



**University of Stuttgart**  
Institute of Industrial Automation  
and Software Engineering

## Smart Assistant

22°C

Heating strategy eco

Indoor temperature



## Home Automation

Based on Large Language Model

# Optimizing LLM Interaction with Domain-Specific Simulation via Model Fine-tuning

Examiner: Prof. Dr.-Ing. Michael Weyrich

Supervisor: Yuchen Xia, M.Sc.

Co-Supervisor: Prof. Dr.-Ing. Nasser Jazdi

Presenter: Zheng Sun



# Content

- Design
  - Introduction
  - Basics
  - Related Works
  - Conceptual Design
  - System Implementation & Testing
- Optimization: Model Fine-tuning
  - Training
  - Evaluation
- Summary & Conclusion

Design

# Introduction

# State of the Art – Existing Products

## Home Automation



Amazon Alexa



Google Home



Apple Siri



Xiaomi Home

## Smart control of home devices

### Limitation:

1. **Clear commands** required, i.e. can not handle indirect commands, not flexible
2. **Rule-based**, not based on reasoning
3. Pre-defined plan

**In the era of GenAI, it is not smart enough.**

# Introduction

## Home Automation



Are LLMs suitable for smart home automation system?

# Basics

## Why LLM can be designed as agent?



Guests arriving in 30 minutes, the living room is at only 15 degree celsius, how should I set my heating device?

Got it—let's warm up your living room quickly and comfortably before your guests arrive.

Here's what you can do:

- ✔ **Set the thermostat to a higher target temperature initially** (e.g., 22–24 °C) to boost heating output quickly. Most heating systems work at full power until they reach the set temperature.
- ✔ **Turn on any fan or circulation mode if your heater has it**—this distributes warm air faster.
- ✔ **Close doors and windows** to retain heat. Cover any drafts if possible.
- ✔ **After about 20–25 minutes**, check the room temperature. Once it approaches your comfort level (around 20–21 °C), you can lower the thermostat to maintain that temperature steadily.

Available information

Reasoning[1]

Logical conclusions

### System Mechanism



LLM-based agent could be used to power this application!

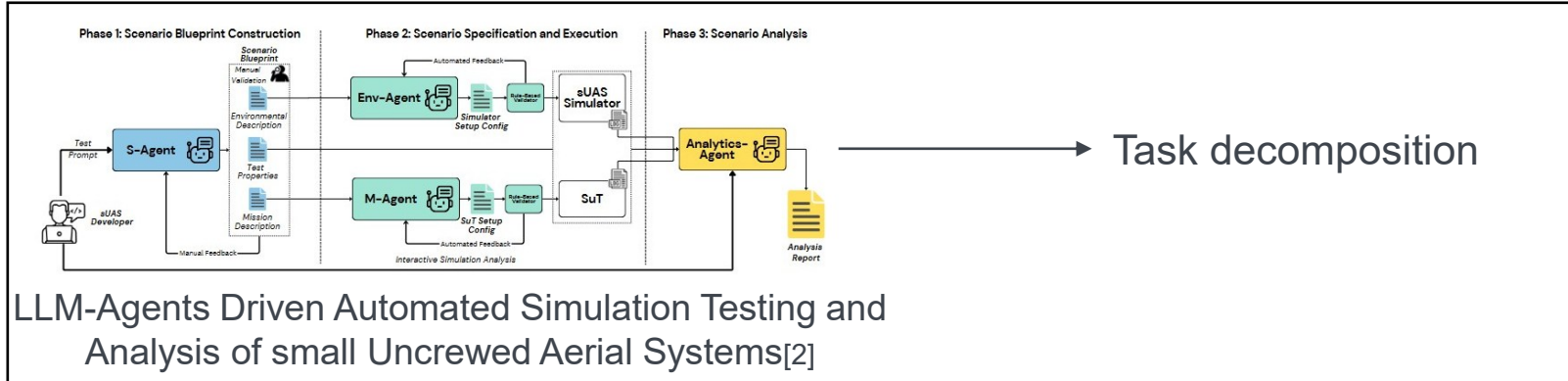
Design

# Related Works

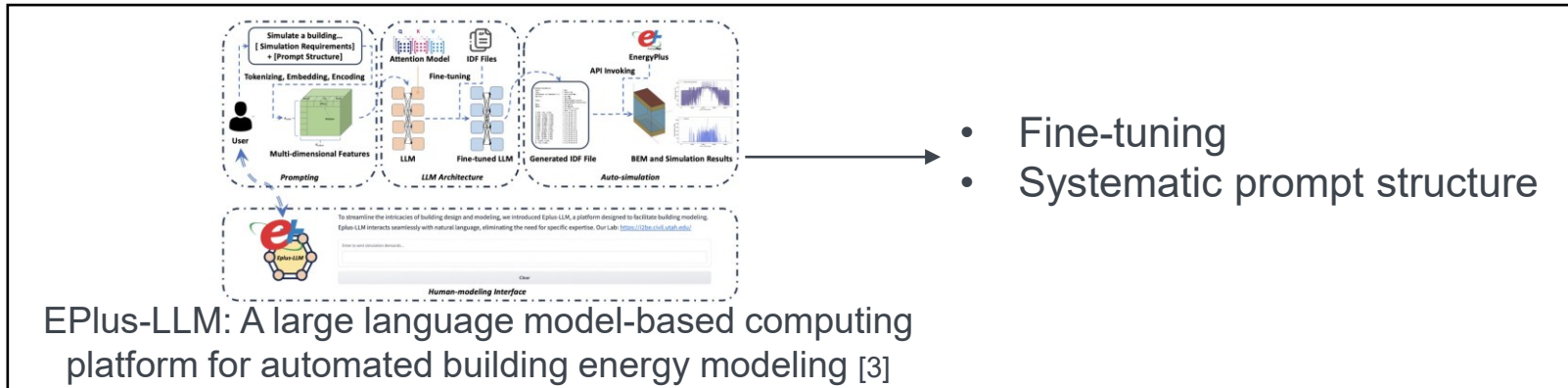
## Related Works

Keywords: LLM agent + automation

1



2

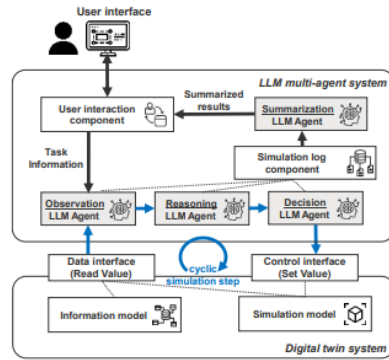




## Related Works

Keywords: LLM agent + automation

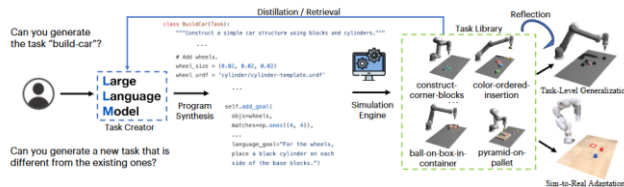
3



Automate simulation parametrization

LLM experiments with simulation: Large Language Model Multi-Agent System for Simulation Model Parametrization in Digital Twins [4]

4

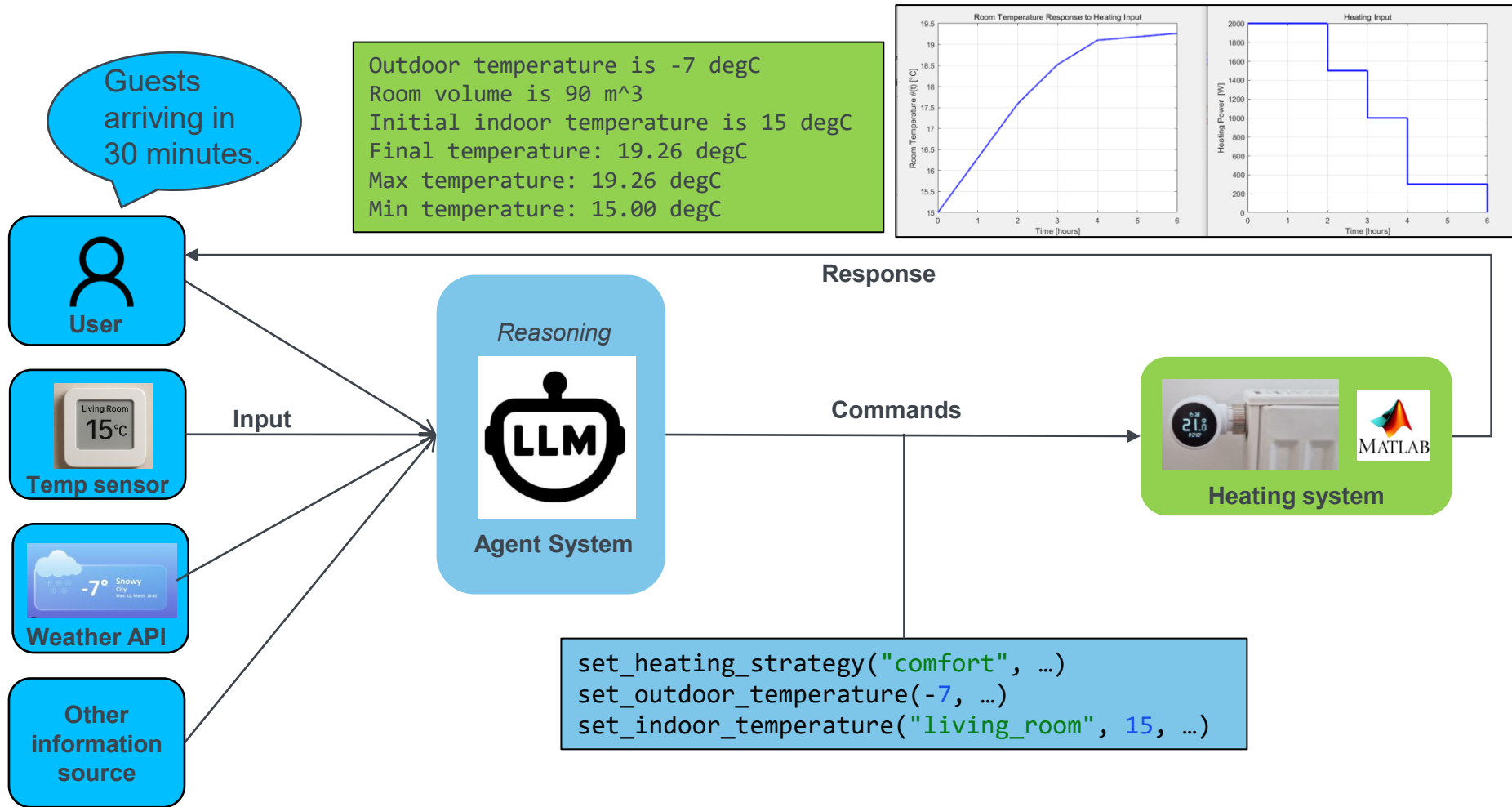


Synthetic training dataset

GenSim: Generating Robotic Simulation Tasks via Large Language Models[5] GenSim: Generating Robotic Simulation Tasks via Large Language Models [5]

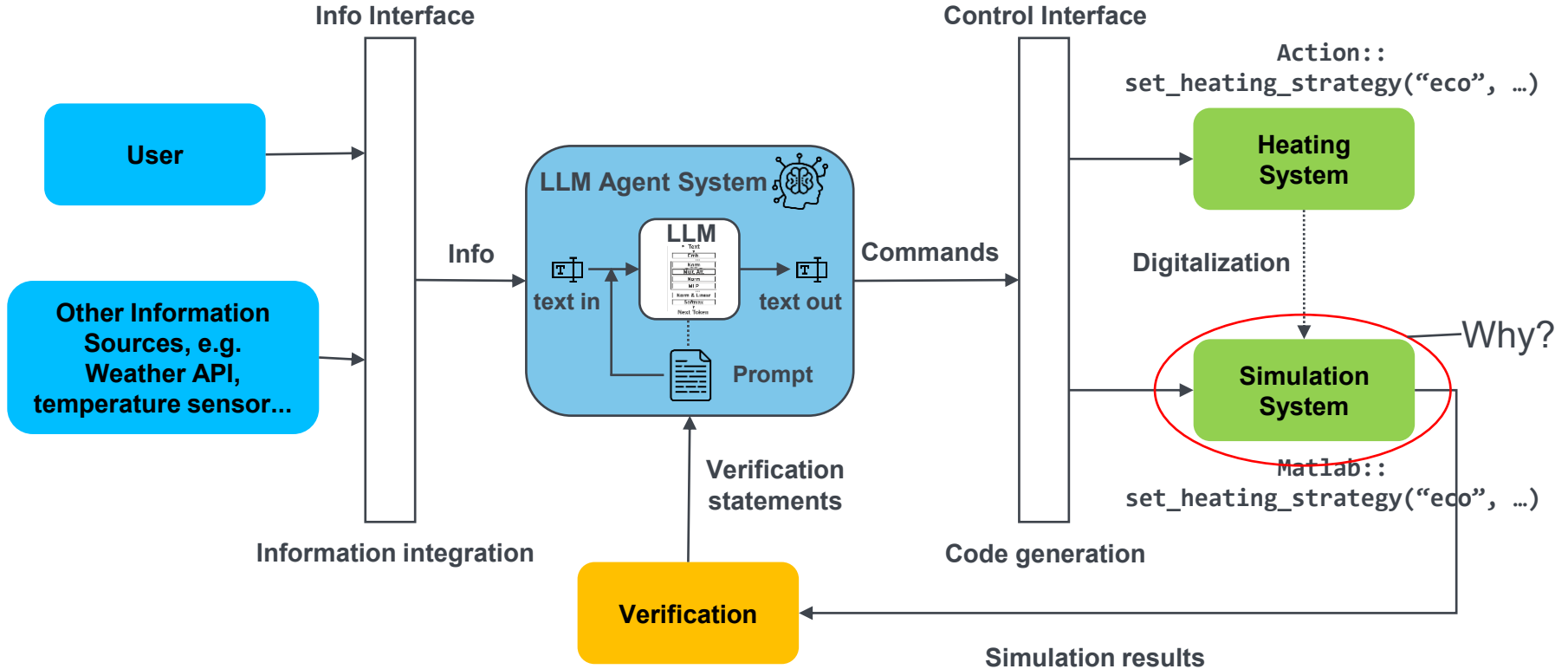
# Conceptual Design

# Use Case Visualization



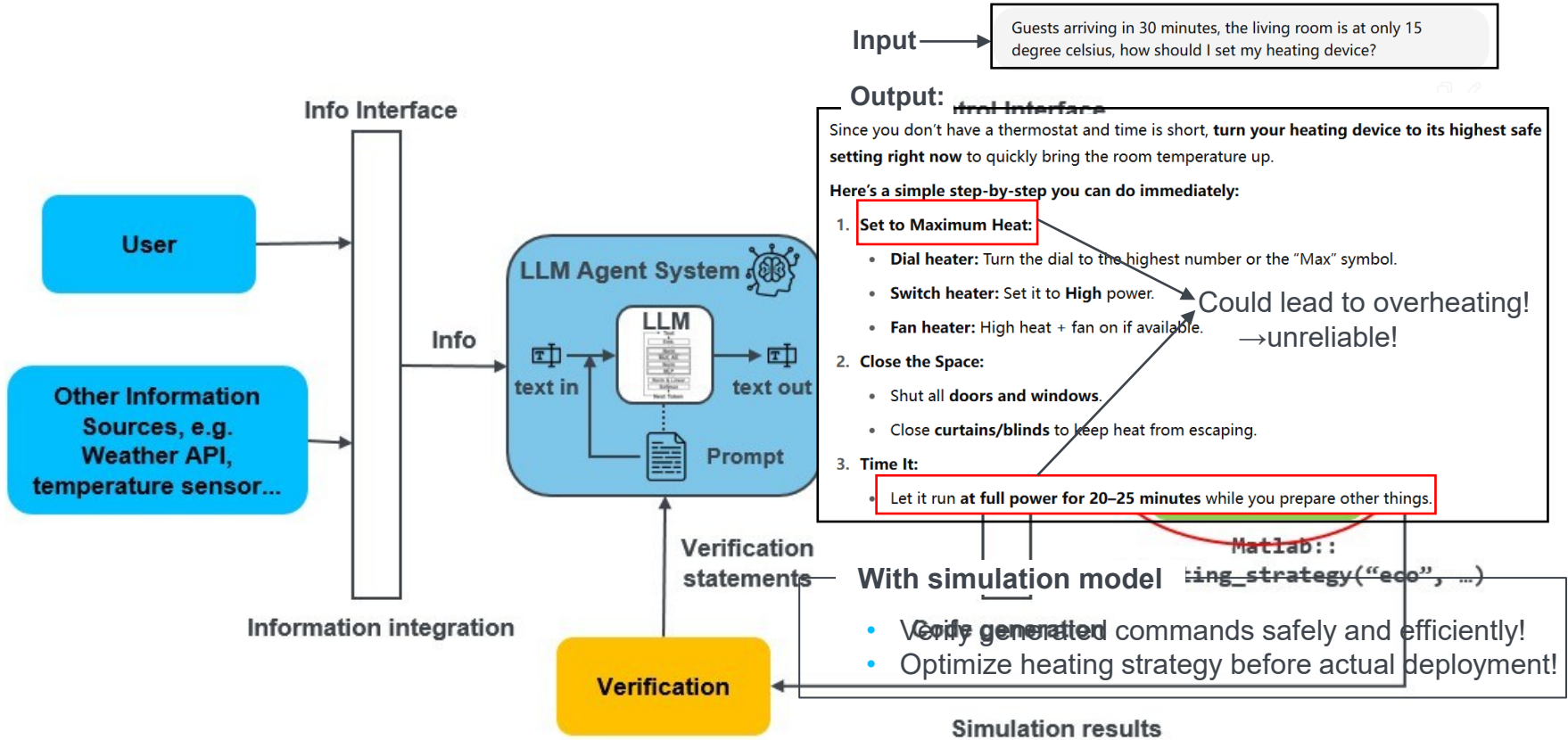
# System Design

## Smart Control in Home Automation Heating



# Conceptual Design

## Why is simulation model necessary?



# Simulation System Design



Figure 3: Simplified assumption in the resistance-capacity model for a simplified system which considers the room and the envelope

Let us suppose that this system is in thermal equilibrium with an initial uniform temperature  $t$ . Making the assumption that the system is subjected to an external constant temperature  $t_{\text{amb}}$  from the time instant  $s = 0$  (step wise in boundary conditions), the temperature change in time  $t$  can be calculated in this way:

$$\frac{\vartheta_{\text{amb}} - t}{R} = C \frac{d\vartheta}{dt} \quad (16)$$

If the thermal characteristics are constant in the considered temperature range, the equation (16) can be expressed as:

$$\frac{d\vartheta}{t_{\text{amb}} - t} = \frac{1}{RC} dt \quad (17)$$

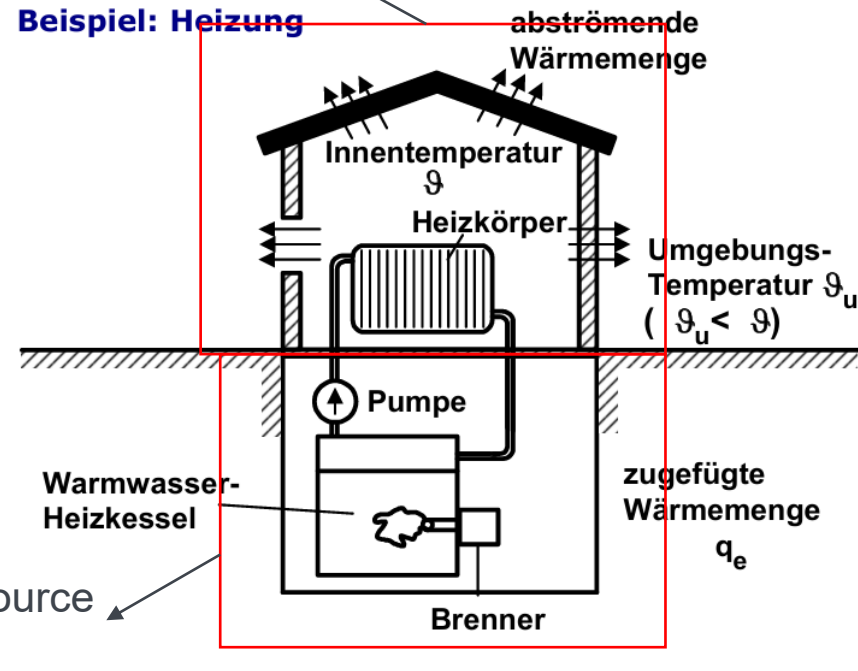
$$\frac{d\vartheta}{dt} = \frac{\vartheta_U - \vartheta}{RC} \quad (1)$$

## Was sind Fließprozesse?

**Fließprozesse:** Technische Prozesse mit kontinuierlichen Vorgängen

- Zeitkontinuierliche Prozessgrößen
- Zeitdiskrete Prozessgrößen mit kontinuierlichem Wertebereich

### Beispiel: Heizung



# Conceptual Design

## Simulation theory

Modeling equation:

$$\frac{d\vartheta}{dt} = \frac{-G}{C} (\vartheta - \vartheta_U) + \frac{1}{C} q_e(t) \quad (2)$$

Equation based on energy balance:

- Heat loss: proportional to  $\Delta$  temperature and coefficient  $G$
- Heat gain: from external heater  $q_e$

### Physical Properties

```
% Air properties
c_air = 1005;           % Specific heat capacity of air [J/(kg·K)]
rho_air = 1.2;          % Air density [kg/m^3]
room_volume = 30;       % Room volume [m^3]
mass_air = rho_air * room_volume; % Mass of air [kg]
C_air = c_air * mass_air; % Thermal capacity of air [J/K]

% Wall properties (estimated)
C_wall = 40000;         % Thermal capacity of walls/furniture [J/K]
C_wall = 5e6;

C_total = C_air + C_wall; % Total thermal capacity of room [J/K]

% Heat transfer through wall
wall_area = 10;          % Area of outer wall [m^2]
wall_thickness = 0.3;     % Wall thickness [m]
lambda_wall = 0.8;        % Thermal conductivity of wall [W/(m·K)]
lambda_wall = 0.21;

R_wall = wall_thickness / (lambda_wall * wall_area); % Thermal resistance [K/W]
```



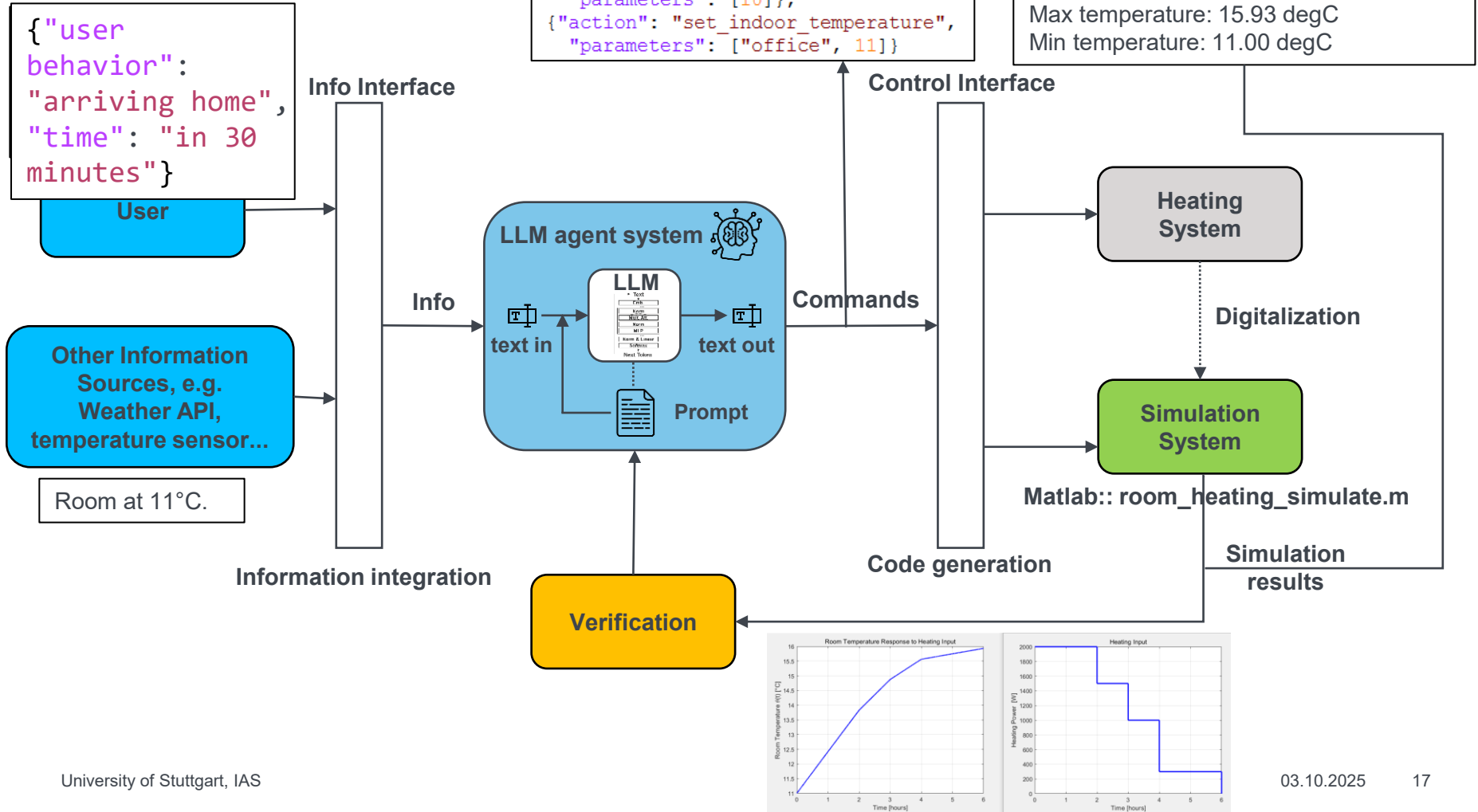
### %% --- Euler Integration using physical energy balance ---

```
for k = 1:steps-1
    heat_loss = G_wall * (theta(k) - theta_out); % heat loss [W]
    net_power = q_e(k) - heat_loss;              % net power [W]
    d_theta = (1/C_total) * net_power;           % delta theta [°C/s]
    theta(k+1) = theta(k) + d_theta * dt;
end
```

# **System Implementation & Testing**

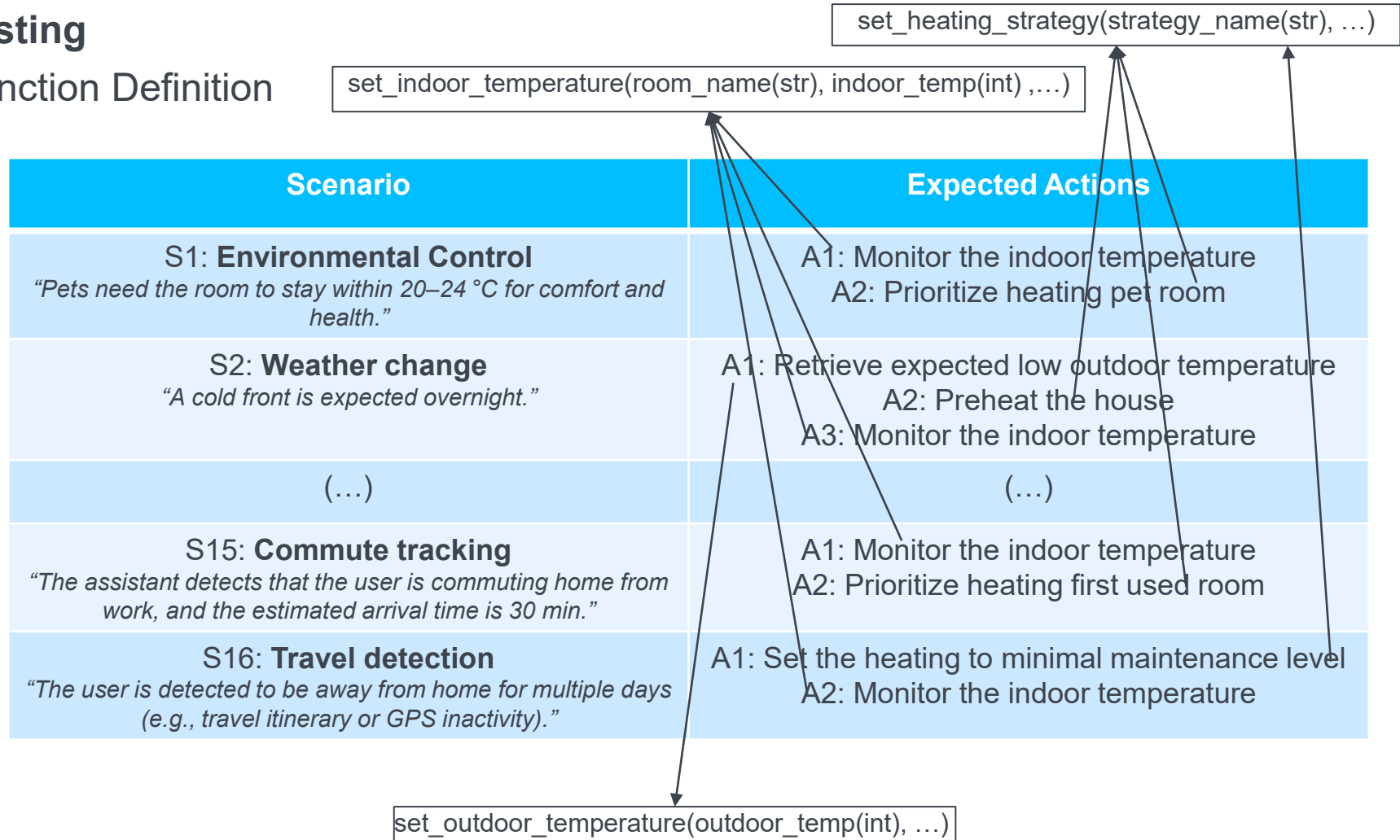


# Visualization of System In



# Testing

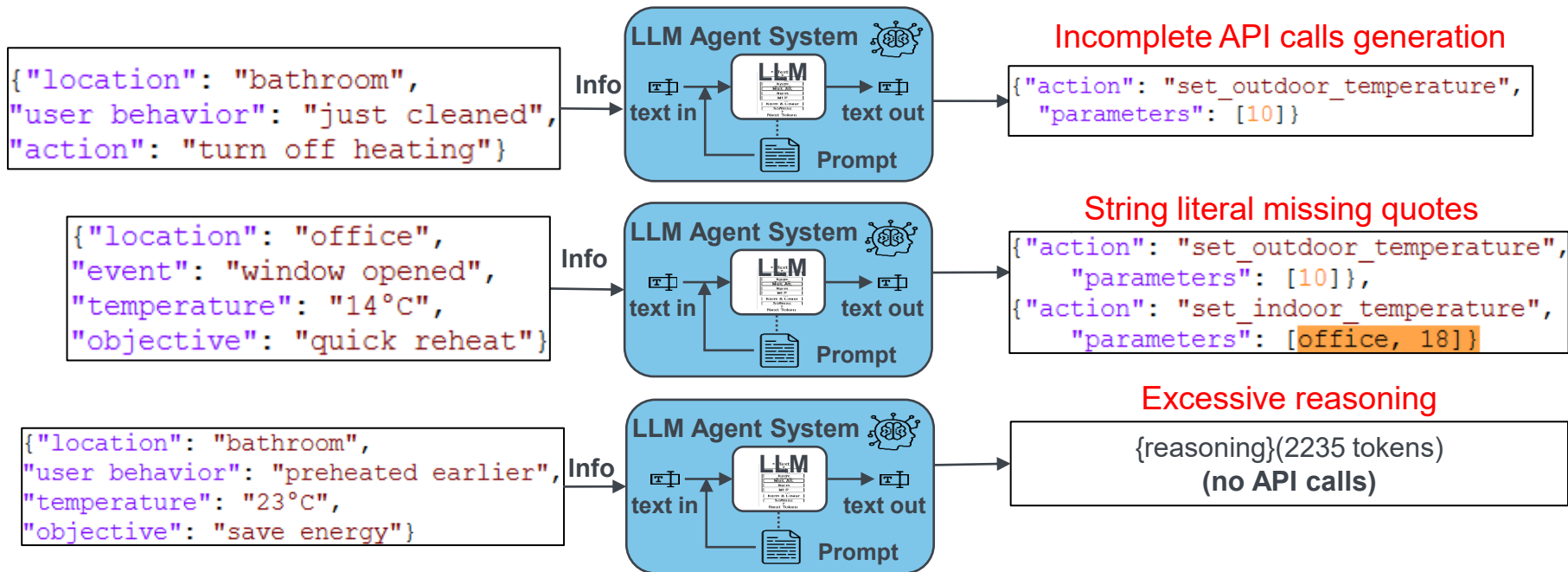
## Function Definition



# Evaluation

## API Calls for simulation models

## Typical output errors:



30 test data * 3 API	LLM Agent
Correct Rate	47 (52%)

Not good enough!  
→ Need further optimization!

# Model fine-tuning

# Model Fine-tuning

- What is Model fine-tuning?
    - Training on **domain-specific dataset**  
→ **better performance** on tasks while keeping core knowledge.
  - Why fine-tuning?
    - Rapid adaption with **limited data**
    - **Optimize model behavior**, such as output format / style through supervised fine-tuning or reinforcement fine tuning
- Considering the errors were mostly **syntax errors** and **not comply** with software API:

**Exactly what is required!**

**Training**

# Testing

## Dataset Generation

- synthetic dataset
  - generated by ChatGPT-4o

Training data: 120

Test data: 30

Task Case No.	System Tasks	Scenario Example	Expected Output
1~50	Plan Heating strategy	<i>Work session planned during snowfall. Ensure cozy room.</i> <pre>{ "user behavior": {   "calendar plan": "work session",   "weather": "snowfall",   "objective": "cozy room"}</pre>	<pre>{ "action": "set_heating_strategy",   "parameters": ["comfort"]} { "action": "set_outdoor_temperature",   "parameters": [-2]}, { "action": "set_indoor_temperature",   "parameters": ["office", 18]}</pre>
51~100	(Plan Heating strategy) + Set indoor temperature	<i>Bathroom was preheated earlier. It's 23°C now. Save energy.</i> <pre>{ "location": "bathroom",   "user behavior": "preheated earlier",   "temperature": "23°C",   "objective": "save energy"}</pre>	<pre>{ "action": "set_heating_strategy",   "parameters": ["off"]} { "action": "set_outdoor_temperature",   "parameters": [10]}, { "action": "set_indoor_temperature",   "parameters": ["bathroom", 23]}</pre>
101~150	(Plan Heating strategy) + Set in-/outdoor temperature	<i>Bathroom temp 20°C. Outdoor 8°C. Maintain light heating while airing.</i> <pre>{ "location": "bathroom",   "room_temperature": "20°C",   "outdoor_temperature": "8°C",   "action": "maintain light heating"}</pre>	<pre>{ "action": "set_heating_strategy",   "parameters": ["eco"]} { "action": "set_outdoor_temperature",   "parameters": [8]}, { "action": "set_indoor_temperature",   "parameters": ["bathroom", 20]}</pre>

# Training

Dataset Structure: **Instruction** + **Scenario** + **Action**

## Instruction:

### Role and purpose definition

You are an API call generator. Based on a given natural language scenario, generate valid API calls using the correct function name and parameters. You have access to the following simulation api call for heating:

**set\_heating\_strategy(strategy\_name(str), eng):** You can use the api call to set the heating strategy. You can do so by writing "set\_heating\_strategy(strategy\_name, eng)", where "strategy\_name" can be "eco", "comfort" and "off". "eco" means that it consumes less energy but can warm up slow. "comfort" means that it can warm up quickly while consume more energy. "off" means that turn off the heating.

### Function and parameter definition

**set\_outdoor\_temperature(temp(int), eng):** You can use the api call to set the outdoor temperature. You can do so by writing "set\_outdoor\_temperature(temp, eng)", where temp is the outdoor temperature in the unit of degree Celsius and should be integer.

**set\_indoor\_temperature(room\_name(str), temp(int), eng):** You can use the api call to set the indoor temperature. You can do so by writing "set\_indoor\_temperature(room\_name, temp, eng)", where "room\_name" can be "living\_room", "bathroom", "bedroom", "study". Temp is the initial indoor temperature in the unit of degree Celsius and should be integer.

### Default values definition and output format specification

If a required parameter is clearly stated or implied in the scenario, use it. If a required parameter is not provided, use "eco" as "strategy\_name", use 10 as outdoor temperature, use "bedroom" as "room\_name" and use 18 as indoor temperature. Always return lines of API call in code format. Do not explain your reasoning or provide any extra text.

Here is the scenario. Scenario: {"user behavior": "calendar plan "work session"", "weather": "snowfall", "objective": "cozy room"}




Action: ["set\_heating\_strategy(\"comfort\", ...)", "set\_outdoor\_temperature(-2, ...)", "set\_indoor\_temperature(\"study\", 18, ...)"]

Train the model to output these text



# Training

## Model Selection

	 Gemma 3 1B [7]	 Qwen3 4B [8]	 Qwen3 8B [8]
<b>Key Strength</b>	<ul style="list-style-type: none"> <li>• Lightweight, efficient</li> <li>• Improved instruction-following</li> </ul>	<ul style="list-style-type: none"> <li>• Strong in coding benchmarks</li> <li>• Dense architecture (simpler fine-tuning)</li> </ul>	<ul style="list-style-type: none"> <li>• Superior performance in function-calling tasks</li> <li>• Outperforms some larger models</li> </ul>
<b>Why selected</b>	<ul style="list-style-type: none"> <li>• Small size for <b>efficient fine-tuning</b></li> <li>• <b>Solid reasoning</b></li> <li>• Easier deployment</li> </ul>	<ul style="list-style-type: none"> <li>• <b>Excellent code generation</b></li> <li>• Great cost-performance tradeoff</li> </ul>	<ul style="list-style-type: none"> <li>• Top-tier performance</li> <li>• Efficient inference for size</li> </ul>

These combined characteristics allow:

- Task-specific fine-tuning
- API calling capacity
- Efficient deployment

Small size language model



Locally efficient deployment

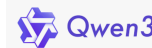
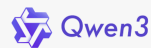
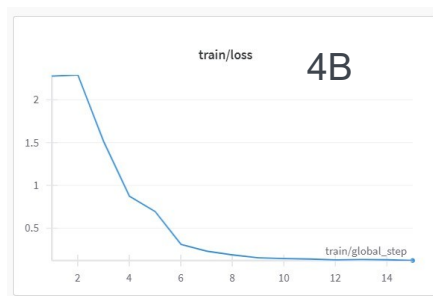
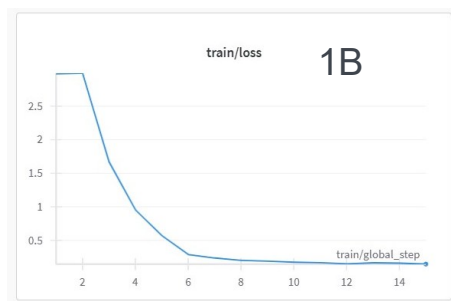


Address privacy concerns

# Training

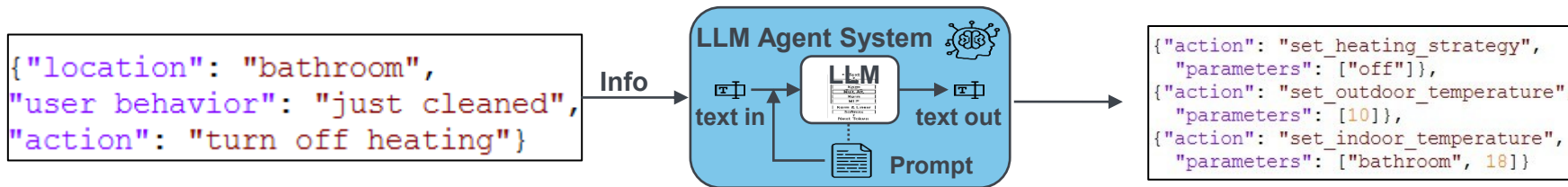
## Training method

Training Method	SFT	Training Type	Full parameter tuning	Learning rate scheduler	Cosine
Epoch	1	Batch size	8	Learning rate	0.000002
Max gradient norm	1	Weight decay	0.001	Scheduler cycles	0.5



# Evaluation

## API Calls for simulation models



30 test data*3 API	1B	4B	8B
Before fine-tuning	18%	52%	67%
After fine-tuning	74%(+319%)	92(+77%)	93(+40%)

Observe:

1. Fine-tuning with small amount of training data can **achieve great progress**, fine-tuned 1B model surpass the pre-trained 8B model.
2. More parameters means better performance, but the **margin is smaller** when it comes to fine-tuned model(4B vs. 8B).

# Evaluation

## API Calls for simulation models – Improvement after model fine-tuning

Error Type	Before fine-tuning Output	After fine-tuning Output
Incomplete API calls generation	<pre>{   "action": "set_outdoor_temperature",   "parameters": [10] }</pre>	<pre>{   "action": "set_heating_strategy",   "parameters": ["eco"] }, {   "action": "set_outdoor_temperature",   "parameters": [10] }, {   "action": "set_indoor_temperature",   "parameters": ["living_room", 15] }</pre>
String literal missing quotes	<pre>{   "action": "set_outdoor_temperature",   "parameters": [10] }, {   "action": "set_indoor_temperature",   "parameters": [office, 18] }</pre>	<pre>{   "action": "set_heating_strategy",   "parameters": ["comfort"] }, {   "action": "set_outdoor_temperature",   "parameters": [10] }, {   "action": "set_indoor_temperature",   "parameters": ["office", 14] }</pre>
Excessive reasoning	{reasoning}(2235 tokens) (no API calls)	<pre>{   "action": "set_heating_strategy",   "parameters": ["eco"] }, {   "action": "set_outdoor_temperature",   "parameters": [10] }, {   "action": "set_indoor_temperature",   "parameters": ["living room", 15] }</pre>

- **Observe:** Syntax error and model overthinking problem

# Evaluation

## API Calls for simulation models – Remained errors

Error Type	Scenario as Input	Expected Output	Model & Example Output
Language ambiguity	<pre>{   "user behavior": "work session",   "time": "rainy afternoon",   "location": "office",   "temperature": "18°C",   "objective": "bump temperature up a bit" }</pre>	<pre>{   "action": "set_heating_strategy",   "parameters": ["eco"],   "action": "set_outdoor_temperature",   "parameters": [10],   "action": "set_indoor_temperature",   "parameters": ["office", 18] }</pre>	<p>Fine-tuned 1B/4B/8B:</p> <pre>{   "action": "set_heating_strategy",   "parameters": ["comfort"],   "action": "set_outdoor_temperature",   "parameters": [10],   "action": "set_indoor_temperature",   "parameters": ["office", 18] }</pre>
Input misunderstanding	<pre>{   "location": "bathroom",   "user behavior": "just cleaned",   "action": "turn off heating" }</pre>	<pre>{   "action": "set_heating_strategy",   "parameters": ["off"],   "action": "set_outdoor_temperature",   "parameters": [10],   "action": "set_indoor_temperature",   "parameters": ["bathroom", 18] }</pre>	<p>Fine-tuned 1B:</p> <pre>{   "action": "set_heating_strategy",   "parameters": ["eco"],   "action": "set_outdoor_temperature",   "parameters": [10],   "action": "set_indoor_temperature",   "parameters": ["bathroom", 18] }</pre>

- **Observe:** 1. **All fine-tuned** model: **ambiguity of inputs** → false heating strategy
- 2. **Fine-tuned 1B** model: limited ability → misunderstanding of inputs

# Evaluation

## Catastrophic Forgetting

A phenomenon that a model may **lose some knowledge or abilities** obtained before during the process of **gaining new knowledge**.

**MMLU: Multi-choice questions** cover a wide range, here: nutrition.

**GSM8K: school math problems** with diverse complexity.

Q: Choose the best answer from the options below. Only output the letter (A, B, C, or D) of your chosen answer. Do not explain your choice. Question: Which foods tend to be consumed in lower quantities in Wales and Scotland (as of 2020)?

A: Meat

B: Confectionary

C: Fruits and vegetables

D: Potatoes

Answer: B

Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for \$2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?

Answer: 18

# Catastrophic Forgetting

MMLU(200)	1B (Gemma)	4B (Qwen)	8B (Qwen)
Before fine-tuning	85	154	156
After fine-tuning	76(↓10.5%) 😞	120(↓22%) 😞	145(↓7%) ✓

GSM8K(200)	1B (Gemma)	4B (Qwen)	8B (Qwen)
Before fine-tuning	146	191	191
After fine-tuning	129(↓11.6%) 😞	185(↓3.1%) ✓	192(↓0%) ✓

Observe:

1. Catastrophic Forgetting do exist.
2. The ability of math solving is harder to forget/lose than simply remembering.
3. The less parameters that model consists of, the catastrophic forgetting is greater.

# Conclusion

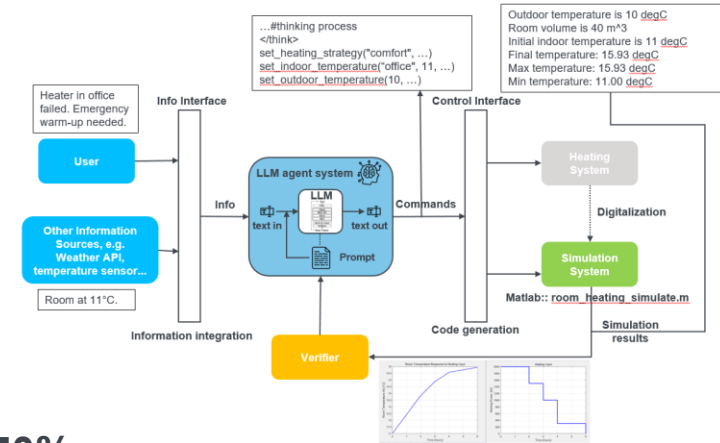
- Task completed
- Future work



# Conclusion

## Summary:

- Home Assistant with LLM
  - Proof of Concept
- Simulation integrated
  - Pre-trained model achieved an **accuracy of 50%**.
- Fine-tuning LLM for improved simulation application
  - Fine-tuned models achieved **93% of accuracy** (8B) on the test dataset.
  - Catastrophic forgetting exists, but math reasoning ability remains.



## Conclusion:

- The designed **system is effective**.
- Model fine-tuning **improves performance**.

## Future Work:

- Use real-world measured data to calibrate the simulation parameters
- Integrate with user-friendly interface



**University of Stuttgart**  
Institut of Industrial Automation  
and Software Engineering

# Thank you!



**Zheng Sun**

e-mail [st181999@stud.uni-stuttgart.de](mailto:st181999@stud.uni-stuttgart.de)

phone +49 (0) 711 685-

fax +49 (0) 711 685-

University of Stuttgart  
IAS



## Citation

- [1]J. Huang and K. C.-C. Chang, “Towards Reasoning in Large Language Models: A Survey,” May 26, 2023, *arXiv*: arXiv:2212.10403. doi: 10.48550/arXiv.2212.10403.
- [2]V. S. A. Duvvuru, B. Zhang, M. Vierhauser, and A. Agrawal, “LLM-Agents Driven Automated Simulation Testing and Analysis of small Uncrewed Aerial Systems,” Jan. 21, 2025, *arXiv*: arXiv:2501.11864. doi: 10.48550/arXiv.2501.11864.
- [3]G. Jiang, Z. Ma, L. Zhang, and J. Chen, “EPlus-LLM: A large language model-based computing platform for automated building energy modeling,” *Appl. Energy*, vol. 367, p. 123431, Aug. 2024, doi: 10.1016/j.apenergy.2024.123431.
- [4]Y. Xia, D. Dittler, N. Jazdi, H. Chen, and M. Weyrich, “LLM experiments with simulation: Large Language Model Multi-Agent System for Simulation Model Parametrization in Digital Twins,” Jul. 22, 2024, *arXiv*: arXiv:2405.18092. doi: 10.48550/arXiv.2405.18092.

## Citation

- [5]L. Wang *et al.*, “GenSim: Generating Robotic Simulation Tasks via Large Language Models,” Jan. 21, 2024, *arXiv*: arXiv:2310.01361. doi: 10.48550/arXiv.2310.01361.
- [6]“Energy\_balance\_TEXT\_02.pdf.” Accessed: Jul. 04, 2025. [Online]. Available: [https://elearning.unipd.it/dii/pluginfile.php/280289/mod\\_resource/content/1/Energy\\_balance\\_TEXT\\_02.pdf](https://elearning.unipd.it/dii/pluginfile.php/280289/mod_resource/content/1/Energy_balance_TEXT_02.pdf)
- [7]G. Team *et al.*, “Gemma 3 Technical Report,” Mar. 25, 2025, *arXiv*: arXiv:2503.19786. doi: 10.48550/arXiv.2503.19786.
- [8]A. Yang *et al.*, “Qwen3 Technical Report,” May 14, 2025, *arXiv*: arXiv:2505.09388. doi: 10.48550/arXiv.2505.09388.

# Appendix

## Relevant equations of simulation models

- Heat loss equation

$$\dot{Q}_{\text{loss}} = G_{\text{wall}} (\theta - \theta_{\text{out}}) \quad (1)$$

- Net power balance equation

$$\dot{Q}_{\text{net}} = \dot{Q}_e - \dot{Q}_{\text{loss}} \quad (2)$$

- Temperature change rate (thermal capacitance)

$$\frac{d\theta}{dt} = \frac{1}{C_{\text{total}}} \dot{Q}_{\text{net}} \quad (3)$$

- Euler Integration (discrete temperature update)

$$\theta_{k+1} = \theta_k + \left( \frac{1}{C_{\text{total}}} \dot{Q}_{\text{net}} \right) \Delta t \quad (4)$$

# Appendix

## Hyperparameter

- Epoch: the time that pass the entire training dataset through the learning algorithm.
  - More epochs could lead to overfitting, especially if your dataset is smaller than the pretraining corpus.
- Batch size: number of samples per optimization step.
  - A small batch size can stabilize training and help regularization.
- Max Gradient Norm: Prevents exploding gradients, helps keep training stable.
- Weight Decay: L2 regularization that discourages large weights.
  - Helps avoid overfitting. 0.001 is a commonly used value.

# Appendix

## Hyperparameter

- Learning Rate Scheduler: learning rate following a cosine curve.
  - Smoothly, encourages better convergence, commonly used in fine-tuning
- Learning Rate: Initial step size for optimization.
  - The choice conservative and safe for stability.
- Scheduler Cycles: Number of cosine cycles during training.
  - Ensures the learning rate smoothly decays across the epoch, avoids oscillation back up to a higher learning rate.

# Evaluation

## API Calls for simulation models

30 test data * 3 API	1B	4B	8B
Correct Rate	16 (18%)	47 (52%)	60 (67%)

Error Type	Scenario as Input	Expected Output	Model & Example Output
Incomplete API calls generation	Bathroom was just cleaned, turn off heating for a while.	set_heating_strategy("off", ...), set_outdoor_temperature(10, ...), set_indoor_temperature("bathroom", 18, ...)	1B: set_outdoor_temperature(10, ...)
String literal missing quotes	User planning nap in bedroom. Room currently at 19°C. Maintain eco mode.	set_heating_strategy("eco", ...), set_outdoor_temperature(10, ...), set_indoor_temperature("bedroom", 19, ...)	1B: ..., set_indoor_temperature(bedroom, 18, ...), ...
Excessive reasoning with no API output	Bathroom was preheated earlier. It's 23°C now. Save energy.	set_heating_strategy("off", ...), set_outdoor_temperature(10, ...), set_indoor_temperature("bathroom", 23, ...)	4B: {reasoning}(2235 tokens) (no API calls)



# Testing

## Function Definition

```
set_heating_strategy("comfort", ...)  
set_outdoor_temperature(-10, ...)  
set_indoor_temperature("study", 18, ...)
```

### Combinations of different parameters

1. set\_heating\_strategy(strategy\_name(str), ...)

Heating strategy: "off", "eco", "comfort"

2. set\_outdoor\_temperature(outdoor\_temp(int), ...)

Outdoor temperature: degree Celsius as unit

3. set\_indoor\_temperature(room\_name(str), indoor\_temp(int) ,...)

Room name: "living\_room", "bathroom", "bedroom", "office".  
Indoor initial temperature: degree Celsius as unit

# Training

## Model Selection – Gemma 3 1B(Google)<sup>[7]</sup>, Qwen 3 4B & 8B(Alibaba)<sup>[8]</sup>

- Among the **strongest open-weight** models → available for full parameter fine-tuning
- Pretrained on diverse datasets → **advanced reasoning** capabilities
- 4B & 8B models offer **best-in-class coding performance** per parameters
- Lightweight variants (1B ~ 4B) → **efficient** fine-tuning and deployment
- These combined characteristics allow:
  - Task-specific fine-tuning
  - API calling capacity
  - Efficient deployment



# Conceptual Design

## Model Fine-tuning

- Qwen-3 4B/8B, Gemma-3 1B-it show great potential in instruction following. But they are not good enough to deliver satisfied performance with this task. [4][5]
- Thus we need model fine-tuning to improve the abilities so that the model can generate preferred and correct API calling functions.
- Full parameter fine-tuning is more powerful than parameter-efficient fine-tuning, under same circumstances(model size, same dataset, same hyper-parameters, etc.).[6]  
Although full parameter fine-tuning can take up more calculation resources, with small scale large language models (max. 8 billion parameters), it still very efficient and effective.

# Evaluation

## Catastrophic Forgetting

MMLU: Multi-choice questions cover a wide range, here: nutrition

MMLU(200)	1B	4B	8B
Original	85	154	156
Fine-tuned	76(↓10.5%)	120(↓22%)	145(↓7%)

- Observe: Catastrophic Forgetting exists.
- Insight: 1. The more parameters, the better performance.  
2.
  - 1B models → Original model is weak → Not much to lose
  - 8B models → Original model is strong → Greater potential to learn new stuff
  - 4B models
    - Better than 1B model as original model
    - Not as strong as 8B model→ Forget the most among the three sizes of models

# Evaluation

## Catastrophic Forgetting

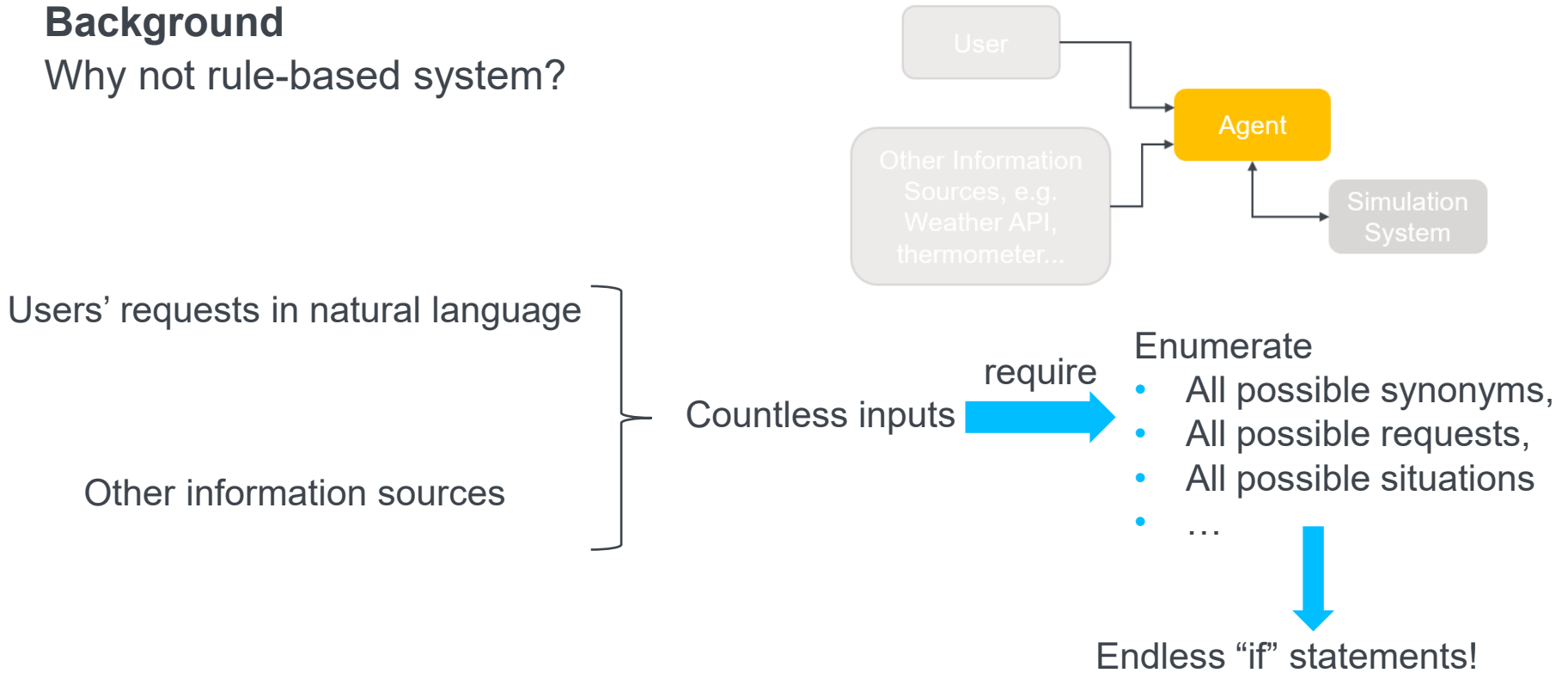
GSM8K: school math problems with diverse complexity.

GSM8K(200)	1B (Gemma)	4B (Qwen)	8B (Qwen)
Original	146	143(*191)	137(*191)
Fine-tuned	129(↓ 11.6%)	172(*185)	131(*192)

- Observe: For 4B and 8B models, there is nearly no catastrophic forgetting with the ability of solving math problems.
- Insight: The ability of math solving / reasoning is harder to forget/lose than simply remembering knowledge.
- \*: These are the cases that the correct answer is included in the thinking process, but due to overthinking, the output has no final answer. But with unlimited output tokens may generate the correct answer eventually.

## Background

Why not rule-based system?



Rule-based system is NOT practical and robust for this task!