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

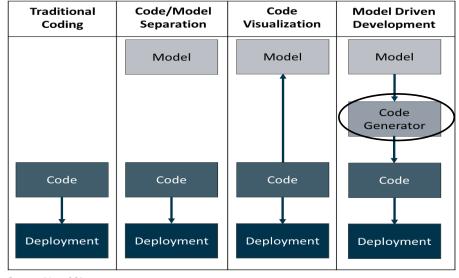
	·							
System	Modern high-end vehicle							
	F-35 jet fighter							
	Boing 787 aircarft							
	Chevrolet volt e-vehicle							
	Mars curiosity rover							
	Military drone							
	Hubble telescope	٥						
	F-22 raptor jet fighter							
	Space shuttle							
		0	20	40	60	80	100	120
		Million Line of Code						
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]

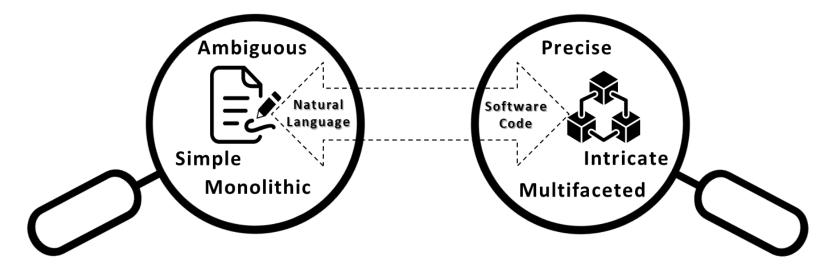
- Model Driven Development (MDD) addresses the solution to manage software complexity
- Current MDD lacks **Agility** as it depends on customized code generators
- Replacing the code generator with a Large Language Model (LLM) enables a novel Agile

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

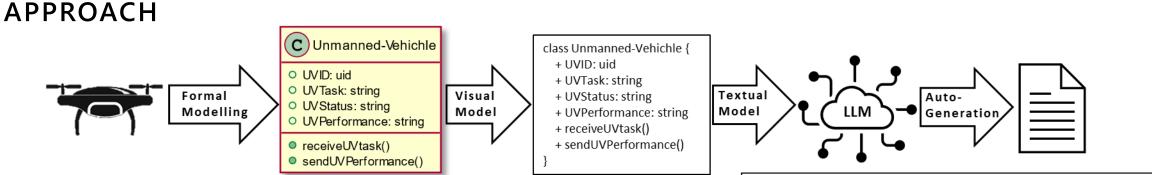


• Code Generation by LLM is commonly achieved via describing the software functionalities

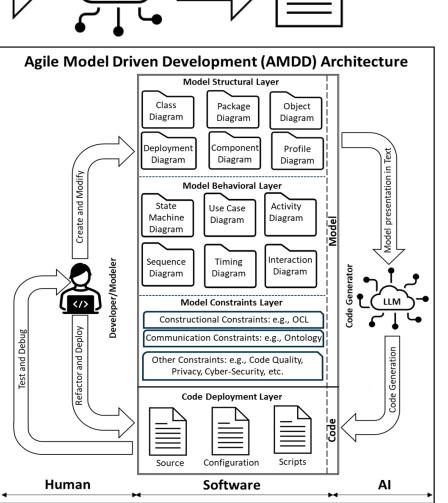
in Natural Language [SCF23]

- 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.



- 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



 \rightarrow 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 \rightarrow the UV agent can only receive a new task if its current status is 'Idle'

Value: ensure that some of the class values are limited to certain threshold.

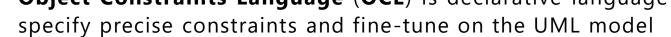
Post-Condition: mandates the new state of an instance after moving from old state

 \rightarrow after a UV agent has received a new task, its status must be updated to be 'Active'

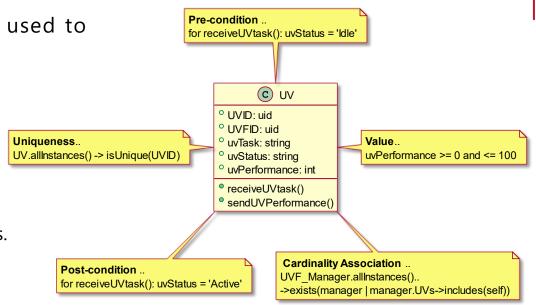
- \rightarrow the UV agent must have a unique identifier across MAS
- **Cardinality:** ensure the association of the class instances with each other's. \rightarrow the UV agent is managed by the UVF-Manager

Object Constraints Language (OCL) is declarative language used to

CASE STUDY: CONSTRUCTIONAL CONSTRAINTS



- Examples:
 - **Uniqueness:** ensure that every class instance is unique.

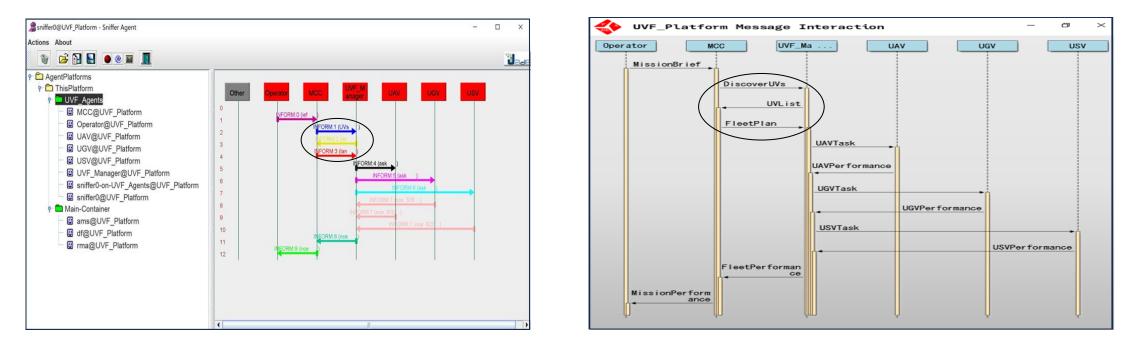


CASE STUDY: COMMUNICATION CONSTRAINTS

- © Operator collaborates CMCC receives receives sends CUVF_Manager C MissionPerformanceSchema C MissionBriefSchema receives sends receives C FleetPerformanceSchema C FleetPlanSchema CUV sends receives receives is lis C UVPerformanceSchema CUG CUAV C UVTaskSchema (c)เ MCC: Mission Control Center **UVF**: Unmanned Vehicle Fleet UV: Unmanned Vehicle
- 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) \rightarrow action performed by a concept

EVALUATION: BEHAVIOURAL DYNAMIC

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



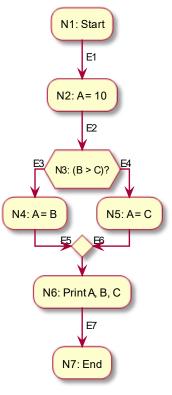
- The auto-generated code is **aligned** with the expected sequence diagram
- LLM enhanced the given activity diagram by adding missing behavior (Discover UVs, UVList)

EVALUATION: STRUCTURAL COMPLEXITY

- Code structure is evaluated through Cyclomatic Complexity (C) = Edges (E) Nodes (N) + 2* Branches (B)
 - C = 1:10 \rightarrow Low risk
 - C = 11:20 \rightarrow Moderate risk
 - C = 21:50 \rightarrow High risk
- > Two models with different constraints levels are used

Generated code from a model with OCL constraints only							
Agent class	Operator	МСС	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		

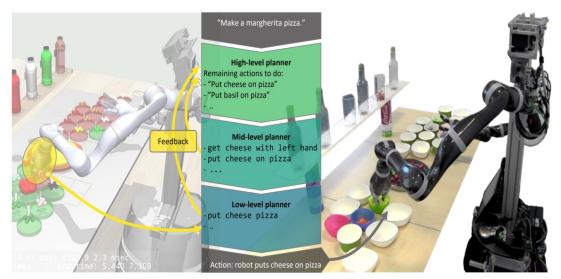
Generated code from a model with OCL and FIPA-Ontology Constraints							
Agent class	Operator	MCC	UVF- Manager	UV	Model		
Edges (E)	12	22	23	12			
Nodes (N)	11	19	19	11			
Branches (B)	1	1	1	1			
Complexity (C)	3	5	6	3	17		

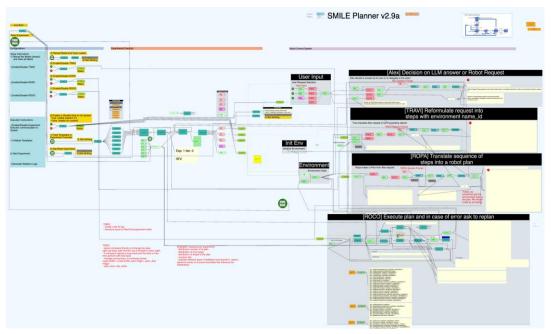


- Introducing ontology brings additional accuracy to the model, while it adds to the complexity cost
- The code complexity is still within the law risk zone, thus more constraints can be included

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 autogenerated 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?



I. INTRODUCTION

CoPAL: Corrective Planning of Robot <u>A</u>ctions with Large Language Models Frank Joublin[®], Antonello Ceravola[®], Pavel Smirnov[®], Felix Ocker[®], Joerg Deigmoeller[®], Anna Belardinelli[®], Chao Wang[®], Stephan Hasler[®], Daniel Tanneberg[®], and Michael Gienger[®]



man and pizza preparation.

large language models, robotics, plan. proposing a hierarchical architecture for robot manipulation tasks involving support for the structure for robot manipulation. tasks involving success/failure checks Classic planning techniques [1]-[3] focus on searching the · A novel closed-loop task planning r Classic planning security is 11-151 tocus on searching the optimal task leaving the problems of natural language grounding and low-level motion planning tasks out of scope 1 multi-level feedback loop (CoPAL). grounding and tow-lever motion planning tasks out or scope. Integrated task and motion planning (TAMP) approaches [4] strive to combine high-level reasoning with motion planning for real robots, but might still be challenged by uncertainties

mutti-level feedback loop (CoPAL). • An evaluation of the planning mechanism demonstrating how different kinds of low-level feedback improve the quality of planning and execution in three different evaluations both in visualizing and on the mail related quarty or pamming and execution in time trace of scenarios, both in simulation and on the real robot.

rear ronors, our ringen sum of characteringen of taxonationers at accounting for failure feedback. Intelligent embodied accounting for failure feedback. Intelligent embodied is however need to be able to adapt and recover from ent kinds of errors. Large language Models are there-increasingly used in robotics [5]-[14] because they II. RELATED WORK The idea of using LLMs for grounding high-level user The nucle of using LLAMS for grounding high-level user goals into mid-level action sequences (skills or action prim-tifives available to a robot) has been recently investigated both very rich commonsense knowledge and imning capabilities [15]-[17]. Advancements and tivies available to a robot) has been recently investigated in [24], backprompating planning: capabilities of LLMs [28], d [29], backprompating [20], [21], and conscription of LLMs [28], is strategies the been studied as well blance studies, are demonstrate to be efficiently utilize LLMs. for planning, they do not studies and the studies of the studies well as task execution feedbacks are taken into account. supplex application scenarios demonstrate this by compact apprication scenarios demonstrate ous ely [18]-[22]. Especially when dealing with under-Itaj-1421, Expectanty when dearing with under-ormation, LLMs are a promising tool to enable behavior, as they can leverage their inherent world nisms, with advances in feedback handling

> state. The approach relies on training skills and their policies sate, the approach tenes on stating same and the powers, via reinforcement learning, which could be considered a ... The via remotentient rearing, which could be considered a coally step in the era of large language models like GPT-4. The authors state that SayCan receives environme The autors scale that only an receives communication feedback only at a current decision step but in case a skill Offenhauk, Germany recurstance, omy an a current decision step rut in case a skill althout, in/icons/ althout, in/icons/ is not available. In this sense, feedback obtained from our

to provide alternative solutions. Incorporating when attenuitive solutions, incorporating were as task execution feedbacks are taken into account, monetal charges is a creater to potentially. Here, we aim to adhere exactly this gap, Below we review a number of statisk trageting closed-loop planning metha-nism, with advances in *Grades* become planning methamely reevaluate plans and react to potentially robots to deal with a wide range of situations. ability of LLMs to generate different action nisms, win aurances in reconsic naming. Lehter et presented SayCan [33] - an approach for grounding LLMs in alfordance functions, by capturing proband accounting for previous experience (in pdates) appears as a sensible approach grounding LLMs in arrorance functions, by capturing prob-abilities of possible skills to be useful for high-level goal ration and utilization [23]: "The that LLMs are generating potential achievement and to be executed successfully from the current fined by external solver ions to close this gap by

ebach, Germany

[SB023] 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