University of Stuttgart
Institute of Industrial Automation
and Software Engineering

Synthetic training data creation for supervised fine-tuning of large language models for autonomous production planning and control

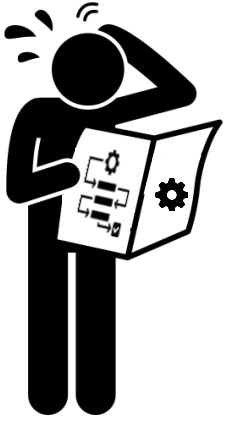
Jize Zhang

Supervisor: Yuchen Xia

Examiner : Prof. Michael Weyrich

Operating a complicated automation equipment?

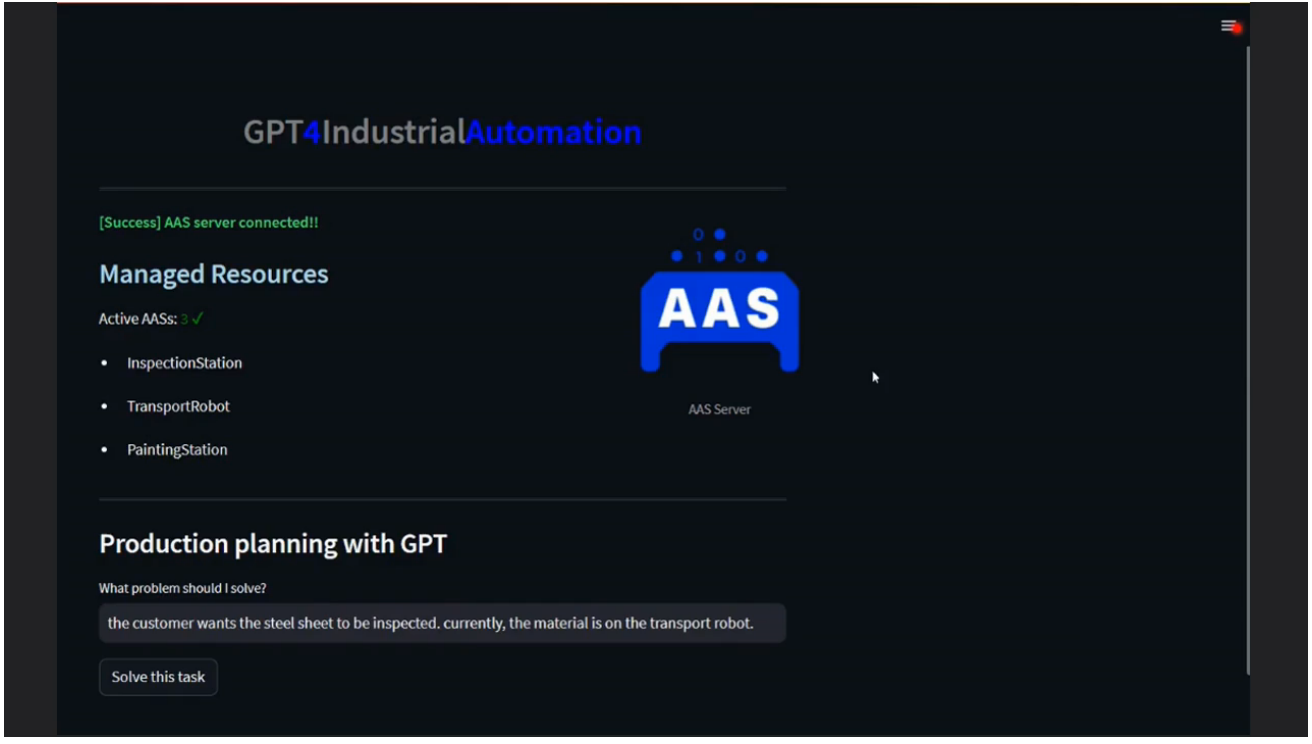
Lower using complexity?



Solution: using natural language commands to interact with autonomous systems

Previous Research

Task:
inspect the steel sheet



[1] Y. Xia, M. Shenoy, N. Jazdi and M. Weyrich, "**Towards autonomous system: flexible modular production system enhanced with large language model agents**," in 2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA), 2023.

Solution: using natural language commands to interact with autonomous systems

Previous Research

Event Log

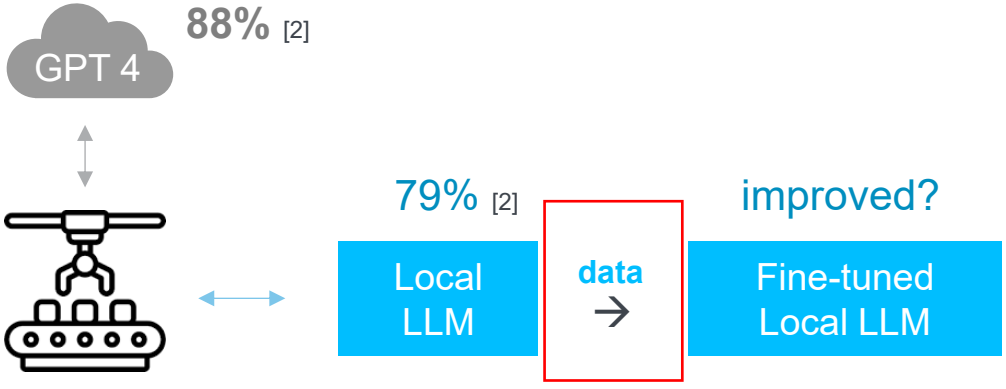
[00:00:14] Sensor BG56 detects an object at the entrance.

LLM Generated Commands

[2] Y. Xia, J. Zhang, N. Jazdi and M. Weyrich, **Incorporating Large Language Models into Production Systems for Enhanced Task Automation and Flexibility**, arXiv preprint arXiv:2407.08550, 2024.

Motivation

Problem Statement



- data security
- network reliability
- accuracy / performance

→ **development with local models**

→ **improve performance**

Synthetic training data creation for **supervised fine-tuning** of large language models for autonomous production planning and control

Outline

- Domain-specific **Tasks of production planning and control**
- Creation of Domain-specific synthetic **Dataset** for fine-tuning
- **Fine-tuning** and **Testing**

Domain-specific Tasks

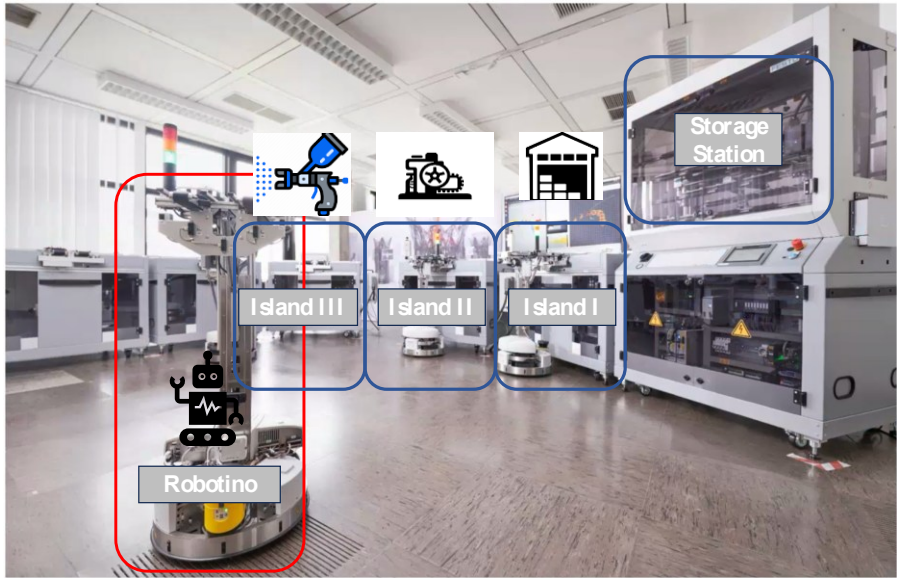
Domain-specific synthetic Dataset for fine-tuning

Fine-tuning and Testing

Domain-specific Tasks

Agents and Tasks

- **Tasks**
  - **Quality Inspection** task (Island I)
  - **Machining process** task (Island II)
  - **Painting process** task (Island III)
  - **Transport** task (Transport Robot)
  - **Coordination** task (Coordinator)

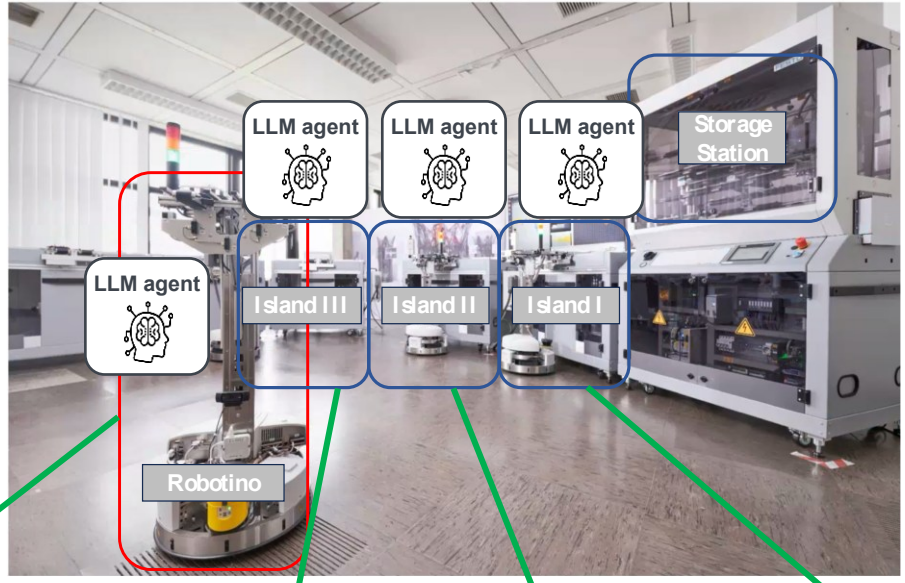


Agents

Definition:
Agent:
A (software) component that is responsible for a specific sub-task

Multi-Agent System Design

Conceptual System Design



Automation Module

Transport Robot Agent transport
data
control

Island III Agent painting
data
control

Island II Agent machining
data
control

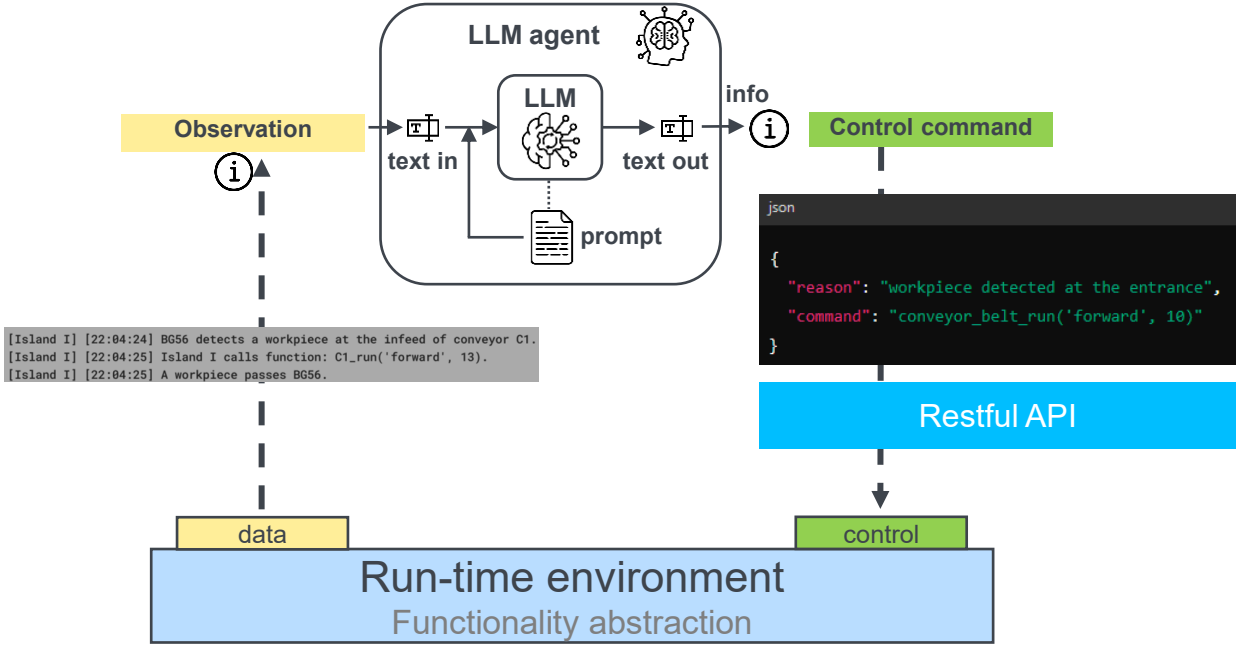
Island I Agent inspection
data
control

Run-time environment

Data / functionality abstraction interface

Multi-Agent System Design

Preliminary Data Interpretation and Control



Synthetic Dataset Creation

Data Interpretation and Commands

a sample

```
[Island I] [22:04:24] BG56 detects a workpiece at the infeed of conveyor C1.
[Island I] [22:04:25] Island I calls function: C1_run('forward', 13).
[Island I] [22:04:25] A workpiece passes BG56.
```

json

```
{
  "reason": "workpiece detected at the entrance",
  "command": "conveyor_belt_run('forward', 10)"
}
```

→ input

→ output / label

Dataset for
Supervised Fine-tuning

one event log-command pair

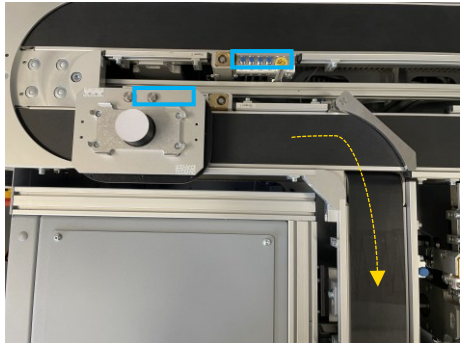
Domain-specific Tasks

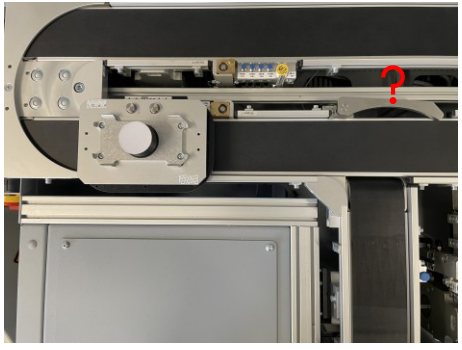
Domain-specific synthetic Dataset for fine-tuning

Fine-tuning and Testing

Synthetic Dataset Creation

Example: Workpiece at the Branch





Types

- **Normal operation**
- **Actuator Failure**
- **Unexpected event**

Events

[Island I] [22:04:28] TF81 reads information from the workpiece. The RFID tag on the workpiece is authorized.

[Island I] [22:04:28] TF81 reads information from the workpiece. The RFID tag on the workpiece is authorized.
[Island I] [22:04:28] Branch fails to divert.

[Island I] [22:04:28] TF81 reads information from the workpiece. The RFID tag on the workpiece is authorized.
[Island I] [22:04:28] Island I called function: branch_divert()
[Island I] [22:04:30] Branch was directed straight.

Command

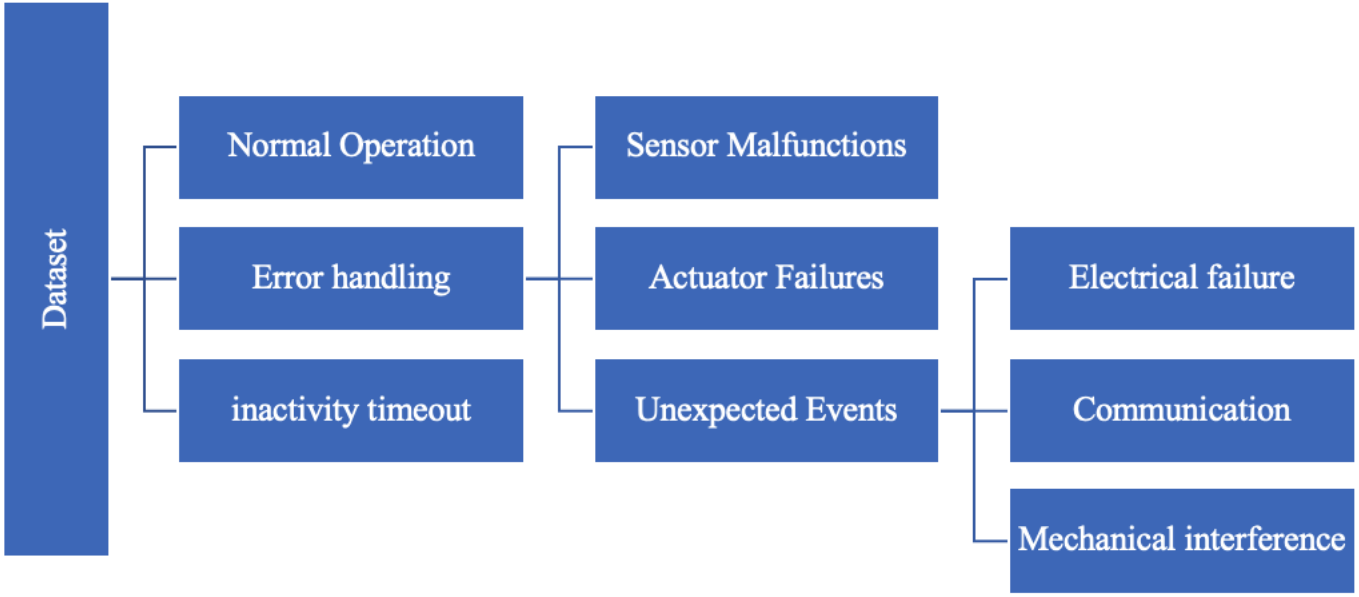
call branch_divert()

call alert_to_supervisor (‘branch failure’)

call branch_divert()

Synthetic Dataset Creation

Types of test samples



Synthetic Dataset Creation

Overview and Statistic

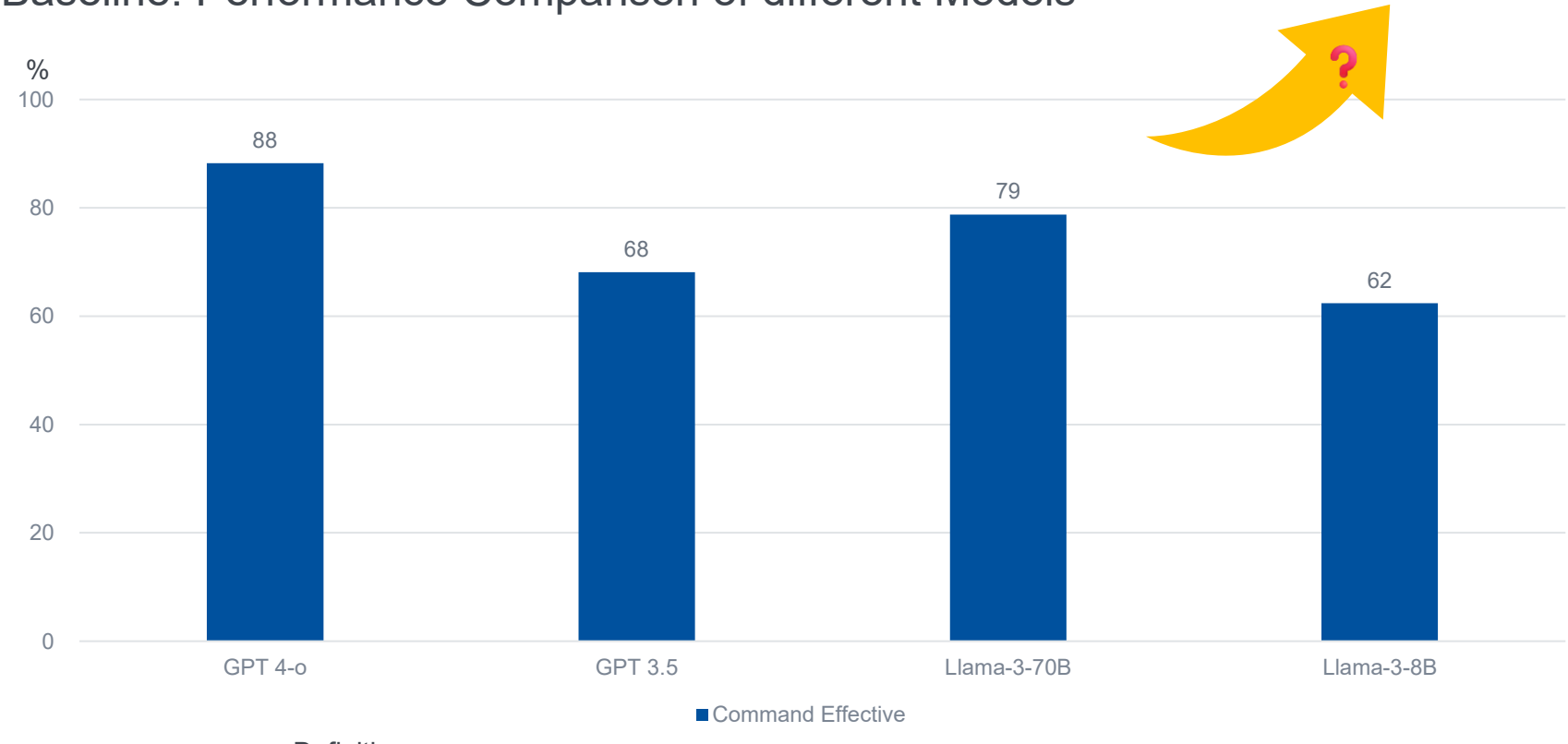
- **Overview:**
  - 120 test cases in total, 24 test cases for each agent
- **Task distribution**

Normal operation (70 samples, 62%) | Error handling(50 samples, 38%)

- **Usage of the dataset**
  - Fine-tuning
  - Evaluation **(before** and **after** fine-tuning**)**

Dataset Usage 1: Model evaluation before fine-tuning

Baseline: Performance Comparison of different Models



Definition:

Command Effective: evaluates if the model can output correct command functions based on observed events

Dataset Usage 2: Fine-tuning and Testing

Fine-tuning methods

- **Fine-tuning methods**
  - SFT (**S**upervised **F**ine-**T**uning)
    - training the model on labeled datasets with human-provided examples to improve its performance on specific tasks
  - RLHF (**R**einforcement **L**earning from **H**uman **F**eedback)
    - refines LLMs by using human feedback to guide the model towards generating more desirable outputs through a reinforcement learning process, optimizing its responses based on reward signals.
- **OPENAI**

  - proprietary
- **Llama3**

  - Full-parameter SFT

Dataset Usage 3: Evaluation after Fine-tuning

Different Test Strategies

LLM input	Generated output	Correct output
[00:05:51] CNC processing is finished. [00:05:51] 'Island II' calls function unload_workpiece(). [00:05:52] Failed to unload the workpiece.	Unload_workpiece()	Alert_to_supervisor ('failed to unload the workpiece')

- **Effectiveness** → **Wholly fine-tuning**
  - how well a model performs on specific tasks after fine-tuning
- potential problems like **Catastrophic Forgetting** and **Overfitting** → **K-Fold fine-tuning**
  - "forgets" how to perform previously learned
  - too tailored to the fine-tuning data and fails to generalize well to unseen or slightly varied inputs
- **Transfer Learning ability** → **Partly fine-tuning**
  - apply knowledge learned from one task or dataset to another, potentially related, task without extensive additional training

Three different fine-tuning methods

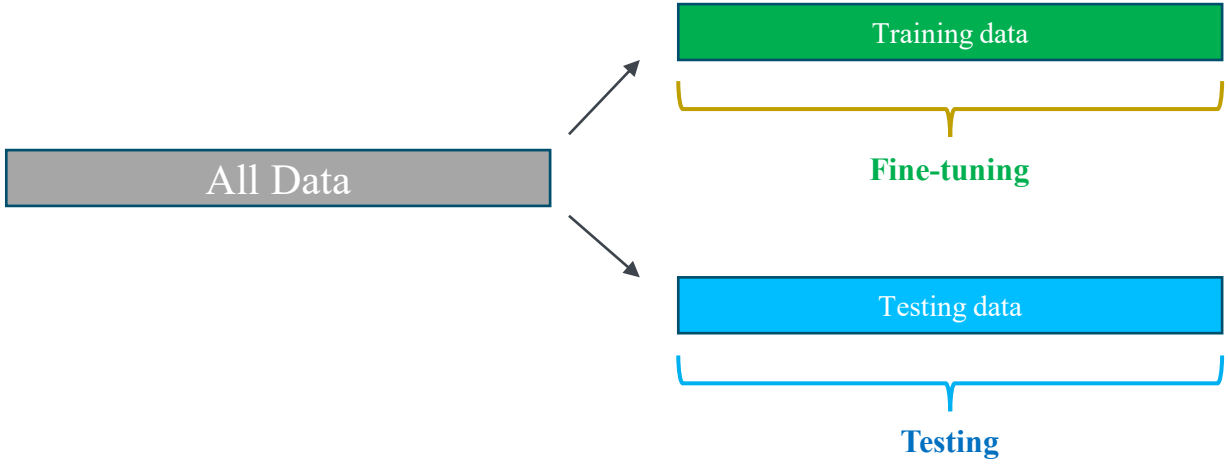
1 Wholly fine-tuning

2 K-Fold fine-tuning

3 Partly fine-tuning

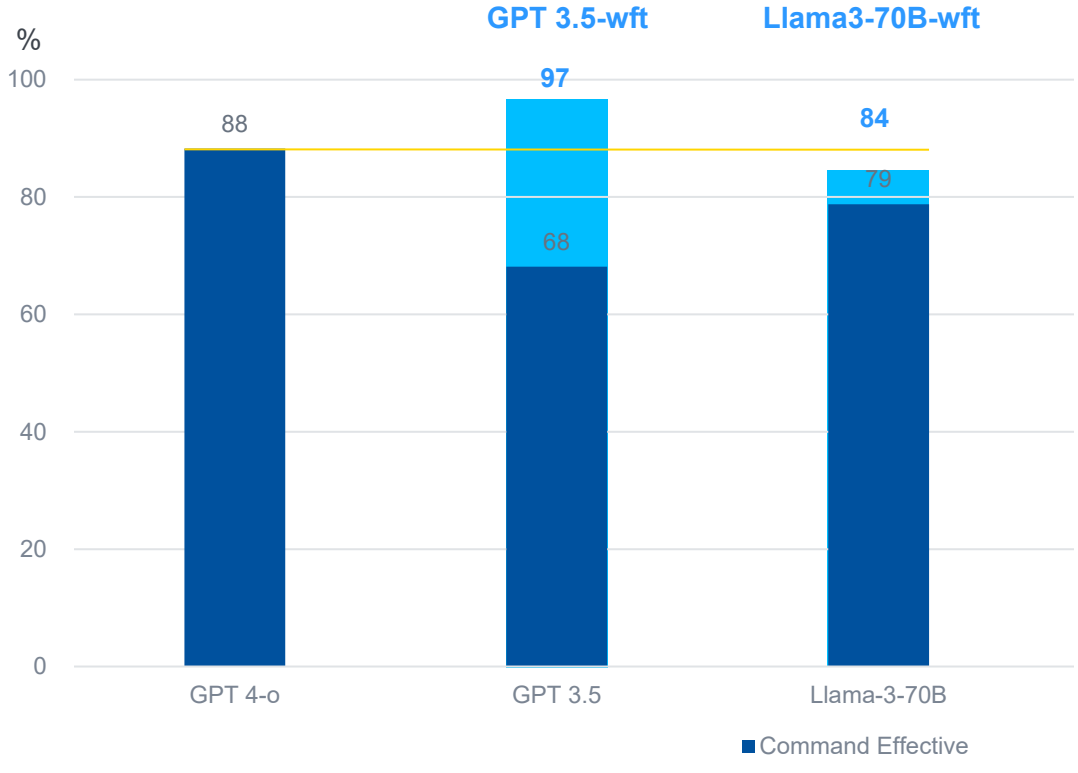
Fine-tuning and Testing

Wholly fine-tuned



Performance comparison

Wholly fine-tuned



Both GPT 3.5 and Llama3-70B have gained much improvement after the fine-tuning.

Effectiveness validated

However: overfitting?

Three different strategies

1 Wholly fine-tuning

2 K-Fold fine-tuning

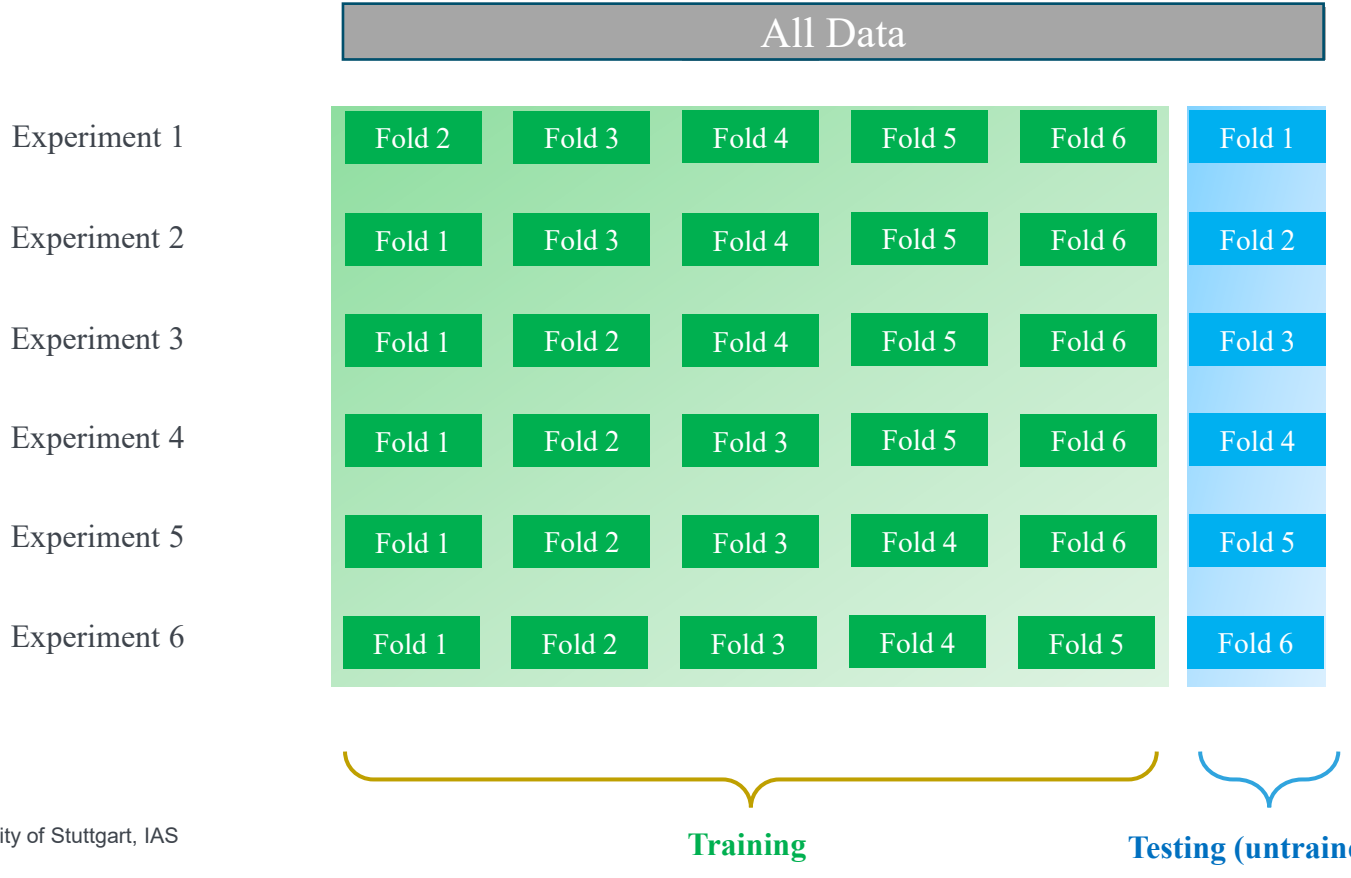
3 Partly fine-tuning

Fine-tuning and Testing

K-Fold fine-tuned

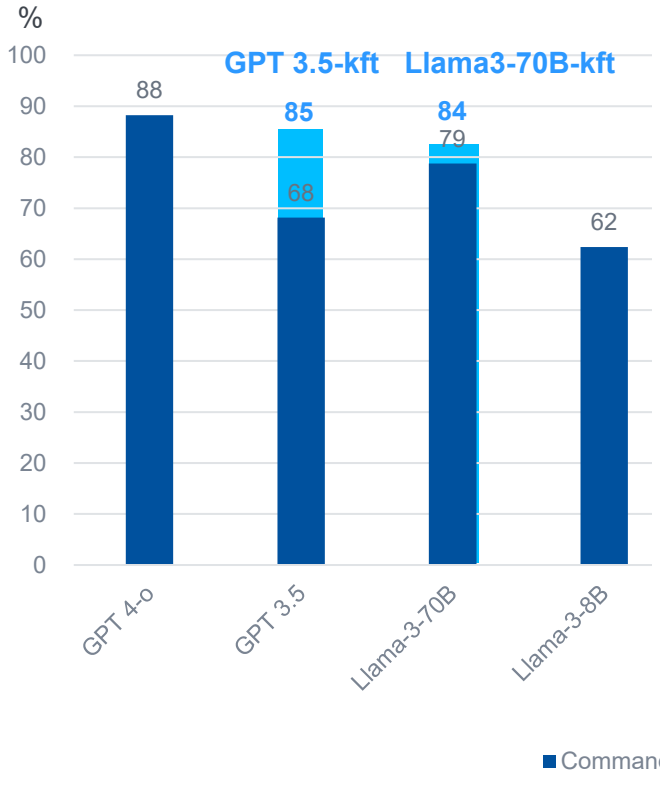
Advantages:

- Maximal utilization of limited dataset
- Test data was not used for fine-tuning



Performance comparison

K-fold fine-tuned

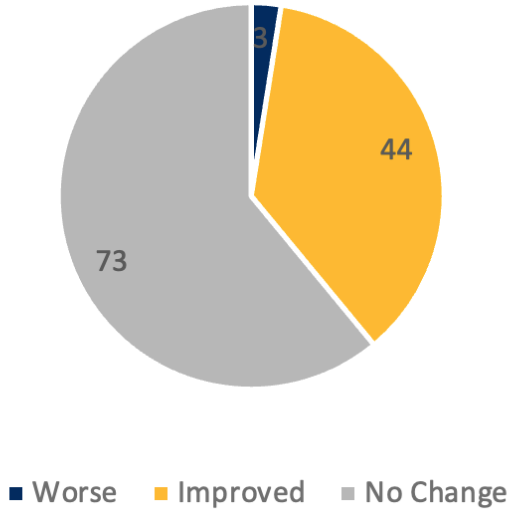


Improved? Worse?

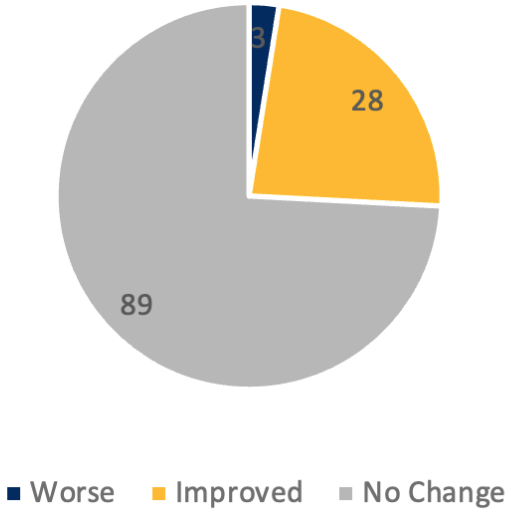
Performance comparison

K-Fold fine-tuned

GPT 3.5 (OpenAI proprietary method)



Llama3-70B (full fine-tuning)



Both GPT 3.5 and Llama3-70B have gained some improvement after the fine-tuning.

The test points that get worse are neglectable

no overfitting

Three different strategies

1 Wholly fine-tuning

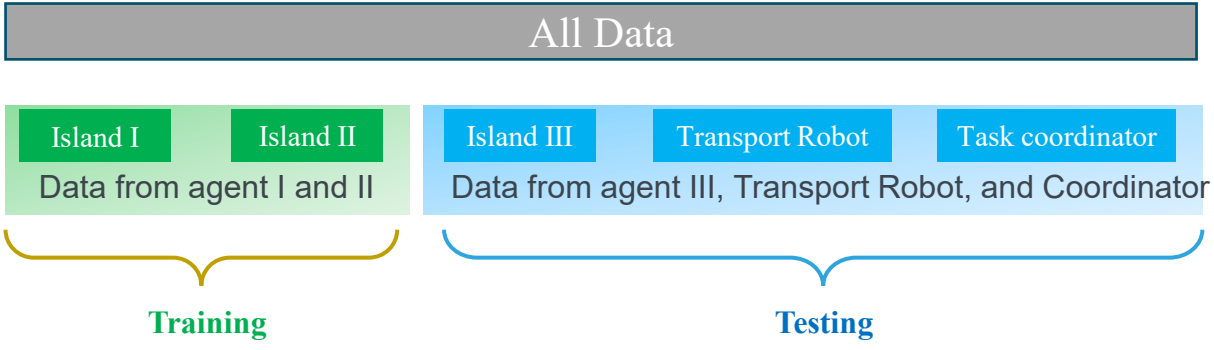
2 K-Fold fine-tuning

3 Partly fine-tuning

Transfer Learning ability across different tasks

Partly fine-tuned

- data from tasks of the agent Island I and II

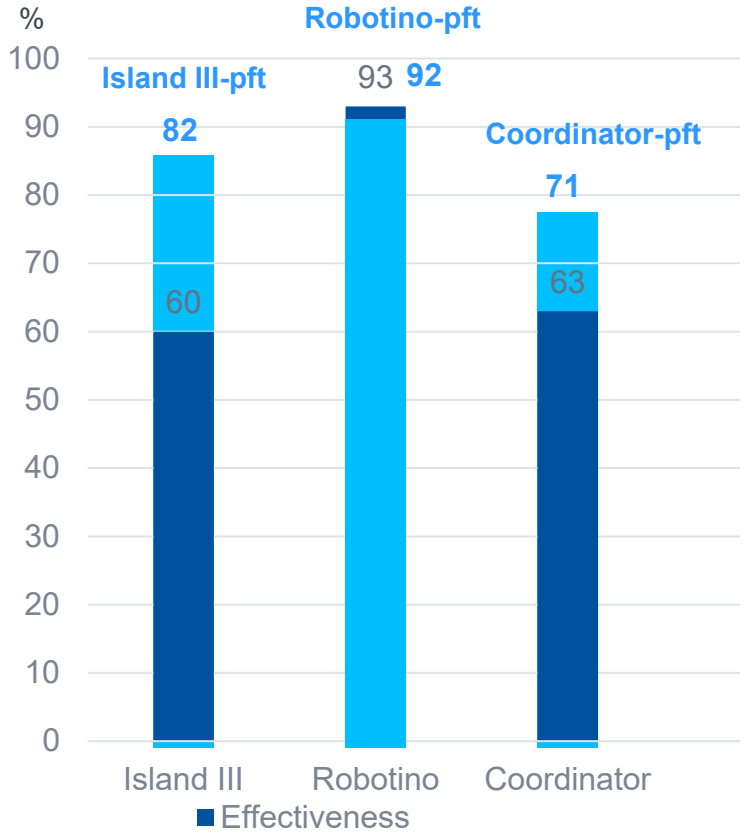


hierarchical task content-similarity with Island I & II:

- Island III: **highly similar** procedures
- Transport Robot: **different procedures**, yet **same format of output**
- Task Coordinator: **different procedures** and **different format of output**

Performance comparison

Partly fine-tuned



Island III

gained **much** improvement after fine-tuning because of **highly similar procedures**

Transport Robot

gained **no** improvement after fine-tuning because **different procedures**

no catastrophic forgetting

Task Coordinator

gained **some** improvement after fine-tuning, possibly due to the **learned knowledge about the automation procedures**, which is **helpful** for plan coordination

Transfer learning ability depends heavily on task content similarity!

Conclusion and Future Work

Conclusion

Fine-tuning effectiveness ?	**Effective** to enhance the performance
Problems like **catastrophic forgetting** and **overfitting?**	**Neglectable**
Transfer learning ability ?	**validated** but highly depends on the **content-similarity** of tasks

Future Work

- extend this study's fine-tuning methodologies to other operational contexts
- improve the simulation environment
- explore other methods of function calling

University of Stuttgart
Institut of Industrial Automation
and Software Engineering

Thank you!

Jize Zhang

e-mail st171260@stud.uni-stuttgart.de

University of Stuttgart

Attachment

Example in K-Fold fine-tuning

[11:48:45] TF81 reads information from the workpiece. Information for machining process is retrieved: [{"material": "titanium" },
[11:48:48] BG51 detects a workpiece at stopper S2 on conveyor C1, indicating that the workpiece arrives at the CNC station.
[11:48:50] Island II calls function: load_workpiece().
[11:48:52] The workpiece is loaded into CNC machine.
[11:48:56] Island II calls function: choose_tool('drill').
[11:48:58] PG51 detects the tool at initial position.
[11:49:00] Island II calls function: start_spindle(1200), apply_coolant('pulsed').
[11:49:02] The spindle starts to rotate.
[11:49:02] The coolant is applied.
[11:49:04] PG52 detects the correct placement of the workpiece.
[11:49:05] Island II calls function: start_cutting(1.0, 12).
[11:50:05] PG54 reads temperature 50°C.

>45 → continuous

Temperature: 50

Ti	Al
continuous	pulsed

[11:48:45] TF81 reads information from the workpiece. Information for machining process is retrieved: [{"material": "aluminium" },
[11:48:48] BG51 detects a workpiece at stopper S2 on conveyor C1, indicating that the workpiece arrives at the CNC station.
[11:48:50] Island II calls function: load_workpiece().
[11:48:52] The workpiece is loaded into CNC machine.
[11:48:56] Island II calls function: choose_tool('drill').
[11:48:58] PG51 detects the tool at initial position.
[11:49:00] Island II calls function: start_spindle(1200), apply_coolant('pulsed').
[11:49:02] The spindle starts to rotate.
[11:49:02] The coolant is applied.
[11:49:04] PG52 detects the correct placement of the workpiece.
[11:49:05] Island II calls function: start_cutting(1.0, 12).
[11:50:05] PG54 reads temperature 50°C.

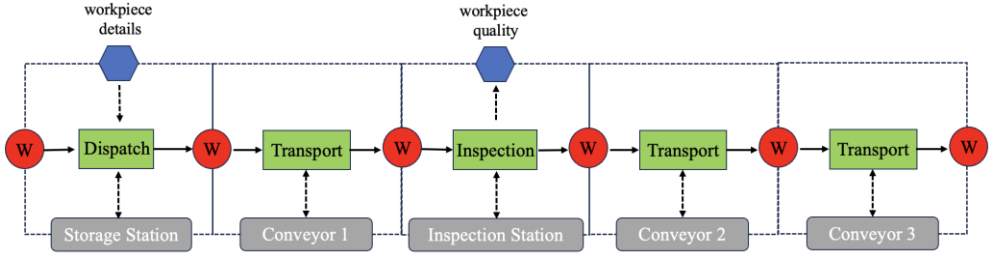
>55 → continuous

- worse --- > fine-granular domain-specific
- fail to deal with these details
- **chain-of-thought**

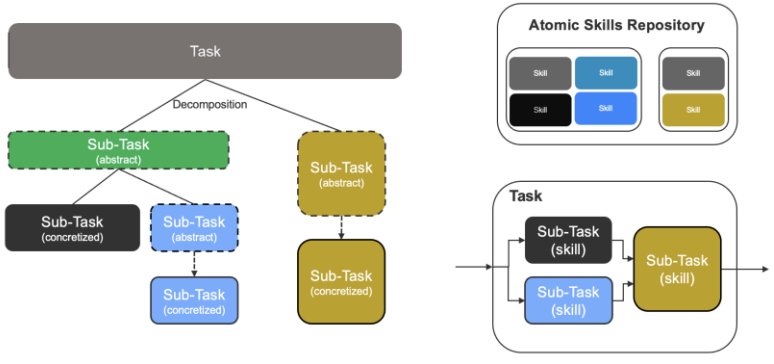
Attachment

Modelling from physical to text

- 1. Product-Process-Resource-Level Procedures



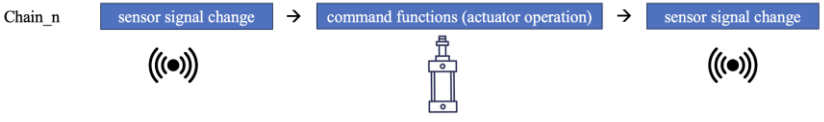
- 2. Task decomposition

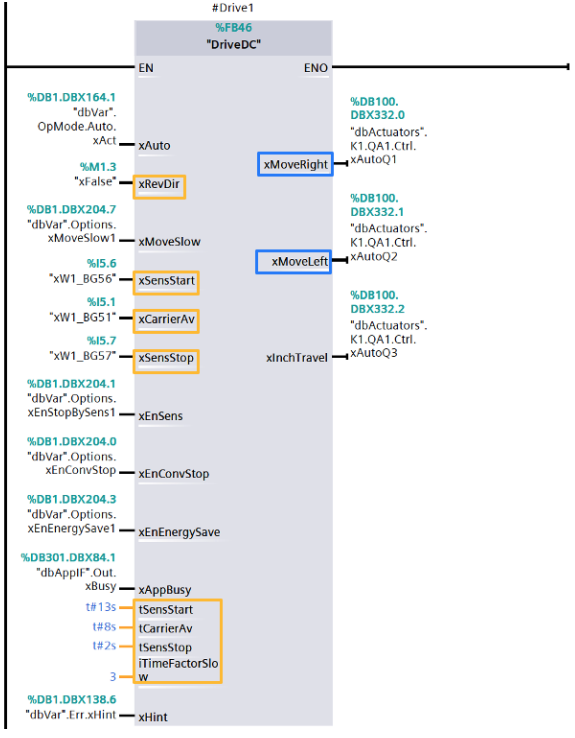


Attachment

Modelling from physical to text

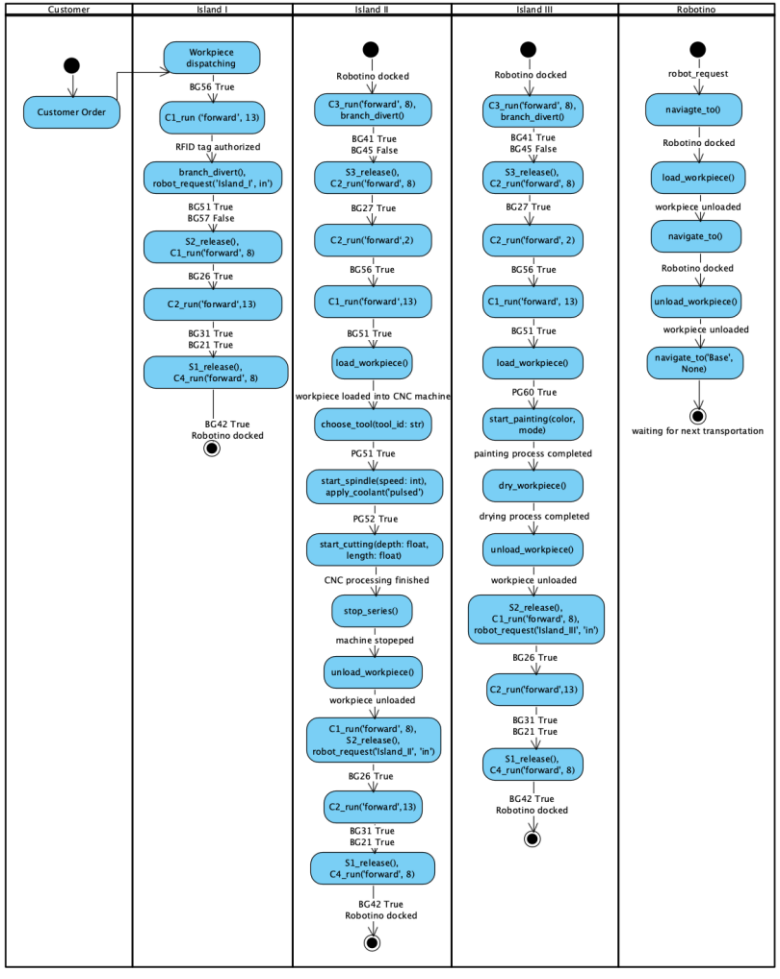
- 3. signal-actuator-level activity





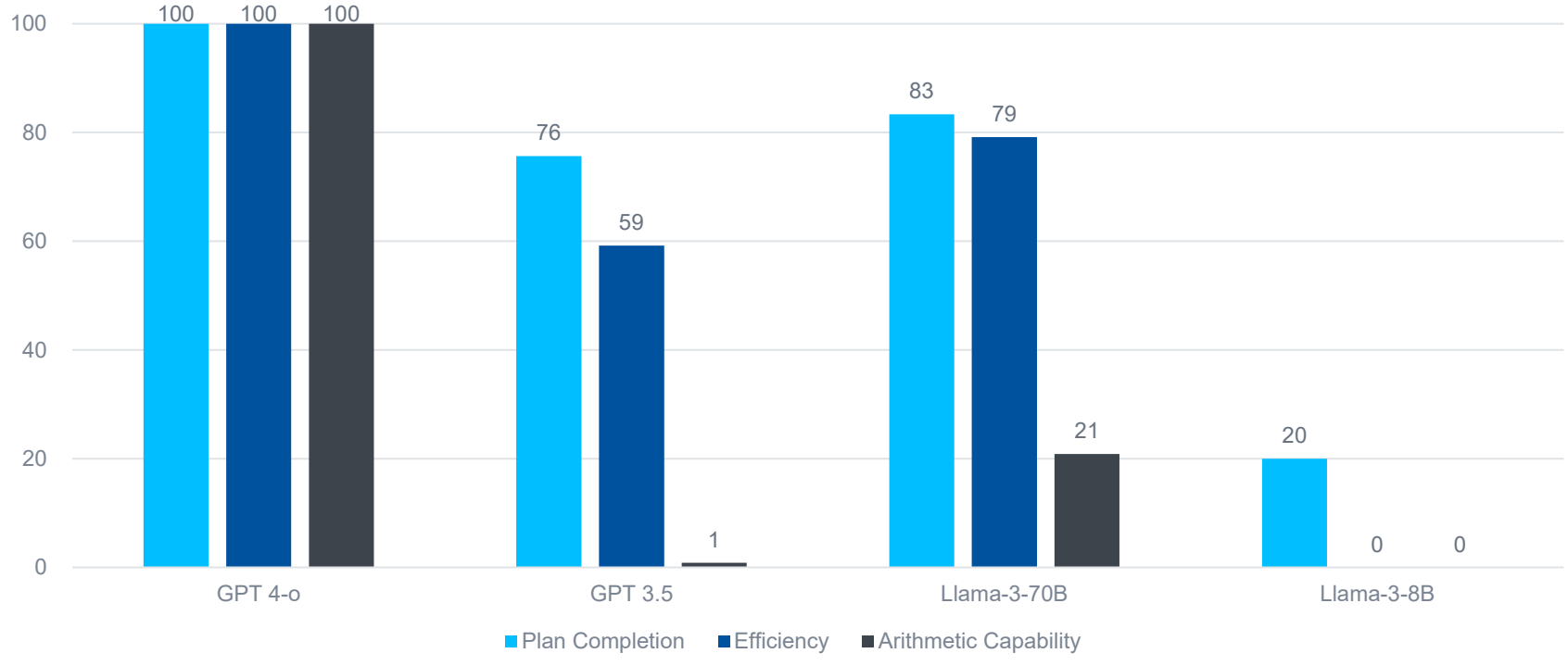
Attachment

Modelling from physical to text



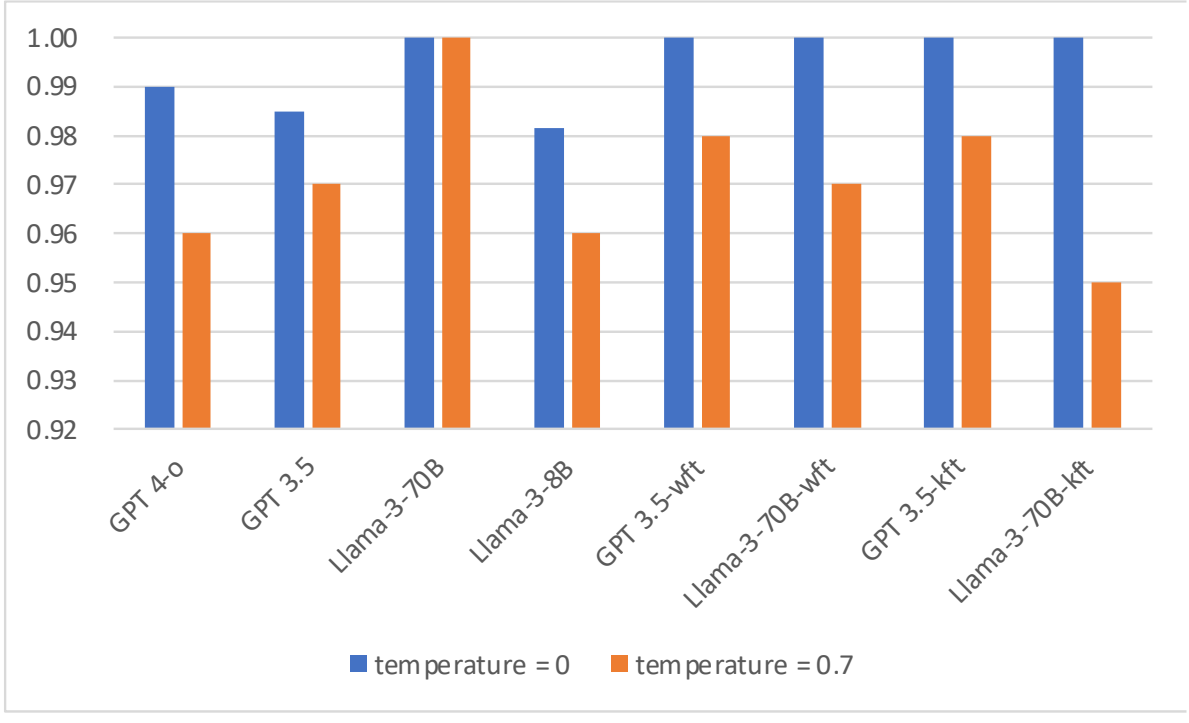
Attachment

Manager Performance Baseline



Attachment - Temperature setting

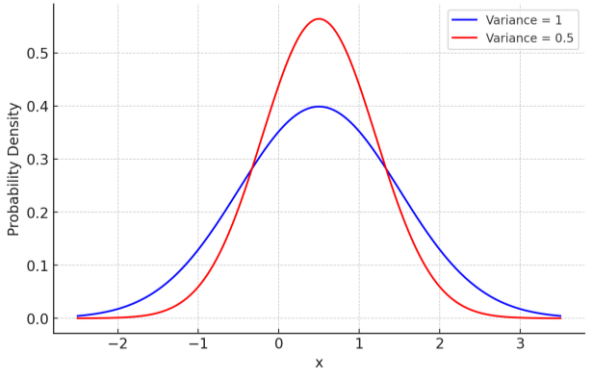
Determinism

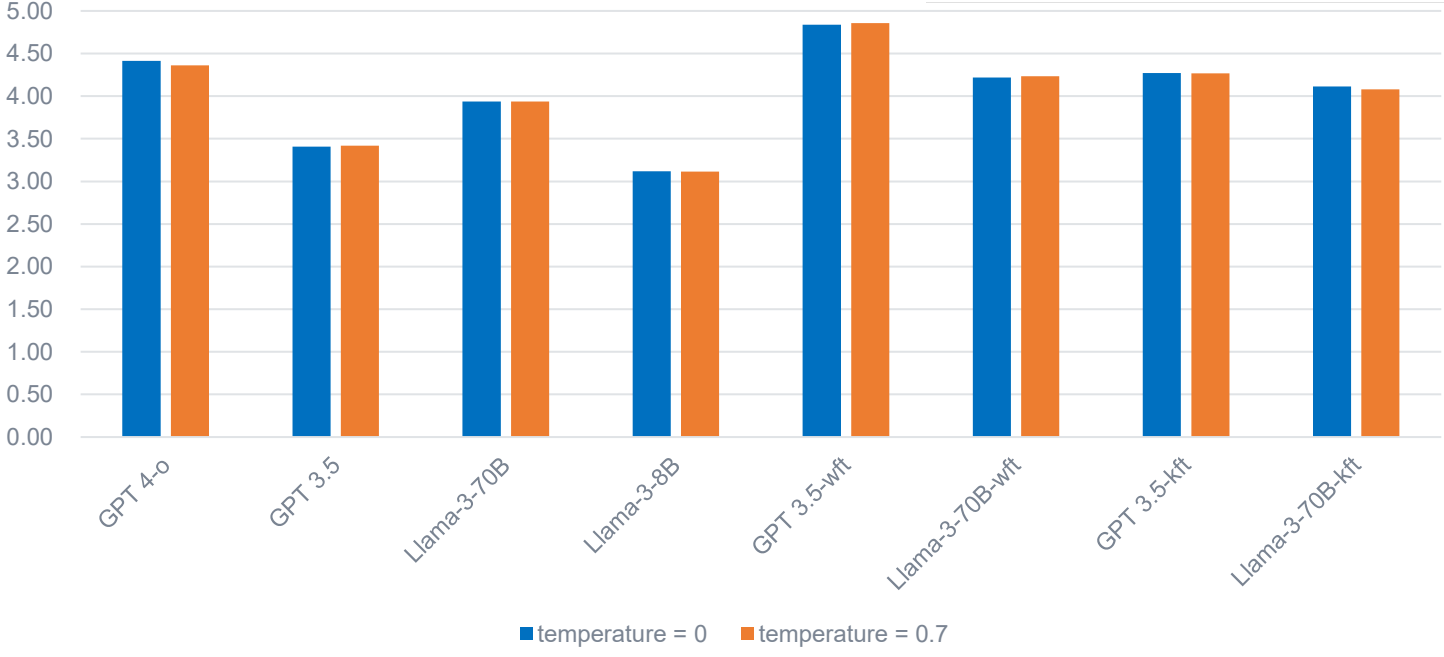


For each test cases, 5 tests were executed

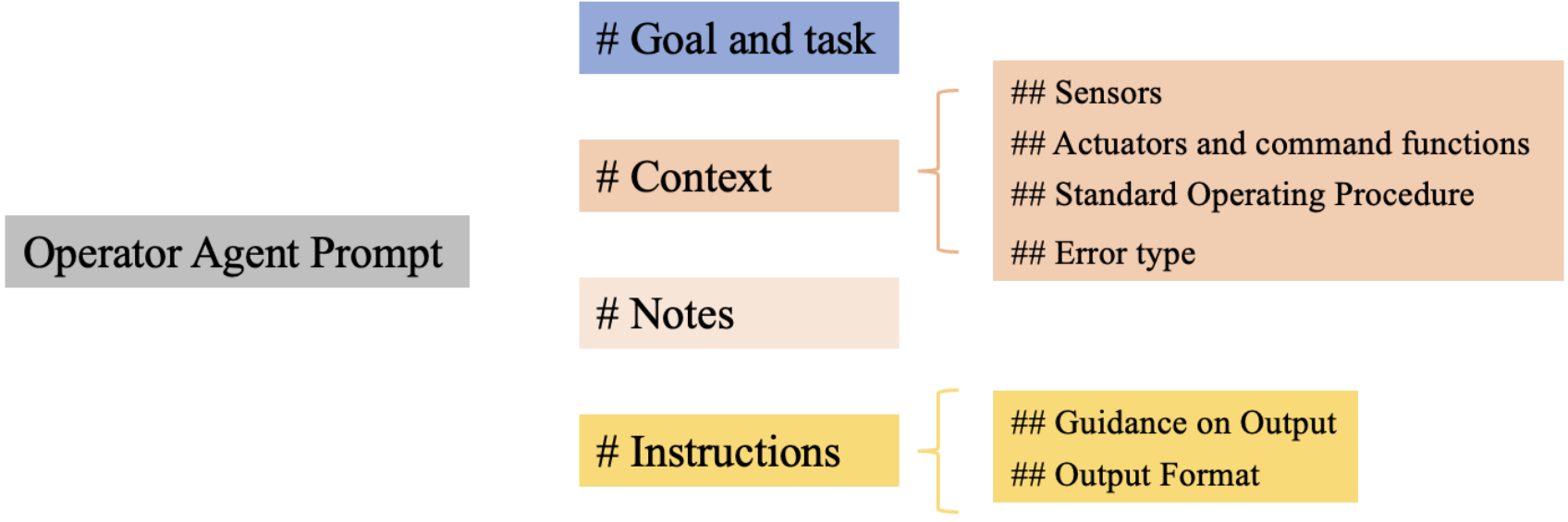
Attachment - Temperature setting

Command Effectiveness Comparison





Prompt design



Dataset

System Prompt	User Prompt (event log input)	Assistant (label)
- Goal and task - Context - Notes - Instructions	[00:05:51] CNC processing is finished. [00:05:51] 'Island II' calls function unload_workpiece(). [00:05:52] Failed to unload the workpiece.	**Command**: alert_to_supervisor ('Failed to unload the workpiece') **Reason**: …

Prompt design

```
Prompt for Island I:

# Goal and task:
You are 'Island I', an agent in an automation system responsible for the quality inspection of workpieces and managing material transport on conveyors within 'Island I'
```

```
# Context
## Sensors:
- BG56, BG51, BG57, BG26, BG21, BG27, BG45, BG41 and BG42 are proximity sensors for localizing workpieces.
- BG20, BG40 and BG50 are sensors for detecting the status of the stopper, which can be either "released" or "raised".
- BG30 and BG31 are sensors for detecting the status of a branch.
- TF81 reads RFID tags from workpieces as they pass by.
- KG35 and GF38 are a light receiver and sender, used to check if Robotino has docked with the island.
## Actuators and functions you can call:
### C1:
- function C1_run('forward',13): move conveyor C1 forward for 13 seconds, after which it will automatically stop.
- function C1_run('forward',8): move conveyor C1 forward for 8 seconds, after which it will automatically stop.
- function C1_run('forward',2): move conveyor C1 forward for 2 seconds, after which it will automatically stop.
- function C1_run('backward',2): move conveyor C1 backward for 2 seconds, after which it will automatically stop.
### C2:
- function C2_run('forward',13): move conveyor C2 forward for 13 seconds, after which it will automatically stop.
- function C2_run('forward',8): move conveyor C2 forward for 8 seconds, after which it will automatically stop.
- function C2_run('forward',2): move conveyor C2 forward for 2 seconds, after which it will automatically stop.
- function C2_run('backward',2): move conveyor C2 backward for 2 seconds, after which it will automatically stop.
### C3:
- function C3_run('forward',8): move conveyor C3 forward for 8 seconds, after which it will automatically stop.
- function C3_run('forward',2): move conveyor C3 forward for 2 seconds, after which it will automatically stop.
- function C3_run('backward',2): move conveyor C3 backward for 2 seconds, after which it will automatically stop.
### C4:
- function C4_run('forward',8): move conveyor C4 forward for 8 seconds, after which it will automatically stop.
- function C4_run('forward',2): move conveyor C4 forward for 2 seconds, after which it will automatically stop.
- function C4_run('backward',2): move conveyor C4 backward for 2 seconds, after which it will automatically stop.
### S1:
- function S1_release(): release stopper S1 to allow the workpiece to pass.
### S2:
- function S2_release(): release stopper S2 to allow the workpiece to pass.
### S3:
- function S3_release(): release stopper S3 to allow the workpiece to pass.
### Branch:
- function branch_divert(): divert the workpieces to conveyor C4 , which subsequently leads them to the Dock.
- function branch_straight(): keep the workpieces moving straight along conveyor C2, which subsequently leads them to the Storage Station.
## Additional functions you can call:
- function no_action(): no functions need to be called
- function robot_request('Island I','in'): sends request to Robotino to come to the dock and transport the workpiece to next island.
- function manager_request(): sends request to Manager Agent for a updated plan. For example, when a Robotino fails to dock with an island, Manager Agent will then assign another Robotino to dock with the island.
- function emergency_stop(): immediately stops all machine activity.
- function alert_to_supervisor("reason"): sends an alert message to a human supervisor detailing the reason for the alert, enabling quick resolution of issues that require human judgment.
## Error type
- Sensor Malfunctions: when sensors fail to work, it will return a failure notification.
- Actuator failure: when actuators fail to operate, it will return a failure notification.
- Inactivity timeout: no events happen for a duration, usually because of sensor or actuator failure. Call function alert_to_supervisor('timeout').
- Communication error: loses connection with other agents.
- Docking error: Robotino cannot dock with the island. Call function manager_request() for an updated plan and function alert_to_supervisor('Docking error').

Under normal conditions, you follow the standard operation procedure (SOP) to plan and execute actions. If deviations are required due to specific circumstances, you have the flexibility to employ alternate functions.
## Standard operation procedure:
1.When BG56 detects a workpiece at the infeed of conveyor C1, call function C1_run('forward', 13).
2.TF81 can be seen as Inspection Station. It will read the information from the workpiece when it passes. If the RFID tag on a workpiece is authorized and confirms the workpiece meets quality standards, it is going to be dispatched to Dock, call functions [branch_divert(), robot_request('Island I', 'in')]; If the RFID tag on a workpiece is not authorized and confirms the workpieces does not meet quality standards, it is not going to be dispatched to Dock, call function branch_straight() to send it back to Storage Station.
3.When BG51 detects a workpiece at stopper S2, if BG57 doesn't detect the other workpiece, indicating that the sub-section is not jammed, call function [S2_release(), C1_run('forward', 8)]. If BG57 detects the other workpiece, indicating that the sub-section is jammed, the workpiece will wait at stopper S2. After the other workpiece passes BG57, call functions [S2_release(), C1_run('forward', 8)].
4.When BG26 detects the workpiece at the infeed of conveyor C2, call function C2_run('forward',13).
5.When Branch is already set to divert the workpieces and BG21 detects workpieces at stopper S1, call functions [S1_release(), C4_run('forward', 8)]; When Branch is already set to direct the workpieces straight and BG21 detects workpieces at stopper S1,call function [S1_release(), C2_run('forward', 8)].
6.When BG42 detects the workpiece on conveyor C4, if Robotino has not docked with the island, call function C4_stop(); if Robotino has docked with the island, no functions need to be called.
7.When BG27 detects the workpiece at the outlet of conveyor C2, call function C2_run('forward', 2).
```

Prompt design

```
# Notes:
1. Each change of sensor signal will be recorded in the event log.
2. Respond using the functions specified above.
3. When mutiple functions need to be called, they are enclosed in [] in SOP. Do not leave out functions.
4. Output your response in JSON format, providing a simple, short reason for your action.
```

```
# Instructions:
You will observe an event log in the input section and you shall generate your response in the output section.
You should follow the following input and output pattern to generate a response in JSON format and give the reason to your action, keep the reason simple and short.

Input:
// An event log will be given here.
Output:
{"command":"function_()", "reason":"reason_for_action"}
Now, you should generate a response based on the event log:
```

Simulation

```
[Island I] [22:04:24] BG56 detects a workpiece at the infeed of conveyor C1.
[Island I] [22:04:25] Island I calls function: C1_run('forward', 13)
[Island I] [22:04:25] A workpiece passes BG56.
[Island I] [22:04:28] TF81 reads information from the workpiece. The RFID tag on the
workpiece is authorized.
[Robotino] [22:04:29] PG73 receieves request from Island I to transport workpieces.
[Island I] [22:04:29] Island I calls function: branch_divert(), robot_request('Island
I', 'in')
[Robotino] [22:04:30] Robotino calls function: navigate_to('Island I', 'in')
[Island I] [22:04:30] Branch is already set to divert the workpieces.
[Island I] [22:04:30] BG51 detects a workpiece at stopper S2 on conveyor C1.
[Island I] [22:04:31] Island I calls function: S2_release(), C1_run('forward', 8)
[Island I] [22:04:31] stopper S2 is realeased.
[Island I] [22:04:32] A workpiece passes BG51.
[Island I] [22:04:33] stopper S2 is raised.
[Island I] [22:04:33] BG26 detects a workpiece at the infeed of the conveyor C2.
[Island I] [22:04:35] Island I calls function: C2_run('forward', 13)
[Island I] [22:04:35] A workpiece passes BG26.
[Island I] [22:04:35] BG21 detects a workpiece at stopper S1 on conveyor C2.
[Island I] [22:04:36] Island I calls function: S1_release(), C4_run('forward', 8)
[Robotino] [22:04:42] Robotino has docked with Island I.
[Robotino] [22:04:43] Robotino calls function: load_workpiece()
[Island I] [22:04:36] stopper S1 is realeased.
[Island I] [22:04:37] A workpiece passes BG21.
[Island I] [22:04:38] stopper S1 is raised.
[Island I] [22:04:41] BG42 detects a workpiece at the infeed of conveyor C4.
[Island I] [22:04:43] Robotino has docked with Island I.
[Island I] [22:04:45] Island I calls function: no_action()
[Robotino] [22:04:45] PG71 confirms the workpiece is securely loaded.
[Robotino] [22:04:46] Robotino calls function: read_info()
[Robotino] [22:04:46] The next destination is retrieved: Island II.
[Robotino] [22:04:46] Robotino calls function: navigate_to('Island II', 'out')
[Robotino] [22:04:58] Robotino has docked with Island II.

[Island II] [22:04:58] Robotino has docked with Island II.
[Island II] [22:04:59] Island II calls function: C3_run('forward', 8),
branch_divert()
[Robotino] [22:04:59] Robotino calls function: unload_workpiece()
[Robotino] [22:05:01] The workpiece is unloaded.
[Island II] [22:05:01] Robotino unloads the workpiece.
[Island II] [22:05:01] BG41 detects a workpiece at stopper S3.
[Island II] [22:05:02] Branch is already set to divert the workpieces.
[Robotino] [22:05:02] Robotino calls function: navigate_to('base', None)
[Island II] [22:05:03] Island II calls function: S3_release(), C2_run('forward', 8)
[Island II] [22:05:03] stopper S3 is realeased.
[Island II] [22:05:05] A workpiece passes BG41.
[Island II] [22:05:06] stopper S3 is raised.
```