

MOSAICO

Challenges and opportunities in integrating multi-agent AI into model-driven engineering of dependable systems

Antonio Garcia-Dominguez - University of York
a.garcia-dominguez@york.ac.uk



Funded by the European Union

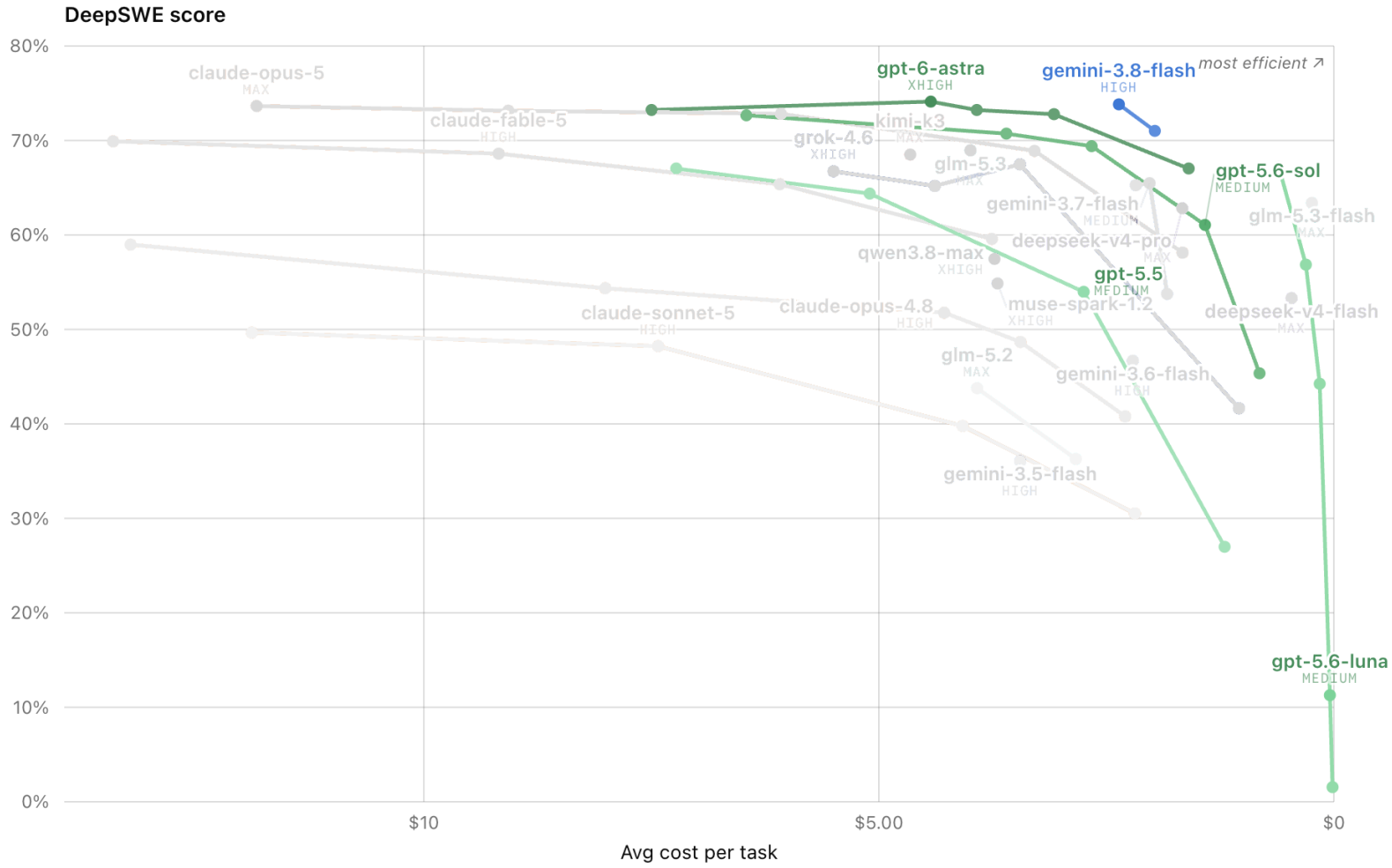
Funded by the European Union under the Grant Agreement No 101189664. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Health and Digital Executive Agency (HADEA). Neither the European Union nor the granting authority can be held responsible for them.

Motivation: what are the challenges?

Potential of LLMs to accelerate system modelling: looking at software development

MOSAICO

- Frontier LLMs are rapidly improving their performance at software engineering tasks
- GPT-6-Astra passes 74% of the 113 original tasks in DeepSWE
- Cost-score ratio keeps improving (compare GPT 5.5, vs 5.6, vs 6)
  - Gemini 3.8 Flash similar to GPT-6-Astra, at 1/3 of cost
- Can we bring this to dependable systems engineering?



Risks: non-determinism, data control, and lack of specialisation in foundational models

- LLMs are non-deterministic: same inputs can yield different outputs, even with temperature = 0 [1]
- LLM providers vary in data retention and training policies
  - Brokers like OpenRouter can enforce Zero-Data Retention, but data still leaves the organisation
  - Local LLMs keep data in-house, but models competitive with cloud models require strong hardware (Kimi K3 has 2.8T weights [2])
- Foundational LLMs are trained on what is out on the Web: they will not be familiar with less-popular languages, concepts, and formalisms

[1] S. Ouyang, J. M. Zhang, M. Harman, and M. Wang, 'An Empirical Study of the Non-Determinism of ChatGPT in Code Generation', *ACM Trans. Softw. Eng. Methodol.*, vol. 34, no. 2, p. 42:1-42:28, Jan. 2025, doi: 10.1145/3697010.
[2] https://huggingface.co/moonshotai/Kimi-K3#1-model-introduction

Risks: preserving properties as LLMs speed up change beyond human review

- LLMs are very good at generating code, but in the end, we are the ones responsible for what the product does
- LLM-generated code can exhibit biases from its training set, including the use of insecure / outdated coding practices [1]
- Developers surveyed in [2] combined manual code review, manual testing, checking outputs, and running external tests
- Significant load on the human reviewer: needs to be supplemented with automated processes (e.g. mix of LLM-as-judge and static analysis via CodeQL [3] for security)

[1] B. Tabarsi, H. Reichert, S. Gilson, A. Limke, S. Kuttal, and T. Barnes, 'LLMs' reshaping of people, processes, products, and society in software development: a qualitative exploration with early adopters', *Empir Software Eng*, vol. 32, no. 1, p. 21, Aug. 2026, doi: 10.1007/s10664-026-10917-0.
[2] R. Krasniqi, D. Xu, and M. Vieira, 'SE Perspective on LLMs: Biases in Code Generation, Code Interpretability, and Code Security Risks', *ACM Comput. Surv.*, vol. 58, no. 5, p. 137:1-137:16, Dec. 2025, doi: 10.1145/3774324.
[3] A. Khare, S. Dutta, Z. Li, A. Solko-Breslin, R. Alur, and M. Naik, 'Understanding the Effectiveness of Large Language Models in Detecting Security Vulnerabilities', in *2025 IEEE Conference on Software Testing, Verification and Validation (ICST)*, Mar. 2025, pp. 103–114. doi: 10.1109/ICST62969.2025.10988968.

Focusing on the modeling problem

MOSAICO

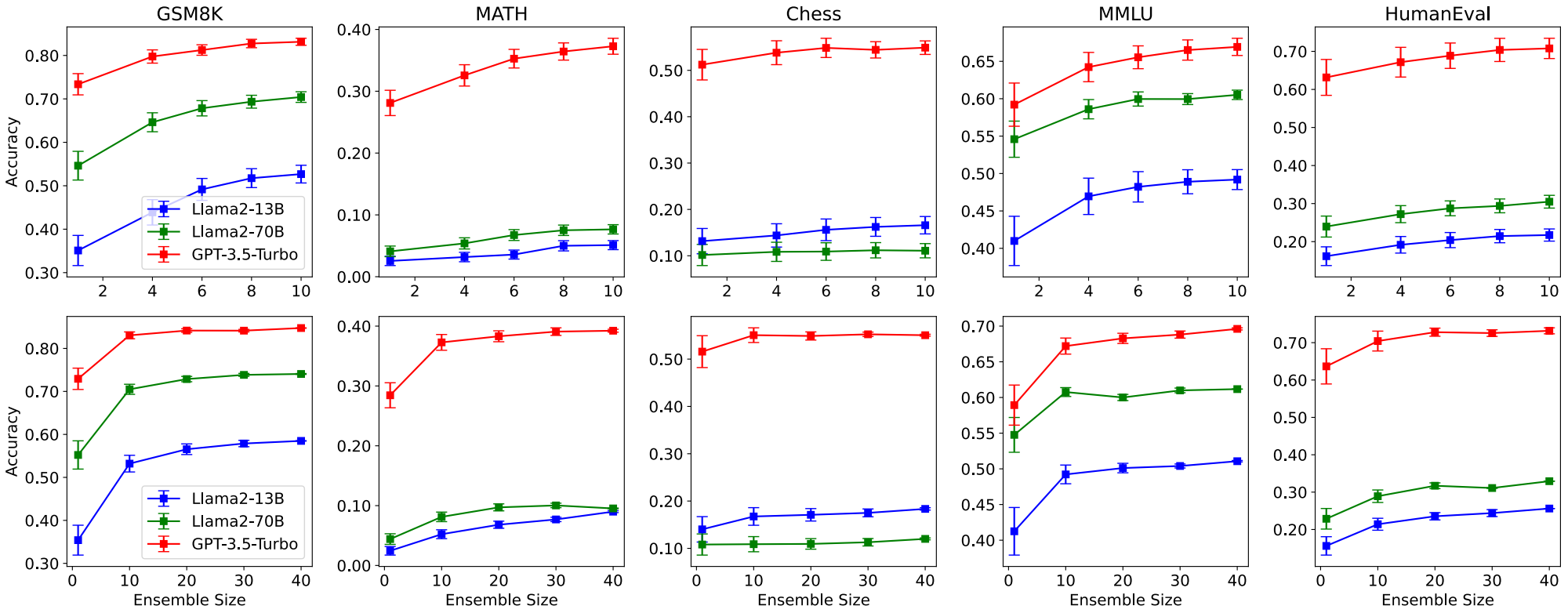
- Two parts to the problem:
  - Syntax – can the agents produce syntactically valid models?
  - Semantics – do the generated models make sense?
- Syntax can be easier for popular output formats
  - For example, popular programming languages like Python, or popular modelling languages like PlantUML
  - Agents need help for formats outside their training set
- Semantics also depends on the LLM's training dataset
  - Again, well-known domains are usually handled well
  - For specialised domains, we need to explain what their concepts mean and what are the valid ways to use and combine them
- Is it enough to throw more agents at the problem?

More agents can help…

MOSAICO

More Agents Is All You Need (Li et al, 2024)

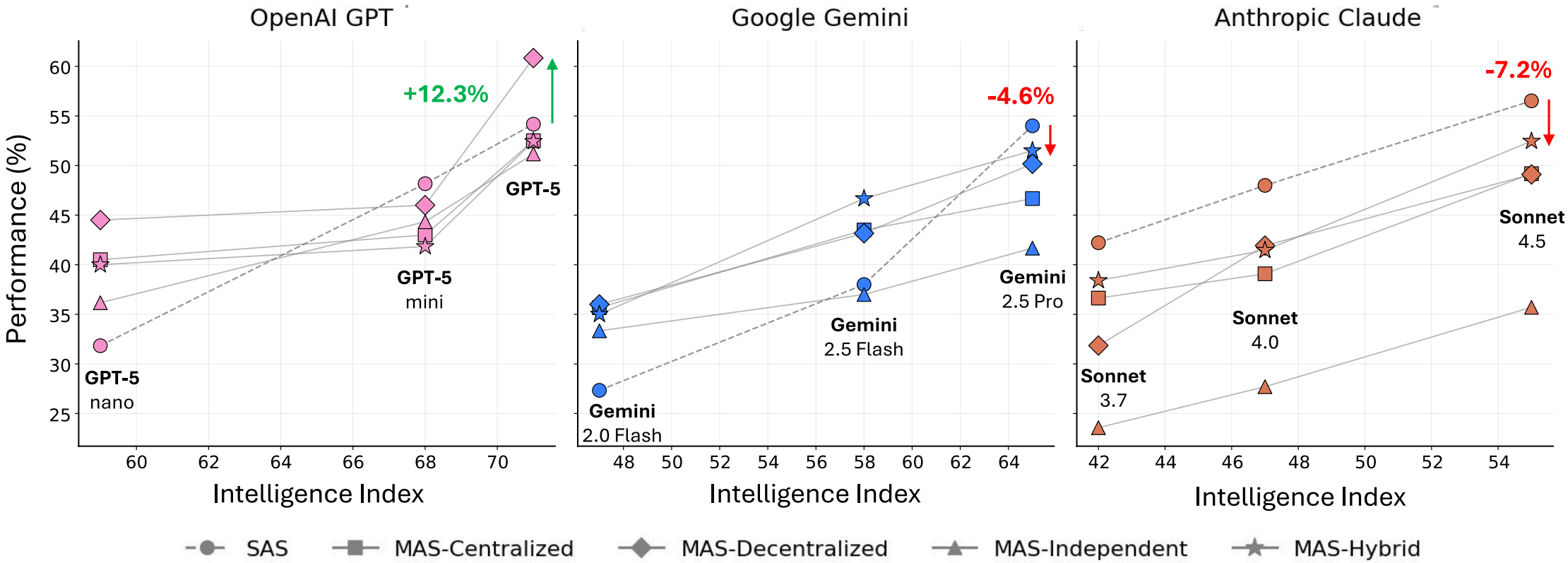
By sampling-and-voting, performance of LLMs increases with the number of agents instantiated (e.g. 10x Llama2-13B got to 70B levels of performance)



… but not always better

Towards a Science of Scaling Agent Systems (Kim et al, 2025)

Unstructured multi-agent networks amplify trace-level errors up to 17 times compared to single-agent baselines



State of the art: what is being done?

Expanding context: prompt engineering, Retrieval Augmented Generation



- As usual, need to craft prompts to clearly define the task, provide guidelines and rules, and set a persona for the LLM
- Beyond this, for domains and syntaxes outside the LLM's training set, we will need to expand the context with supplementary information
  - **One/few-shot prompting**: provide examples of input/output pairs
  - **Chain-of-thought**: ask to think in steps (built into "reasoning" LLMs)
  - **Structured outputs**: provide a JSON schema of output shape
  - **RAG**: complement prompt with information fetched from elsewhere
- There are limits to how much context we can use, and input tokens still cost us (even if cached, and lower-priced than output ones)

Feedback loops and planning: ReAct architecture, Tree of Thought

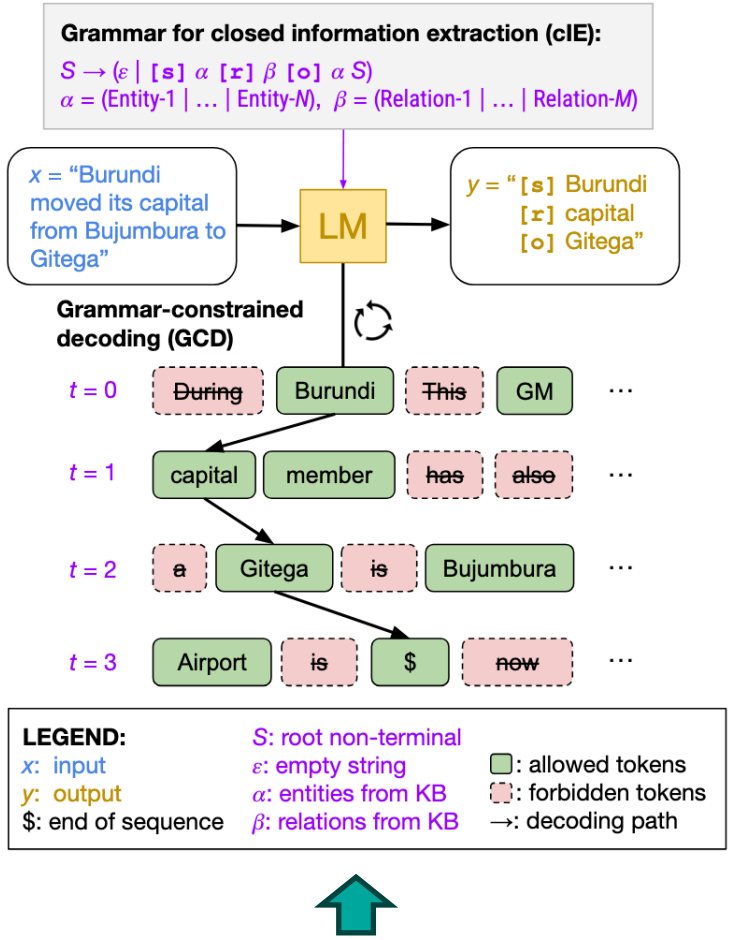
MOSAICO

- We can manually define workflows over LLM invocations, but this means we need to be able to anticipate the required steps
- If we cannot anticipate those steps (e.g. we're required to handle arbitrary problems thrown our way), we can let the LLM plan its work
- ReAct [1]: let the LLM run in a thought-action-observation loop, given a set of tools to work with, and an initial prompt
  - This is the core of most agentic coding tools nowadays!
- We can combine this with planning-oriented prompting strategies
  - Tree of Thought [2]: ask the LLM to come up with different alternatives and compare them, before committing to one

[1] S. Yao *et al.*, 'ReAct: Synergizing Reasoning and Acting in Language Models', in *Proceedings of ICLR 2023*, Kigali, Rwanda, May 2023. Accessed: Jul. 17, 2025. [Online]. Available: https://par.nsf.gov/biblio/10451467-react-synergizing-reasoning-acting-language-models
[2] S. Yao et al., 'Tree of Thoughts: Deliberate Problem Solving with Large Language Models', in Advances in Neural Information Processing Systems, Curran Associates, Inc., 2023, pp. 11809–11822. doi: 10.52202/075280-0517.

Enforcing syntax: grammar-constrained decoding, construction by code / tool calls

- To produce output in a format, grammar-constrained decoding [1] (e.g. GBNF) can filter tokens coming out of a local LLM
- Alternatively, instead of generating the output directly, we can have the LLM build it up by generating code (Smolagents) / calling tools
- At York, we produce agents that generate Python code building object-oriented models
- Modeling tool vendors (e.g. Capella, Sirius Web, MagicDraw) are building MCP servers to let LLMs interact with models



[1] S. Geng, M. Josifoski, M. Peyrard, and R. West, 'Grammar-Constrained Decoding for Structured NLP Tasks without Finetuning', in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, H. Bouamor, J. Pino, and K. Bali, Eds, Singapore: Association for Computational Linguistics, Dec. 2023, pp. 10932–10952. doi: 10.18653/v1/2023.emnlp-main.674.

Reasoning and abstraction for LLMs: neuro-symbolic and MDE-based approaches

MOSAICO

- By themselves, LLMs generate text that follows procedural thinking, but does not really reason in an analytical / deductive way [1]
- Symbolic approaches (i.e. automated reasoning systems) work in a deterministic way, but formalising is expensive
  - LLMs can help with this (e.g. LLM+P [2] formalises problems as PDDL for a classical planner)
- Similarly, we can use LLMs to work on high-level system models we can validate more easily, then use deterministic code generators

[1] P. Shojaee, I. Mirzadeh, K. Alizadeh vahid, M. Horton, S. Bengio, and M. Farajtabar, 'The Illusion of Thinking: Understanding the Strengths and Limitations of Reasoning Models via the Lens of Problem Complexity', in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2025, pp. 108018–108059. doi: 10.52202/085713-3603.
[2] B. Liu et al., 'LLM+P: Empowering Large Language Models with Optimal Planning Proficiency', arXiv.org. Accessed: Apr. 06, 2026. [Online]. Available: https://arxiv.org/abs/2304.11477v3

Trade-offs: no single technique wins

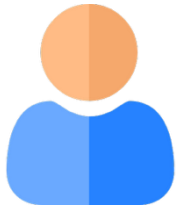
Technique	Execution cost	Development cost	Performance
Few-shot & chain-of-thought prompting	**Low–Med** extra input / reasoning tokens	**Low** write examples and guidelines	**Modest gains** helps with new formats; no guarantees
Retrieval-Augmented Generation	**Med** retrieval + longer prompts	**Med–High** build and maintain knowledge base	**Adds in-house knowledge** only as good as the retrieval
Structured outputs / grammars *JSON schema, GBNF*	**Low** negligible overhead	**Low–Med** schema / grammar (GBNF: local LLM)	**Syntax guaranteed** semantics are not
ReAct / Tree of Thought	**High** many calls, growing context	**Med** define the tools	**Self-corrects from tool feedback** handles open-ended tasks
Construction via code / tools *Smolagents, MCP*	**Med–High** iterative steps, sandboxed execution	**Med–High** modelling API or MCP server	**Valid syntax by construction** plays to LLMs' strength in Python / JSON
LLM + deterministic generators *neuro-symbolic / MDE*	**Low** generation step is deterministic	**High** metamodels + generators	**Most reliable artefact** validate at the model level
Multi-agent workflows	**High** sum of all agents involved	**High** orchestration and observability	**Best when structured [1]** unstructured: up to 17× errors [2]

Can we curate our own community of agents, using the best approaches for each task?

[1] D. Huang et al., 'AgentCoder: Multi-Agent-based Code Generation with Iterative Testing and Optimisation', arXiv:2312.13010, 2024.
[2] Kim et al., 'Towards a Science of Scaling Agent Systems', 2025.

What are we doing in MOSAICO?

A simple task for an AI agent



Please generate a domain model in Eclipse Emfatic for a simple library management system, with two main concepts: library and book. Each library has a name and contains a collection of books. Each book has a title. The domain model has to be syntactically and semantically correct.

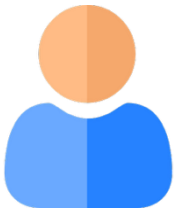


```
@namespace(uri="http://www.example.org/libmanagement", prefix="p")
package LibraryManagement;

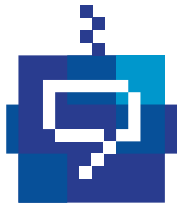
class Library {
  attr String name;
  val Book[*] books;
}
class Book {
  attr String title;
}
```

Direct LLM call

MOSAICO divides it across multiple agents



Please generate a domain model in Eclipse Emfatic for a simple library management system, with two main concepts: library and book. Each library has a name and contains a collection of books. Each book has a title. The domain model has to be syntactically and semantically correct.



Please generate a domain model in Eclipse Emfatic for a simple library management system, with two main concepts: library and book. Each library has a name and contains a collection of books. Each book has a title.



ReAct + code

Please syntactically validate this Emfatic model …



Deterministic parser

Please semantically validate this Emfatic model against this description …



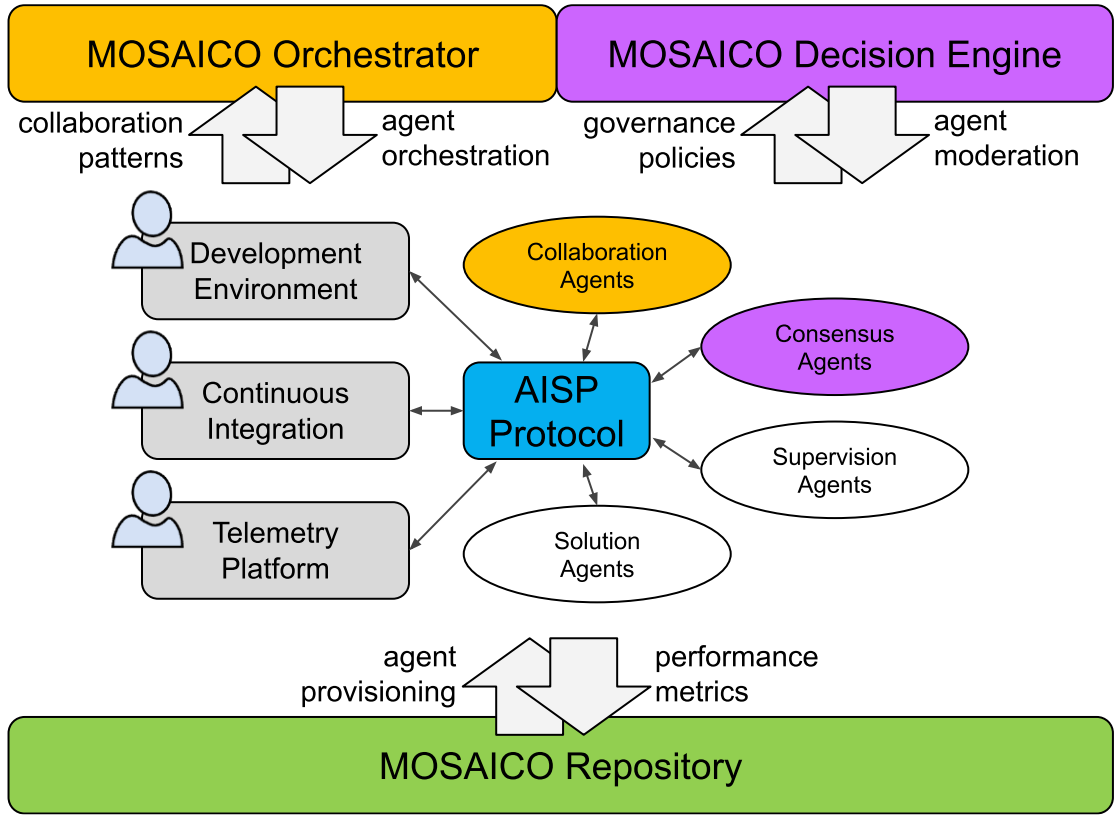
LLM-as-judge

MOSAICO: key details

- **Key objective**: Enabling and studying the cooperation of a (possibly large) number of LLM-based agents, for increasing the reliability of LLMs in software engineering
- **Project type**: Horizon Europe - Research and Innovation Action
- **Consortium**:
  - 4 universities (U of York, IMT Atlantique, LIST, U of L'Aquila)
  - 8 industrial partners across Europe

MOSAICO: key components

- Four kinds of agents:
  - Solution
  - Supervision
  - Consensus
  - Collaboration
- Agents communicate via a **common protocol** (AISP)
- Agents + benchmarked / live KPIs discoverable via a **repository**
- Generic orchestrator and decision engines for **collaboration** and **consensus**



AISP as an A2A profile

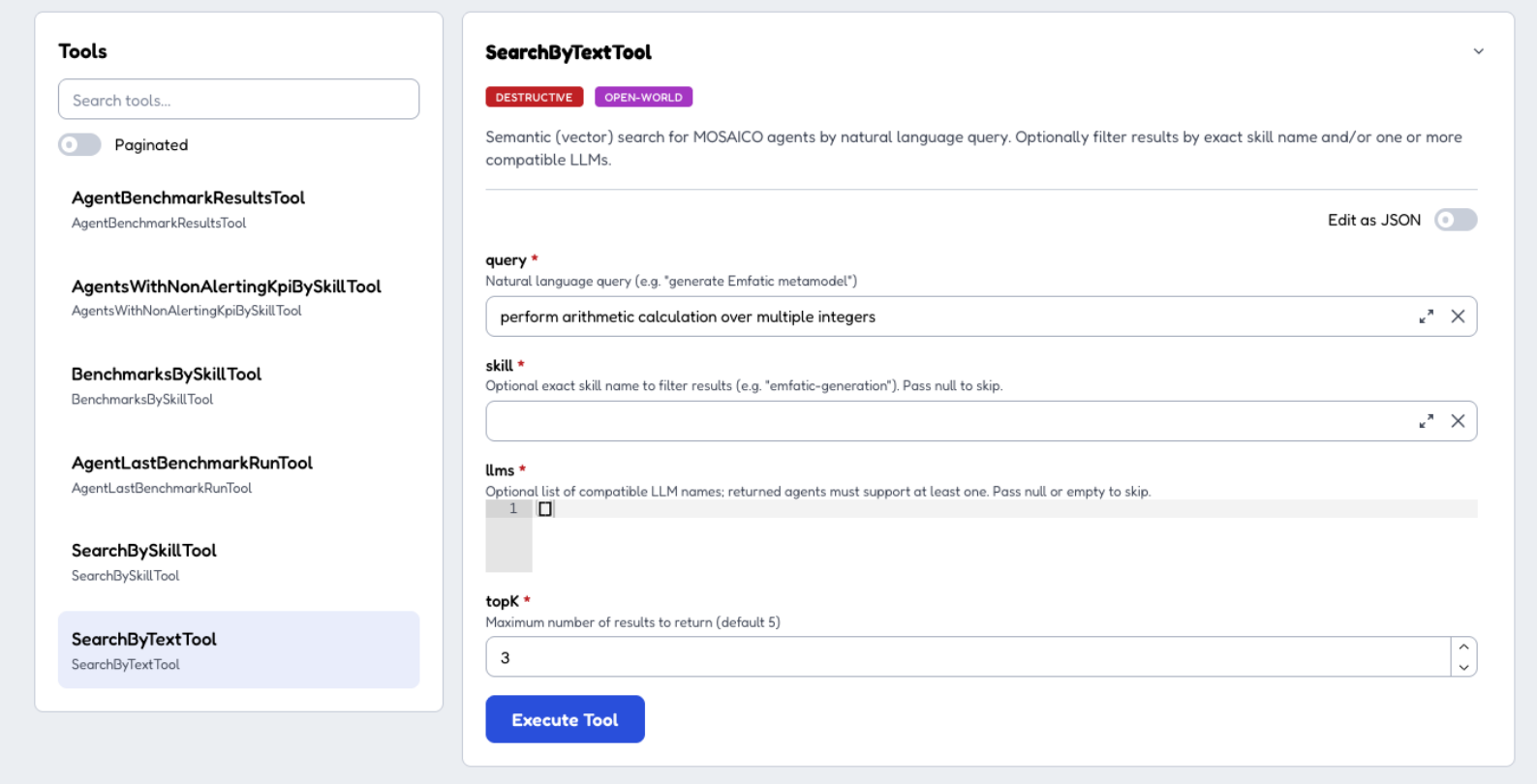
- We need a common protocol so the MOSAICO agents can converse
  - Agents can be developed independently and use different technologies / LLMs / strategies
- We prototyped an in-house protocol, and compared against protocols from industry:
  - Anthropic Model Context Protocol
  - Google Agent-to-Agent Protocol
- MOSAICO AISP is an A2A profile:
  - Rules for each agent type
  - Cross-collaboration observability extension
  - Agent-directed info retrieval extension
  - Planned: fine-grained feedback extension

Name	Description	Date	Reference
ANP	Agent Network Protocol	Oct 2024	(ANP Team, 2025)
MCP ⋆	Model Context Protocol	Nov 2024	(Spahr-Summers et al., 2025)
LMOS	Language Model Operating System	Dec 2024	(Eclipse Foundation, 2025)
AGNTCY ACP	Agent Connect Protocol	Feb 2025	(Duminuco et al., 2025)
IBM ACP	Agent Communication Protocol	Mar 2025	(BeeAI, 2025)
A2A ⋆	Agent-to-Agent	Apr 2025	(Linux Foundation, 2025)

Table 2: Recently proposed protocols for agent collaboration, ordered by date of first release. Items marked with ⋆ are the protocols chosen for comparison against creating a new protocol.

MCP-enabled repository for agent search

- MOSAICO has a single entrypoint: the reference agent
- Reference agent always looks first in the repo: its LLM sums up the main task into a single sentence
- Repo has both REST and MCP endpoints
- Semantic similarity is used to find top N agents
- If candidates are close, user can pick among them (or leave it up to the LLM)



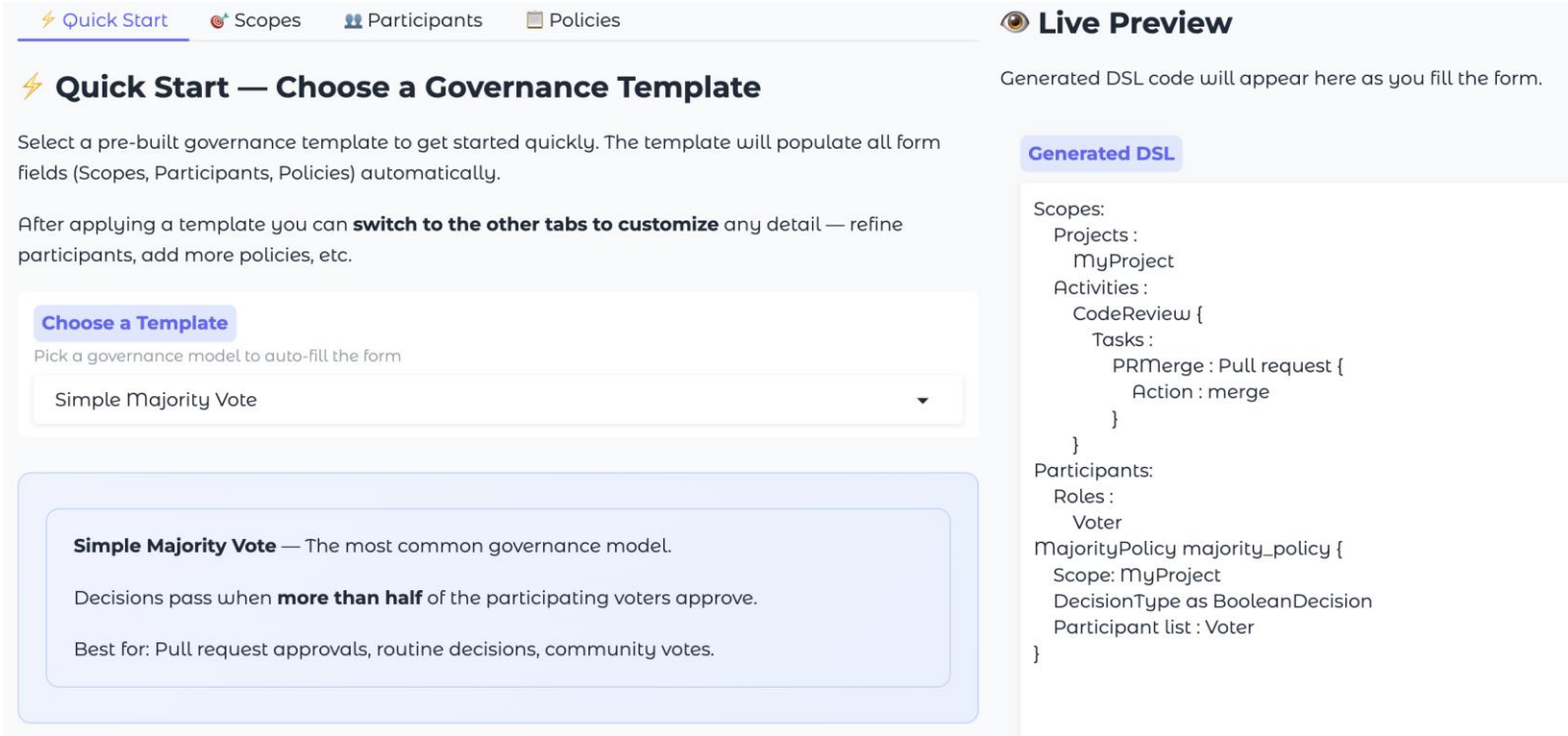
SysML2-based agent collaborations, and explicit consensus/governance rules

MOSAICO

- The SysMLv2 textual notation is being adapted to define action flows for modeling behavioral workflows and system behaviors.
- Workflows could be manually- or LLM-defined.

- MOSAICO is defining a policy language for consensus rules over solution / supervision agents.
- Textual DSL, with form-based UI for easy instantiation via templates.

Let's see how MOSAICO works.



Demonstration

Challenges and opportunities

MOSAICO

- Challenges:
  - LLMs are non-deterministic, and we need to stay dependable
  - LLMs can overwhelm human reviewers with large amounts of output
  - LLMs may not be trained in your domain, processes, and/or tools
- Opportunities:
  - LLMs are strong at translating: rather than ask them to produce, they can help formalise natural language to deterministic artifacts
  - Domain-specific modelling languages can help in providing a higher-level abstraction that the LLM can target (instead of code)
  - In MOSAICO, we are researching infrastructure to curate your own in-house community of specialised agents



MOSAICO

Thank you!

Visit us at **www.mosaico-project.eu**

Funded by the European Union

Funded by the European Union under the Grant Agreement No 101189664. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Health and Digital Executive Agency (HADEA). Neither the European Union nor the granting authority can be held responsible for them.