Services in Industry 4.0
Modeling and composition

Francesco Leotta, Massimo Mecella, Flavia Monti, Luciana Silo

{leotta,mecella,monti,silo}@diag.uniroma1.it

SAPIENZA
UNIVERSITÀ DI ROMA

DIPARTIMENTO DI INGEGNERIA INFORMATICA,
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

This work has been supported by MICS (Made in Italy – Circular and Sustainable) Extended Partnership and received funding from the European Union Next-GenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4 COMPONENTE 2

CITE THESE SLIDES AS:

LEOTTA F., MECELLA M., MONTI F., SILO L. (2023) SERVICES IN INDUSTRY 4.0. MODELING AND COMPOSITION. IN: SERVICE-ORIENTED COMPUTING
17TH SYMPOSIUM AND SUMMER SCHOOL, SUMMERSOC 2023, HERSONISSOS, CRETE, GREECE, JUNE 26–JULY 1, 2023, TUTORIAL PROGRAM

Who are we ?

Part of the Processes, Services and Software Engineering group of Sapienza

- ***Francesco Leotta***
  - *Assistant professor*
  - Main research interests in Ambient intelligence, HCI and Software architectures
  - Wide experience in Industry 4.0 projects (DESTINI, Electrospindle 4.0, Fustella 4.0)

- ***Massimo Mecella***
  - *Full professor*
  - Main research interests in SOA, BPM, smart applications (AI applied to new scenarios), software engineering
  - Wide experience in EU and National projects
  - Chair of various conferences in the last years (lately CAiSE 2019 and BPM 2021 in Roma)
  - Chair of **ICSOC 2023 in Roma – deadline for papers 10 July**

- ***Flavia Monti***
  - *PhD student*
  - Main research interests in Industry 4.0, AI-based methods

- ***Luciana Silo***
  - *PhD student*
  - Main research interest in Industry 4.0, formal methods

Outline of the talk

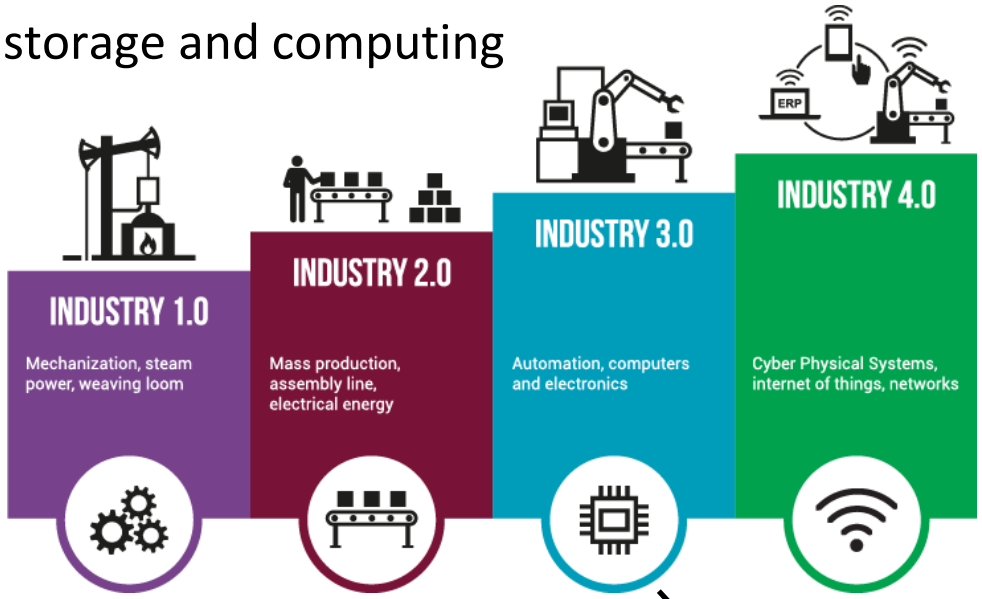
- Introduction to Smart Manufacturing
- Modeling I4.0 systems as services and their composition
- Approaches for automated reasoning
- AIDA – **A**daptive **I**n**D**ustrial **A**PIs: case study and demo

AN INTRODUCTION TO SMART MANUFACTURING

Smart Manufacturing (Industry 4.0)

Dalenogare, L. S., Benitez, G. B., Ayala, N. F., & Frank, A. G. (2018). The expected contribution of Industry 4.0 technologies for industrial performance. International Journal of production economics, 204, 383-394.

- Evolution of the traditional world of industrial automation
  - continuously evolving technologies for networking, storage and computing
- Functional to
  - increase productivity and quality
  - ease workers' lives
  - define new business opportunities



- Digital factory is a key concept
  - a multi-layered integration of the **information related to activities** along the factory and related resources

Introduction of Fieldbus (e.g., EtherCAT)

Smartness in manufacturing

Manufacturing smartness is:

- Gaining line of sight, optimizing use of dispersed resources & expertise, and planning a coordinated response to individual (partner) & collective (network) manufacturing needs
- Increasing intelligence of machinery and production lines decreases the need for human intervention in manufacturing processes

Emerging Trends

- Factories and machines increasingly complex
  - Human-centricity and robot collaboration are crucial
- Mass customization
  - Customer requires customized services and products
- Business Processes are not considered isolated
  - Multiple stakeholders involved

Zero Defect Manufacturing

- Strategy for improving process and product quality considering production planning, quality management and maintenance strategies
- Leverages heterogeneous data generated by a company (e.g., shop floor data, product functioning data, supplier data)
- Self-correcting system predicting and detecting defects, enhancing product design and quality

Powell, D., Magnanini, M. C., Colledani, M., & Myklebust, O. (2022). Advancing zero defect manufacturing: A state-of-the-art perspective and future research directions. Computers in Industry, 136, 103596

Smart Products

- Objectives
  - Real-time monitoring
  - Transparency, i.e., keeping an history of data
  - Remote control, e.g., firmware updates, parameter updates
  - Predictive capacity
  - Adaptability
- *Smart products for an Industry 4.0 company may take the role of machines in other Industry 4.0 companies*

Manufacturing Networks

- A Mfg Network is a permanent or temporal coalition comprising production systems of geographically dispersed OEMs or 1st/2nd tier suppliers that collaborate in a shared value-chain to conduct joint manufacturing.
- Enhancing mfg network visibility, information sharing & mfg process integration are major contributors to effective managing manufacturing networks.
  - Each partner in the network produces one or more product part(s) assembled into final service-enhanced products under the control of joint production schedule, while keeping its own autonomy.
  - Production schedules are monitored & optimized collectively to accomplish a shared manufacturing goal.

Manufacturing Barriers to Achievement

KEY CHALLENGES	EXPLANATION
Manufacturers need to track, and trace products from "cradle to grave", i.e., from raw materials and work in progress to finished products.	• **Manufacturing data, operations & processes should be integrated to link toward a shared manufacturing goal.** • **Dislocated assembly lines must be directly related to mfg management processes, e.g., production planning & control.**
Disconnection between Enterprise & Shop-floor Applications	• ERP and Production Management are complementary • Absence of a logical interface offering analytics for executive decision support
Disconnection between PLM & Manufacturing	• PLM Strategies have traditionally centered around Product Development
Limited point-to-point connections between Business IT & Factory Automation	• **No global visibility & no timely reaction to problems & changes**
Lack of visibility into plan to produce process to help eliminate bottlenecks	• Limited information on production attainment to plans, plant performance • Quickly identify exceptions in current performance & root cause analysis to identify bottlenecks in the production process • Monitor and track resource and material usage and variances
Unable to identify work order performance related to on-time completions and quality	• No knowledge of critical work orders that are stuck and need to be expedited • Little visibility into work order cycle times and work order aging of open orders • Track execution of "perfect work orders" that are on-time and high quality

Papazoglou, M. P., van den Heuvel, W. J., & Mascolo, J. E. (2015). A reference architecture and knowledge-based structures for smart manufacturing networks. *IEEE Software*, *32*(3), 61-69.

7 Steps to Smart Manufacturing Networks

Smart Manufacturing Networks: The Road-Map

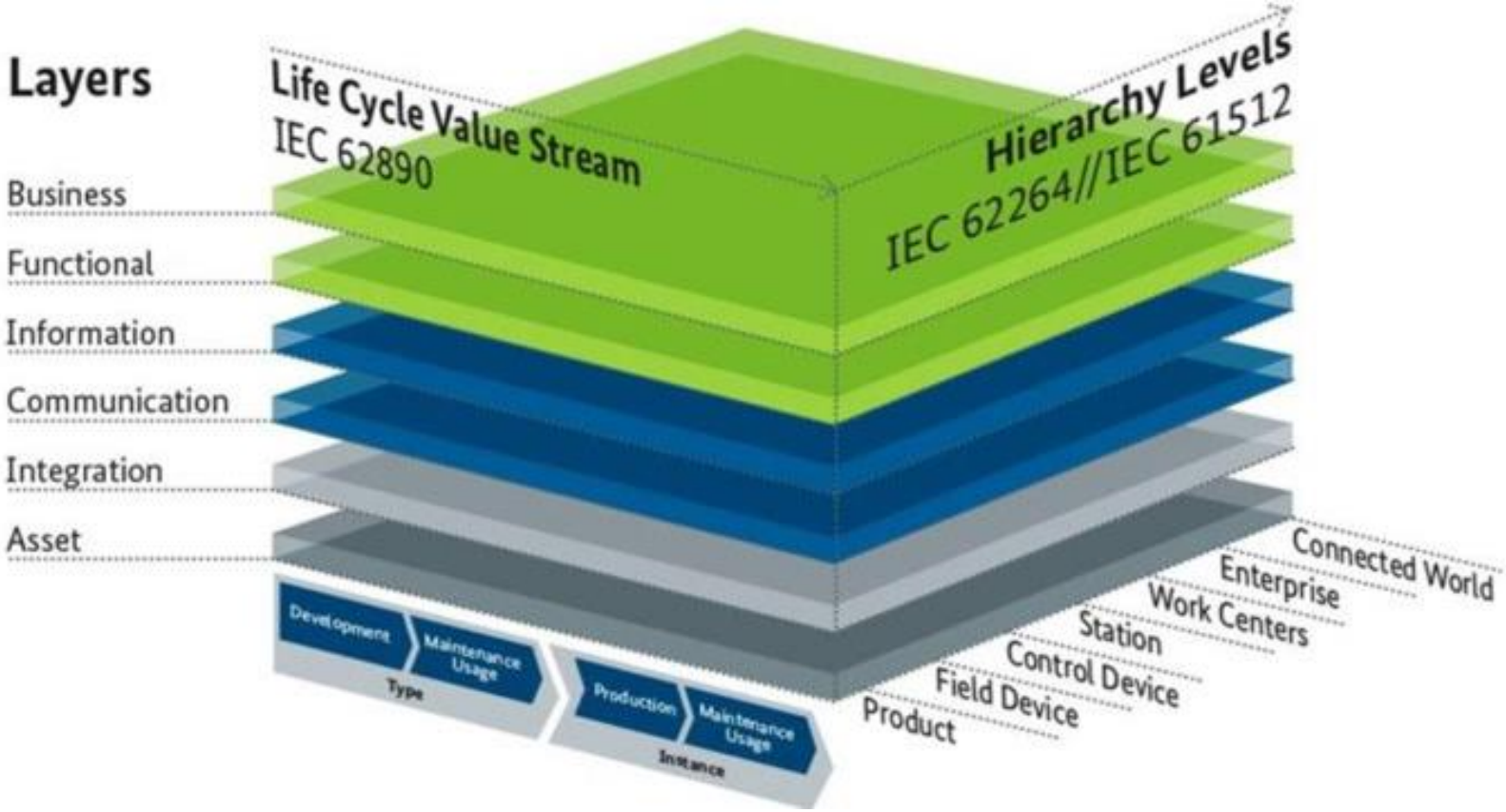
1. Reference Architecture
2. Connectivity of Enterprise and Plant Floor Systems
3. **Raising the Level of Manufacturing Abstraction**
4. Enabling Manufacturing “Smartness” and Analytics
5. Presentation: Human-Interaction
6. Manufacturing Network Lifecycle Management
7. Leveraging Open Standards

Smart Manufacturing Actors

- Humans
  - Manufacturing line operators
  - Manufacturing line supervisors
  - Managers
  - Software development teams (for information systems and machines)
  - Product Designers
  - Process Designers
  - …
- Machines
  - Including new (Industry 4.0 native), updated and extended ones
- Organizations
  - Also including suppliers and public authorities
- Information Systems
- Smart products

Introducing RAMI

- RAMI 4.0 - Reference Architectural Model for Industry 4.0
  - German initiative developed adopted at EU level
  - a 3-dimensional model
  - life-cycle/value-stream vs. organizational hierarchy vs logical layers



Hankel, M., & Rexroth, B. (2015). The reference architectural model industrie 4.0 (rami 4.0). ZVEI, 2(2), 4-9.

Logical Layers in RAMI

- **Assets layer**: physical components such as robots, but also non-physical objects such as software and ideas.
- **Integration layer:** information for assets in a form that can be digitally processed
- **Communication layer**: standardization of communication using uniform data format and predefined protocols. E.g., MQTT.
- **Information layer**: transforms available data into useful information.
  - Also called **context extraction**
  - **Digital Twins (DT)** belong to this layer
- **Functional layer:** includes formal descriptions of functions.
- **Business layer:** includes mapping of the business model and links between different business processes.

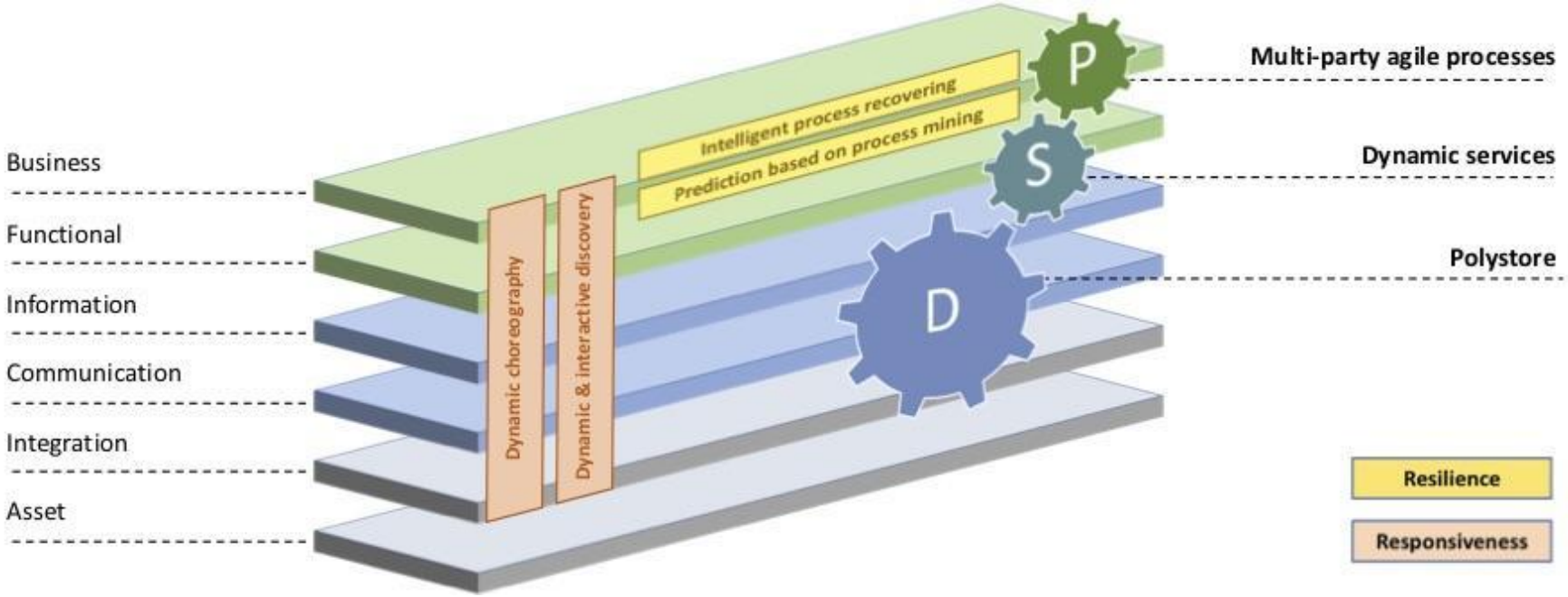
INDUSTRY 4.0 SUPPLY CHAIN

Smart Supply Chain

- **Agility** as responsiveness and adaptivity
  - **Responsiveness**: quickly react to disruptions along the supply chain
  - **Adaptivity**: ability to adapt to changes in the market while remaining competitive → **business opportunity**
- **Dynamic situations** must be managed during the product lifecycle
    - Natural disasters, climate change, sociopolitical shifts, economic crises, wars
- Responsiveness and ability to cope with Dynamic Situations contribute to **resilience**

A Possible Vision of RAMI

- RAMI defines logical functions of the different layers
  - How RAMI is implemented is up to the designer



Bicocchi, N., Cabri, G., Mandreoli, F., & Mecella, M. (2019). Dynamic digital factories for agile supply chains: An architectural approach. Journal of Industrial Information Integration, 15, 111-121.

Extending AI to Supply Chain Adaptivity

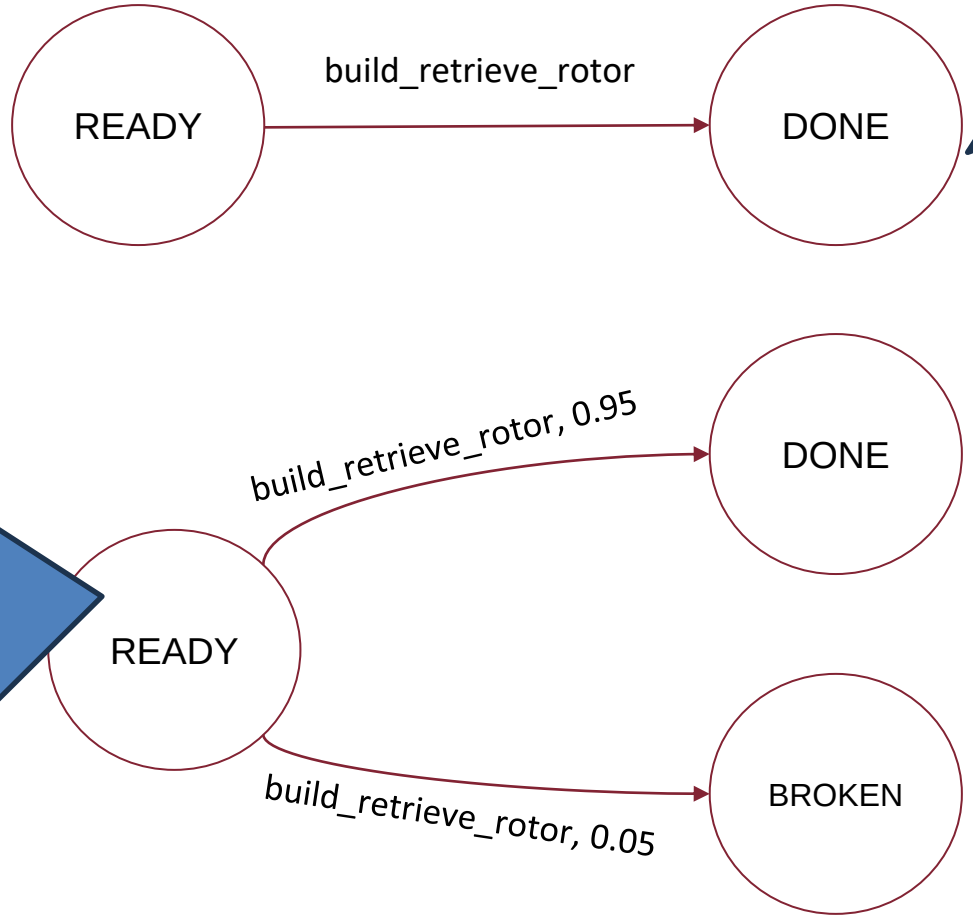
- Techniques for adaptivity explored in the context of single factories can be extended to the supply chain
- The supply chain may substantially increase the number of services and choices thus making manual decisions difficult to compute
- Model complex geo-political issues
- Model machine leasing

Modeling I4.0 systems as services

Actors in I4.0 are services

- Humans, robots, machines … each one of them performs actions and changes its state, ready to execute the next action(s) among those which are designed
- (optional) during the execution of an action, it may fail (with a given probability)
  - Here ML can support the evaluation of such a probability, for simplicity now we assume it is given
- The execution of an action may lead potentially in different states

Example: `rotor_builder` machine



Simpler *deterministic model*: actions lead to a unique state and assume execution is always correct

Stochastic model: actions lead to different states according to different probabilities. The probability represents in a combined way possible failures

I4.0 system as a composition of services

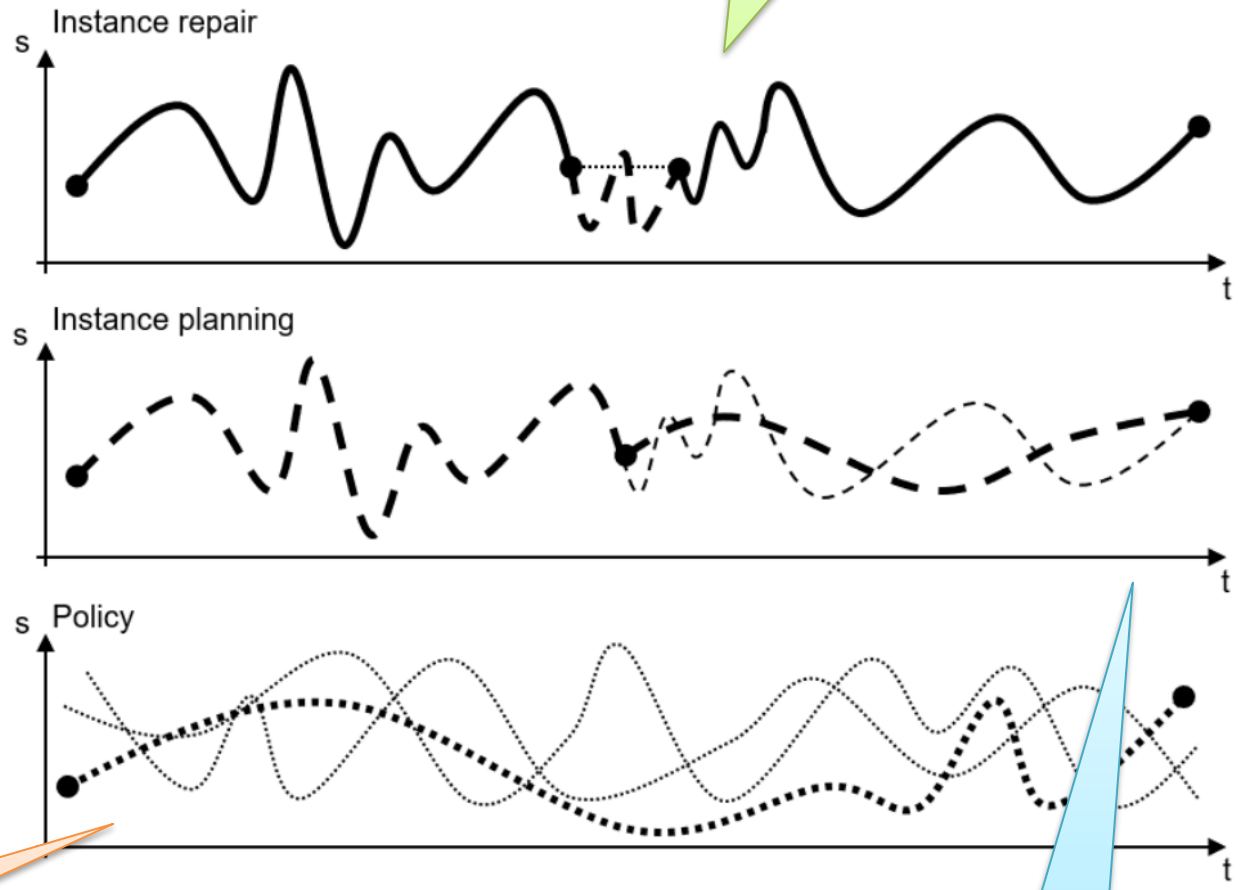
- Once upon a time there was the «Roman model»
- Processes can be modeled as orchestration of services
- Different variants can solve different issues
  - Adaptiveness as a re-planning problem over already specified orchestrations
  - Adaptiveness as a synthesis problem given a target orchestration and/or goals

- G. De Giacomo, M. Mecella, F. Patrizi (2014): Automated Service Composition Based on Behaviors: The Roman Model. Web Services Foundations 2014: 189-214
- D. Calvanese et al. (2008): Automatic Service Composition and Synthesis: the Roman Model. IEEE Data Eng. Bull. 31 (3), 18-22
- D. Berardi et al (2003): Automatic Composition of E-services That Export Their Behavior. ICSOC 2003 *(~ 600 citations)*

Adaptive strategies

- **Instance repair**
  - Process is well defined and upon exceptions the state of resources is taken back to the expected one employing (automated reasoning) techniques
- **Instance planning**
  - (automated reasoning) Techniques are employed every time a new process instance is needed and upon disruption(s)
- **Policy-based**
  - (automated reasoning) Techniques are employed every time a new process instance (policy) is needed. All the possible states are computed.

Andrea Marrella, Massimo Mecella, Sebastian Sardiña: Supporting adaptiveness of cyber-physical processes through action-based formalisms. AI Commun. 2018
Andrea Marrella, Massimo Mecella, Sebastian Sardiña: Intelligent Process Adaptation in the SmartPM System. ACM Trans. Intell. Syst. Technol. 2017



Giuseppe De Giacomo, Marco Favorito, Francesco Leotta, Massimo Mecella, Luciana Silo: Digital twin composition in smart manufacturing via Markov decision processes. Comput. Ind. 2023

Agnes Koschmider, Francesco Leotta, Jerin George Mathew, Massimo Mecella, Flavia Monti: On the Application of Process Management and Process Mining to Industry 4.0. Submitted to SOSYM, special issue on BPMDS 2022

IMPLEMENTATION HINTS

The importance of data abstraction

- IIoT data are very fine-grained, not at the proper granularity level typical of automated reasoning approaches
- IIoT data can be continuous values over huge domains, not good if you like to adopt automatic reasoning / verification techniques
- A proper abstraction layer solves such granularity-mismatch, and it should be devised in order to solve the mismatch semi-automatically and not necessarily ad-hoc

Physical-to-digital interface (where data abstraction happens)

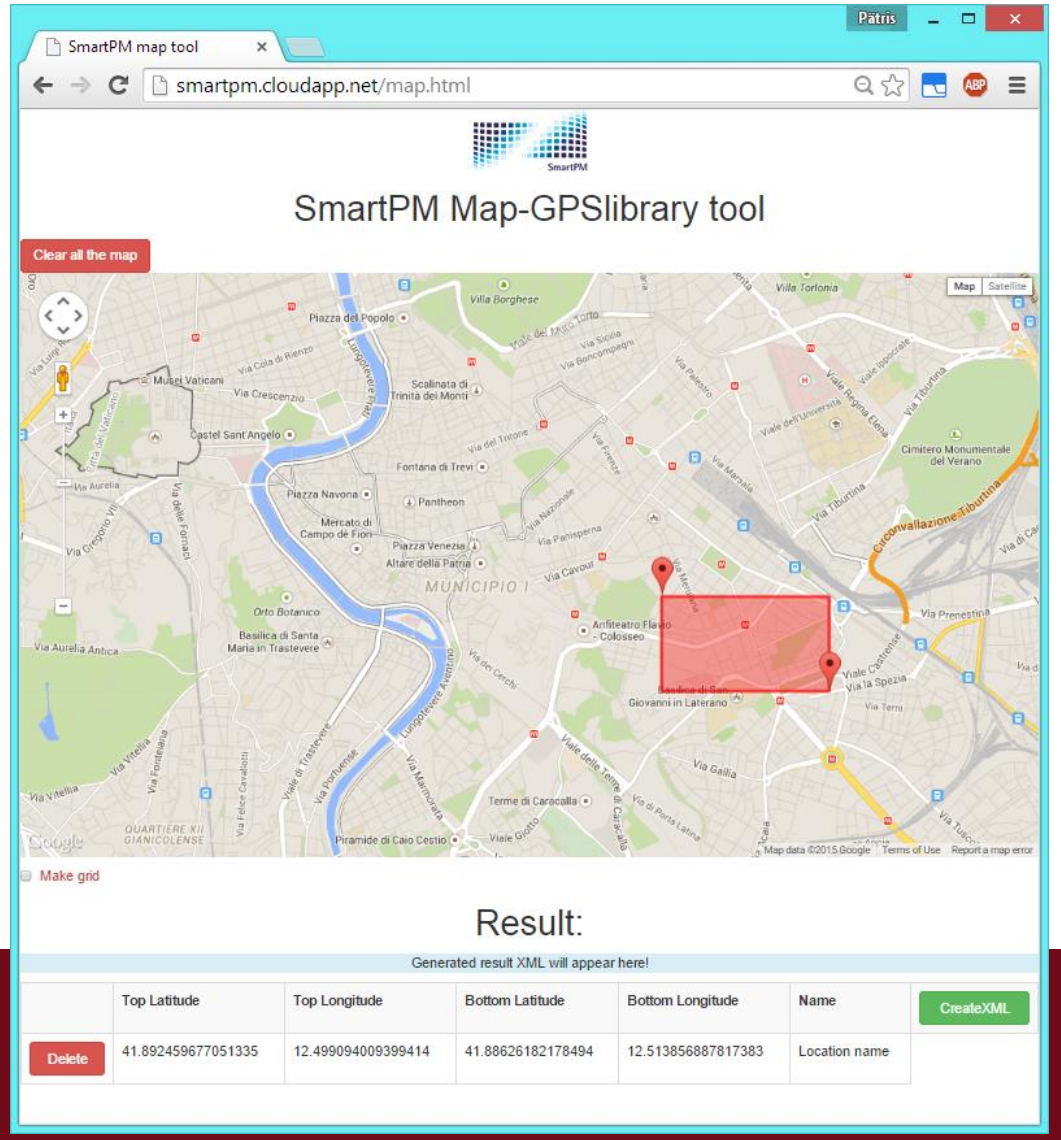
- Sensors data are often **continuous values** over huge domains.
- To exploit automatic reasoning/verification techniques, such data must be abstracted as **discrete variables** grounded into finite domains.

Physical-to-Digital interface

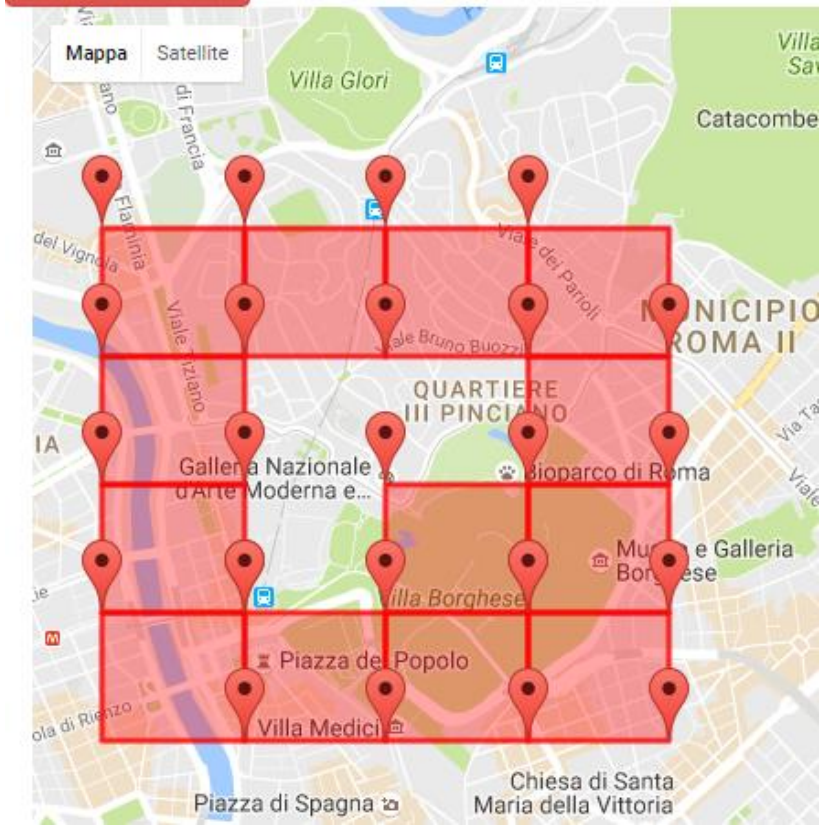
- Some prototypes of ours provide some **web tools** that allow us to associate some of the data objects defined in the domain theory with the continuous data values collected from the environment.

The SmartPM Location Tool

SmartPM provides a **Location web tool** (as a Google Maps plugin) that allows a process designer to mark areas of interest from a real map (by selecting latidude/longitude values) and **associate them to the discrete locations** (e.g., loc00, loc01, etc.) defined during the design stage of a process

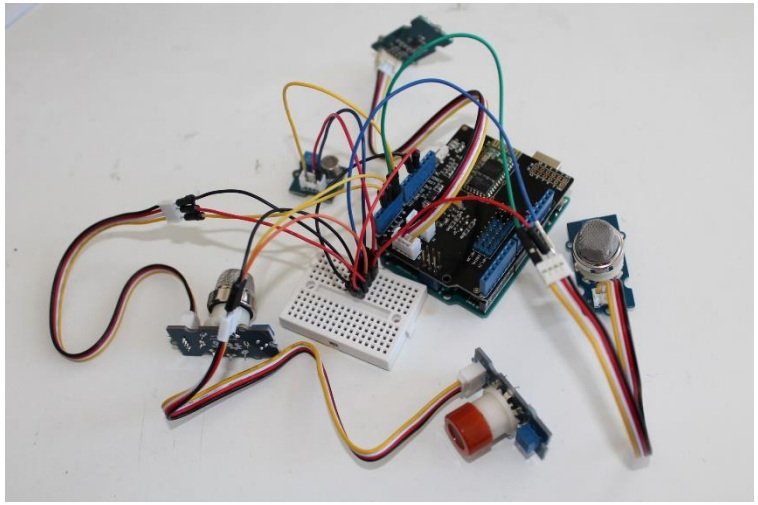


Clear all the map

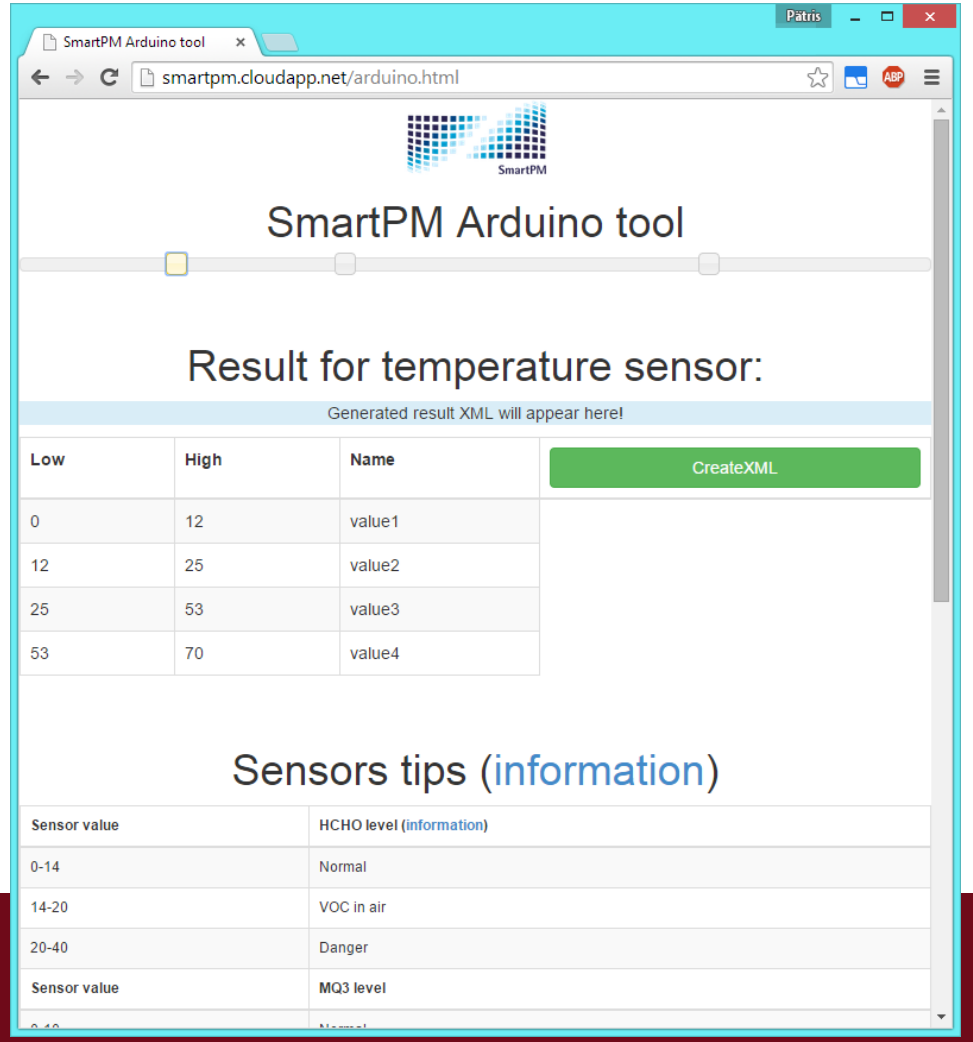


	Top Latitude	Top Longitude	Bottom Latitude	Bottom Longitude	Name
Delete	41.92578147109541	12.468795776367188	41.92131071657068	12.475447654724121	loc00
Delete	41.92578147109541	12.475447654724121	41.92131071657068	12.482099533081055	loc01
Delete	41.92578147109541	12.482099533081055	41.92131071657068	12.488751411437988	loc02
Delete	41.92578147109541	12.488751411437988	41.92131071657068	12.495403289794922	loc03

The SmartPM Arduino Tool

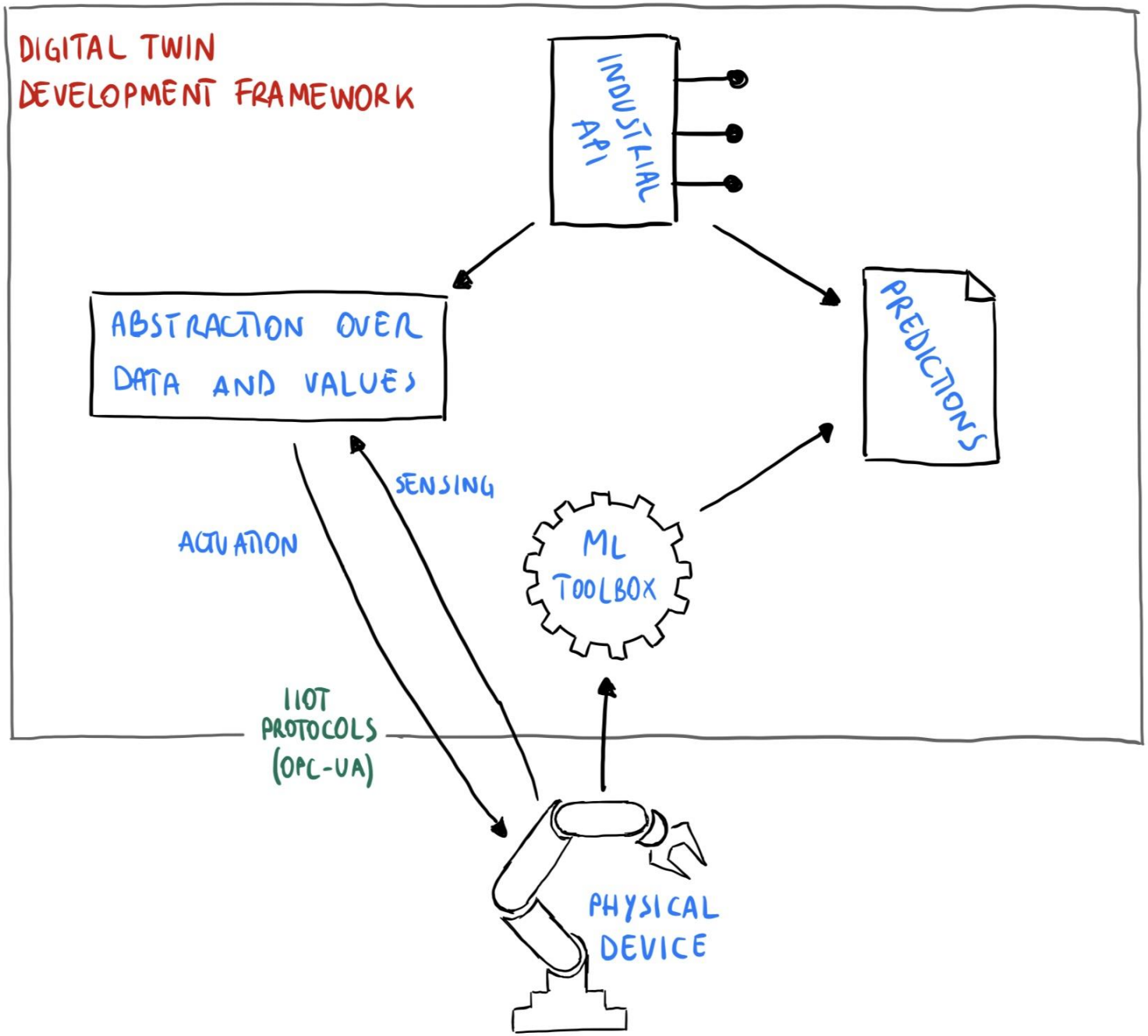


Arduino has a large variety of sensors available to measure different environmental values, for example different gas levels in the air, water quality, radiation level, etc. Arduino can be connected with Android via Bluetooth for transferring the data.



Wrappers/proxies

- Actors are services if appropriately wrappred/proxied
  - Industrial REST APIs
  - ML/predictors are inside such wrappers
- Service-based frameworks available for development
  - Eclipse Ditto
  - AWS IoT TwinMaker
  - Microsoft Azure Digital Twins



Intermezzo: speculating over smart systems and closed-loop approaches

A smart system

A system that can:

- Sense the state of its context and detect relevant changes in that context or in its own internal state, as they occur (i.e., in real time)
- Respond to such changes in a timely manner in a way that is consistent with or conducive to its intended purpose
- Adjust its behavior to deal with previously unknown or unexpected situations based on available data and/or its history

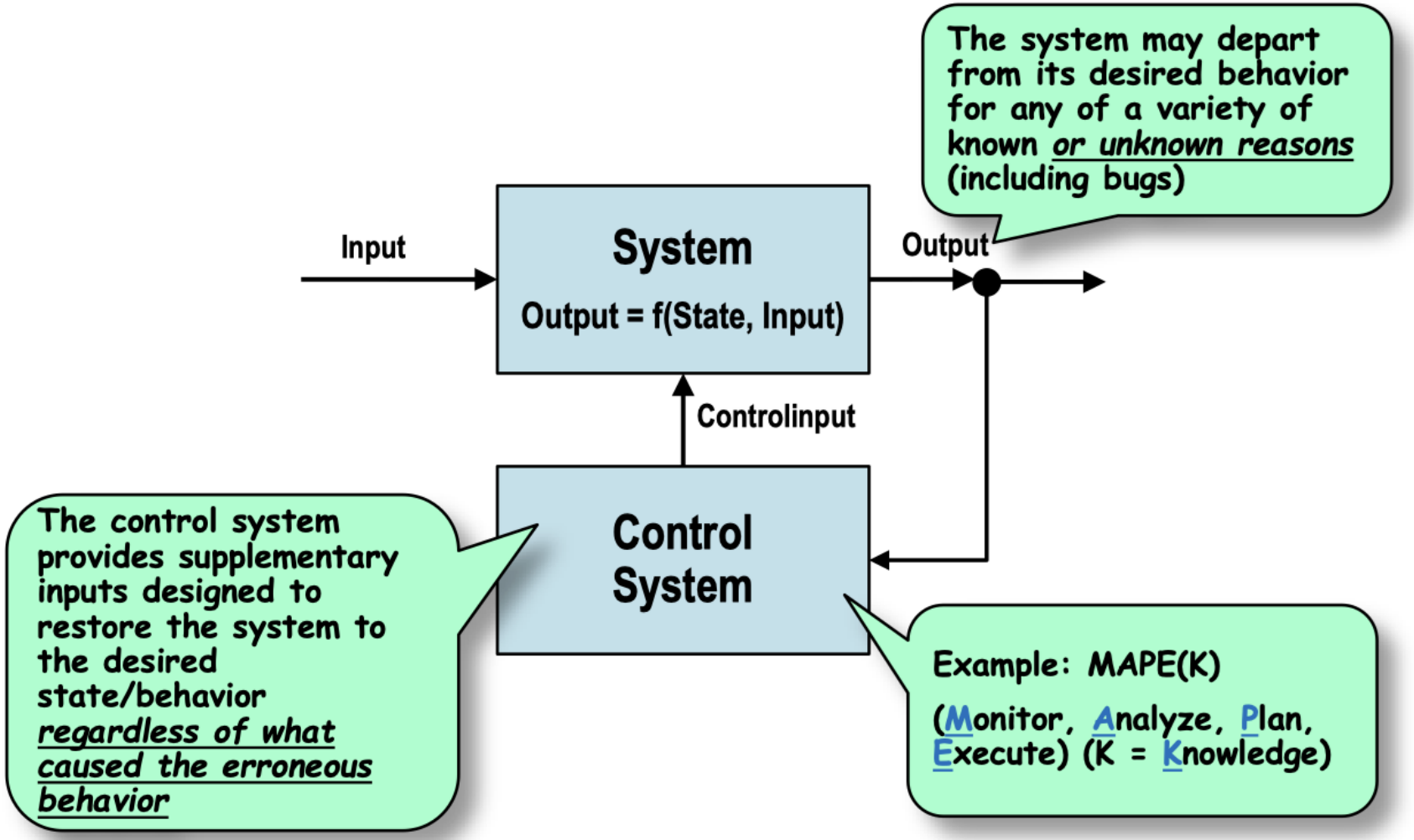
Definition by Bran Selic (https://scholar.google.com/citations?user=bzxh-TsAAAAJ)
cf. Matching Software with Reality. A Reflective Back to Front View. CAiSE 2023 Keynote speech

Smart systems are complex

- *It is safe to conclude that in these kinds of complex systems it will never be practically feasible to identify, in advance, all possible feature interactions that can (and, invariably, will) occur*

Closed-loop approaches

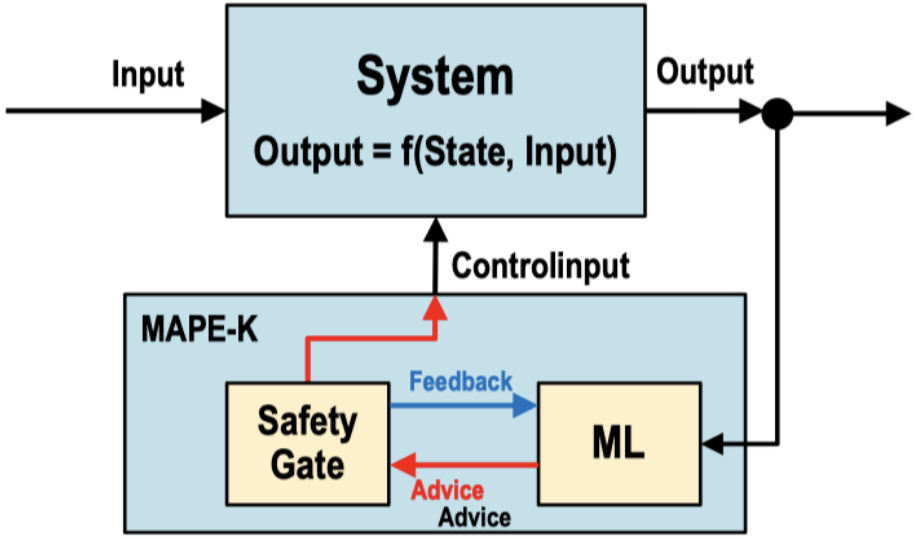
- Based on classical feedback control theory



The Role of AI Components

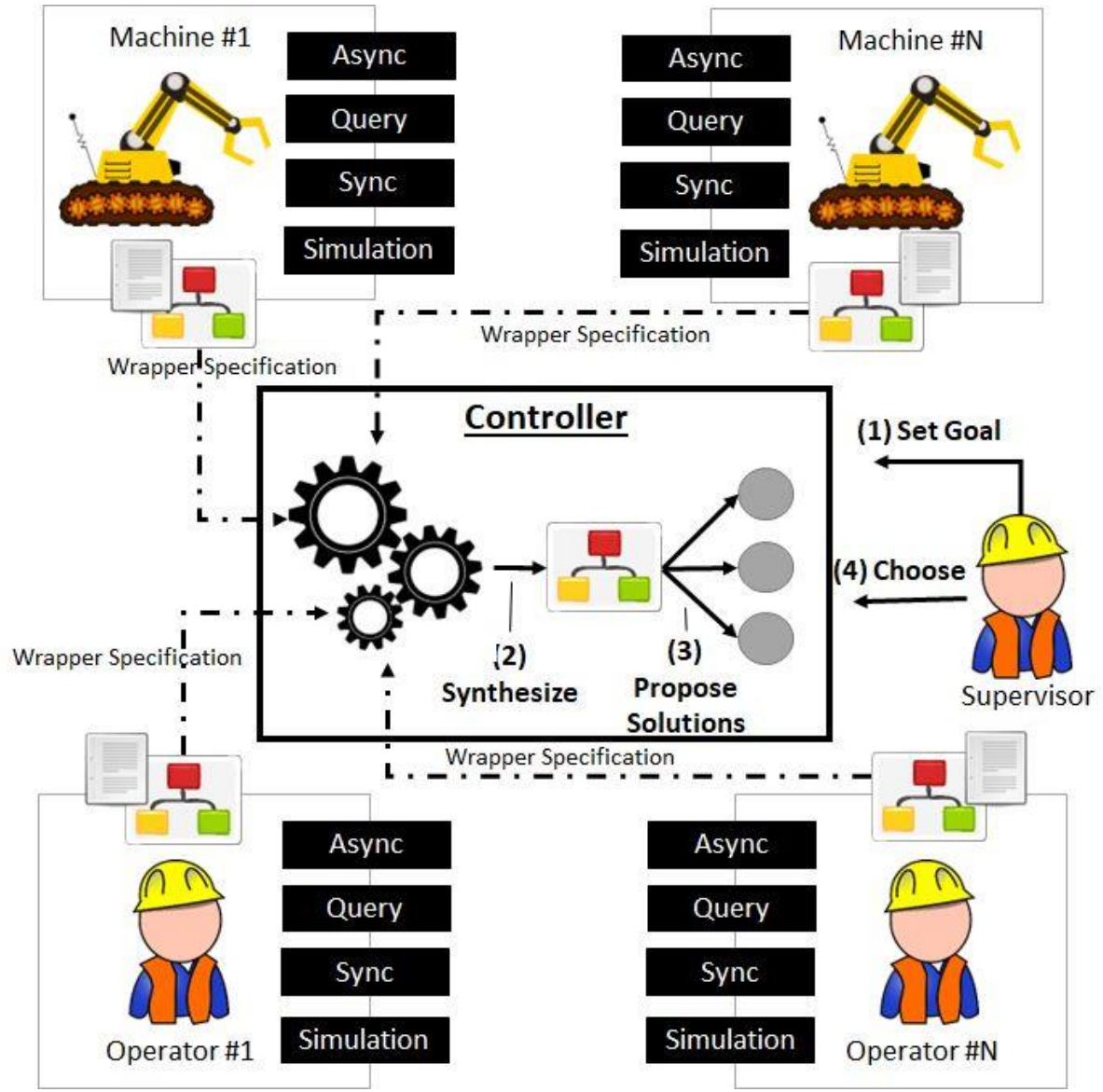
MONASH University

- The "K" in MAPE-K provides the adaptation capability through Machine Learning (ML)
- But AI/ML is itself characterized by uncertainty!
  - Its outputs may or may not be appropriate to a given situation
- A possible architecture for dealing with this:



AIDA reference framework

- A human supervisor provides target(s) / goal(s)
  - A **controller** proposes solutions satisfying the goals
  - Multiple solutions, given the current context
- New solutions proposed at runtime for resilience and responsiveness and adaptivity



Catarci, T., Firmani, D., Leotta, F., Mandreoli, F., Mecella, M., & Sapio, F. A conceptual architecture and model for smart manufacturing relying on service-based digital twins. In *IEEE ICWS 2019*

TECHNICALITIES

Target description language (. tdl)

- Specification of the manufacturing goal
- JSON file

```
target_template.tdl
{
  "target": "_target definition"
}
```

Examples

MDP-based

```
{
  "target": ["exactly_once:build_retrieve_stator",
             "exactly_once:build_retrieve_rotor",
             "exactly_once:build_retrieve_inverter",
             "exactly_once:running_in",
             "exactly_once:assemble_motor",
             "absence_2:electric_test",
             "absence_2:painting",
             "absence_2:static_test",
             "alt_succession:build_retrieve_stator, assemble_motor",
             "alt_succession:build_retrieve_rotor, assemble_motor",
             "alt_succession:build_retrieve_inverter, assemble_motor",
             "alt_succession:assemble_motor, running_in",
             "alt_precedence: assemble_motor, painting",
             "alt_precedence: assemble_motor, electric_test",
             "alt_precedence:assemble_motor, static_test",
             "not_coexistence:electric_test, static_test"
           ]
}
```

planning-based

```
{
  "target": ["run_in, motor, true",
             "el_test, motor, true",
             "st_test, motor, true",
             "painted, motor, true"
           ]
}
```

Service description language (. sdl)

- Specification of the manufacturing actors
- JSON file

```
service_template.tdl
{
  "id": "<service_name>",
  "attributes": {
      "type": "<service_type>",
      "_comment": "static properties"
  },
  "features": {
      "_comment": "dynamic properties"
  }
}
```

`attributes` are used to model types and static immutable values, e.g., definitions of actions, serial number of the device, etc.
Note that if the type definition should be initialized, this is modeled in `attributes`

`features` are used to model dynamic values that may change during runtime.
Wrappers continuosly update this section of the `.sdl` files in order to keep the controller updated

Examples

MDP-based


```
{
  "id": "rotor1",
  "attributes": {
    "type": "service",
    "transitions": {
      "ready": {
        "config_rotor_builder": [
          {
            "configured": 1.0
          },
          [-1.0, 0.0]
        ]
      },
      "configured": {
        "checked_rotor_builder": [
          {
            "executing": 0.95,
            "broken": 0.05
          },
          [-2.0, 0.0]
        ]
      },
      "executing": {
        "build_retrieve_rotor": [
          {
            "ready": 0.95,
            "broken": 0.05
          },
          [-2.5, 2.425]
        ]
      },
      "broken": {
        "restore_rotor_builder": [
          {
            "repairing": 1
          },
          [-5.0, 0.0]
        ]
      },
      "repairing": {
        "repaired_rotor_builder": [
          {
            "ready": 1
          },
          [-3.0, 0.0]
        ]
      }
    },
    "initial_state": "ready",
    "final_states": [
      "ready"
    ]
  },
  "features": {
    "transition_function": {
      "ready": {
        "config_rotor_builder": [
          {
            "configured": 1.0
          },
          [-1.0, 0.0]
        ]
      },
      "configured": {
        "checked_rotor_builder": [
          {
            "executing": 0.95,
            "broken": 0.05
          },
          [-2.0, 0.0]
        ]
      },
      "executing": {
        "build_retrieve_rotor": [
          {
            "ready": 0.95,
            "broken": 0.05
          },
          [-2.5, 2.425]
        ]
      },
      "broken": {
        "restore_rotor_builder": [
          {
            "repairing": 1
          },
          [-5.0, 0.0]
        ]
      },
      "repairing": {
        "repaired_rotor_builder": [
          {
            "ready": 1
          },
          [-3.0, 0.0]
        ]
      }
    },
    "current_state": "ready"
  }
}
```


planning-based


```
{
  "id": "rotor_builder1",
  "attributes": {
    "type": "ServiceRotor",
    "actions": {
      "building_rotor": {
        "properties": {
          "type": "operation",
          "command": "build_rotor",
          "cost": 2,
          "parameters": [
            "ObjectRotor - o"
          ],
          "requirements": {
            "positive": [
              "o.built:false",
              "status:available"
            ]
          },
          "effects": {
            "added": [
              "o.built:true"
            ],
            "deleted": [
              "o.built:false"
            ]
          }
        }
      }
    },
    "features": {
      "status":{
        "properties": {
          "value": "available"
        }
      }
    }
  }
}
```

APPROACHES

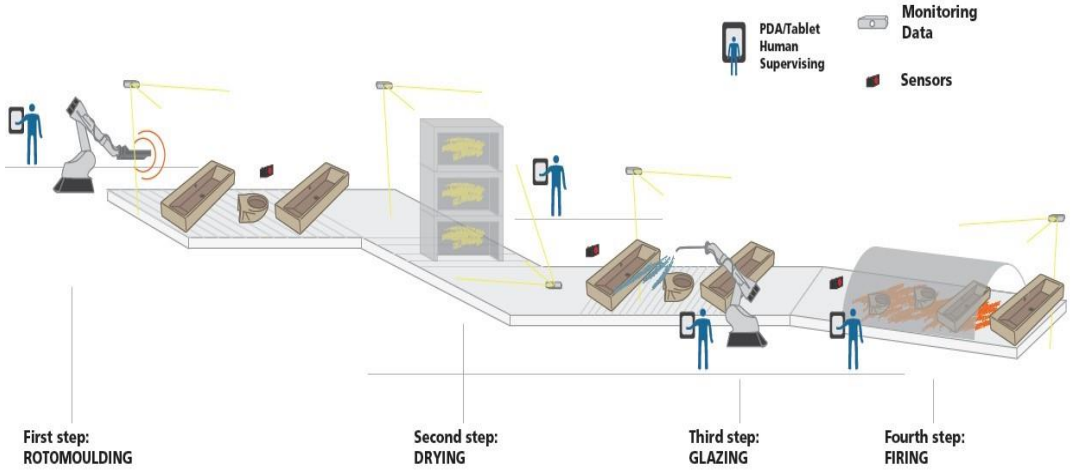
AI and The Control Problem

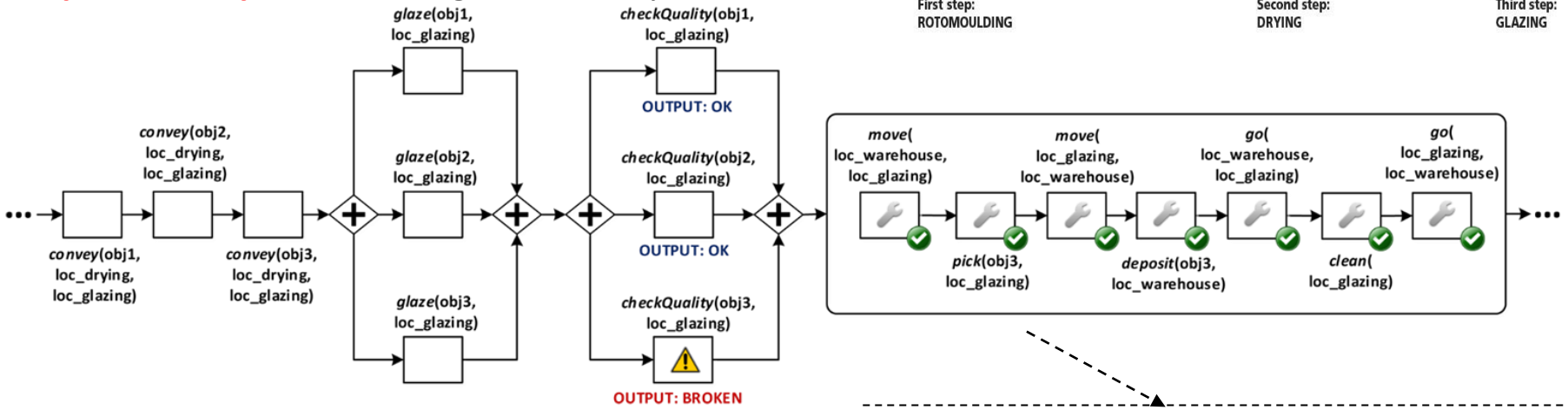
The key problem is to select the action to do next. This is the so-called control problem. Three approaches to this problem:

- Programming-based: Specify control by hand
  - Advantage: domain-knowledge easy to express
  - Disadvantage: cannot deal with situations not anticipated by programmer
- Learning-based: Learn control from experience
  - E.g., Reinforcement learning
- Model-based: Specify problem by hand, derive control automatically
  - **Automated Planning**
- Approaches not orthogonal though

Industrial Process Responsiveness

- It is the ability of a process to **cope with exceptions** and **deviate at run-time** from the execution path prescribed by the process.
- Different kinds of exceptions:
  - **anticipated exceptions**, captured in the process model at design-time
  - **unanticipated exceptions**, managed manually at run-time.





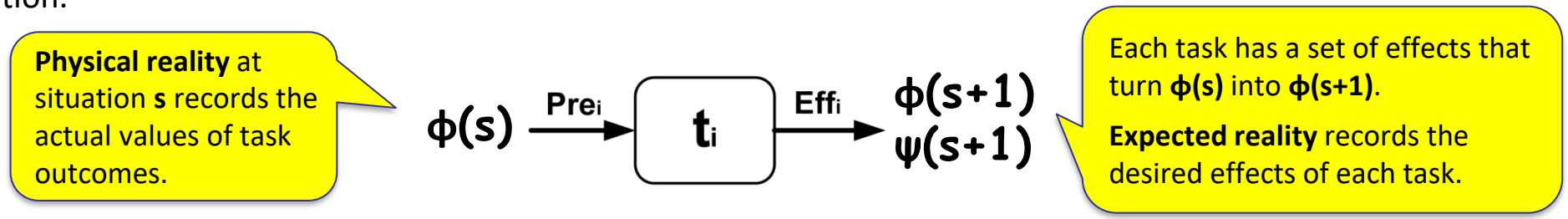
If a deformation after the glazing step is evaluated as critic and not anymore "adjustable", it is useless to proceed to the next step of the process.

Recovery procedure: A moving robot pick up and deposit the broken element in a warehouse and a technician clean the conveyor belt from debris.

Responsiveness through classical planning

Modeling approach towards a **declarative specification** of process tasks.

- Each task is described with the needed preconditions for executing it and the expected effects produced after the task execution.



Process Adaptation*:* the ability to reduce the gap from the expected reality **ψ(s)** – the (idealized) model of reality used to reason – and the physical reality **ϕ(s)**.

Intuition: ***for each execution step***
if **ϕ(s+1)** is different from **ψ(s+1)**
then **adapt**

The aim is to find a recovery procedure that turns ϕ(s) (the faulty physical reality) into ψ(s) (the desired expected reality). **This can be done, for example, by classical planning**.

Automated Planning

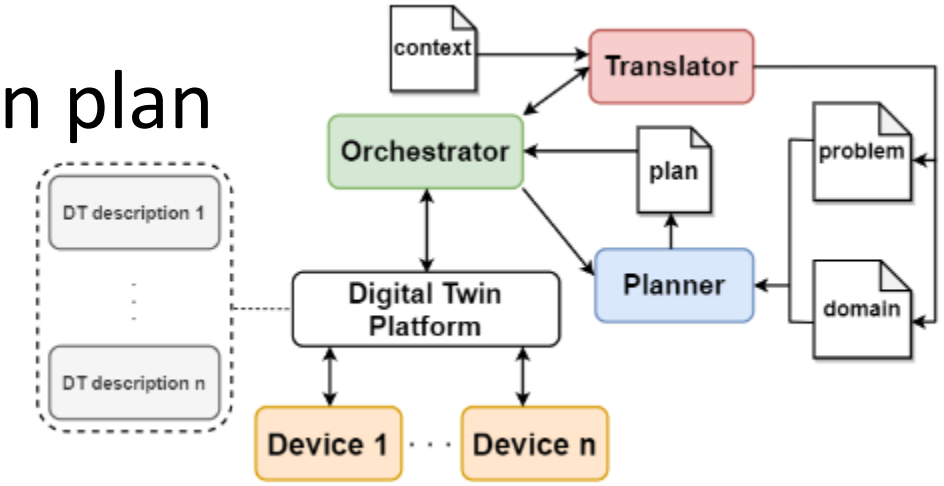
(Classical) Planning concerned with finding a sequence of actions that transforms an initial state into a goal state. This is called a plan

- States are truth assignments, represented by the atoms that are true
- Actions add certain atoms and delete others, provided their preconditions hold
- A planner is a solver that takes a planning problem (initial and goal states, and actions) and outputs a plan
- The cost of a plan is given by the sum of action costs (default is 1)

Init, Actions, Goals ⇒ Planner ⇒ *Plan*

Planning-based AIDA

- Apply classical planning to Industrial APIs
- Translate service descriptions into a PDDL domain
- Describe a production goal
- Use planning for a complete production plan
- Continuously monitor
- Decide whether to fix the current plan or to completely replan
  - Difference between mass- and custom- production



```
{"thing":{
    "thingId":"com.myThings:pr",
    "attributes":{
      "type":"printer"},
    "features":{
      "printing":{
        "properties":{
          "type":"operation",
          "command":"print",
          "cost":1,
          "parameters":[
            "product - p",
            "position - x"],
          "requirements":{
            "positive":[
              "at:x"],
            "negative":[
              "p.processed"] },
          "effects":{
            "added":[
              "p.processed",
              "p.at:x"] } } },
      "at":{
        "properties":{
          "type":"state",
          "value":{
            "type":"position",
            "current":"pos32" } } },
      "status":{
        "properties":{
          "value":"free" } } } } }
```

```
(:types
  position
  typeA typeB - piece
  piece product - movable
  printer cutter assembler - service
  printing cutting assembling - capability
  movable service  - object)

(:objects
    p - product
    ta - typeA
    pr - printer
    printing - capability
    pos11 pos12 pos13 pos21 pos22 ... - position
    ...)


 (:action PRINT
   :parameters    (?srv - service
                   ?p - product
                   ?x - position)
   :precondition (and (provides ?srv printing)
                   (not(processed ?p))
                   (at ?srv ?x))
   :effect        (and (processed ?p)
                   (at ?p ?x)))


  (:predicates (at ?o - object ?p - position)
               (cut ?p - piece)
               (processed ?p - product))


  (:goal (and (processed p)(at p pos33))
```

Planning-based AIDA

- Limitations
  - Preconditions and effects of actions as conjunctions of boolean predicates;
    - Discretization and explicit grounding
  - Different Industrial APIs must share a common vocabulary;
    - Data integration techniques and schema matching
      - Runtime or translation time?
  - **Planning takes into account neither non-determinism nor probabilities in the execution of actions, thus resulting in potentially non-optimal or risky plans**
      - **Responsiveness but not really adaptivity → no real agility!!!**
      - **Possible solution: Decision-Theoretic Planning**

Decision-Theoretic Planning

- Previous approaches relies on determinisms of actions
- Addressing uncertainty can be helpful in smart manufacturing as in such environments action outcome is not granted
- Two approaches:
  - Not determinism
  - Probability theory
- No observability, partial or full observability
- PPDDL – Probabilistic PDDL allows to describe uncertain domains
  - Extends PDDL2.1 with probabilistic effects
  - PPDDL extension with non deterministinc «oneof» op.
  - RDDL also supports partial observability

Blythe, J. (1999). Decision-theoretic planning. *AI magazine*, *20*(2), 37-37.

Non Deterministic Planning

- finite and discrete state space S
- a **set of possible initial state** $S\,0 \in S$
- a set $S\,G \subseteq S$ of goal states
- actions $A(s) \subseteq A$ applicable in each $s \in S$
- a **non-deterministic transition function** $F(a, s) \subseteq S$ for $a \in A(s)$
- uniform action costs c(a, s)

If the world is not observable, we have **Conformant Planning**. A solution is still an action sequence but must achieve the goal for any possible initial state and transition

If feedback is available (partial – POND or full observability - FOND), we have **Contingent Planning** → actions depending on observations

- E.g., FOND-SAT is a PDDL planner with “oneof” operator for FOND planning based on SAT

Example: Multi-Tier FOND

```
(:action walk
  :parameters
    (?o - Cell ?d - Cell)
  :precondition (and (at ?o)
    (adj ?o ?d) (not (broken)))
  :effect (and
    (not (at ?o)) (at ?d)) )

(:action run
  :precondition
    (and (at c2) (not (broken)))
  :effect (and
    (not (at c2)) (at c0)) )

(:goal (and (at c0)
            (not (scratch))
            (not (broken)) ))
```

(a) In model $\mathcal{D}_3$, any running and walking always succeeds.

```
(:action walk
 :parameters
   (?o - Cell ?d - Cell)
 :precondition (and (at ?o)
   (adj ?o ?d) (not (broken)))
 :effect (oneof
   (and (not (at ?o)) (at ?d))
   (and (not (at ?o)) (at ?d)
   (scratch))) )

(:action run
  :precondition
    (and (at c2) (not (broken)))
  :effect (oneof
    (and (not (at c2)) (at c0))
    (and (not (at c2)) (at c0)
    (scratch)) ))

(:goal (and (at c0) (not
(broken))) )
```

(b) In model $\mathcal{D}_2$, agent may move successfully but suffer minor scratch damage.

```
(:action walk
  :parameters (?o - Cell ?d - Cell)
  :precondition (and (at ?o)
    (adj ?o ?d) (not (broken)))
  :effect (oneof
    (and (not (at ?o)) (at ?d))
    (and (not (at ?o)) (at ?d)
    (scratch))
    (scratch) ))

(:action run
  :precondition
    (and (at c2) (not (broken)))
  :effect (oneof
    (and (not (at c2)) (at c0))
    (and (not (at c2)) (at c0)
    (scratch))
    (broken) ))

(:goal (and (at c2) (not broken)) )
```

(c) In model $\mathcal{D}_1$, movements may actually fail and may even leave the robot broken.

- If assumptions of plan Di fail, degrate to Di+1
- Translate a MTD (Multi-Tier Domain) in a corresponding Dual FOND problem
  - Actions are supposed to be unfair, so a strong policy is required to obtain a strong cyclic plan
- Apply FOND-SAT

Ciolek, D., D'Ippolito, N., Pozanco, A., & Sardiña, S. Multi-Tier Automated Planning for Adaptive Behavior. In *ICAPS 2020*.

Synthesis with Markov Decision Processes

An MDP $\mathcal{M} = \langle \Sigma, A, P, R, \lambda \rangle$ is a **fully observable, probabilistic** state model:

- a set of states Σ
- a set of actions A
- a transition function $P : \Sigma \times A \rightarrow Prob(\Sigma)$
- a reward function $R : \Sigma \times A \rightarrow \mathbb{R}$
- discount factor λ

Synthesis with Markov Decision Processes

- A solution to an MDP is a function *policy* ρ that assigns an action to each state
- The *value of a policy* $v_\rho(\sigma)$ is the expected sum of rewards when starting at certain state σ and selecting actions based on the policy ρ
- Every MDP has an *optimal policy* ρ^* that maximizes the value of a policy

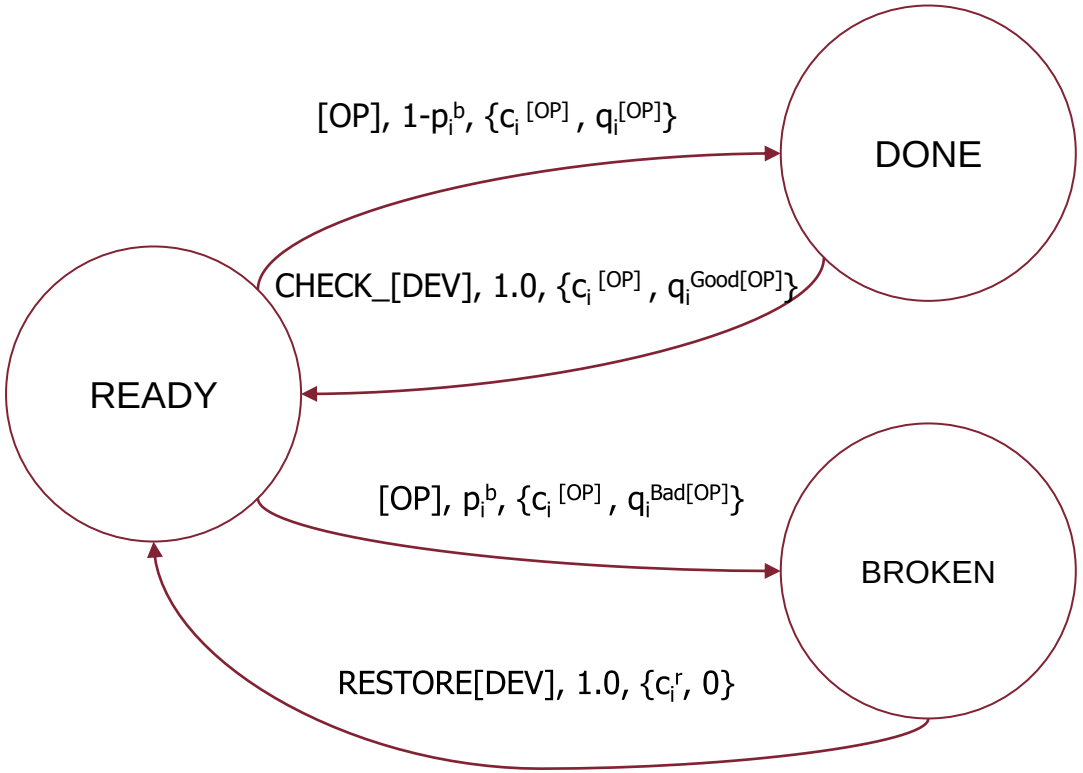
Lexicographic Markov Decision Processes

- An **LMDP** maximizes the expected cumulative rewards following a *lexicographic preference* over the reward function
- Reward function now is a *vector of reward function*

$$\mathbf{R} = [R_1(s, a, s'), \ldots, R_k(s, a, s')]$$

- An optimal solution to an LMDP is a policy ρ^* which assigns an action to each state and maximizes the expected cumulative rewards

Example - actors



Two objectives:
minimizing the cost and maximizing the overall quality of the product.
Vector of rewards has two different costs:
-the *economic cost* ($c_i^{[OP]}$)
-the *quality cost* ($q_i^{[OP]}$)

The execution of [OP] may take the actor$_i$ to the **BROKEN** state with probability p_i^b with a bad $q_i^{[OP]}$ or may take the actor$_i$ in the **DONE** with a probability 1-p_i^b, then the actor return ready to perform the next operation with a good $q_i^{[OP]}$

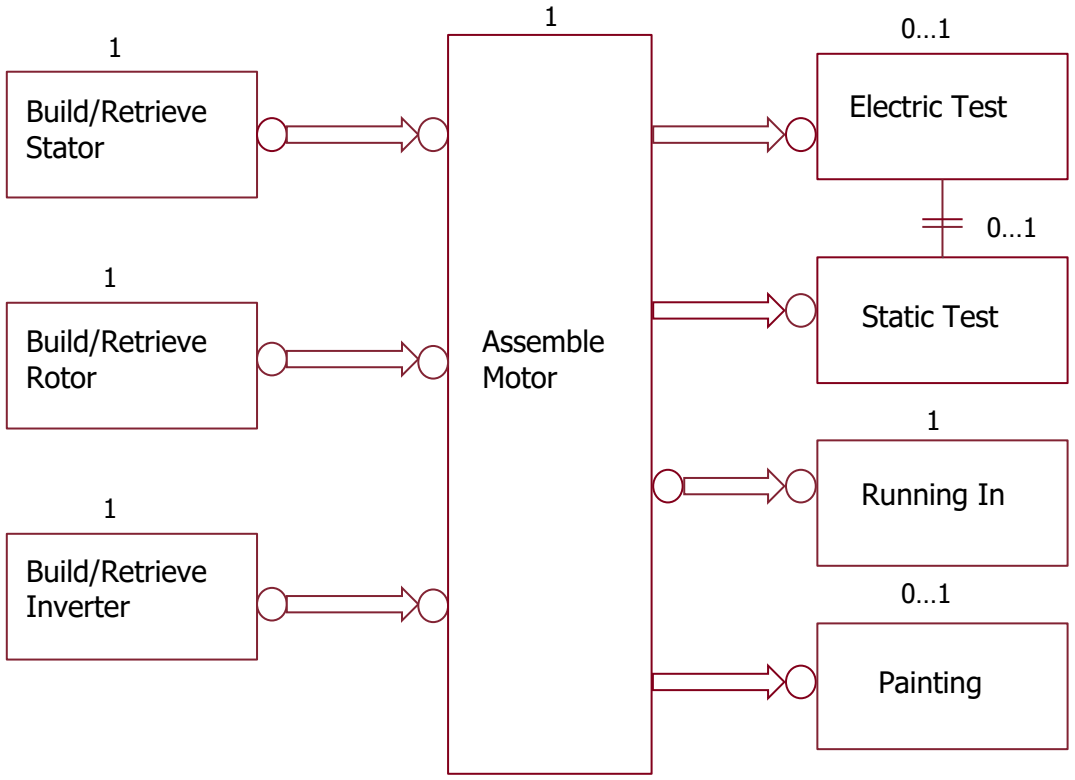
Problem formalization

- The manufacturing process is modeled using **DECLARE**
  - language and framework for the declarative, constraint-based modeling of processes and services
- DECLARE assumption can be transformed in a **LTL_f formula** φ from which we can generate **the target DFA[1]** obtained by:
  - trimming the DFA generating from the formula, by removing every state from which no final state is reachable

[1] DFA – Deterministic Finite-state Automata
Lydia tool:https://github.com/whitemech/lydia

Example - manufacturing process

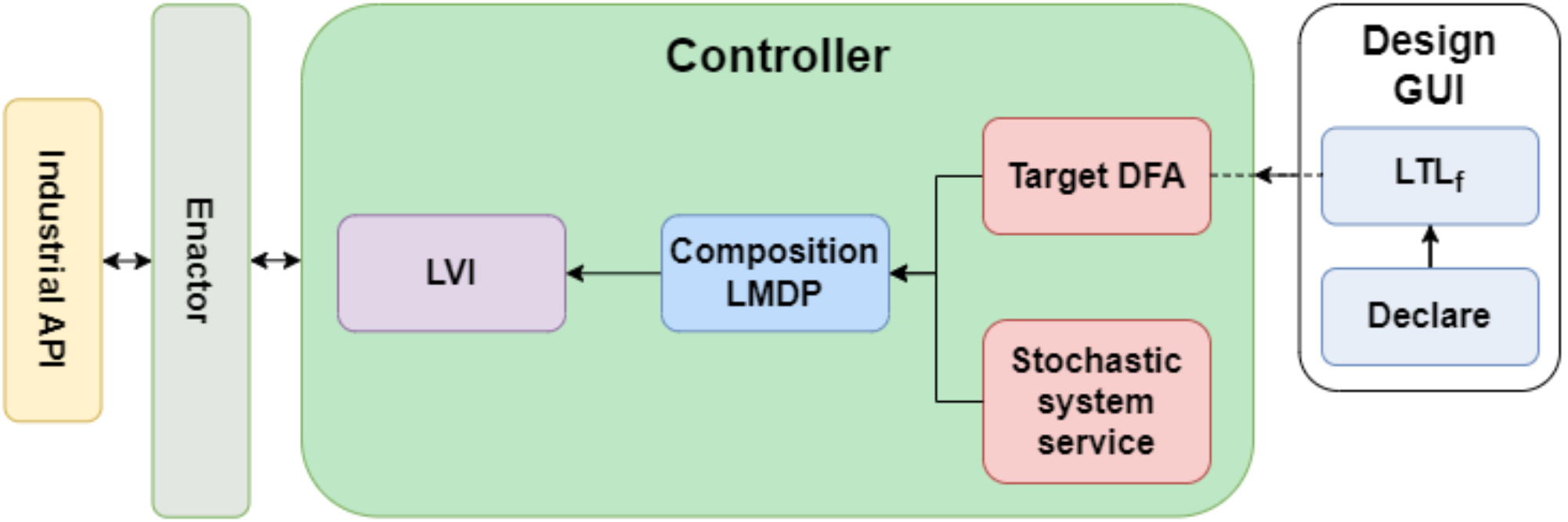
- Electric motor case study



Solution technique through LMDPs

- Given the *target dfa* $\mathcal{A}_\varphi = \langle \mathcal{P}, Q, q_0, F, \delta \rangle$ and the *stochastic system service* $\mathcal{Z} = \langle \Sigma_z, \sigma_{z0}, F_z, A_z, P_z, R_z \rangle$ compute the **composition LMDP** $\mathcal{M}(\mathcal{Z}, \mathcal{A}_\varphi) = \langle S_\mathcal{M}, A_\mathcal{M}, T_\mathcal{M}, \mathbf{R}_\mathcal{M}, \lambda \rangle$:
  - $S_\mathcal{M} = \Sigma_z \times Q$
  - $A_\mathcal{M} = A_z \times \{1, \ldots, n\}$
  - $T_\mathcal{M}((\boldsymbol{\sigma}, q), \langle a, i \rangle, (\boldsymbol{\sigma'}, q')) = P_z(\boldsymbol{\sigma}' \mid \boldsymbol{\sigma}, \langle a, i \rangle)$
  - $\mathbf{R}_\mathcal{M}((\boldsymbol{\sigma}, q), \langle a, i \rangle, (\boldsymbol{\sigma}', q'))$ is a vector of *m+1* elements

Architecture of the solution

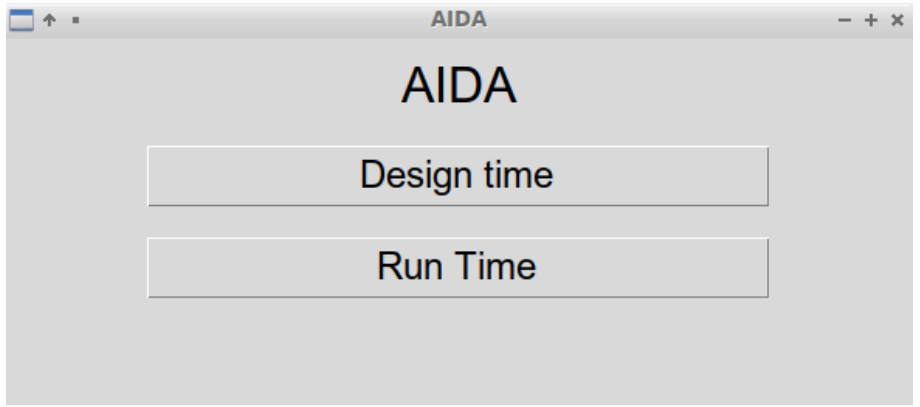


CASE STUDY AND DEMO

Case Study: manufacturing of an electric motor

- Components:
  - rotor, stator, inverter
- Types of actors:
  - warehouse, builder machine, assembler machine, smart tester, human operator
  - each type
    - supports a specific set of actions (e.g., assembler machine supports the *assemble* operation)
    - defines a specific set of states (e.g., ready, configuring, executing, broken, repairing)
- Multiple instances of each type of actors
  - E.g., `rotor_builder1`, `rotor_builder2`, `rotor_warehouse`
- The manufacturing process imposes a set of rules. E.g., all components must be built before assembling.

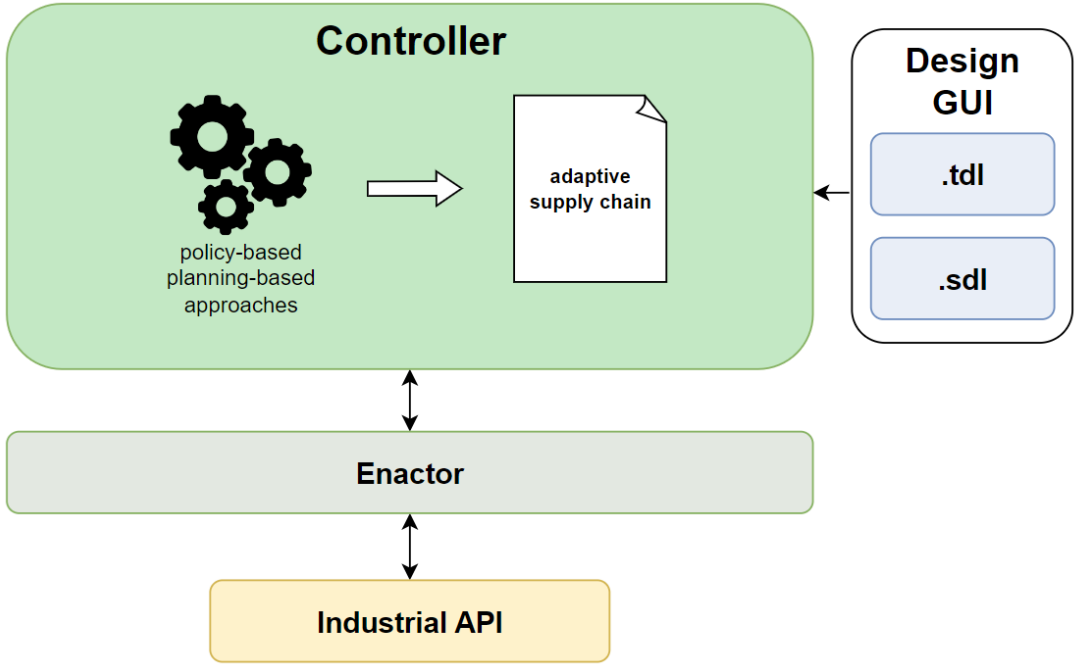
AIDA - Adaptive InDustrial APIs tool



- Design-time for the modeling of the services
- Run-time for the execution phase of the planning-based and policy-based approaches
  - Deployment of services through an Industrial API
  - Production plan (consisting of service actions) executed step by step
  - Current status of services highlighted with colors
    - ready, configuring, executing, broken, repairing

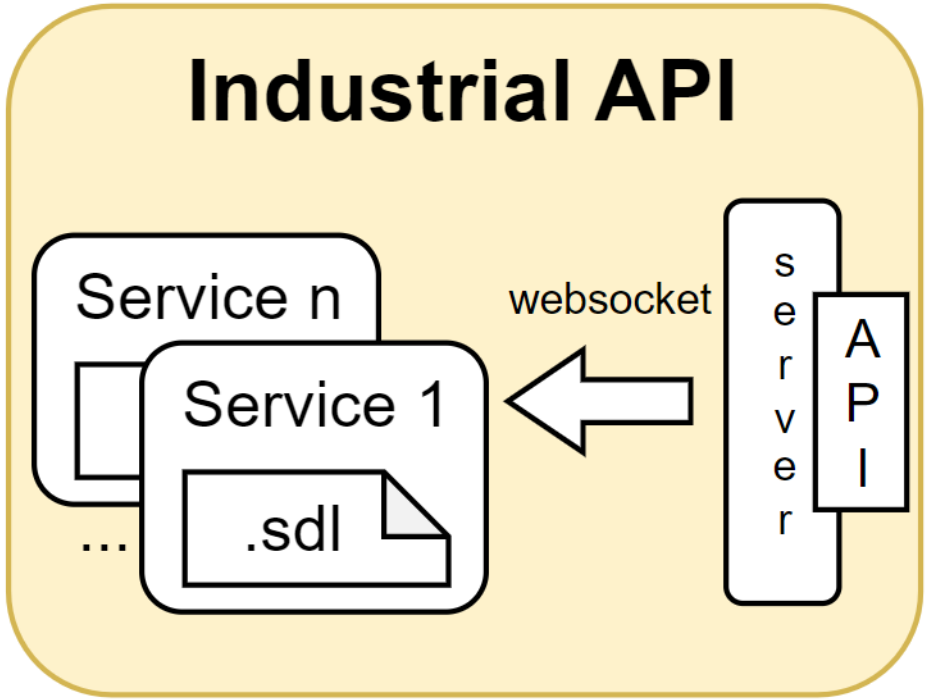
AIDA architecture

- **Design GUI**: graphic tool to model manufacturing actors and goal
- **Controller**: composing the manufacturing actors
- **Enactor**: acting as a middleware between controller and Industrial API
- **Industrial API**: realizing the services wrapping the manufacturing actor

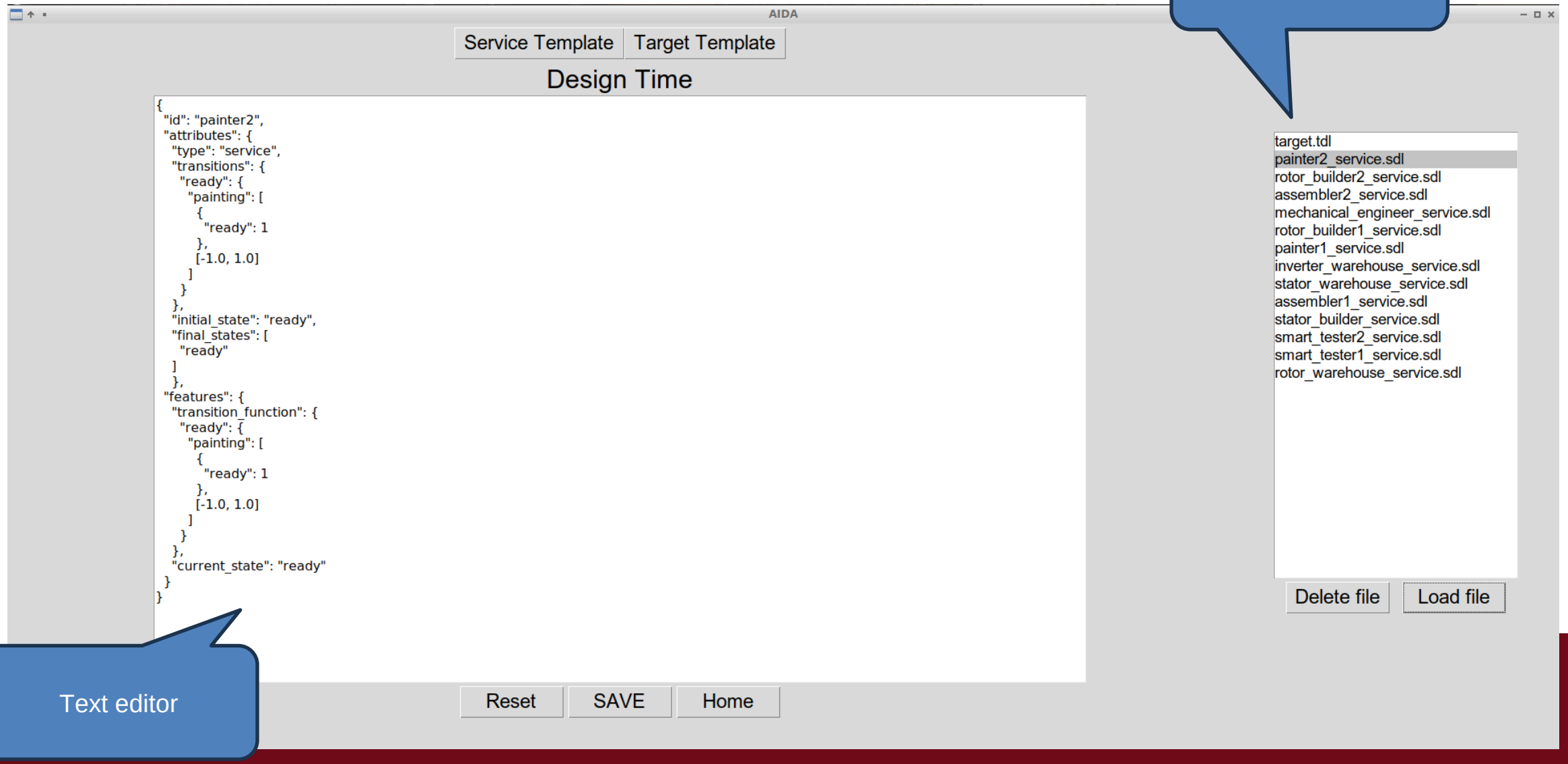


Industrial API

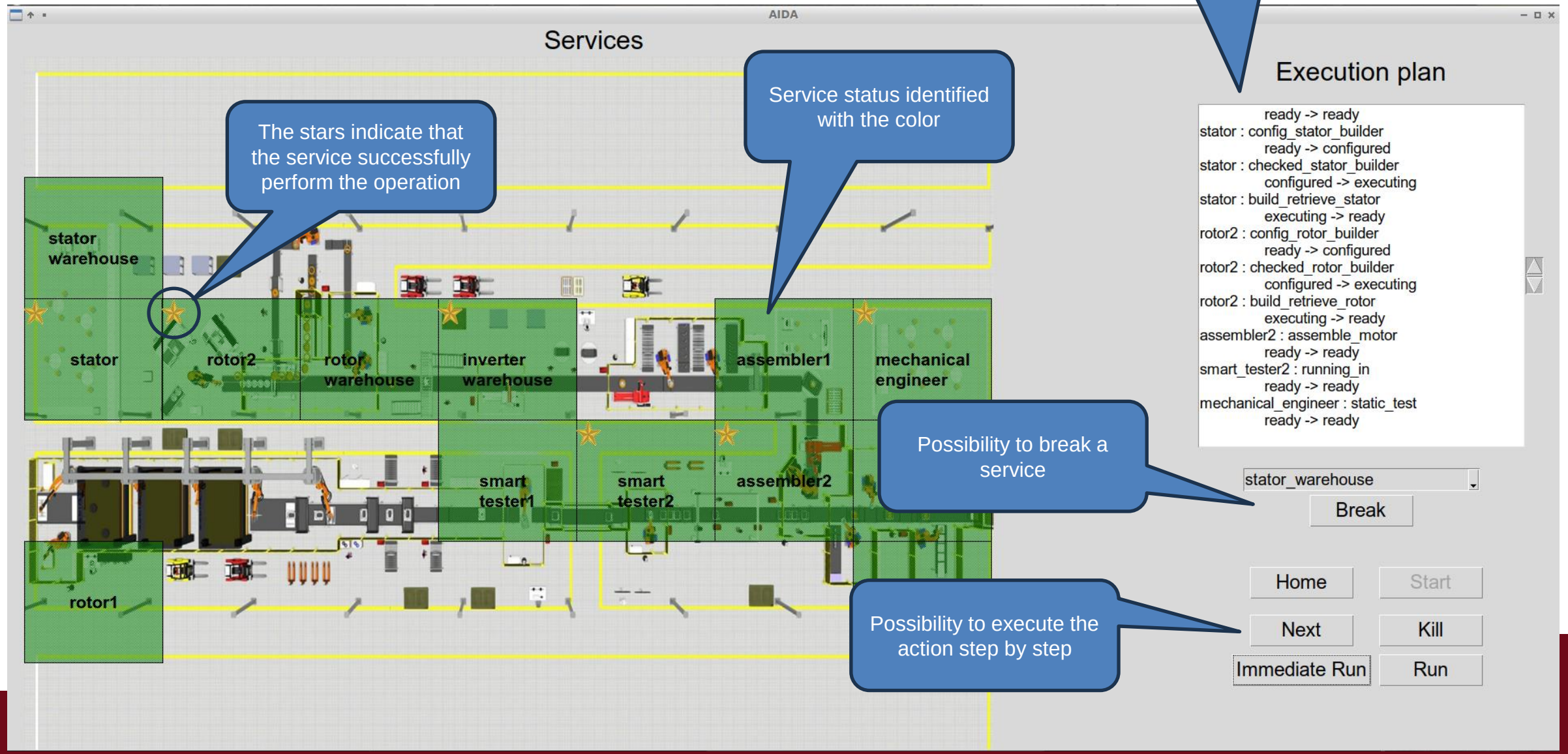
- The *controller* interact with the Industrial API through REST API
  - Retrieve service specification and current status
  - Request the execution of an action by a service
- The services are connected to the server via websocket
- Services connect to the server
  - to register
  - to execute actions



Design GUI

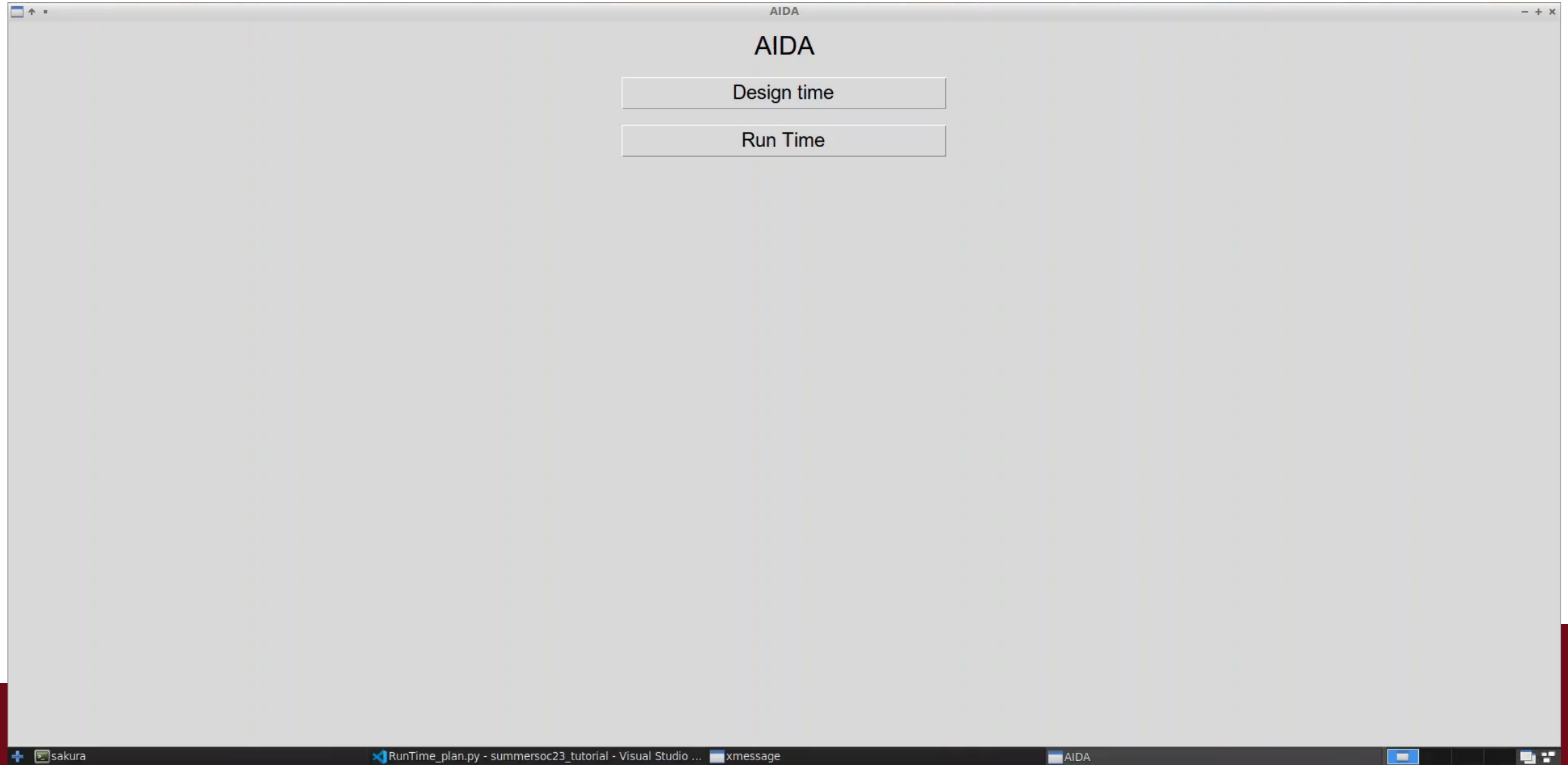

AIDA
Service Template
Target Template
Design Time
{
"id": "painter2",
"attributes": {
"type": "service",
"transitions": {
"ready": {
"painting": [
{
"ready": 1
},
[-1.0, 1.0]
]
}
},
"initial_state": "ready",
"final_states": [
"ready"
]
},
"features": {
"transition_function": {
"ready": {
"painting": [
{
"ready": 1
},
[-1.0, 1.0]
]
}
},
"current_state": "ready"
}
}
Reset
SAVE
Home
List of .sdl and .tdl files
target.tdl
painter2_service.sdl
rotor_builder2_service.sdl
assembler2_service.sdl
mechanical_engineer_service.sdl
rotor_builder1_service.sdl
painter1_service.sdl
inverter_warehouse_service.sdl
stator_warehouse_service.sdl
assembler1_service.sdl
stator_builder_service.sdl
smart_tester2_service.sdl
smart_tester1_service.sdl
rotor_warehouse_service.sdl
Delete file
Load file
Text editor

Run-Time


AIDA
Services
The stars indicate that the service successfully perform the operation
Service status identified with the color
List of performed action
Execution plan
ready -> ready
stator : config_stator_builder
ready -> configured
stator : checked_stator_builder
configured -> executing
stator : build_retrieve_stator
executing -> ready
rotor2 : config_rotor_builder
ready -> configured
rotor2 : checked_rotor_builder
configured -> executing
rotor2 : build_retrieve_rotor
executing -> ready
assembler2 : assemble_motor
ready -> ready
smart_tester2 : running_in
ready -> ready
mechanical_engineer : static_test
ready -> ready
stator warehouse
stator
rotor2
rotor warehouse
inverter warehouse
assembler1
mechanical engineer
smart tester1
smart tester2
assembler2
rotor1
Possibility to break a service
stator_warehouse
Break
Home
Start
Next
Kill
Immediate Run
Run
Possibility to execute the action step by step

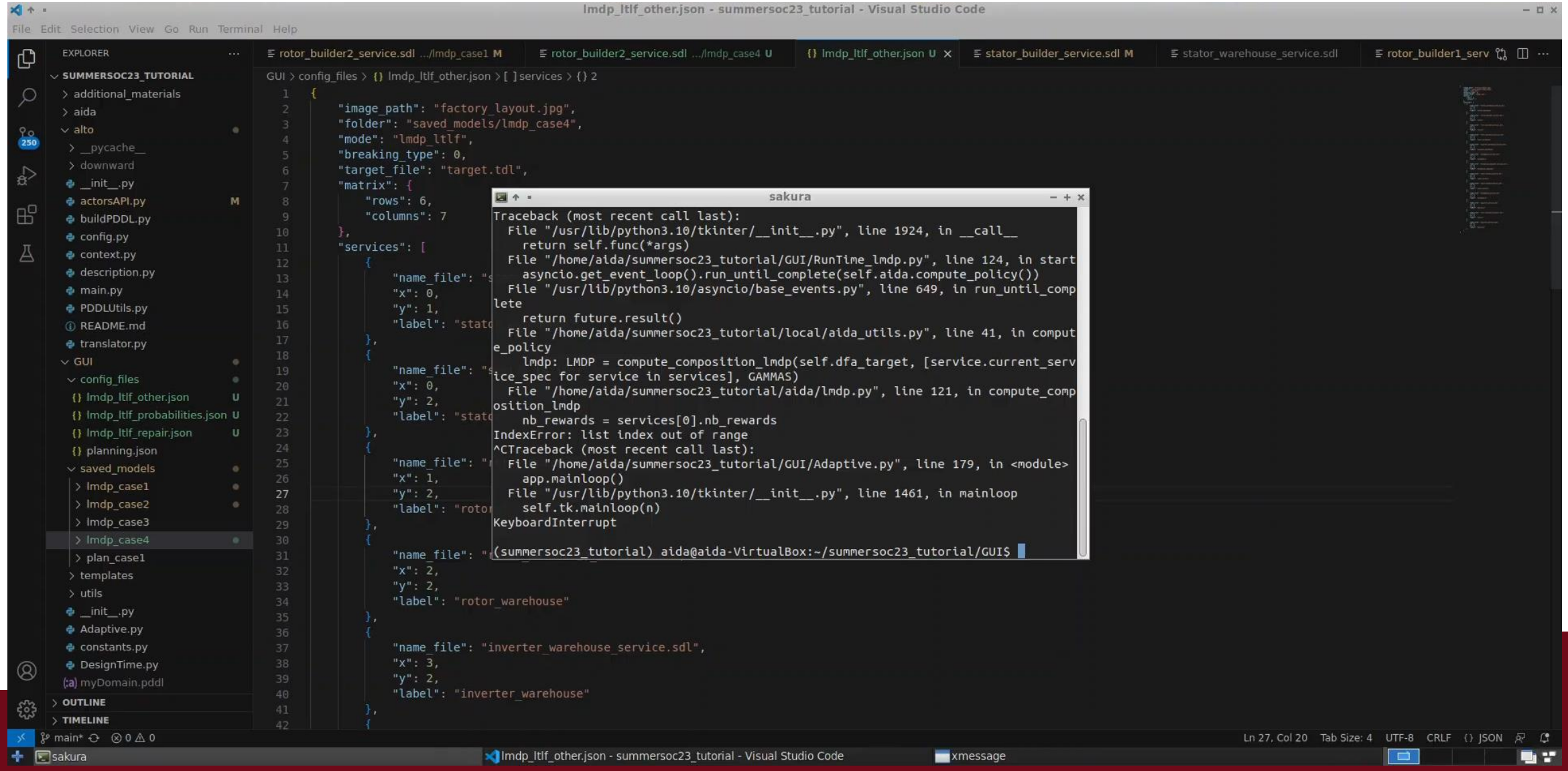
DEMO: planning-based AIDA (re-plan upon exception)

https://drive.google.com/file/d/1a8-p26mCT8TphWGN5Gknsml_k5oYwCbK/view?usp=sharing



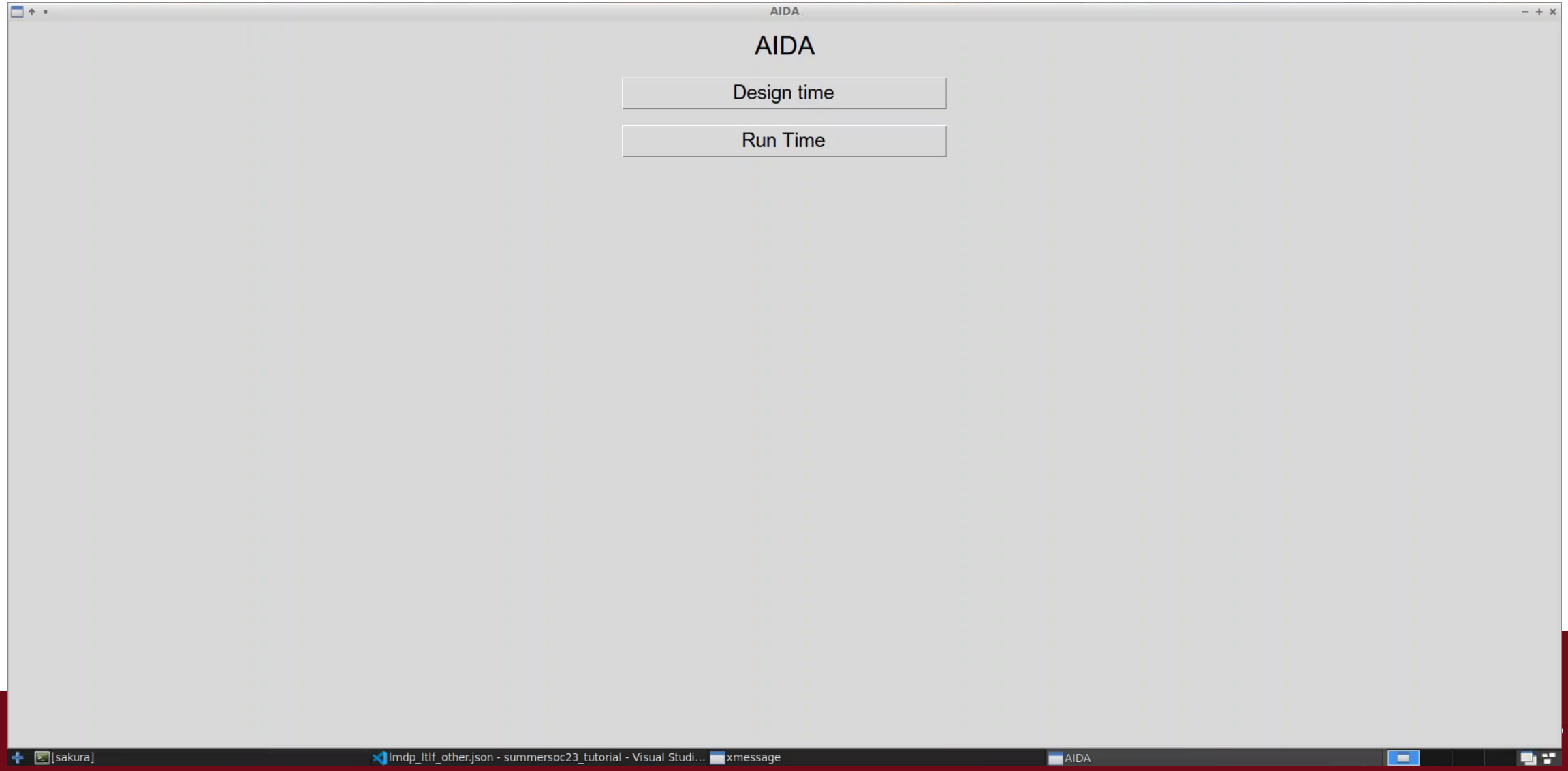
DEMO: policy-based AIDA (Choose an alternative)

https://drive.google.com/file/d/1aEQFftG_DWsHJtqyfyY7l4Opwk18b9Iu/view?usp=sharing



DEMO: policy-based AIDA (Choose to repair)

https://drive.google.com/file/d/1aGf_E6uiUxZ5a19r2n5urUHLSAuG8Pzo/view?usp=sharing



AIDA Tool

- GitHub repository: iaiamomo/summersoc23_tutorial (github.com)

CONCLUDING REMARKS

An experimental analysis (cf. the paper by Monti et al presented in the afternoon)

- Preliminary experiments based on *automated planning* and *stochastic* approaches
- Automated planning – classical planning
  - Take into account neither non-determinism nor probabilities in the execution of the operation
  - Resulting in potentially non-optimal or risky plans
  - Slowly growing execution time and memory consumption when increasing the number of services employed thanks to efficient heuristics
- Stochastic approaches – Markov Decision Processes (MDPs) and DECLARE formulas
  - Take into account the stochastic behaviors of the actors
  - Solutions are functions (policies) mapping states into actions and minimize expected cost to goal
  - Memory consumption and execution time increase impractically with the number of services employed

	Small case		Manageable case #1		Manageable case #2		Complex case	
	Time (s)	Memory (MiB)	Time (s)	Memory (MiB)	Time (s)	Memory (MiB)	Time (s)	Memory (MiB)
Instance planning	0.14	26.4	0.14	26.5	0.18	26.4	0.31	26.7
Stochastic policy	16.43	2 245.53	905.51	119 057.86	7 834.65	845 834.16	Failed	Failed
Stochastic constraint-based policy	85.86	134 721.56	4 067.72	293 288.93	Failed	Failed	Failed	Failed

…

- We have shown how to model and compose services at different levels of modeling expressiveness, both for the services and the targets/goals
- Approaches based on «precise» automated reasoning techniques can lead to unpractical problems to be solved
- It’s time to include ML, and more generally, generative AI in the «game», but …

…

- ML/genAI will not provide solutions that can be proved to be exact and perfect, is this acceptable in a manufacturing production environment ?
- chatGPT-like approaches can be successfull for service composition in general …
  - Aiello, M., Georgievski, I. Service composition in the ChatGPT era. SOCA (2023). https://doi.org/10.1007/s11761-023-00367-7
- … but are not useful (so far) for I4.0, as data and service descriptions are not in the public Web

…

- Specialized LLMs merged with KR-based / AR-based methods might be a promising solution
- Requirements
  - Training/customization data sets
    - Availability ? Confidentiality ? Committments from industries
  - Layered approach, in which techniques are designed for industrial sectors and then specific needs
  - Integration with MESs – Manufacturing Execution Systems and safety considerations for *humans-in-the-loop*

...

- Robotics (or service robotics) is present in large corporations and specific sectors with high-automation
- But the reality, especially in Europe, is the one of SMEs, which needs specialized and more gradual approaches
  - Already adopting industrial APIs and modeling services and allowing manual composition would be a tremendous improvement