CAIDAS-WORKSHOP: AI FOR SOFTWARE ENGINEERING

CODING BY DESIGN: LLM EMPOWERS AGILE MODEL DRIVEN DEVELOPMENT

Dr.-Ing. Ahmed Sadik → 01 October 2024

Senior Scientist
ahmed.sadik@honda-hri.de

Honda Research Institute Europe
Germany

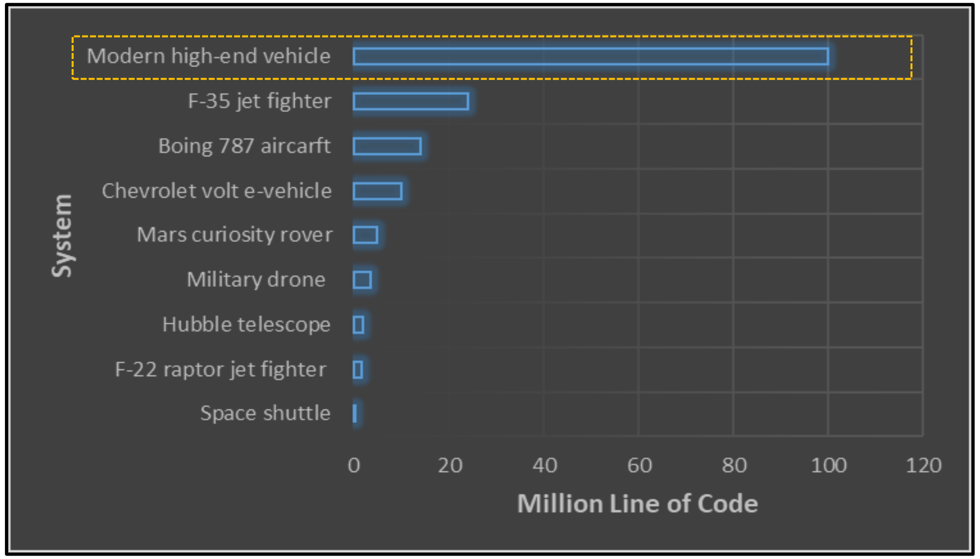
WHO ARE WE

- Honda Research Institute (**HRI**) focuses on “**Innovation through Science**”
- **Three locations** (Germany, Japan, USA) ~ 200 researchers
- **Honda Global Network** with universities and research institutes
- **Advanced research** in Automotive, Robotics, Machine Learning, Optimization, and System Engineering

OUTLINE

- ❑ MOTIVATION –> SOFTWARE COMPLEXITY
- ❑ CHALLENGES –> CODE GENERATION AND LANGUAGE AMBIGUITY
- ❑ APPROACH –> AGILE MODEL DRIVEN APPROACH
- ❑ CASE STUDY –> UNMANNED VEHICLE FLEET MODEL
- ❑ EVALUATION –> GENERATED CODE BEHAVIOR AND STRUCTURE
- ❑ CURRENT WORK –> HYPERGRAPHOS FRAMEWORK

MOTIVATION

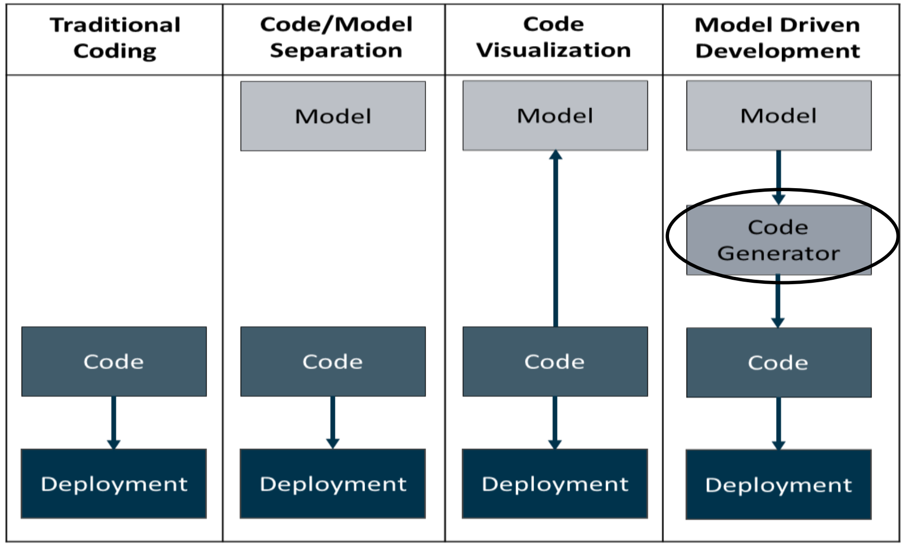


Source **[freecodecamp.org]**

- **Software based System Complexity →** Innovative solution to navigate system complexity
- **Large Scale System →** Agile approach to develop the system
- **Technology Transfer →** Architecting toolchain to craft precise system blueprints

CHALLENGES

➢ System Complexity



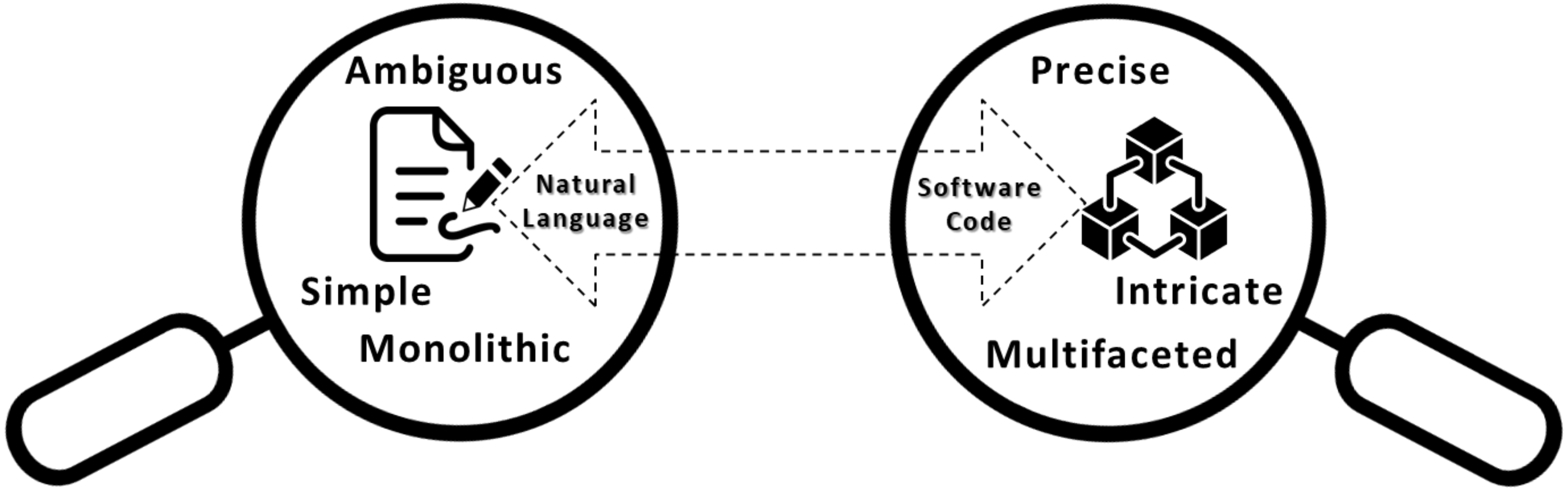
Source **[Jam08]**

- **Model Driven Development (MDD)** addresses the solution to manage software complexity
- Current MDD lacks **Agility** as it depends on customized code generators
- Replacing the code generator with a **Large Language Model (LLM)** enables a novel **Agile Model Driven Development (AMDD)** architecture

[Jam08] S. Kelly, J.P. Tolvanen; **"Domain-Specific Modeling: Enabling Full Code Generation"**. Wiley-IEEE Computer Society Pr., 2008.

CHALLENGES

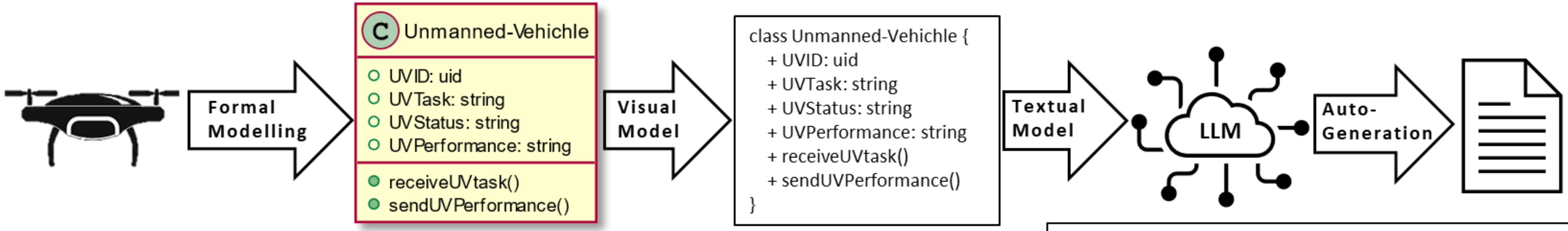
➢ **Natural Language Vs Software Code**



- Code Generation by LLM is commonly achieved via describing the software functionalities in **Natural Language** **[SCF23]**
- Generating **Deployment-Ready** software artifacts
- Generating **Intricate and Synergistically Structured** code

[SCF23] A. Sadik, A. Ceravola, F. Joublin. **"Analysis of ChatGPT on Source code"**, arXiv:2306.00597.

APPROACH

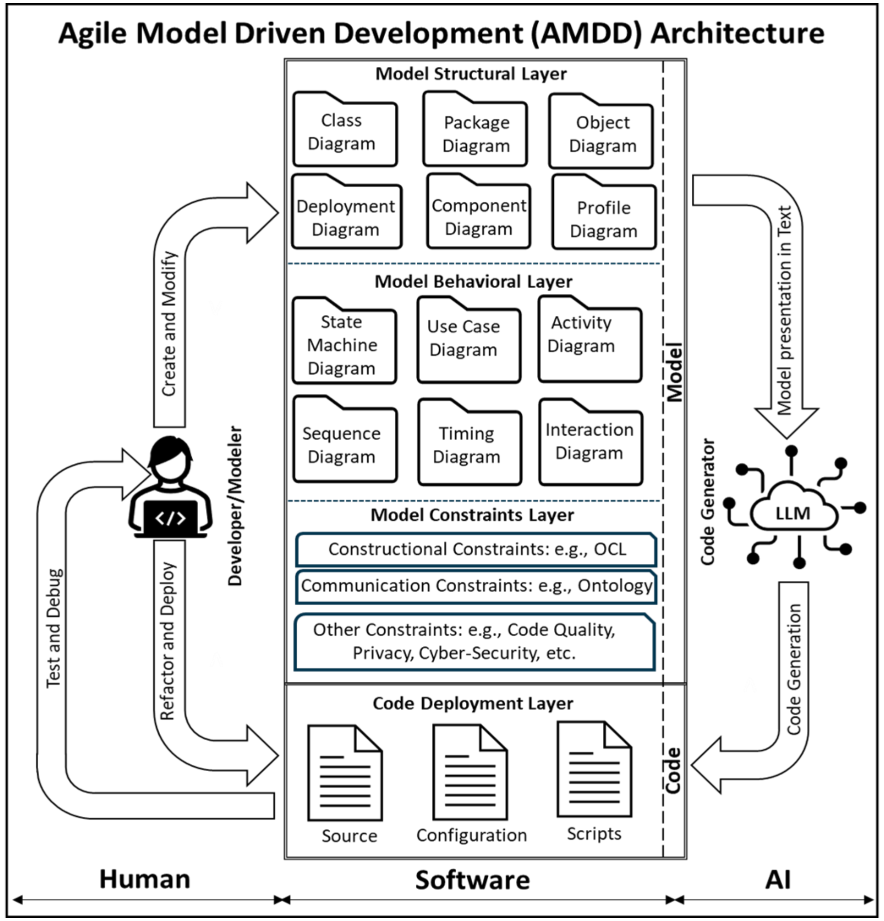


- Utilizing **Formal Modeling Languages** to sidestep ambiguity in natural language

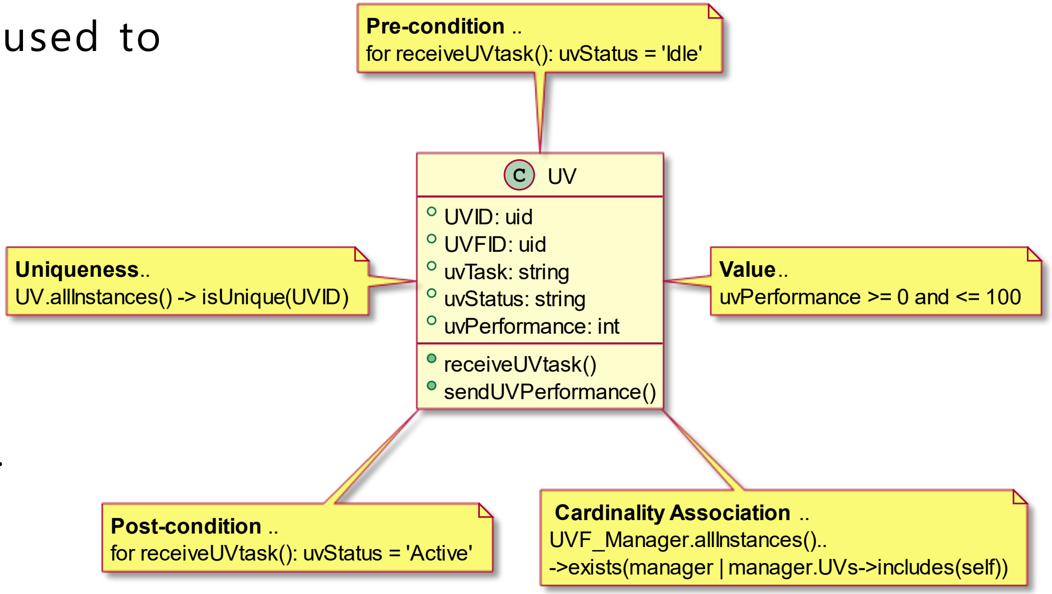
- Using **LLM to auto-generation** of deployment-ready software

- An **AMDD** framework leveraging formal constraints to enhance model semantic clarity and reduce its ambiguity

- Advancing **Collaborative-AI** in software engineering by **Integrating Human** in the loop to refine the auto-generated code



CASE STUDY: CONSTRUCTIONAL CONSTRAINTS



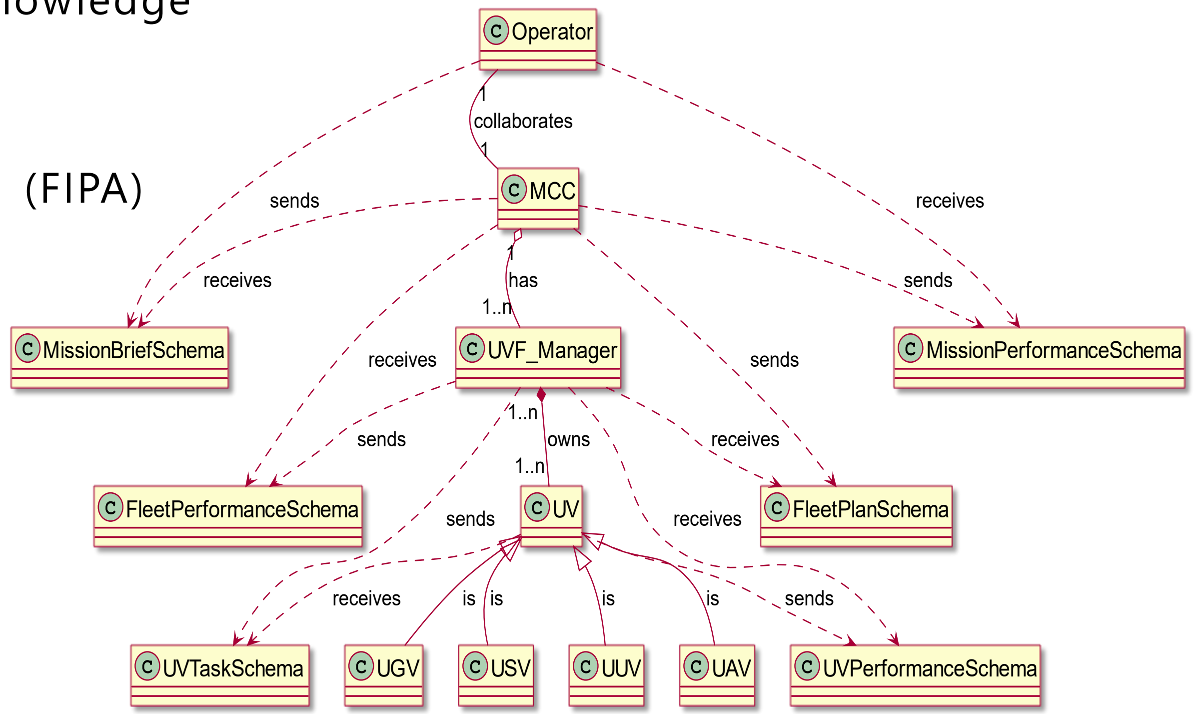
- **Object Constraints Language** (**OCL**) is declarative language used to specify precise constraints and fine-tune on the UML model
- Examples:
  - **Uniqueness:** ensure that every class instance is unique.
    → the UV agent must have a unique identifier across MAS
  - **Cardinality:** ensure the association of the class instances with each other's.
    → the UV agent is managed by the UVF-Manager
  - **Value:** ensure that some of the class values are limited to certain threshold.
    → the performance value of any UV agent is within the 0 to 100 range.
  - **Pre-Condition:** guarantee the state consistency of an instance before triggering the next state
    → the UV agent can only receive a new task if its current status is 'Idle'
  - **Post-Condition:** mandates the new state of an instance after moving from old state
    → after a UV agent has received a new task, its status must be updated to be 'Active'

CASE STUDY: COMMUNICATION CONSTRAINTS

- **Ontology** enables the common understanding of knowledge that are exchanged

- Examples: Foundation for Intelligent Physical Agents (FIPA) ontology

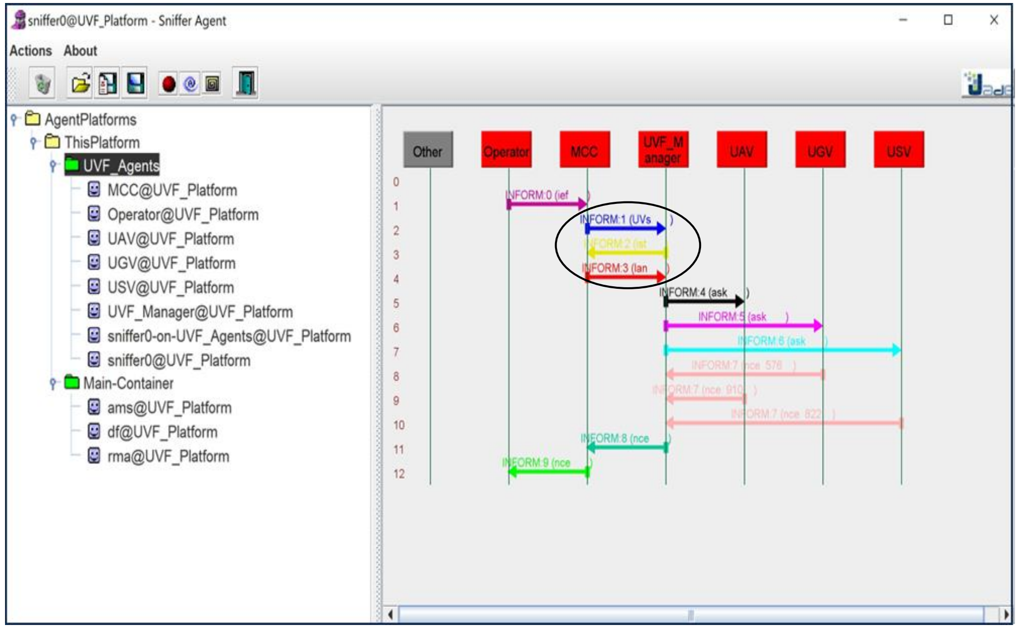
  - **Concepts:** Mission-Brief → an entity within the ontology
  - **Predicates**: (agent-x) <collaborates> (agent-y) → customized relationships among agents including their communication concepts
  - **Actions:** send (schema-x) → action performed by a concept

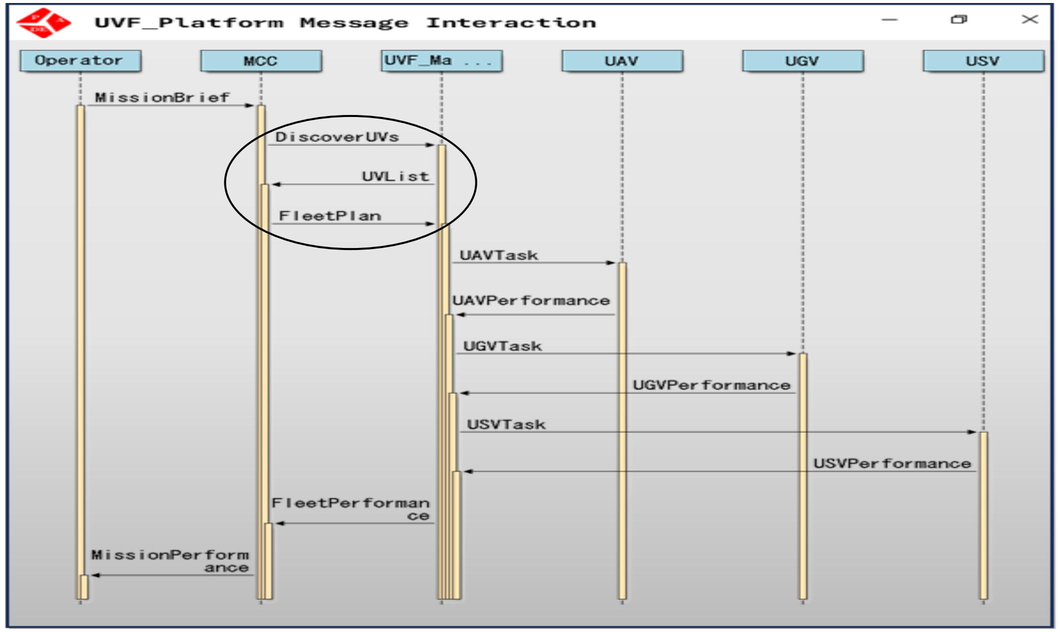


MCC: Mission Control Center **UVF**: Unmanned Vehicle Fleet **UV**: Unmanned Vehicle

EVALUATION: BEHAVIOURAL DYNAMIC

- Deployment in **Java** and **Python** is used to assess the code behavior



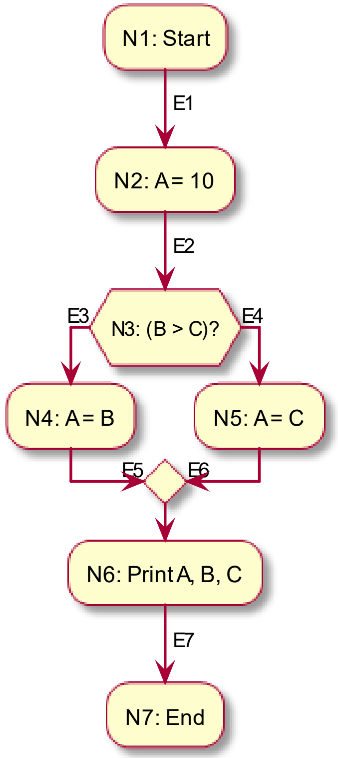


- The auto-generated code is **aligned** with the expected sequence diagram

- LLM enhanced the given activity diagram by **adding missing behavior** (Discover UVs, UVList)

EVALUATION: STRUCTURAL COMPLEXITY

- Code structure is evaluated through **Cyclomatic Complexity** (C) = Edges (E) - Nodes (N) + 2* Branches (B)
  - C = 1:10 → Low risk
  - C = 11:20 → Moderate risk
  - C = 21:50 → High risk



- Two models with different constraints levels are used

Generated code from a model with OCL constraints only

Agent class	Operator	MCC	UVF-Manager	UV	Model
Edges (E)	8	15	16	8	
Nodes (N)	8	13	14	8	
Branches (B)	1	1	1	1	
Complexity (C)	2	4	4	2	12

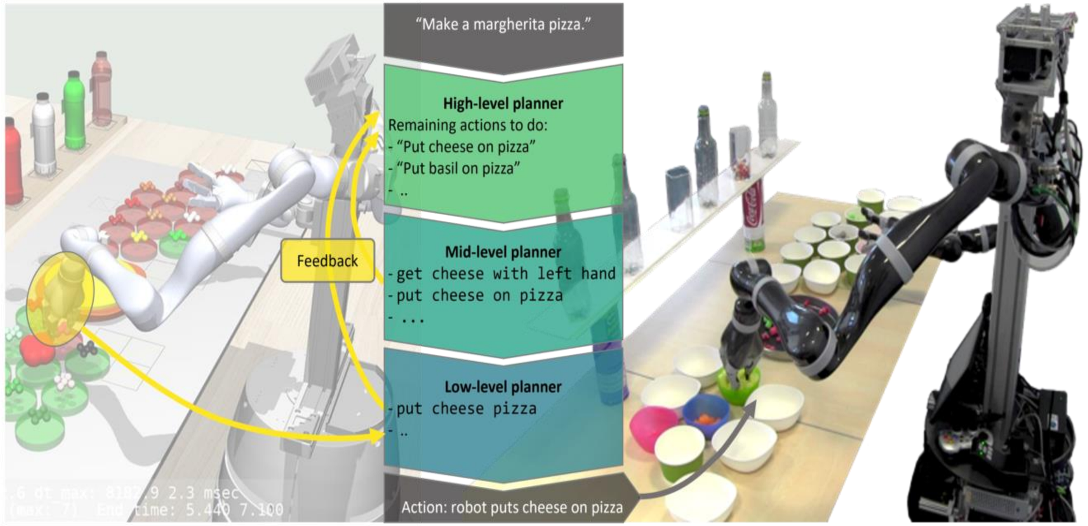
Generated code from a model with OCL and FIPA-Ontology Constraints

Agent class	Operator	MCC	UVF-Manager	UV	Model
Edges (E)	12	22	23	12	
Nodes (N)	11	19	19	11	
Branches (B)	1	1	1	1	
Complexity (C)	3	5	6	3	17

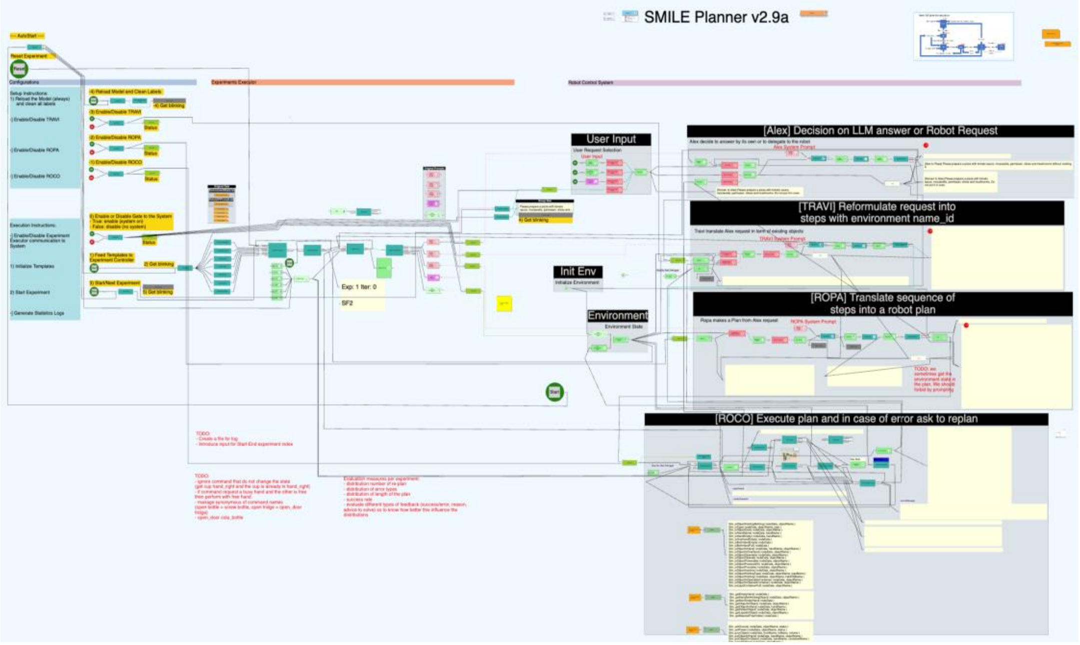
- Introducing ontology brings **additional accuracy** to the model, while it **adds to the complexity** cost
- The code complexity is still within the law risk zone, thus **more constraints can be included**

CURRENT WORK – HYPERGRAPHOS

- Objective: Cooperative planning of robot actions with LLM
- Approach: Model-based Multi-Agent System
  - 3 Main Agents (LLM):
    - Robot-Human Natural Language interface
    - Robot Motion Planner Specification Generator
    - High Level Plan Generator



- Features
  - Real-time feedback Loop: The robot generates real-time feedback while manipulating objects.
  - Corrective Feedback: On-Line Feedback used to correct/fix and problem while manipulating.
  - Real world test scenarios: Tested in making pizza, cocktail and stacking cubes.



SUM UP

- LLMs can enable agile transformation of MDD, where models become the primary code artifacts.
- The natural language ambiguity challenges LLMs in generating intricate, synergistically structured code
- Mitigate challenges by employing formal language models, and enhance auto-generated code quality through consideration of diverse system constraints
- LLMs enhance auto-generated code by introducing new behaviors. However, human supervision must be essential to prevent undesired code behavior
- Employing constraints boost auto-generated code complexity, yet increases its structural clarity
- A market gap in the AMDD toolchain requires further investigation

Thank you and happy to answer your Questions?

[SBO23] Ahmed R. Sadik, Sebastian Brulin, and Markus Olhofer. "Coding by design: Gpt-4 empowers agile model driven development." *arXiv preprint arXiv:2310.04304* (2023)

[Jou23] Frank Joublinet al. "CoPAL: corrective planning of robot actions with large language models." *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024