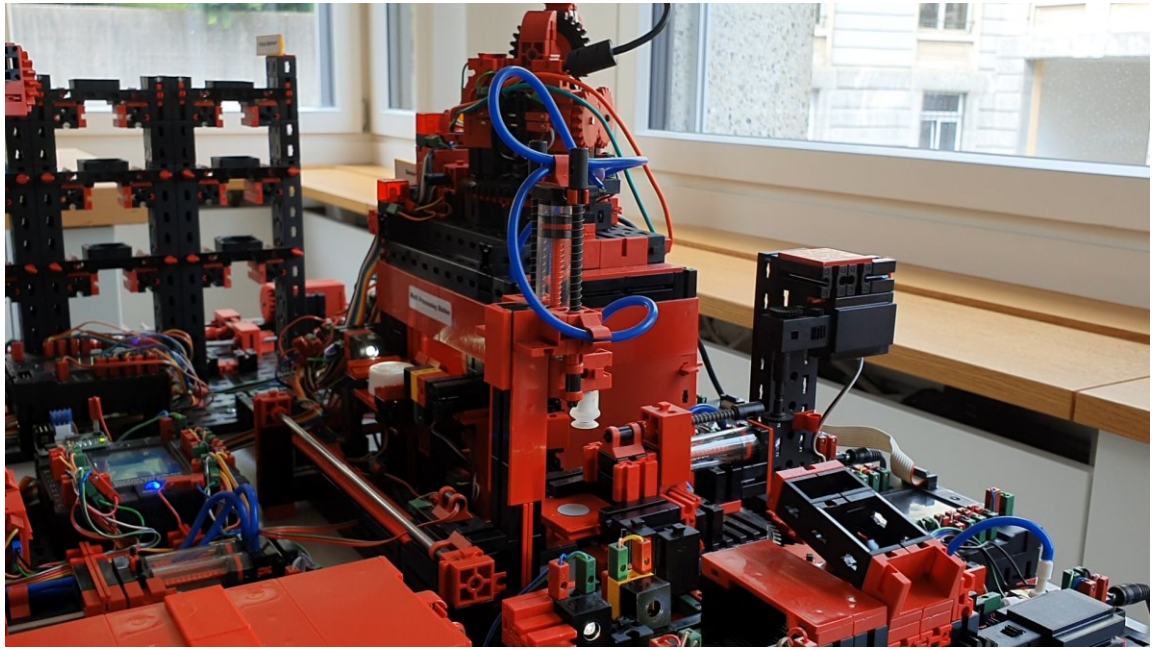


University of St.Gallen

Universität Trier

German Research Center for Artificial Intelligence

Going Object-oriented: Example



```
oven1 = Oven(12)
oven1.burn(5)

class Oven:
    def __init__(self, txt_number):
    ...

    def burn(self, time_in_seconds: int = 2) -> None:
        self._drive_into_oven()
        self.txt.play_sound(15)
        for i in range(time_in_seconds*8):
            if i%2==0:
                self.o8.setLevel(512)
            else: self.o8.setLevel(0)
            time.sleep(0.3)
        self.o8.setLevel(0)
        self._drive_out_of_oven()
        self._adjust_platform_for_workstation_transport()
        return
    ...

    def _adjust_platform_for_workstation_transport(self) -> None:
        self.m1.setDistance(100)
        self.m1.setSpeed(-1 * self.m1_speed)
        time.sleep(0.3)
        self.m1.stop()
        return
```

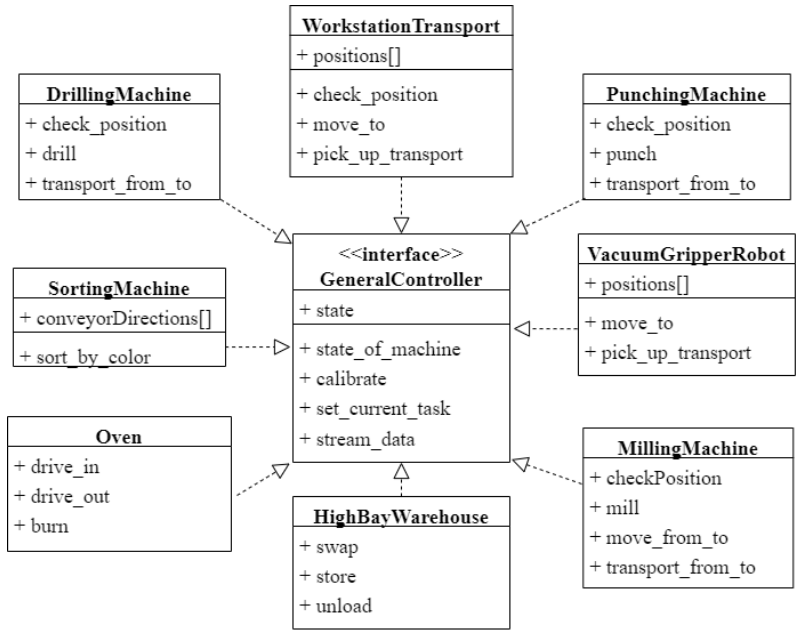
University of St.Gallen

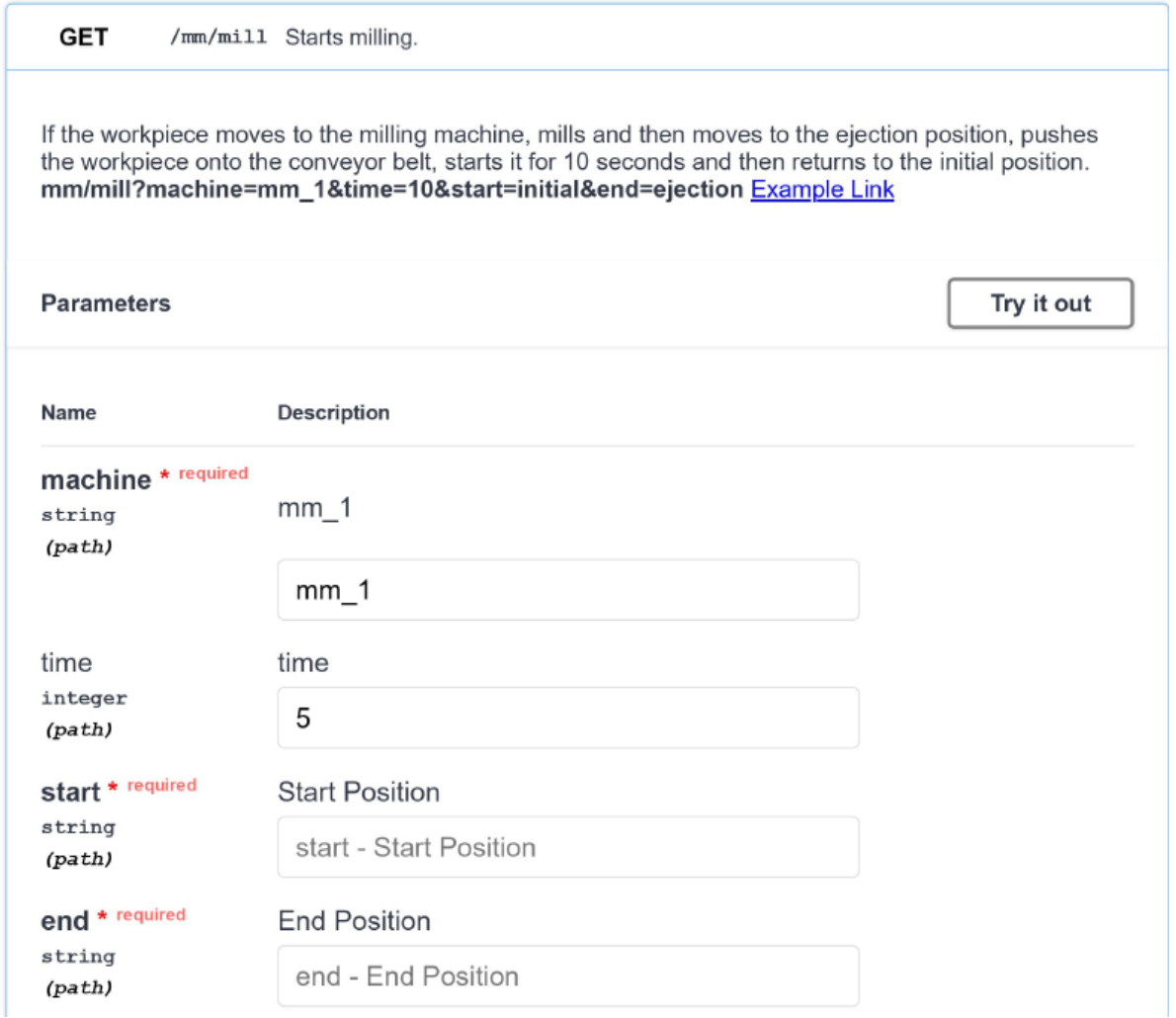
Universität Trier

German Research Center for Artificial Intelligence

Going Service-based

- Expose high-level methods of OOP model 1:1 via RESTful web service
- HTTP GET with parameters
- Python Flask



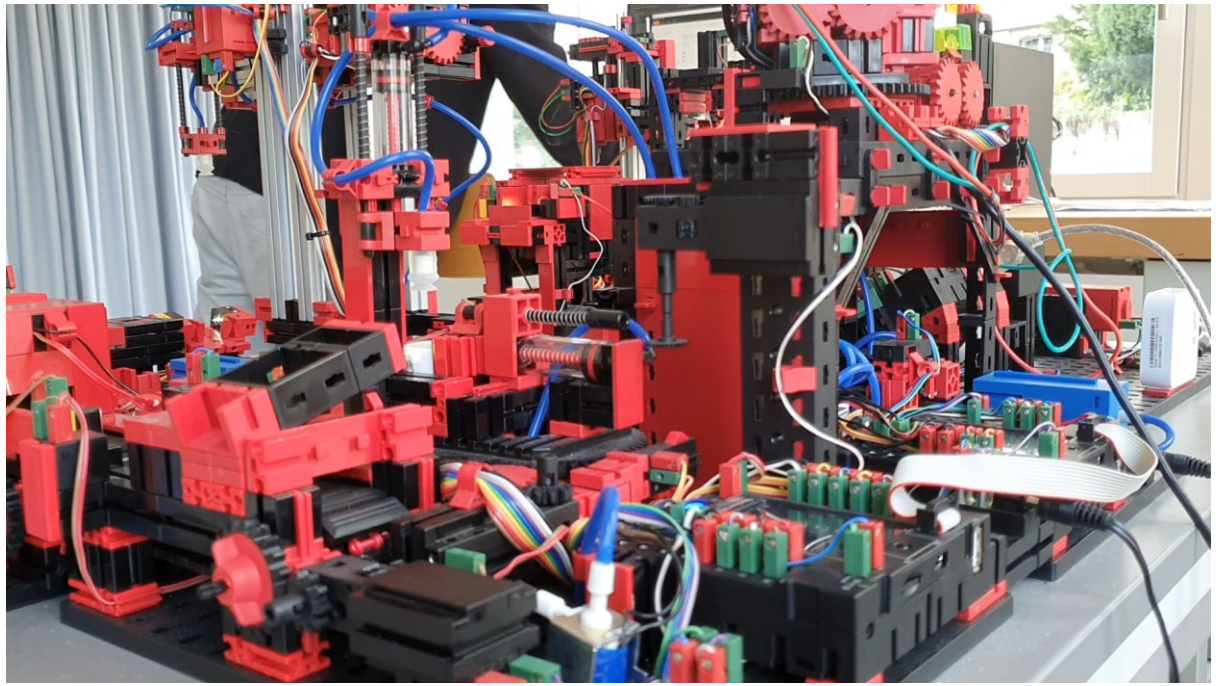


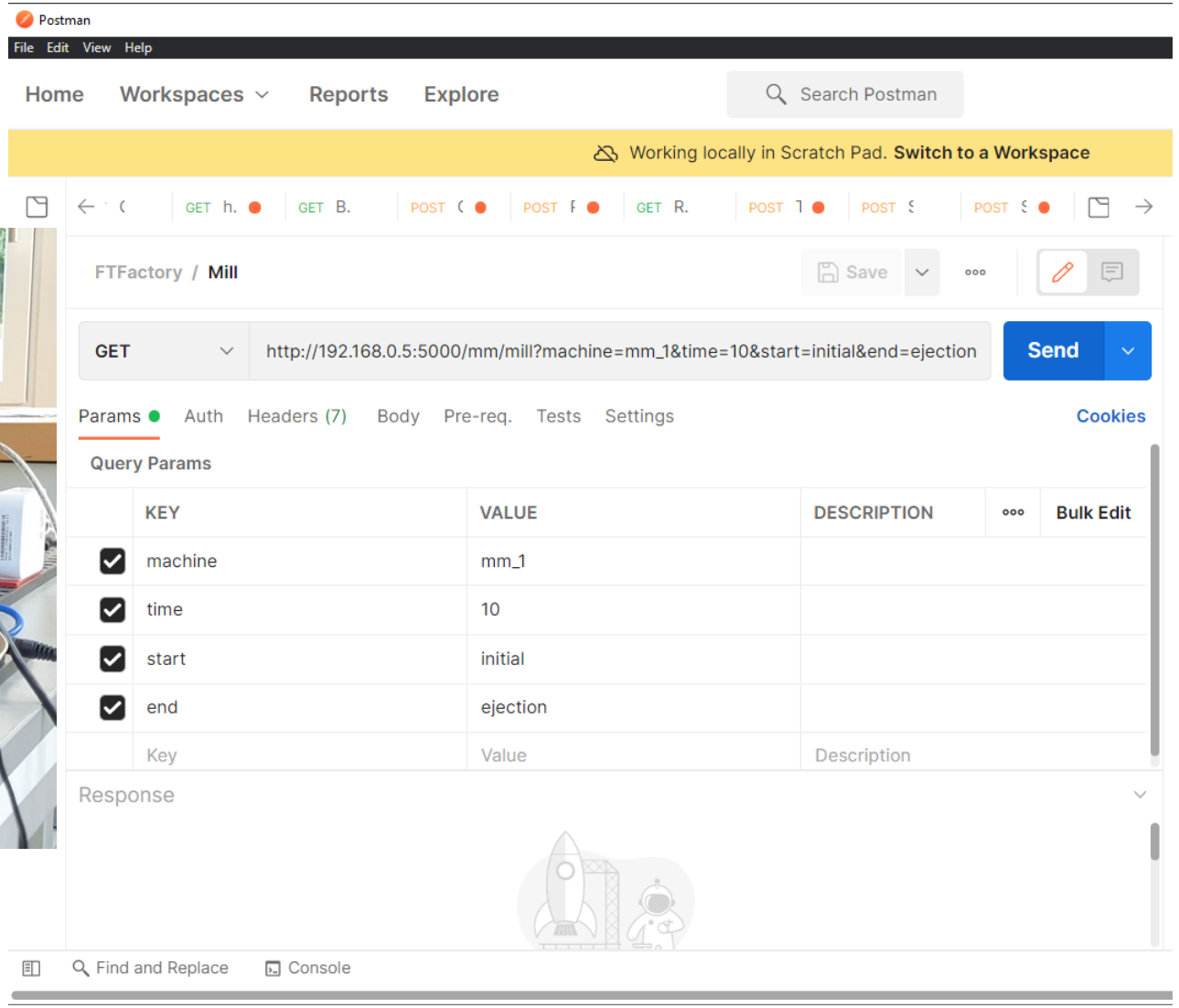
University of St.Gallen

Universität Trier

German Research Center for Artificial Intelligence

Going Service-based: Example





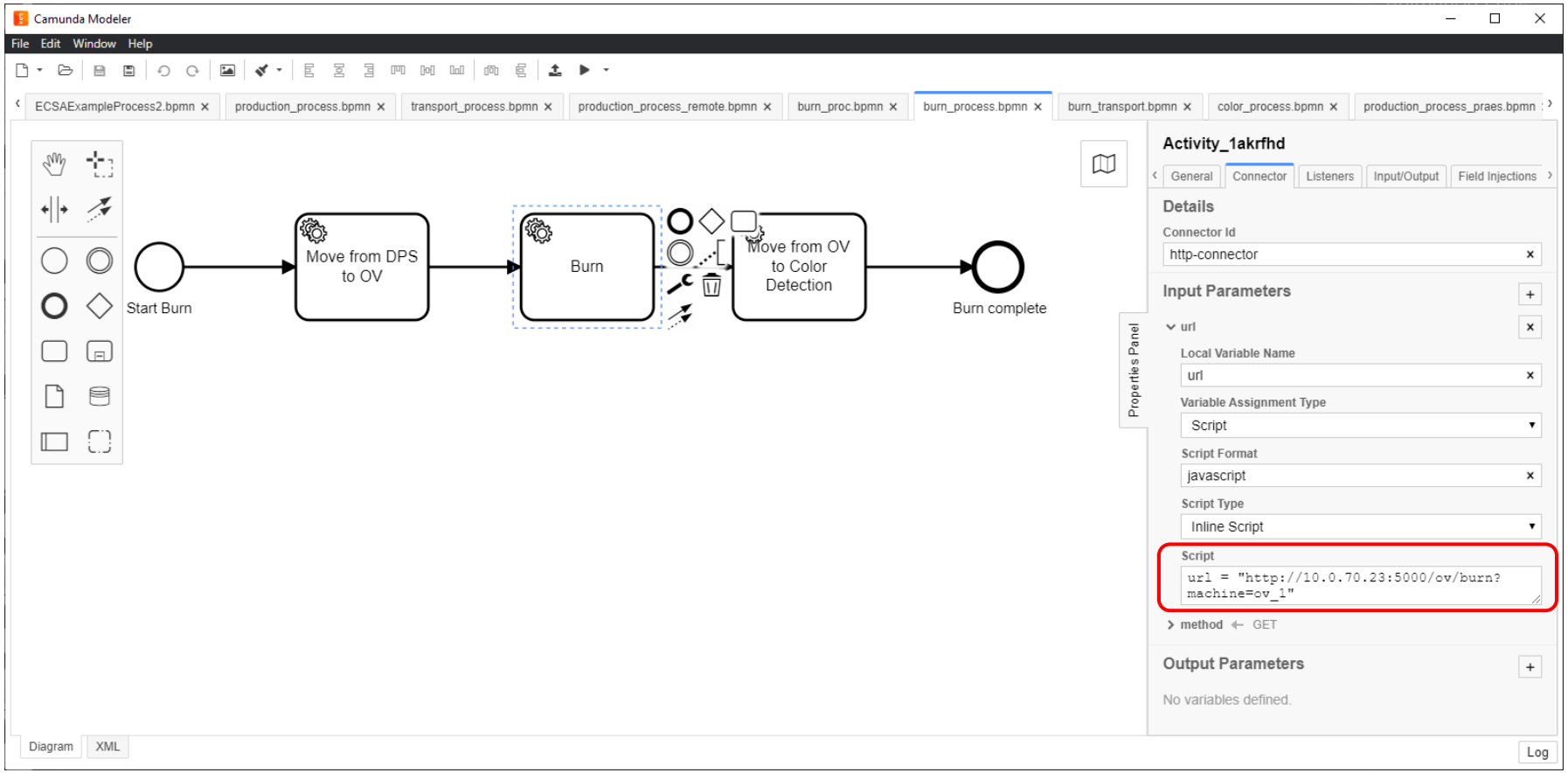
University of St.Gallen

Universität Trier

German Research Center for Artificial Intelligence

Going Business Processes

- Call web service via BPMN Service Tasks using http-connector



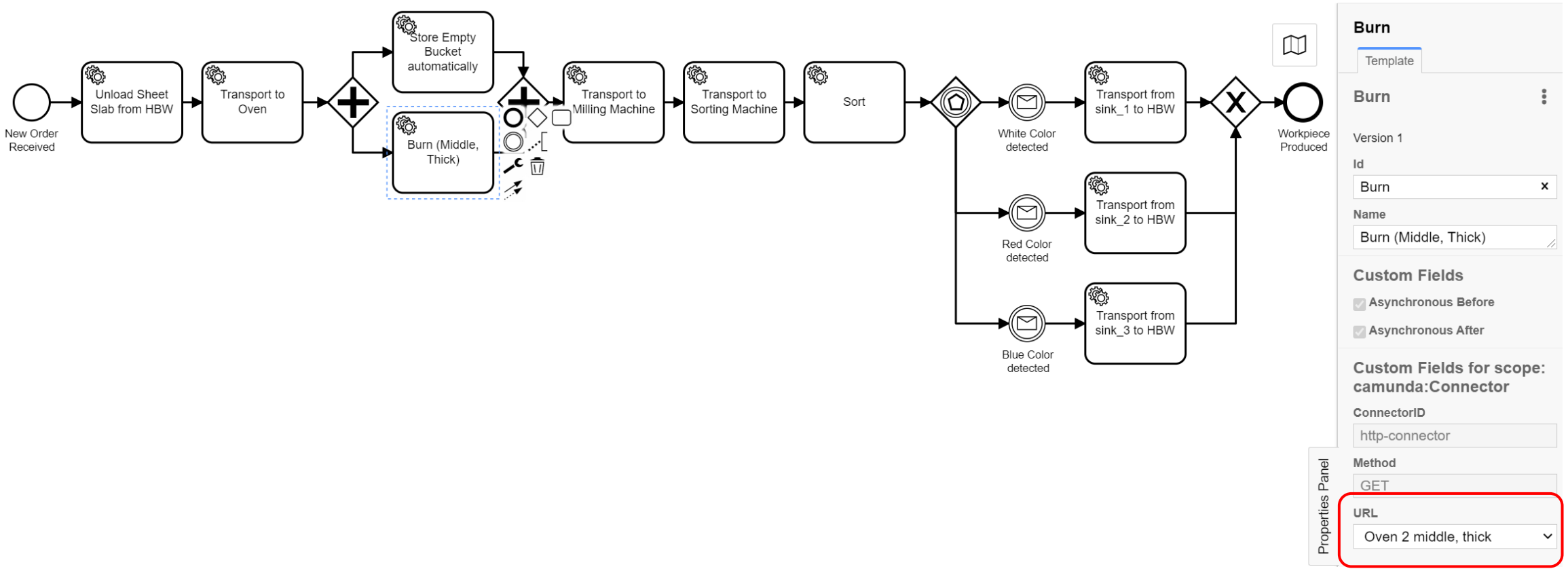
University of St.Gallen

Universität Trier

German Research Center for Artificial Intelligence

Going Business Processes

- Model processes with element templates

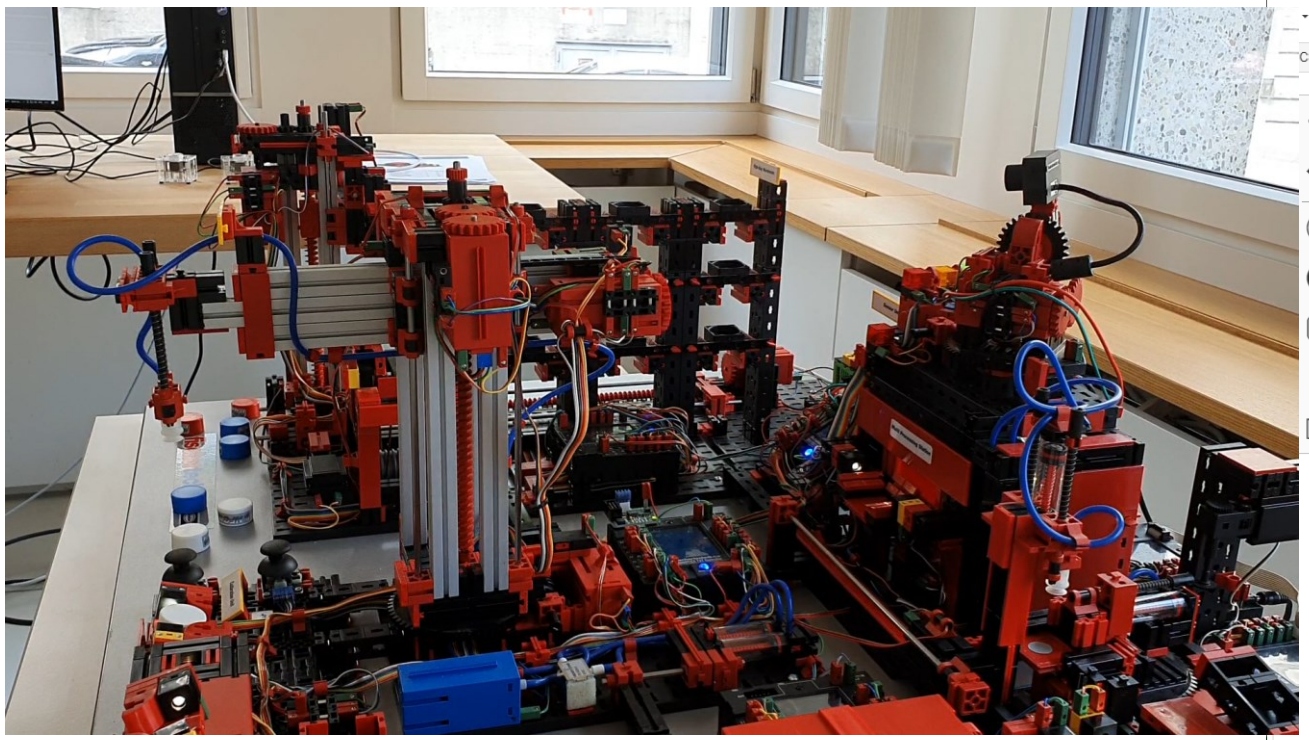


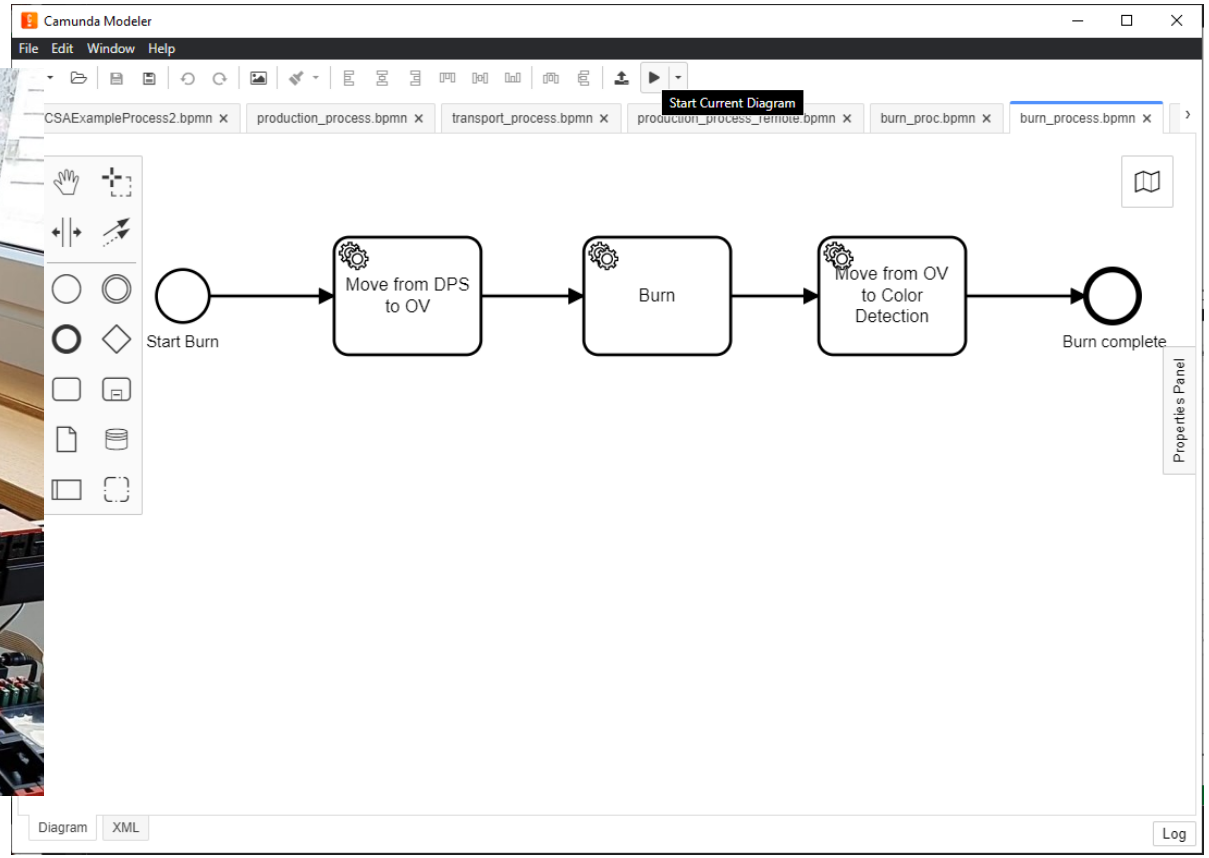
University of St.Gallen

Universität Trier

German Research Center for Artificial Intelligence

Going Business Processes: Example



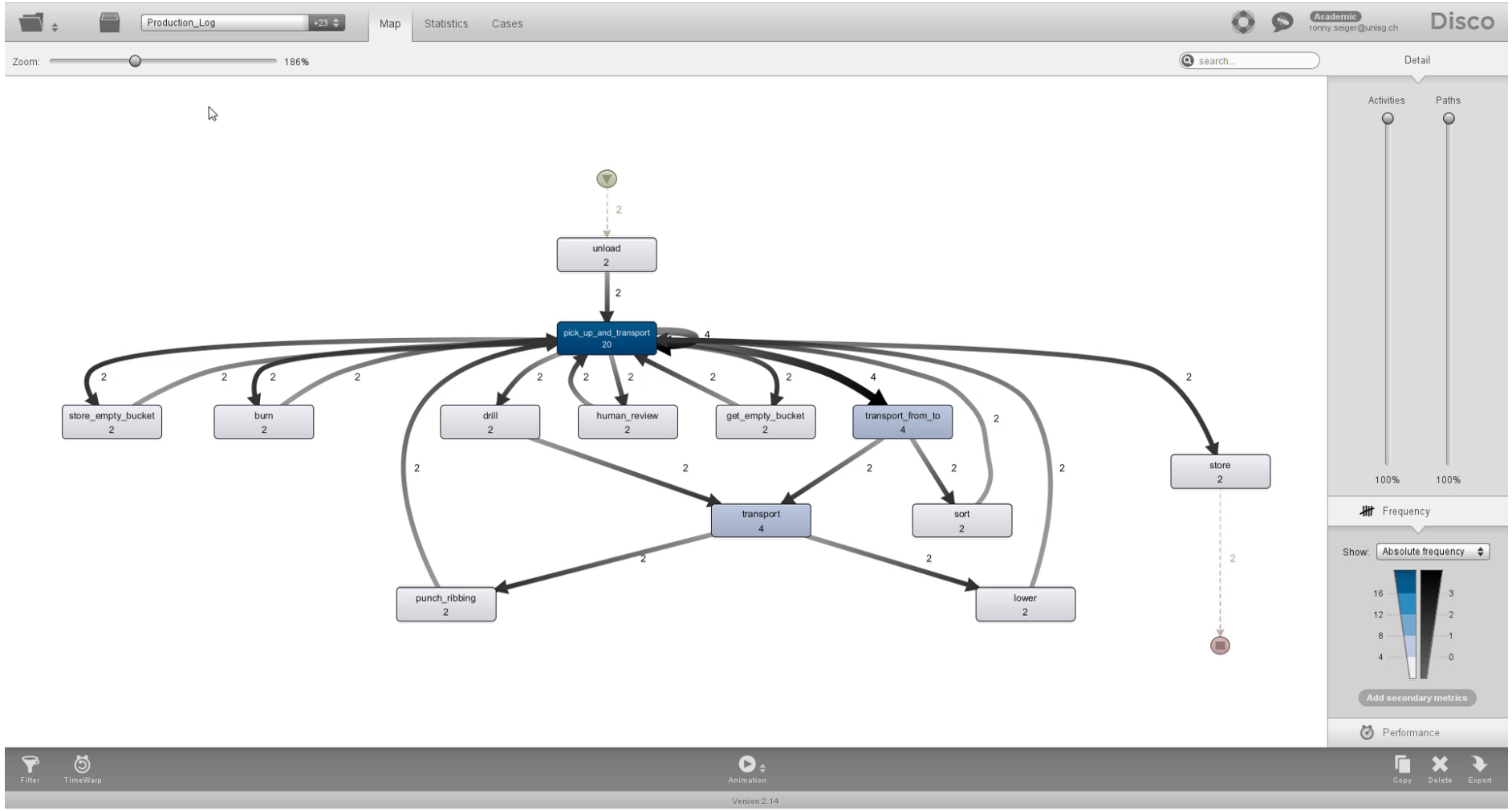


University of St.Gallen

Universität Trier

German Research Center for Artificial Intelligence

Process Mining almost for Free

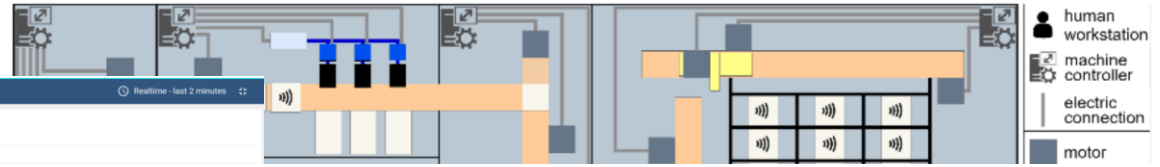


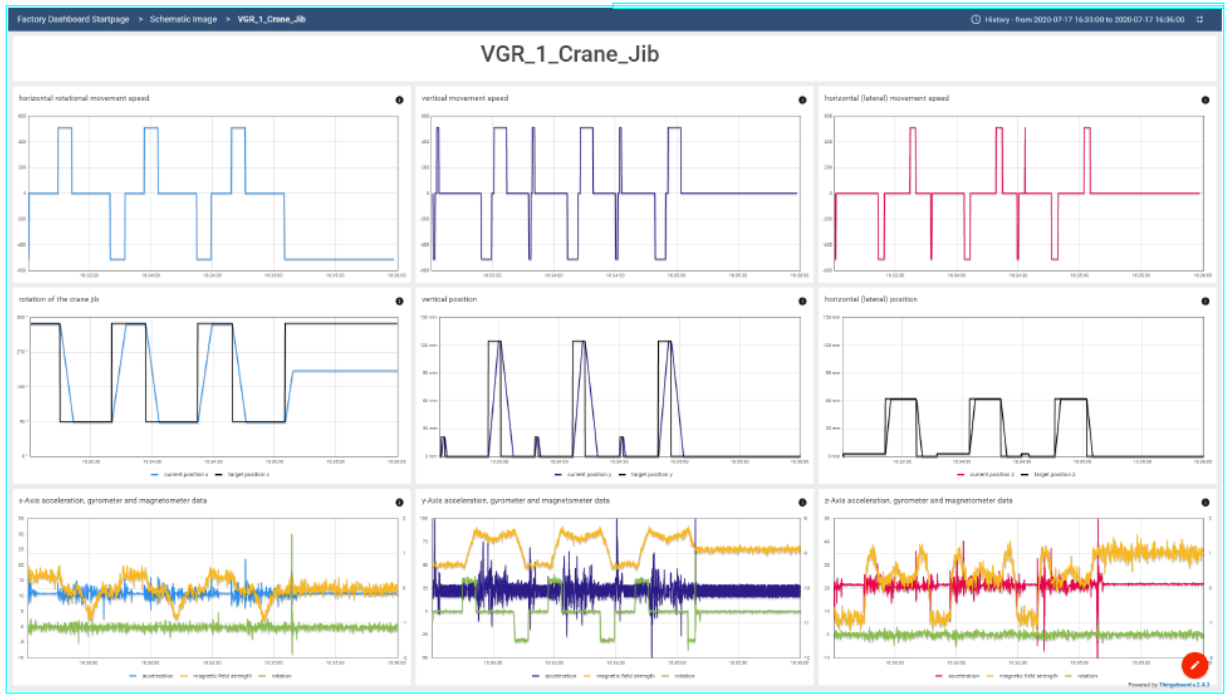
University of St.Gallen

Universität Trier

German Research Center for Artificial Intelligence

What about Sensors?



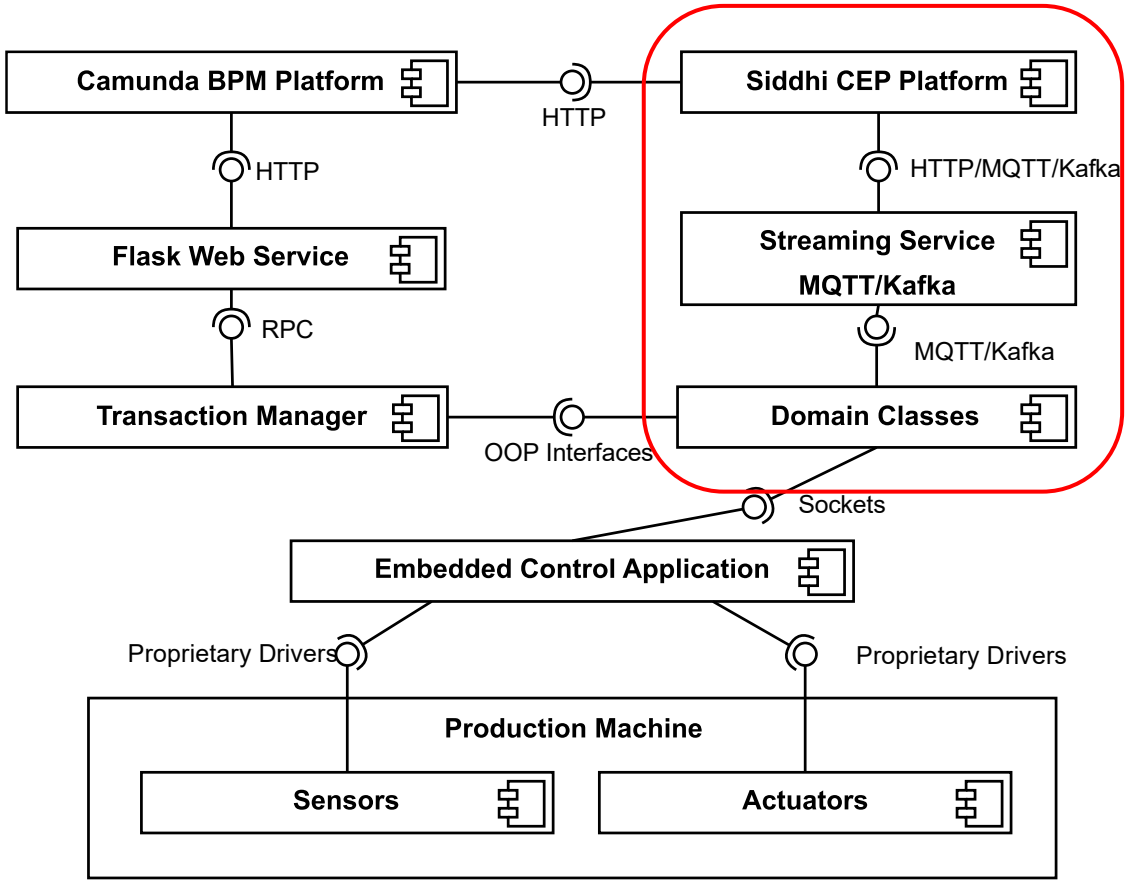


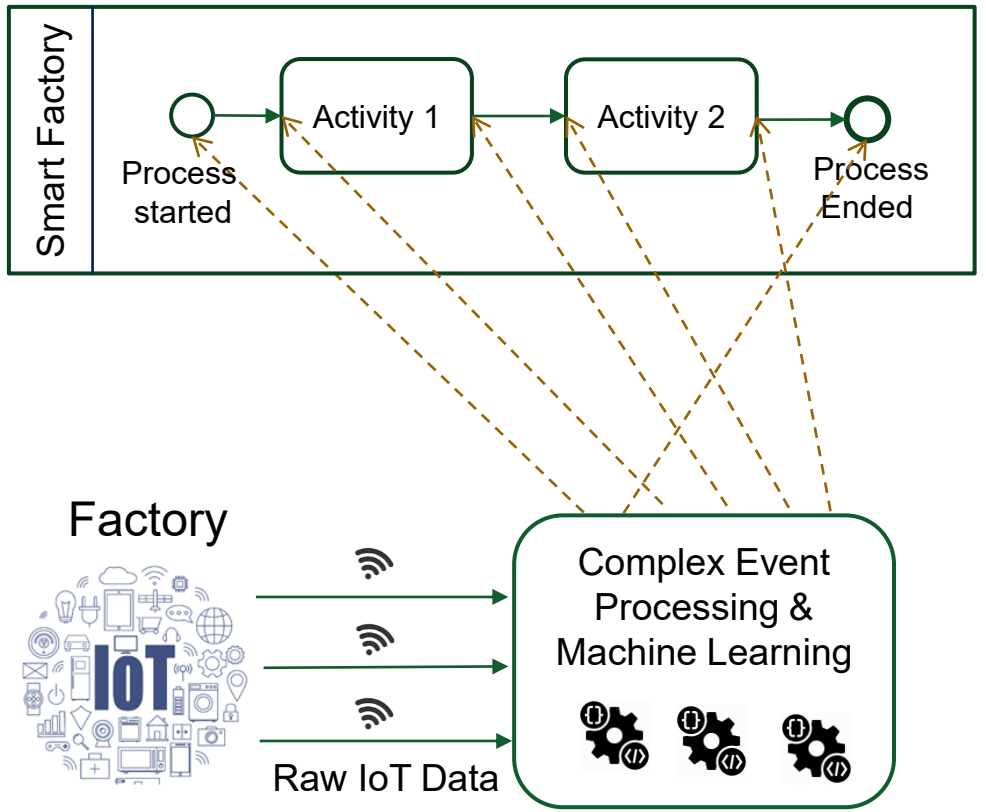
University of St.Gallen

Universität Trier

German Research Center for Artificial Intelligence

Going Event-driven



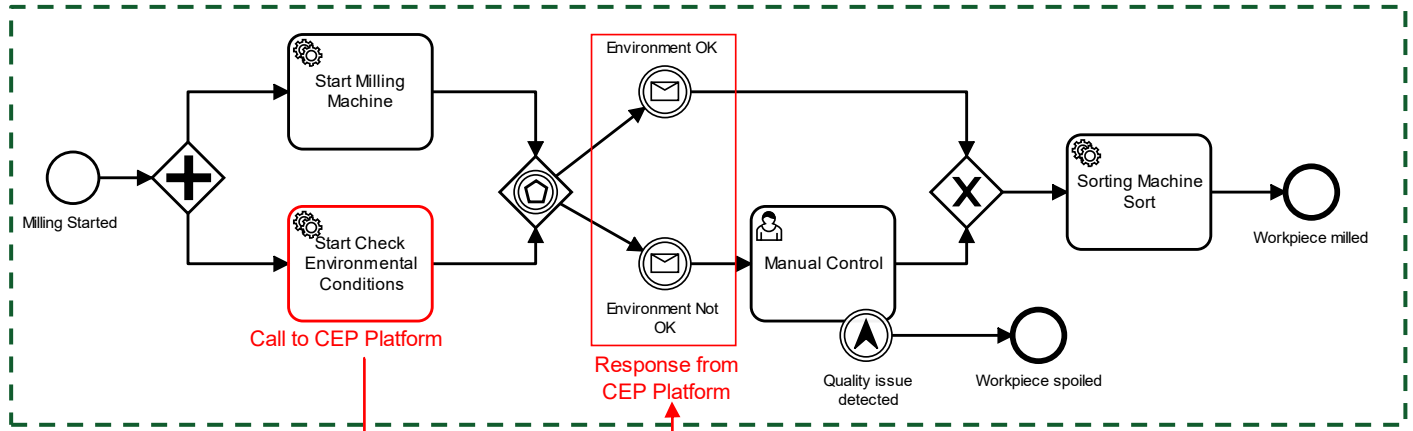


University of St.Gallen

Universität Trier

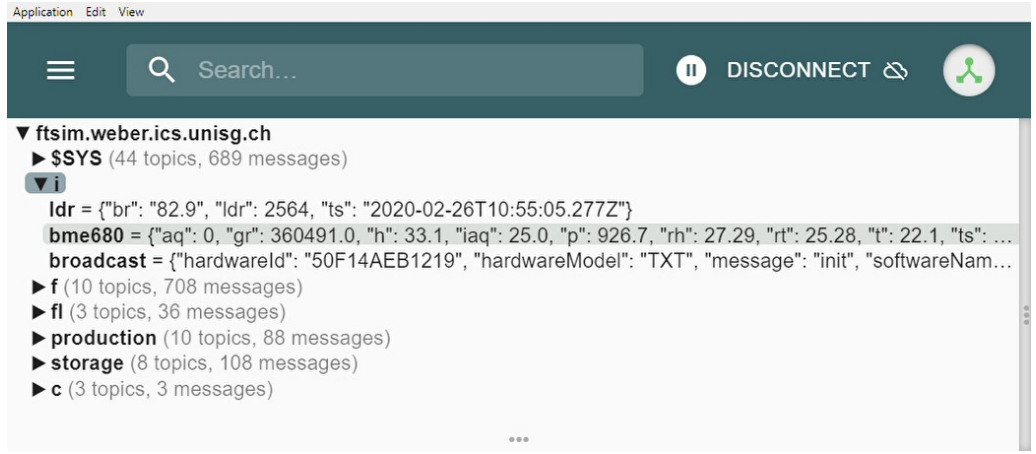
German Research Center for Artificial Intelligence

Going Event-driven: Quality Check Example



Camunda BPMS

MQTT Streaming



HTTP POST

HTTP POST

```
@info(name='EnvCheckOk')
from EnvAnalysisStream#window.timeBatch(10sec)
select "env_ok" as msgname,  avg(t) as avgtemp,
  avg(br) as avglight
having (avgtemp >= 22 and avgtemp<=25) and
  (avglight >= 80 and avglight <=85)
insert into EnvCheckStream;
```

Siddhi CEP

University of St.Gallen

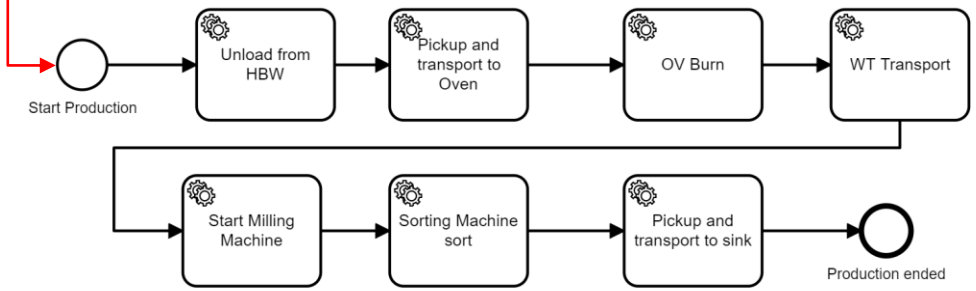
Universität Trier

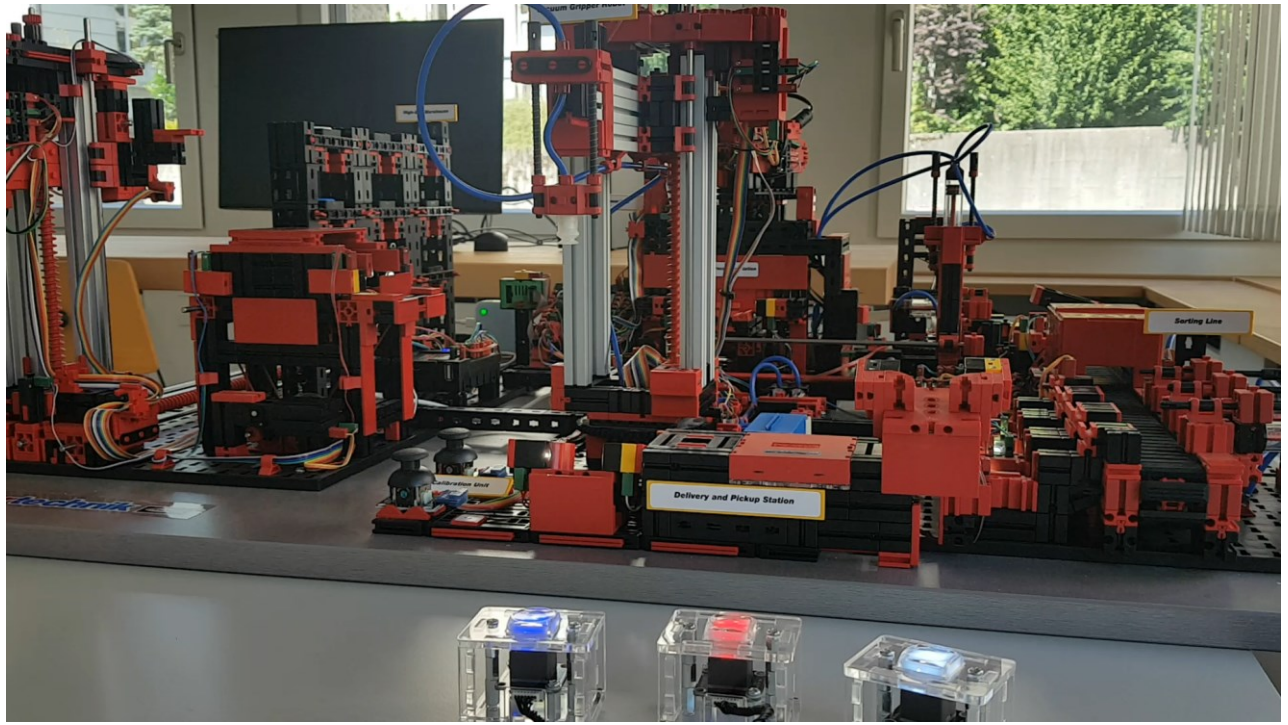
German Research Center for Artificial Intelligence

Going Event-driven: Starting a Process

```
@source(type = 'mqtt', url =
"tcp://broker.hivemq.com", client.id = “unisg", topic =
"TF_Buttons/M3X", @map(type = 'json’))
define stream BlueButtonStream (type string, state
string, ts string);

@info(name=‘bluebutton’)
from BlueButtonStream[state == 'pressed’]
select str:lower(type) as color
insert into ProcessStartStream;
```

HTTP POST



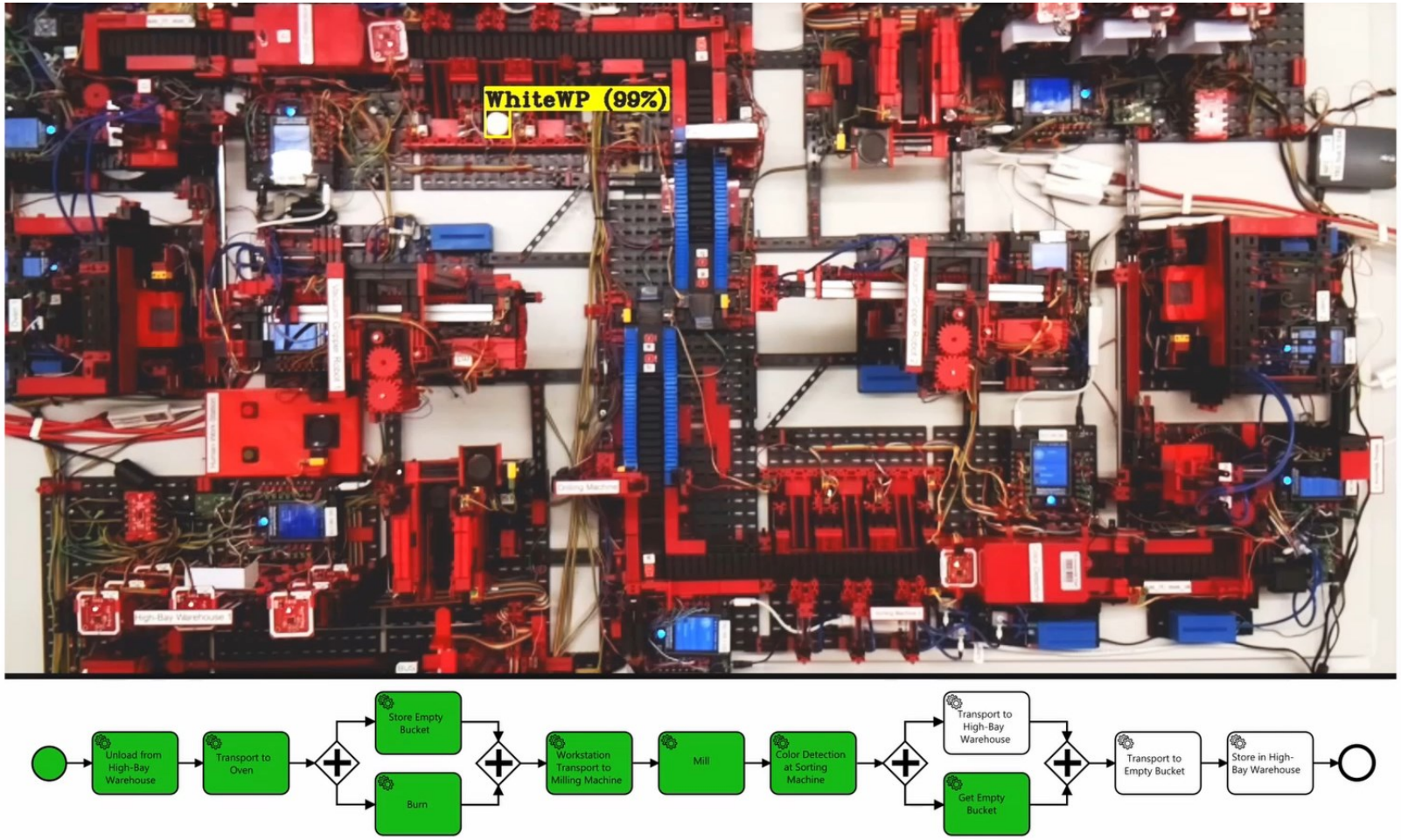


University of St.Gallen

Universität Trier

German Research Center for Artificial Intelligence

AI is Everywhere!



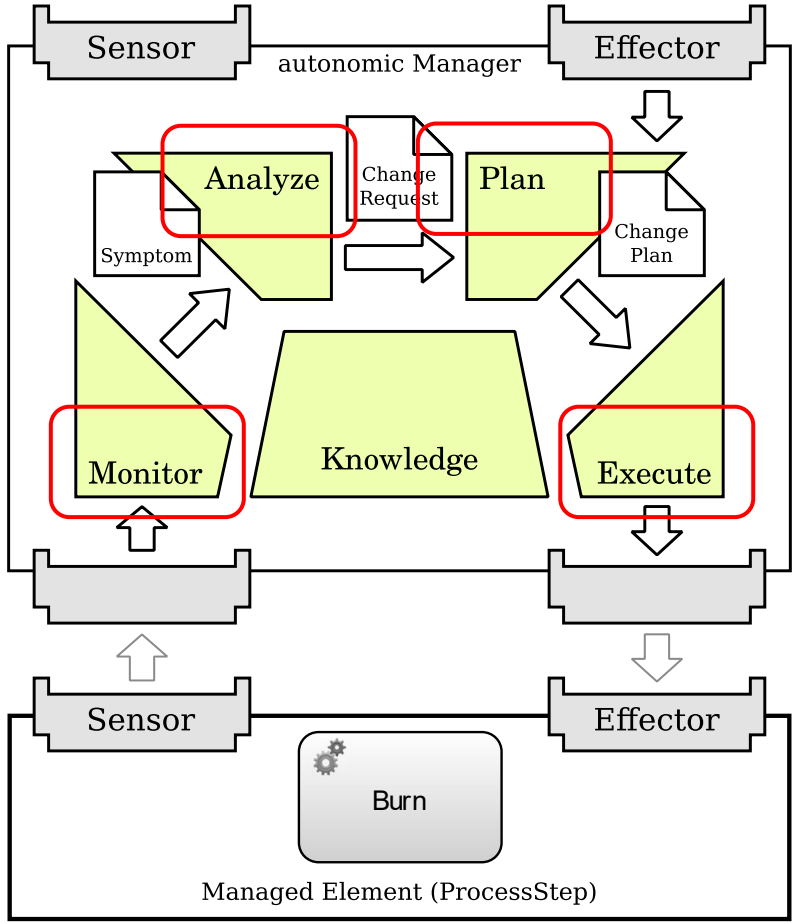
University of St.Gallen

Universität Trier

German Research Center for Artificial Intelligence

Now the Research

- **Error Handling & Autonomous Behavior**
- Monitor: Sensor Streaming
- Analyse: Complex Event Processing & Machine Learning
- Plan: Case-based Reasoning (Learn from the Past) & Automated (AI) Planning
- Execute: Adapt the Process
- Repeat

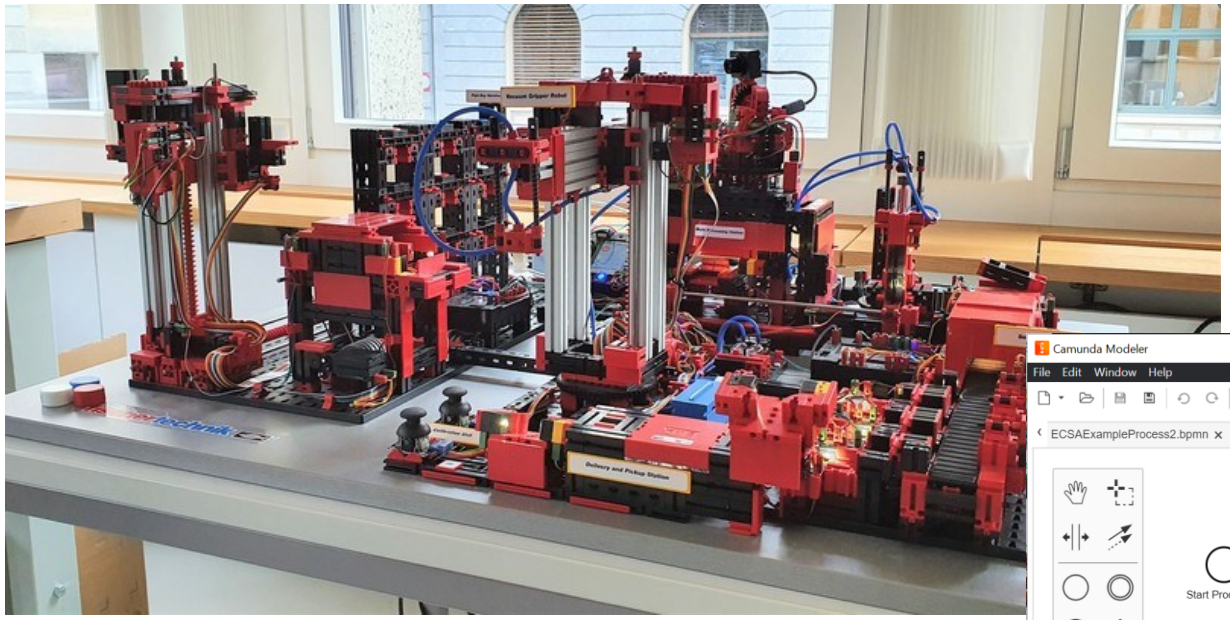


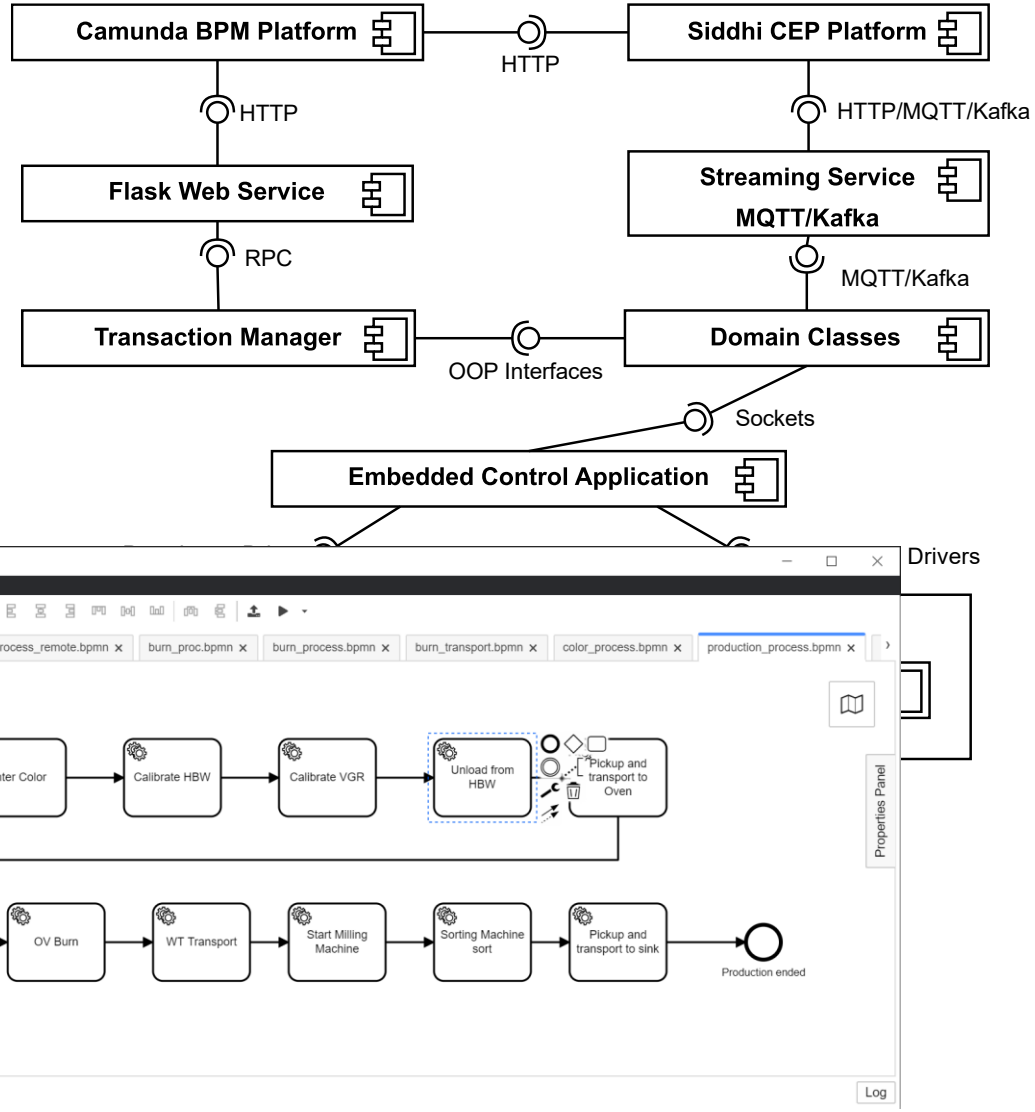
University of St.Gallen

Universität Trier

German Research Center for Artificial Intelligence

The Whole Picture

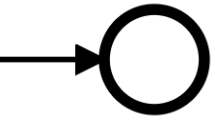




University of St.Gallen

Universität Trier

German Research Center for Artificial Intelligence



Presentation finished

Controlling a Smart Factory with Processes in Camunda

Get in Touch

Ronny Seiger, Barbara Weber
University of St.Gallen, Switzerland
ronny.seiger@unisg.ch

Lukas Malburg, Ralph Bergmann
University of Trier, Germany
German Research Center for AI (DFKI)
malburgl@uni-trier.de

University of St.Gallen

Universität Trier

German Research Center for Artificial Intelligence

Questions?