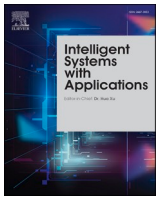
Intelligent Systems with Applications 29 (2026) 200626



Contents lists available at ScienceDirect

Intelligent Systems with Applications

journal homepage: www.journals.elsevier.com/intelligent-systems-with-applications





Generative AI for autonomous data analytics

Mattheos Fikardos [a], Katerina Lepenioti [a], Alexandros Bousdekis [a], Dimitris Apostolou [a,b], Gregoris Mentzas [a,*,1]

[a] Information Management Unit of the Institute of Communication and Computer Systems, School of Electrical and Computer Engineering, National Technical University of Athens, Greece
[b] Department of Informatics at the University of Piraeus, Greece

ARTICLE INFO

Keywords:
Large language models
Data science
Analytics
Autonomous agents

ABSTRACT

Recent advancements in Large Language Models (LLMs) and Generative Artificial Intelligence (GenAI) have revolutionised software engineering (SE), augmenting practitioners across the SE lifecycle. In this paper, we focus on the application of GenAI within data analytics—considered a subdomain of SE—to address the growing need for reliable, user-friendly tools that bridge the gap between human expertise and automated analytical processes. In our work, we transform a conventional API-based analytics platform into a set of tools that can be used by AI agents and formulate a process to facilitate the communication between the data analyst, the agents and the platform. The result is a chat-based interface that allows analysts to query and execute analytical workflows using natural language, thereby reducing cognitive overhead and technical barriers. To validate our approach, we instantiated the proposed framework with open-source models and achieved a mean overall score increase of 7.2 % compared to other baselines. Complementary user-study data demonstrate that the chat-based analytics interface yielded superior task efficiency and higher user preference scores compared to the traditional form-based baseline.

1. Introduction

Artificial General Intelligence (AGI) is a controversial and yet not fully defined term that aims to describe artificial intelligence systems that have capabilities equal to the level of humans (Morris et al., 2023). Even though this term is loosely defined and can be dated years ago (Goertzel and Pennachin, 2007), research breakthroughs in recent years have raised its popularity. Advances in Generative Artificial Intelligence (GenAI), which is a subset of AI, and specifically innovation in the field of large language models (LLMs), significantly increased the interest in AI, as LLMs indicate vital signs of AGI (Bubeck et al., 2023). Since the introduction of GPT-3 (Brown et al., 2020), the usage of LLMs in areas such as education (Kasneci et al., 2023) and medicine (Thirunavukarasu et al., 2023) has increased, due to the opportunities that the models can offer. The field where the technologies of GenAI had the biggest impact was in software engineering, where LLMs have been extended to support a variety of software engineering tasks (Hou et al., 2023) by both performing specific tasks autonomously, such as unit testing (Schäfer et al., 2023), or acting as an intelligent assistant and supervisor, empowering software engineers and enhancing the efficiency of software engineering tasks. In contrast, the introduction of such models in applications also comes with challenges that are constantly increasing and evolving alongside the technology, creating limitations. The main challenges affect the reliability and accuracy of the models (Kaddour et al., 2023, Zhang et al. 2025), where their capabilities must be formed and guided to follow specific processes (Moncada-Ramirez et al., 2025), especially when their impact is critical, and work with sensitive data.

Our work focuses on automating data science software engineering tasks. Data Science can be formally defined as the process of extracting knowledge from data by collecting, analysing, interpreting, and gaining insights using scientific methods, algorithms, and systems to make data more understandable and valuable (Cao, 2018). Given the complexity of data science projects and related demand for human expertise, automation has the potential to transform data science software engineering tasks (De Bie et al., 2021). Our broader goal is to design, develop, and implement a framework that incorporates AI, for automating or

* Corresponding author at: Information Management Unit, National Technical University of Athens, 9, Iroon Polytechniou Str., 157 80 Athens, Greece.
E-mail addresses: mfikardos@mail.ntua.gr (M. Fikardos), klepenioti@mail.ntua.gr (K. Lepenioti), albous@mail.ntua.gr (A. Bousdekis), dapost@unipi.gr (D. Apostolou), gmentzas@mail.ntua.gr (G. Mentzas).
[1] http://imu.ntua.gr.

https://doi.org/10.1016/j.iswa.2026.200626
Received 29 January 2025; Received in revised form 2 December 2025; Accepted 1 January 2026
Available online 2 January 2026
2667-3053/© 2026 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

supervising and assisting data science software engineering tasks, such as: developing software systems for organizing and storing data so that they can be accessed and used when needed; software tools for processing and visualizing data; as well as creating and testing applications that extract, process, visualize and analyze insights from data.

Existing approaches primarily focus on integrating AI within platforms as copilots or addressing specific stages of the analytics lifecycle. Most of these methods leverage AI to automate technical and software engineering aspects of data science (or analytics). In contrast, our work differentiates as it (1) considers the whole lifecycle of analytics, (2) investigates the exposure of platform functionalities as tools accessible to external systems and (3) presents a systematic framework to design agentic workflows related to analytics. In this sense, our approach focuses on transforming the interaction of a traditional API-based analytics platform into a conversational interface that employs multiple LLM-based agents. The platform's services are converted into tools for the agents, which grant them access to several entities, algorithms, and functionalities related to analytics. We propose a framework to implement agentic workflows for the analytics lifecycle, composed of states, tasks, agents and tools, in order to enable a humane communication with the analytics platform. Specifically, our research aims to augment users' interactions in multiple analytics lifecycle steps, allowing them to request results, retrieve platform entities, build analyses and get algorithm explanations in natural language. The main contributions of our work are:

- A methodology to systematically design agentic workflows to augment data analytics
- Enhanced analytics platform with function-calling capabilities based on existing protocols that enable the integration with external GenAI systems
- Experiments that compare three architectures across different cases
- Feedback from users in the context of a real-world use case

The rest of the paper is structured as follows: Section 2 presents an overview of related work, introduces the analytics platform, and provides a detailed analysis of the challenges and enablers associated with LLMs. Section 3 describes the proposed framework, outlining its key components and the adaptations made to the analytics platform. In Section 4, experiments are conducted to evaluate an instantiation of the proposed framework against two baselines, highlighting some key findings as well as a study to receive feedback from users. Section 5 discusses the outcomes, limitations and future research directions. Finally, Section 6 concludes with a summary of our work and contributions.

2. Background

2.1. Related work

As mentioned earlier, there is a huge effort to combine GenAI and specifically LLMs with the fields of Computer Science and Software Engineering, with the ultimate goal of automating procedures and making autonomous systems. Researchers have already started assessing and exploring the usage of AI-powered systems in the field of data science in general. For example, Diaz-de-Arcaya et al. (2024), performed an extensive systematic survey about MLOps and AIOps, highlighting the challenges and opportunities of those fields while also mentioning the field of Data Science. From their findings, several limitations and solution directions arise, especially in the field of data science, which to some extent we try to address and approach with our work. Another similar study examined challenges in data science assisted by LLMs, focusing on the conversational aspects, and evaluating the performance of data analysts in executing 4 Tasks while using ChatGPT (Chopra et al., 2023). The authors, based on their experiments found obstacles in the communication between the subjects and the model and tried to suggest possible solutions by providing some recommendations.

Despite the plethora of related work in the field of AI and data science, proposed approaches specifically incorporating LLMs in the analytics lifecycle are quite limited and usually only address a small section of the procedure. Starting from the first step of the analytics lifecycle, which is Data Preprocessing, Zhang et al. (2023), tried to use LLMs and experiment with them on different data preprocessing tasks. By combining different methods such as Few-shot and batch prompting, they tested several models on different datasets and examined their scores. Hollmann et al. (2023), proposed a pipeline where the LMM can be tasked with the feature engineering part of the data, where it iteratively applies data manipulation actions and tests the performance of ML models. Li et al. (2023), introduced a framework of a multi-agent system empowered by an LLM delegating the process of training an ML model to the various agents needed. Additionally, researchers proposed frameworks that cover data science processes close to the analytics lifecycle, covering common steps. For example, Kotaru (2023), proposes a copilot that helps with tasks such as retrieval and analytics tasks on operator data while also executing code. John et al. (2023), proposed DataChat, which is an intuitive and interactive platform for data analytics. Its main goal is to remove the necessity to use code in data analytics and provide an architecture that handles data processing and machine learning through natural language. Finally, it is important to mention that already a lot of software solutions related to analytics have included in their features tools and assistants powered by AI which help the user with tasks related to the available tools. Despite their numerous functionalities and features those solutions are mostly focused on data science and their scope is limited to a subset of the analytics lifecycle, usually either to the technical and coding parts (Microsoft, 2023; OpenAI, 2023; Project Jupyter, 2025) of the process or at the visualizations and descriptive analytics (Databricks, 2023; DataChat Inc, 2025; Polymer Search, 2024; Qlik, 2025; Salesforce, 2025).

2.2. Challenges

LLMs have shown promising results in areas such as SE, but they carry some limitations regarding their usage. Research has already been conducted to identify such problems as the impact of LLM-based systems and their ethical challenges, such as bias, privacy and security, overreliance, explainability, reliability, and many more, with some scientists focusing primarily on the field of education (Hadi et al., 2023; Kasneci et al., 2023; Ray, 2023). In this work, we focus on technical challenges that impact their performance and capabilities as described by Kaddour et al. (2023). Specifically, the challenges of interest are around the areas of (1) Context length, (2) Prompting, (3) Hallucinations and misalignment, and (4) Outdated or Limited Knowledge, which hinder their usage in specific applications. The first two areas are somewhat correlated, as the prompting capabilities are mainly limited by the size of the model's input, in the case of LLMs, the context window. The size of the context window limits the amount of input and information the model can accept, restricting its ability to get external information and be guided with instruction prompting. The latter two areas are also related, as the limited and outdated training datasets on which the model was trained might lead to hallucinations and misinformation. Although the lack of updated knowledge can be a factor for hallucinations, it is not the only one, as LLMs can also have faulty reasoning and misunderstanding of the input data and instructions (Maynez et al., 2020) and even perform worse depending on the location of information inside the context window (Sun et al., 2023).

As previously discussed, the main functionality of LLMs is the prediction of the next token or word of a sequence, which is rather simple and shallow considering their emerging capabilities. Researchers in the field saw great opportunities to leverage that simplistic – at the start - functionality of LLMs; thus, a vast number of methodologies and techniques have already been developed to shape and extend their functionality. In our work, we refer to all those techniques as "enablers" as

they enable the technology of LLMs to be integrated and adjusted to the proposed framework. The rest of this section focuses on those enablers, which are categories of methodologies relevant to our work grouped: (1) Prompt engineering (how to instruct the model), (2) Knowledge injection (providing the model with specific information), (3) Memory expansion (how to manage the memory limitations), (4) Tool usage / Function calling (allowing the model to interact with external tools), (5) Fine-tuning (adjusting the model through training), and (6) AI Agents (having multiple AI agents, covering different aspects). Even though there is some overlap and correlation between the enablers, we try to group them based on their main contribution and how they are implemented.

2.3. Enablers

Prompt engineering: One of the first methodologies that researchers experimented with was under the umbrella of the term "Prompt engineering" which is used to describe how to create and manage the input text that is given to the LLM to be programmed (White et al., 2023). Before diving into prompt engineering, it is important to note that this enabler is mainly performing better to a category of LLMs called Instruction fine-tuned, which are fine-tuned to follow instructions of the user and are going to be analyzed in more detail in the Fine-tuning section. With prompt engineering, the models are given specific prompts (inputs) that are, in a sense, instructions on how to proceed. Thus, the instruction fine-tuned models are better suited for this. The main principle of prompt engineering is to format and shape the input given to the LLM to improve its performance and alignment with the user's request. The final prompt given to the model does not have to be the same as the one given initially by the user, but rather an expanded prompt that injects instructions or relevant information for the LLM and incorporates the user's input. This is quite handy when the model has to fulfil the user's request but also has to follow a specific system approach or consider any limitations and boundaries. In addition, prompt engineering can be used to guide the model on how to "think" and reason, pushing it to work closer to what a human would think and act. Several general prompt engineering methodologies have been created that help the model reply and think with a structured flow, resulting in better-performing results (Wei et al., 2022; Yao et al., 2023a). Moreover, the construction of prompts can also help minimise the risk of challenges such as hallucinations and is also used by other enablers such as knowledge injection and AI agents that are further analysed in the following subsections.

Knowledge injection: One of the key downsides of LLMs is the limited or outdated data that they were trained on, which can drive them to produce hallucinations or outdated information. As mentioned earlier, LLMs are trained on a huge corpus of data, which makes their training expensive in terms of both time and computational power. This means that their re-training to incorporate updated or specific knowledge is not a viable option. One of the abilities of LLMs is to adapt their output based on information given through their context window. With the help of prompt engineering, requests can be enhanced to include specific information that provides context to the LLM and helps it produce reasonable and more accurate answers. One of the first approaches for providing knowledge to LLMs was the method of Retrieval-Augmented Generation (RAG) proposed by Lewis et al. (2020). The authors argue that utilising hybrid models with parametric and non-parametric memory can solve some of the problems of Pre-trained Language Models (PLM), by retrieving relevant information from external documents based on a similarity search on the input tokens. A similar approach that will be covered in depth later in this section is the utilisation of function calling, where models can use code to execute functions and retrieve external information. Similarly, with RAQ, the models can get external information but are not limited to documents or the similarity search, as they can also get the information from other sources like databases, the internet, etc. Another approach that tries to diminish the problems of LLMs related to knowledge is to connect them with Knowledge Graphs (KG). Researchers have already developed novel methods for connecting the two components, with different views on how they are connected or how one supplements the other. Based on this paper (Pan et al., 2023), LLMs and KGs should complement each other in a way that is synergised to cover each other's weaknesses.

Memory expansion: The two previous enablers contribute to guiding the model and providing specific information aiming to increase its performance on a task, but unfortunately, both inherit and also contribute to another limitation of LLMs, which is the limited context window. Custom prompts and external knowledge can work well with the LLMs, but what happens when there is insufficient space? The context window is the input and what the mode can "see" and has a finite size that cannot be changed, contributing as a constraint factor on the available space that the prompts and knowledge can be applied. One logical assumption is to increase the context window size and thus the model size, which is rather tricky as the transformer architecture doesn't scale linearly in terms of needed memory, making its continuing expansion not feasible or reasonable after a certain point. Larger models are appearing gradually in terms of size and context window at a point where from a few hundred tokens' sizes, there are now models with more than a hundred thousand tokens, but this is still a considerable constraint. Even though some researchers tried to optimize and enlarge significantly the size of the context window (Ding et al., 2023), just scaling up the input size of the models is not always the solution, as based on recent studies models struggle to utilize and properly comprehend information which is located in the middle of the context window (Liu et al., 2023). Another approach proposed by Packer et al. (2023), which also leverages the function-calling ability of LLMs, is to use the model as an OS. Packer et al., inspired by how computers work and how their memory is managed, also tried to utilise LLMs as an OS. In this case, the OS's random-access memory (RAM) is the context window of the model. Despite the limited size, information can be added or removed on the fly, providing the model with only the needed information. In addition, the model can store, access, and retrieve information from storage (e.g., database or file storage), which can be received or provided through the context window. With this method, limitations of the actual context window can be avoided and give the perception of an infinite-size context window. Another indirect enabler that can also contribute to this memory management is through AI Agents, as it could divide the context window requirements into several roles of the same model that in the end require less memory, utilising their conversation abilities.

Tool usage / Function calling: One of the signs of higher intelligence is considered to be the creation and usage of tools to successfully complete tasks. Fortunately, LLMs have already achieved enough intelligence to not only use but also create tools that can test and keep track for later use (Cai et al., 2023). In the literature, LLMs have displayed impressive programming skills (Chowdhery et al., 2022) and can be paired with programming runtime to create and execute code during inference (Gao et al., 2023), while others used existing tools such as APIs to help the model reason and perform specific tasks such as information retrieval and basic mathematical calculations (Yang et al., 2023; Yao et al., 2023b). Tool usage leverages prompt engineering mentioned earlier, where through the prompt models are given few-shot examples or even zero-shot examples with documentation (Hsieh et al., 2023) that enables the models to use such tools. Usually, this procedure is referred to as function calling (OpenAI, 2024), giving the models instructions on how to use functions and providing the necessary input variables in a structured format as JSON. Then, through some mechanisms that can differ, the model can perform tasks such as knowledge retrieval and provide accurate and external information to the user.

Fine-tuning: Apart from using enablers to enhance the performance of a pre-trained language model for downstream tasks during inference, another straightforward approach is to train a model for a specific domain. Due to the huge cost of computation, time, and effort needed to

train an LLM from scratch, the best solution in this case is fine-tuning the model. This means that a PLM will be fine-tuned by training on a smaller, narrower dataset, which also incorporates specific domain information related to the downstream task that will be used. So far, this evolving field of LMMs has seen much interest from researchers as it can yield many benefits while reducing the limitations of training models from scratch. First of all, via fine-tuning the needed size of a model can be significantly reduced if it is trained for a specific task. Reducing the size of the model while preserving its performance for a certain task can reduce the cost of inference and the needed computation resources.

Furthermore, fine-tuning also advocates the creation and adoption of open-source models, which are vital to privacy and security-intensive tasks. Also, as this field evolves, more efficient practices are created which accelerate the usage of LLMs, thus democratising their power. For example, so far researchers have already tried different methods to efficiently fine-tune models under the Parameter-Efficient Fine-Tuning (PEFT) paradigm, such as LoRa (Hu et al., 2021), QLoRa (Dettmers et al., 2023), and (IA)3 (Liu et al., 2022) where the fine-tuning process is optimised and the training cost is reduced. The aforementioned methods focus on the technical aspects of fine-tuning models on datasets, but efforts are also made to introduce fine-tuning with different mechanisms. Approaches such as utilising bigger models during training as teachers that a smaller model can mimic (Mukherjee et al., 2023) show promising results. Another alternative approach is to put humans in the loop during fine-tuning, a method called Reinforcement Learning from Human Feedback (RLHF), which helps the model adapt to human preferences (Ouyang et al., 2022). Finally, a category of fine-tuned LLMs that are mostly related to our work is the instruction-tuned models that are specifically fine-tuned to follow instructions by the users (Chung et al., 2024; Peng et al., 2023), which can improve the model's responses when prompted by the user to follow certain instructions. In the context of this paper, fine-tuning was not directly implemented due to the aforementioned challenges and the fact that our work is still in the preliminary stages. Instead, we leveraged the capabilities of a model that was fine-tuned to follow instructions, enabling it to be guided and integrated with the framework.

AI Agents: Drawing inspiration from this survey (Xi et al., 2023), the last enabler introduced is called “AI Agents” and refers to the usage of LLMs as “autonomous” agents to perform various tasks. This enabler has only recently attracted attention as it is heavily dependent on and constrained by the previous enablers described in this section. In general, LLM-based agents are formulated by providing the model with attributes such as personality, tasks, capabilities, and responsibilities, which influence the way it perceives its environment and acts accordingly. There are diverse frameworks of such agents that focus on specific tasks and have different capabilities such as AutoGPT (Significant Gravitas, 2023), LangChain Agents (LangChain, 2025), CrewAI (CrewAI Inc., 2025) and Transformers Agent (Hugging Face, 2024). Furthermore, researchers have also introduced multi-agent scenarios where the same model is used to instantiate multiple agents that can also interact, converse, and cooperate with each other and be assigned specific tasks to fulfil (Wu et al., 2023) and even created dynamically based on the given task (Chen et al., 2023). This field is relatively new and rapidly evolving but has shown promising results on how LLM models operate when used as Agents, shaping the behaviour of the model as a single or multiple agents provides a structured yet flexible flow of operation that can follow a specific procedure and be aligned with task-specific requirements.

3. Proposed framework

The paper extends the Analytics-as-a-service platform, referred to as A4A, introduced in our previous work (Lepenioti et al., n.d), which aims to support autonomous analytics throughout their lifecycle: design, build, and deploy. The platform can be accessible to both data analysts who can design analytics and end users, who retrieve their results, through a GraphQL interface that exposes the platform’s functionalities via Application Programming Interface (API) requests. In essence, the analytics platform seeks to streamline and define all the necessary processes and entities of the analytics lifecycle, automating creation, configuration, and deployment of analytics services. Although the platform is capable of managing the analysis lifecycle, it does not inherently guarantee effective usability among its users. A critical limitation arises from its reliance on an API-based query interface, which requires users to formulate code-like requests and possess deep knowledge of configuration parameters, creating barriers for non-technical users. Moreover, the integration of advanced analyses and algorithms may introduce significant technical barriers, particularly for non-technical users, while overloading them with excessive functionalities can lead to overwhelming user experiences.

To address these challenges the proposed framework, as depicted in Fig. 1, follows modern GenAI application patterns where the agents are instructed by users and interact with their environment to fulfil tasks and respond to the user. The objective is to enhance communication between the user and the analytics platform by utilising a GenAI system as an intermediary step. In this setup, the GenAI system processes users’ requests formulated in natural language, which prompts GenAI agents designed to respond effectively. The agents communicate with the analytics platform via a defined protocol, providing them with the necessary context to accurately address to the user’s request. Upon processing the information, the agents generate a concise answer which includes relevant details from the analytics platform along with supplementary comments.

3.1. Building blocks

Firstly, the functionalities of the analytics platform must be properly integrated with the GenAI system in a standardised manner. Secondly, the management of the agents plays a crucial role in limiting their inherent limitations, such as hallucinations and limited knowledge. To systematically ensure that a proper workflow is followed and relevant information is available, a well-structured process must be defined. The creation, configuration, and execution of the process require some essential building blocks, as shown in the conceptual architecture of the framework (Fig. 2). Starting from the right-hand side, all the building blocks are instantiated based on three categories: (1) Agents (entities that can converse), (2) Process (configuration attributes of the process), and (3) Connectors (communication definitions for agents). Through the Process Creation section, in the middle of the figure, all the building blocks are combined and connected to form a unified process. Then, the configured process is executed through the Process Flow, where agents take actions within tasks and traverse through states to fulfil the user’s request. Finally, the user can observe and interact with all the other agents through a chat-based interface.

The architecture is designed as modular and discrete, enabling a variety of configurations that facilitate seamless communication among the entities. Moreover, diverse processes can be created in this way to fit specific requirements, while providing all the necessary attributes to maintain a process that can be strict and flexible at the same time.

3.1.1. Function calling

The first step of integrating any form of AI into the process of the analytics lifecycle is to provide access to the analytics platform functionalities. For this reason, a Function Calling API was created as part of the analytics platform, where several callable functions were exposed to the outside world. Even though those endpoints are static and retrieve information from the analytics, they play a crucial role in the agents’ performance. The designed functions were implemented to contain a short but complete description of their usage that could be provided to the agents as context, as well as a description of all the input parameters that are required. To call the function, the AI must provide two parameters: (1) the name of the function and (2) the function’s input

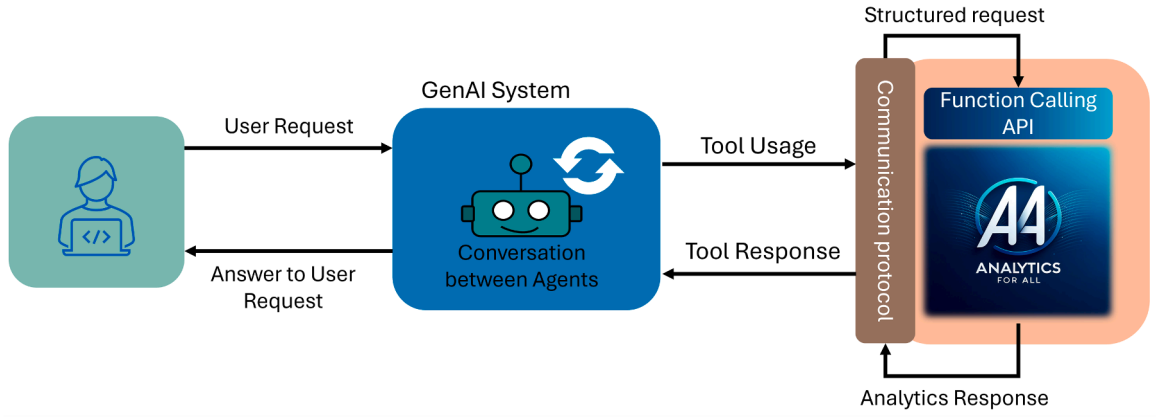


Fig. 1. Proposed framework overview.

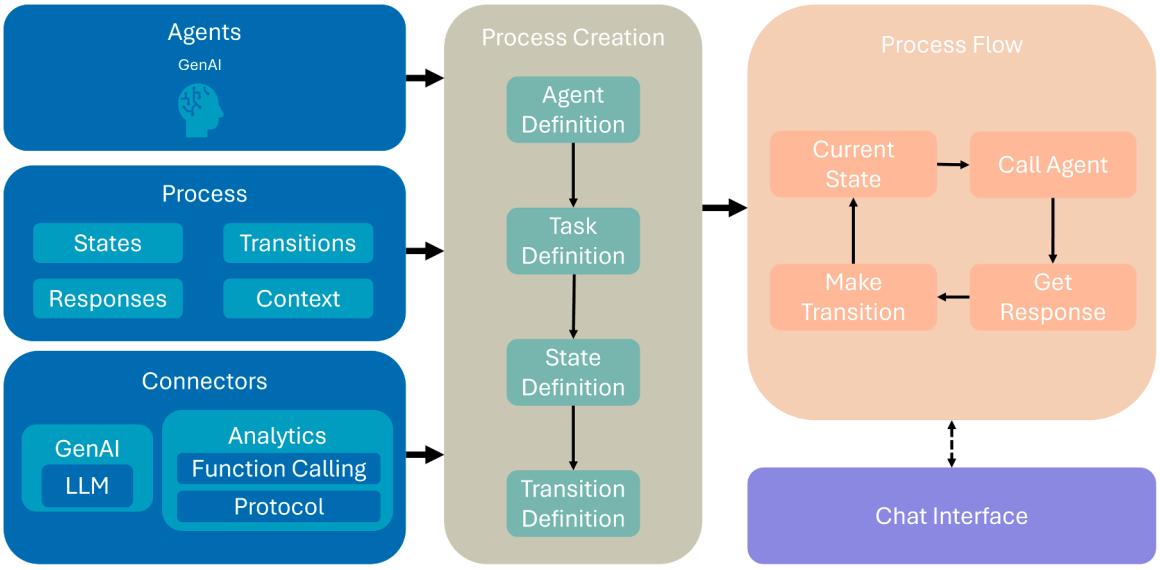


Fig. 2. Conceptual architecture.

parameters in JSON format if there are required inputs. A validation mechanism was implemented to identify potential errors in the incoming requests in order to ensure correct input to the functions. Moreover, to assist the agents in identifying and overcoming the potential misuse, a meaningful text description of the error with explanations is provided upon request failure.

To streamline the usage of the exposed analytics function calling API, the respective endpoints need to be wrapped as tools using a communication protocol. In the literature, many standards have emerged in recent years (Ehtesham et al., 2025; Yan et al., 2025; Yang et al., 2025), such as the Model Context Protocol (MCP) standard introduced by Anthropic (Anthropic, 2024). Since its introduction, MCP has been adopted by the research community and industry as it provides a JSON-based communication standard between LLM applications and external resources. The protocol revolves around three main concepts: the host (the LLM or agent), the client (MCP adapter of the host), and the server (wrapped tools and resources). In our setting, the analytics platform incorporates an MCP server that wraps the function calling into tools, enabling any compatible host to use them. This approach greatly increases interoperability and adherence to best practices.

3.1.2. Process formulation

To ensure that the AI agents can follow a structured and confined procedure to answer the user request and have adequate information for the analytics context, a process was formulated to guide them. The process is broken down into discrete steps called states, which are connected through transitions based on their outcomes. For each state, agents communicate and are given a specific context and instructions to produce responses. Based on the responses, transitions are selected to move between states and navigate through the process with the end goal of fulfilling the user's initial request. Based on the above, the process can be defined as:

$$P = \langle S, A, T, R \rangle$$

as a set of States (S), Agents (A), Tasks (T) and Transitions (R) that are combined to create the process P.

3.1.2.1. Agents. The framework provides a conversational interface as a chat, where multiple Agents converse and exchange information to perform tasks related to the analytics. Each Agent $a \in A$ can be defined as a tuple of three parameters:

$$a = \langle N, D, TL, O \rangle$$

where N is a name, D is a description or personality of the agent and is mainly used for the AI models as part of the system message, *TL* is the set of available tools, and O is the expected output. The human user is considered as an agent entity without parameters but with the ability to

provide input or comments to the GenAI agents. The human can be then introduced in the initial states to provide their input and to subsequent states to implement human-in-the-loop functionalities where approval or additional input is required.

3.1.2.2. State. The states can be interpreted as rooms where a single or multiple agents exist, and converse about a specific task while acquiring related information and producing a final response as the outcome of the conversation. The outcome is then forwarded to the next "room" as context, where the same or other agents follow the same procedure with a different task. Thus, a state $s \in S$ can be defined as a tuple of 5 entities:

$$s = \langle N,\ a^s,\ a^r, t,\ r \rangle$$

where N represents the name of the state s, the next two entities $a^s \in A$, $a^r \in A$ represent the sender and receiver agents respectively, $t \in T$ is the task assigned to the agents, and $r \in R$ are the transitions of the state. The transitions are defined in the following subsection and are embedded in the tasks as instructions.

3.1.2.3. Transition. The transitions form the connection between the states where the outcome of a state is forwarded to the next one. Based on the outcome of the state a decision is made on what transition has to take place. Some transitions are static and are forced to happen while others are dynamically executed based on the state outcome. Specifically, the dynamic transitions are happening from the agents or when the human is kept in the loop and needs to take action. A transition $r_{s_i} \in R$ of state s_i is defined as a tuple of 4 entities:

$$r_s = \langle DM_{t_s},\ K_{t_s},\ F_{t_s},\ OM_{t_s} \rangle$$

where:

- DM_{t_s} is a decision message as text which is used as context and instructions to the agents to make a decision and produce a response that indicates which decision is made.
- K_{t_s} is the keyword as short text that identifies which decision is made by the agent.
- F_{t_s} is a function that produces a Boolean representation of the decision based on the response Re of the agent: $F_{t_s} \times\ Re \rightarrow T\ /F$.
- OM is the output map which defines the mapping of the function output to the next state $s' \in S$: $(F_{t_s} \times\ Re) \times\ OM_{t_s} \rightarrow s'$.

3.1.2.4. Context window. The context window, which represents the AI agent's environment and what they can "see", plays a vital role in how the models respond. It is crucial to provide them with all the needed information while minimising quantity, maximising quality, and defining the instruction to properly respond. To achieve this, the prompt (context window) is dynamically constructed based on the state, the task, the agent, and the accumulated information gathered so far. Thus, the context window cw_i of the receiver agent a^r at state s is defined as:

$$cw = \langle Sys,\ C,\ PI,\ I \rangle$$

Where:

- Sys represents the system message which is the description D_{a^r} of the agent and the task instructions.
- C is the provided context, which consists of the response Re already produced by a^s, and information related to the analytics from the analytics platform.
- PI are the instructions from the decision message DM_{t_s} of the transition r_s.
- I is a set of prompt engineering methods, such as Chain-of-Thought.

3.1.3. Process execution

After all the building blocks of the process are created, the framework moves on to the process flow phase where the process is executed as shown in Table 1. At the start, the current state S is initialised and then the process moves to a loop that runs as long as the user is using the framework. Then, for each iteration, all the parameters are initialised based on the current state and are used to form the context window that is given to the receiver agent of the state. Following, based on the agent's generated response and the transition of the state, the next state S' is calculated. Before the transition is made, the process checks if the mechanisms of self-reflect and human in the loop are enabled and the appropriate corrections are made. The first mechanism can be used when an agent has to self-reflect and generate a response with a specified format and the second one can be used to enable communication with the analytics platform after human approval. In the case where checks were successful, the messages are displayed to the user, the parameters of the next state are updated and finally, the process transitions to the next state.

3.2. Process instantiation

We instantiated an analytics process with a total of 6 states and 5 agents, as depicted in Fig. 3, and refer to it as the *Flow*. The process begins with the *User Input* state, where the human provides their request. Their input is forwarded to the *Manager* state, where the manager agent determines which subsequent states must be executed and provides instructions to the next agent on how to proceed. The *Coder, Analyst,* and *Result Expert* states are derived from the available functionalities of the analytics platform, as shown in Table 2. The *Code* state includes a technical agent equipped with tools related to models and algorithms used in analyses. The *Analyst* state incorporates a data analyst agent responsible for managing and executing the different analyses. The *Result Expert* state handles the retrieval of descriptive results and the generation of predictive analytics. For descriptive results, the respective tool includes a filtering mechanism to aggregate specific results based on attributes and timestamps. Most of the tools require certain inputs; among these, the "analytics goal" which serves as the identifier of the analyses. For this reason, all agents have access to the *Intents* tool that provides a list of all the available analytics goals. This categorisation of tools into respective states and agents enables a more granular approach

Table 1
Process execution algorithm.

Algorithm 1 Process Execution	
1:	$S \leftarrow S_{starting}$
2:	**while** true **do**
3:	$a^r \leftarrow a^r_S$
4:	$a^s \leftarrow a^s_S$
5:	$Sys \leftarrow D_{a^r}$
6:	$C \leftarrow Re_{a^s}$ + Analytics Platform Information
8:	$PI \leftarrow DM_{t_S}$
9:	$I \leftarrow$ Prompt engineering Instructions
10:	$cw \leftarrow Sys + C + PI + I$
11:	Get response from agent $Re_{a^r} \leftarrow a^r(cw)$
12:	Find next state based on transition $S' \leftarrow (F_{t_s} \times\ Re_{a^r}) \times\ OM_{t_s}$
13:	**if** self-reflect **then**
14:	$Re_{a^s} \leftarrow Re_{a^r}$
15:	Jump to line 6
16:	**end if**
17:	**if** human in the loop **then**
18:	Ask for user permission
19:	**if** User Message is NOT "yes" **then**
20:	$Re_{a^r} \leftarrow$ *User Message*
21:	Jump to line 6
22:	**end if**
23:	**end if**
24:	Display messages $(S, D_{a^r},\ R_{a^s})$
25:	$a^s_S \leftarrow a^r_S$
26:	$Re_{a^s_S} \leftarrow Re_{a^r}$
27:	Update state $S \leftarrow S'$
28:	**end while**

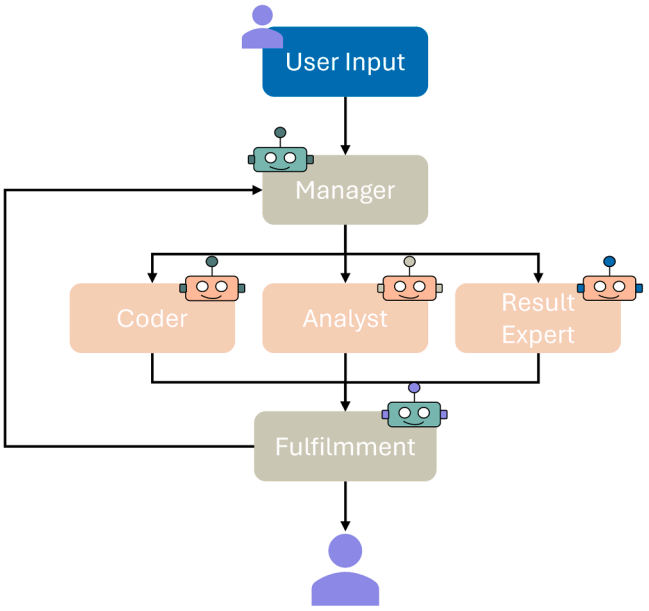


Fig. 3. Instantiation process.

Table 2
Available agent tools.

Tool	Description	Agent
Result Attributes	Provides information about the attributes names and types of the analysis results. Input: analysis goal	Analyst, Result Expert
Build Analysis	Executes an analysis. Input: analysis goal	Analyst
Algorithm Code	Provides the code used to generate the results of an analysis. Input: analysis goal	Coder
Algorithm Description	Provides a text description (docstring) of the algorithm used to generate the results of an analysis. Input: analysis goal	Coder
All analysis	Provides the information of all the analysis entities created.	Analyst
Intents	Provides a list of all the available analysis goals (intents).	Analyst, Coder, Result Expert
Analysis	Provides the details for a specific analysis entity. Input: analysis goal	Analyst
Model	Provides the details for a specific model entity. Input: analysis goal	Coder
Results	Provides the generated results of the analysis. It includes two filter functionalities based their attributes and dates. Input: analysis goal, attribute filters, date filters.	Result Expert
Prediction	Generates results for predictive analyses. Input: analysis goal, prediction specific inputs.	Result Expert

to utilise analytics functionalities and allows for better configuration of agents with targeted descriptions, tasks, and a limited subset of tools. Based on the manager's decision, the respective agent receives the initial user input and instructions on how to proceed. After using the tools and retrieving the necessary information, the three states transition to the final state, *Fulfilment*. In this state, the *Task Fulfilment* expert compares the initial user input with the output of the previous agent to determine whether the response adequately answers the user's input. If the response is deemed sufficient, it is formatted appropriately and displayed to the user. If not, the process transitions back to the *Manager* state, repeating the process cycle until the request is fulfilled. This feedback loop ensures refinement of outputs and correction of potential errors during the process. In addition, we note that each agent uses the self-reflect mechanism and is able to iterate multiple times to use various tools and reason.

4. Experiments

4.1. Experimental setup

The analytics platform is instantiated with data, analyses and models to support an industrial use case that evaluates and validates Electric Vehicle (EV) Li-Ion batteries. Technicians employ specialised equipment—referred to as chambers and battery testers—to conduct assessments of EV battery components. Each chamber transmits different sensor data including temperature, humidity, and electric power consumption. This data is used to create tests for the customers as well as several safety mechanisms that prevent dangerous situations to be occurred (e.g. fire hazards). The battery testers are devices that provide sink and source for high voltage battery packs. The output of these testers is forwarded into the test volume of the chambers, where the Device Under Test (DUT) undergo evaluation. They are managed via Programmable Logic Controllers (PLCs). The readings from various buttons and sensors (maintenance and emergency button, CO2/H2/O2 concentration, and others) are constantly available to the PLC, that uses them to perform certain actions and ensure safety.

Each machine at any moment has a status or "defconlevel" (e.g. Ready, Run, Alarm, Maintenance) that defines its state. These statuses persist over long periods and have a direct impact on machine management and personnel safety. For instance, Maintenance indicates that the machine door is open and a human presence inside or nearby, restricting usage, while an Alarm status signals a malfunction that can compromise personnel safety. For this reason, timely access to insights regarding machines is essential for technicians to efficiently manage machine usage, prevent accidents, and detect recurrent patterns. In addition to the defconlevel, the platform ingests a variety of sensor measurements (e.g. machine names, temperature, humidity) recorded periodically and timestamped. In this setting, the analytics platform supports technicians in the shop floor, delivering five descriptive analyses that group and aggregate the status durations of instruments and one predictive analysis that forecasts the chamber status, given their name. A detailed description of these analyses is provided in Table 3.

Table 3
Configured analyses for the use case. The name shortly describes the analysis. The intent (equivalent to the analysis goal) is used as a unique human-readable identifier, describing its goal.

Analysis Name	Intent	Description
Group by Defconlevel	GROUP_BY_DEFCONLEVEL	It groups and counts the recorded statuses from the Chambers per day.
Fault Intervals of Defconlevel	FAULT_INTERVALS_DEFCONLEVEL	The duration of the flagged fault statuses are calculated.
Duration of Defconlevel	DURATION_OF_CHMABER_DEFCONLEVEL	The duration of all the chamber statuses are calculated.
Percentages of Defonclevel	DEFCONLEVEL_PERCENTAGES	The percentages of all the chamber statuses are calculated inside a certain time frame, in our case a day.
Duration of Emergency	DURATION_OF_TESTER_EMERGENCY	The duration of the flagged emergencies are calculated.
Prediction of next chamber status	DEFONLEVEL_PREDICTOR	Predictive analysis that forecasts the succeeding status of a given chamber.

For the above analyses, the tools could provide their analysis entity, model entity, the algorithm used, and their results or predictions. The analysis and model entities provide details about the respective inputs, outputs, configurations and execution dates.

To evaluate the proposed framework, we compare our approach with two baseline architectures referred to as LLM and Agent. Both had access to all the tools and some generic instructions as prompts for the model. The LLM architecture implemented a single inference request by the model, while the Agent implemented a single agent with self-reflect and the ability of multiple inferences. These distinctions enable the comparison of our approach with naive GenAI systems that either prompt an LLM or have a single agent. To broaden our experimentation, we used two different LLMs, and each case was executed three times to get more representative results. Moreover, we design a total of sixteen cases as experiments. Each case consisted of one question by the user that required information from the analytics platform. The questions were formed to cover all the available functionalities (tools) and analyses of the platform. They vary in terms of complexity, from requesting the intents to retrieving descriptive results based on specific filters. For evaluation purposes, each question was accompanied by an example response (based on the analytics) and the tool that could provide the answer. Combining all the above parameters, a total of 288 experiments were conducted.

To evaluate the outcomes of the three architectures, we defaulted to the LLM-as-a-judge paradigm, where a stronger model is used to judge the output of other models based on some rules and instructions. Using an LLM to evaluate provides scalability, explainability (Zheng et al., 2023), and adaptation to content formats and domain-specific knowledge (Gu et al., 2025). These advantages enabled the evaluation of our experiments, because even though the cases included an example of the expected response, the final outcome of the architectures could vary in terms of format while preserving the details from the analytics. In total, we used five metrics to calculate the results of our experiments provided by the Deepeval[2] evaluation framework, as shown in Table 4**.** The first three metrics use the G-EVAL methodology (Liu et al., 2023), while the rest follow a similar pattern for agent-specific tasks. The output of each metric is the measurement normalised between 0 and 1, with a short reason given by the judge why that value was chosen. Lastly, as the judge, we used the Deepseek-r1 model.

The implementation of the MCP server, the three architectures, and the evaluation were created using the Python programming language with the libraries of LlamaIndex, CrewAI, and Deepeval. The LLM models were hosted using the Ollama tool and specifically used the quantised version of mistral-small:24b (referred to as Mistral), qwen2.5:32b (referred to as Qwen), and deepseek-r1:32b (referred to as Deepseek-r1). The machine used to run the experiments had two RTX 4090 graphics cards with a combined vRAM of 48GB. The models required approximately ~15GB, ~21GB and ~21GB of vRAM each, respectively, and were able to infer in near real time.

Table 4
Metrics used for the evaluation.

Metric	Description
Correctness	Correctness of generated response. Measures how closely the actual output aligns with the expected output.
Coherence	Examines the consistency and clarity of the output, measuring if the style is uniform and if it can be easily understood.
Professionalism	Assesses the level of professionalism and expertise conveyed
Task Completion	Evaluates the effectiveness of an agent to accomplish a task.
Tool Correctness	Measures if the expected tools were called. The metrics values are calculated as the division between the correctly used tools and the total number of tools used.

4.2. Results

Table 5 presents the aggregated results for each architecture and model combination. The best Overall score is achieved by the Flow architecture with the Qwen model, with a score of 0.86 and with a mean difference of 7.2 % compared to the other approaches. The Flow architecture has shown improvement in the Correctness and Tool Correctness metrics, while performing slightly worse than the others. In the Coherence and Professionalism metrics, all combinations achieve high results with small differences. The variation between the results is presented in more detail in Fig. 4 and the calculated coefficient of variation (CV). For the metrics of Coherence and Professionalism, all combinations achieve low variation in their responses. Concerning the architectures, our method indicates greater consistency across the metrics for both models. In contrast, while the Qwen model is consistent in all architectures, the Mistral model indicates obvious variability in the LLM architecture.

From the spider plots in Fig. 5, we observe the differences between the architectures and the models. In Fig. 5**a** our proposed method outperformed the other two architectures in the metrics of Task Completion, Correctness and Tool Correctness. For the other two metrics, Professionalism and Coherence, the proposed method achieved similar performance compared with the baselines. In Fig. 5**b**, the Qwen model outperforms the Mistral model in all metrics and especially the Correctness and Tool Correctness. Comparing the differences observed between the models and the scores of the proposed method in Table 4, it is evident that having a more sophisticated architecture (i.e. Agent, Flow) can overcome model limitations. This can be further observed in Fig. 6, which depicts the percentage improvements between the Qwen and Mistral models. In the LLM architecture, which is more demanding for the model, the large Qwen model dominates the Mistral model except in the metric of Coherence. Moreover, in the other two architectures, the relative improvement percentage decreases from 23.7 % to less than 4 %.

In addition to performance metrics, we recorded the execution times, agent executions and tool calls, for each experiment to compare time and resource costs across the architectures and models. Table 6 presents the details of execution times, agent execution and tool calls per architecture-model combination. As expected, the LMM architecture exhibits the lowest mean execution times and variability, since the model is inferred once. For the Qwen model, mean and median execution times for the LLM and Agent architectures are nearly identical, with the Agent variant showing a slight smaller median deviation. The Flow architecture however shows a noticeable increase in execution time compared to other architectures, which aligns with the expected scaling cost as architecture complexity increases. Specifically, for the Qwen model, this increase was approximately 2.5 times for the mean and 2 times for the median execution times. For the Mistral model the increase is around 3 and 1.5 times, respectively. These findings indicate a trade-off between computational costs across architectures, and between robustness and speed across models. When comparing execution times between models, Mistral is faster for the LLM architecture but has higher mean execution times for the two others, particularly for the Flow. Although median values suggest that, on average, the architectures deliver comparable execution times, the discrepancy between means and medians—alongside substantial standard deviations—indicates the presence of outliers and performance instability across the tested scenarios.

To identify potential outliers in the Flow architecture, we applied the Iner Quartile Range (IQR) method. The IQR is computed as:

$$IQR = Q3 - Q1$$

[2] https://deepeval.com

Table 5
Results per architecture and model types as aggregated values for the cases. Data are presented as mean ± standard deviation. The Overall score is calculated as the mean ± standard deviation across all metrics. Bold values indicate the best-performing combination for each metric.

Architecture	Model	Correctness	Coherence	Professionalism	Task Completion	Tool Correctness	Overall
LLM	Mistral	0.27 ± 0.34	0.95 ± 0.06	0.96 ± 0.07	0.59 ± 0.32	0.52 ± 0.44	0.66 ± 0.25
LLM	Qwen	0.61 ± 0.33	0.90 ± 0.1	**0.98 ± 0.04**	**0.80 ± 0.22**	0.79 ± 0.38	0.82 ± 0.22
Agent	Mistral	0.61 ± 0.36	0.86 ± 0.12	0.94 ± 0.13	**0.80 ± 0.18**	0.79 ± 0.36	0.80 ± 0.23
Agent	Qwen	0.68 ± 0.28	**0.91 ± 0.08**	**0.98 ± 0.05**	0.78 ± 0.19	0.79 ± 0.38	0.83 ± 0.20
Flow	Mistral	0.70 ± 0.37	0.88 ± 0.08	0.95 ± 0.17	0.75 ± 0.24	0.88 ± 0.30	0.83 ± 0.23
Flow	Qwen	**0.73 ± 0.28**	0.90 ± 0.09	**0.98 ± 0.06**	0.78 ± 0.24	**0.92 ± 0.15**	**0.86 ± 0.16**

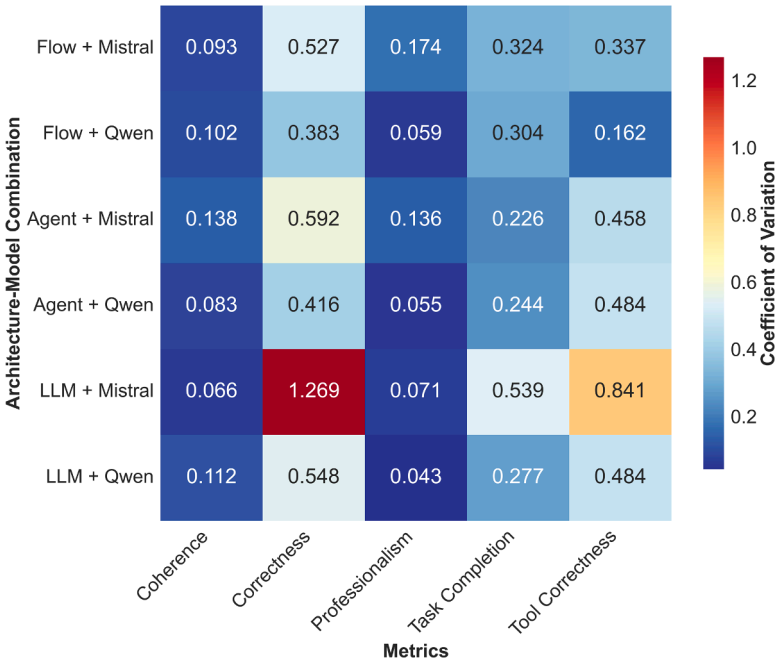
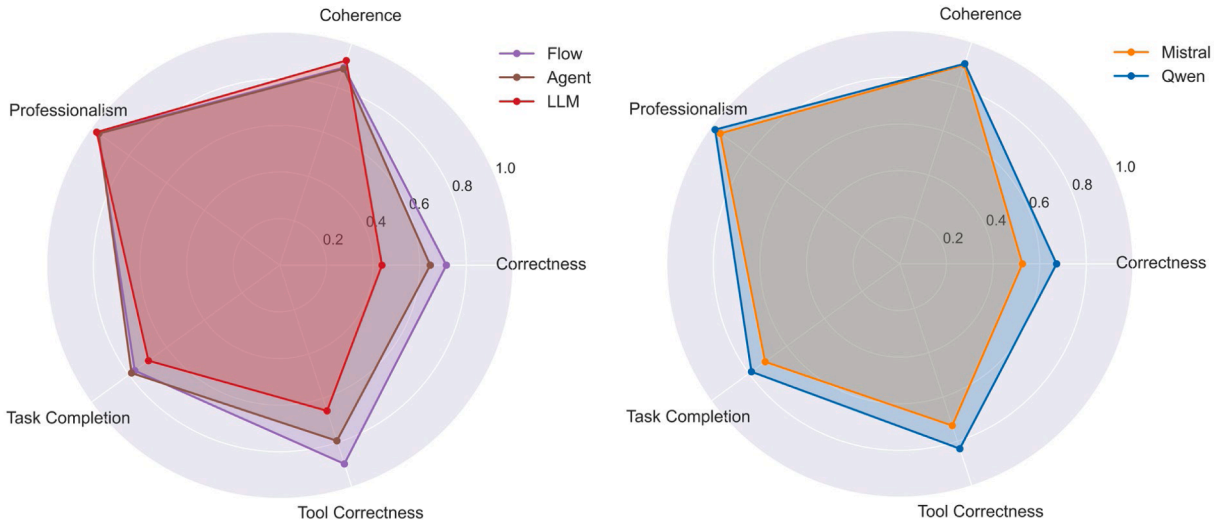


Fig. 4. Heatmap of the coefficient variant between architecture and model pairs per metric.



(a) Architecture performance across the metrics (b) Model performance across the metrics

Fig. 5. Spider plots across the evaluation metrics for the (a) architectures and (b) models. Values are calculated as the mean score of the models per architecture and the mean score of the architectures per model, respectively.

Where Q3 is the 75th percentile (upper quartile) and Q1 is the 25th percentile (lower quartile). An execution time was flagged as an outlier if it fell outside the interval $[Q1 - 1.5 \times IQR,\ Q3 + 1.5 \times IQR]$. Using this criterion, six case-model combinations were identified as outliers: two



Fig. 6. Heatmap of the improvement percentage between the two models, for the different architectures and metrics. Improvement percentages show the relative performance gain when upgrading from Mistral to Qwen, calculated as ((Qwen_Score - Mistral_Score) / Mistral_Score) × 100.

Table 6
Execution times, agents executed and tool calls per architecture and model. For the execution times, data are presented as the mean and median calculation across the experiment cases and seeds ± standard deviation in seconds. For the number of agent executions and tool calls are presented as the mean ± standard deviation.

Architecture	Model	Mean Execution time (seconds)	Median Execution time (seconds)	Agents Executed	Tool calls
LLM	Mistral	10.63 ± 5.00	8.64 ± 5.84	-	1.81 ± 0.64
LLM	Qwen	18.16 ± 8.71	14.79 ± 9.40	-	1.35 ± 0.70
Agent	Mistral	22.64 ± 21.16	19,19 ± 3.08	-	2.56 ± 1.43
Agent	Qwen	18.90 ± 11.67	14.68 ± 3.68	-	1.92 ± 1.22
Flow	Mistral	65.79 ± 84.51	28.50 ± 12.00	6.44 ± 7.19	9.00 ± 14.37
Flow	Qwen	46.35 ± 68.81	28.00 ± 14.00	4.63 ± 7.11	3.27 ± 7.07

for the Qwen model and four for the Mistral model, with a single case flagged for both models. The majority of the outliers were caused due to one seed requiring significantly longer time to execute than the other seeds. A systematic review of the experiment logs did not reveal a definitive root cause. Our hypothesis is that the non-deterministic behavior of the agents contributed to the extended runtimes, possibly due to: 1) case difficulty - certain cases required complex query construction resulting tool usage with missing or wrong parameters and 2) tool overuse - agents occasionally used multiple tools causing them to retrieved a large quantity of excessive context that increased their processing time. After excluding the identified outliers the recalculated mean scores for the Flow architecture were 30.72 ± 0.51 s for the Mistral and 32.76 ± 0.55 s for the Qwen. The updated values contradict our initial observation that the Qwen model was faster than Mistral for the Flow architecture by a slight margin.

Aligned with the observations in previous metrics, we can confirm the superiority of Qwen in terms of agent executions and tool usage. Across all architectures it makes fewer tool calls than Mistral and with less variability, especially for the flow architecture which also make fewer agent executions. It is evident that the Flow architecture requires more tool calls than the other architectures, because of its increased complexity and agent participation. If we exclude the identified outliers for the Flow architecture the agent execution and tool call counts for the Qwen model are 3.36 ± 0.98 and 1.75 ± 1.04, while for the Mistral are 3.50 ± 2.09 and 3.94 ± 5.93. Even though both improved significantly, again the Qwen model outperforms Mistral.

4.3. User study

To strengthen our evaluation, we conducted a comprehensive user study designed to gather feedback from human participants. Adhering to the experimental design outlined by Inkpen et al. (2023) and He et al. (2025), we implemented a dual-condition experiment involving two distinct tools: a manual Form-based and an agent-augmented chat, across various tasks. Furthermore, we consulted similar studies from Farhi et. al. (2023), Siregar et al. (2023), and Awasthi et. al. (2023) to guide the information to be collected from users. Before the trial, we collected demographic data such as gender, age, education and experience adjusted from (Farhi et. al. 2023) while during the experimental trial, quantitative and qualitative metrics were collected in each condition. To obtain quantitative metrics, we logged user actions to derive task execution time, task completion, and task output accuracy. For the qualitative metrics, we focused on usability, satisfaction, and confidence by adapting questions from Siregar et al. (2023) and Awasthi et al. (2023). Participants in the study were selected to include individuals unfamiliar with the specific use case but possessing either relevant academic qualifications or data analytics experience across varying levels of technical expertise.

4.3.1. Survey procedure

The experiment began with an introduction page that presented a short video tutorial, an overview of the use case, a description of the two tools, detailed step-by-step instructions, and a mandatory consent form that needed to be accepted by all participants. After the introduction, users were required to provide demographic information, including their age range, gender, educational qualifications, professional field or occupation, experience level, technical proficiency, familiarity with similar tools, and confidence in executing analytical tasks. Participants had the option of providing additional personal details, such as a name and email address.

Following this preparatory phase, participants were instructed to complete three tasks by addressing corresponding questions utilising both tools. For each question, they first used the form-based tool—implemented with drop-down lists and text inputs that

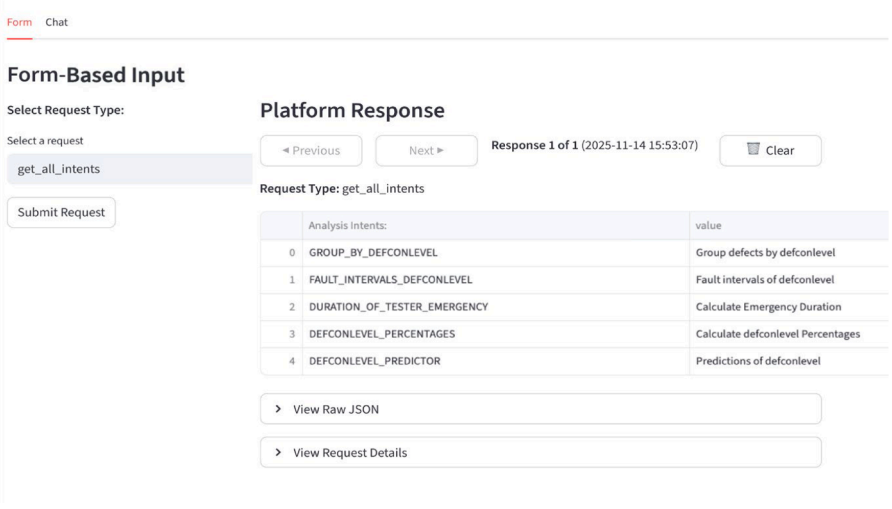
corresponded to the platform's available request parameters—to generate an answer. They subsequently used the chat interface, implementing the Flow architecture with the Qwen model, to obtain the same answer. Both tool interfaces are presented in Figure 7. The use of a structured form was intentional; offering a raw text input would have increased task difficulty by requiring participants to construct low-level requests manually. The questions required analyses result retrieval from the analytics platform. After each tool submission, a brief questionnaire captured usability, satisfaction, and confidence ratings. Upon completing all questions, users were required to fill out a final questionnaire capturing their overall impressions and feedback.

Quantitative metrics were derived from the logged actions, including task execution time, task completion status, and output accuracy. Qualitative data were collected through the post-task and final questionnaires, which assessed usability, satisfaction, confidence, and overall preference. All the questionnaires featured Likert scale statements (1 Strongly disagree – 5 Strongly agree), and the users completed a total of seven questionnaires. A complete list of all questionnaires' statements can be found in Table 7. The collected data enabled analysis of demographic distributions, completion rates per question and interface, questionnaire responses, user answers, session metrics (duration, number of actions, interaction patterns), and comparative preference analysis.

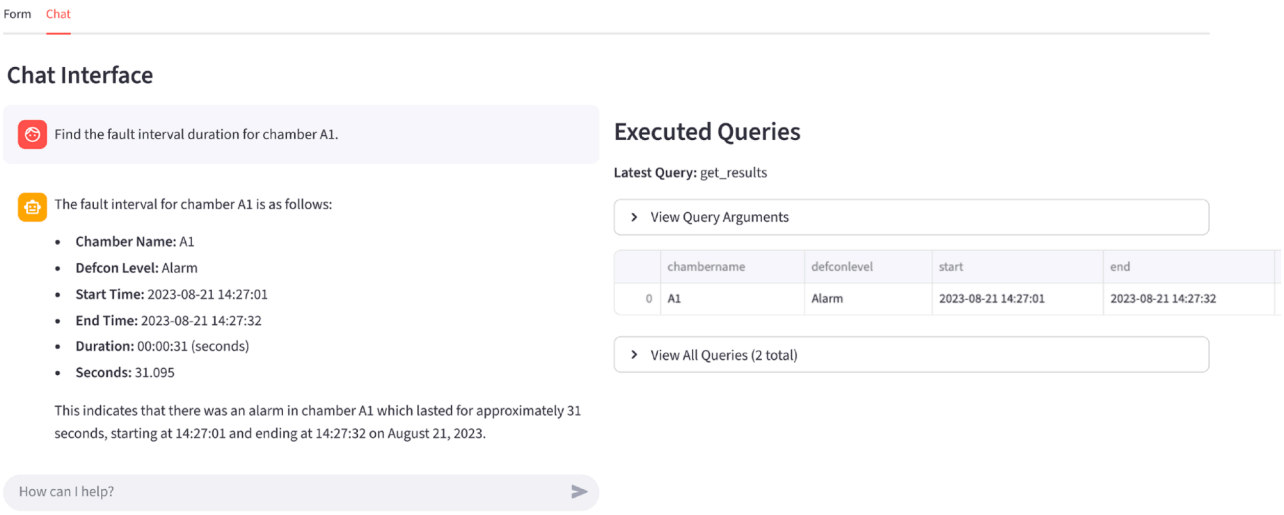
4.3.2. Survey results

This subsection presents the findings from the user study (N=11). Statistical significance was evaluated using the Mann-Whitney U test, suitable for the non-parametric nature and sample size of the survey. Statistically significant values are denoted with "*" where $p < 0.01$. As intended, participants possessed a high degree of technical and academic competence. The group was highly educated, with 9 individuals holding postgraduate degrees (4 Master's, 4 PhDs, 1 Professor) and 2 with a bachelor's degree. Additionally, the majority of the group was technically experienced, with four proficient coders and two advanced users working with APIs/scripts. Similarly, regarding data analysis experience, seven participants were classified as advanced or experts who perform analytical work regularly.

The quantitative results presented in Table 8 indicate that the Chat



(a) Form interface



(b) Chat interface

Fig. 7. Interface screenshots presented to users. (a) Form-based interface displaying input fields and the presentation of executed queries. (b) Chat-based interface showing user messages and agent responses alongside the queries executed by the agents.

Table 7
Survey Likert-scale questionnaires. The Form and Chat questionnaires were presented immediately after each survey question, followed by the Overall questionnaire at the end of the survey.

Questionnaires	Statements
Form	It was easy to complete the task using the form-based input.
	I was satisfied with the result produced via the form-based method.
	I found it easy to navigate and perform the task using the form-based input.
	I was able to complete the task in a reasonable amount of time using the form-based input.
Chat	The response generated by the chat interface was accurate.
	The response was helpful in completing the task.
	The response did not include incorrect or irrelevant information (hallucinations).
	The response from the chat interface was given in a timely manner.
	It was easy to use the chat interface to perform the task.
	I was satisfied with the result produced via the chat interface.
	I was confident in using the information provided by the chat interface.
Overall	I found the chat interface easier to use than the form-based method.
	I would prefer to use the chat interface rather than the form-based method for similar tasks.
	Lack of familiarity with the use case made the form-based method difficult to use.
	Lack of familiarity with the use case made the chat interface difficult to use.
	Lack of familiarity with the analytics platform made the form-based method difficult to use.
	Lack of familiarity with the analytics platform made the chat interface difficult to use.
	Overall, how would you rate the usability of the form based tool?
	Overall, how would you rate the usability of the chat interface tool?

Table 8
Comparison of objective performance metrics and subjective user perceptions (N=11). Arrows in the metric column indicate the desirable direction (↓ lower is better; ↑ higher is better). Values represent mean ± standard deviation (where available). Bold values indicate best performance with and asterisks indicate a statistically significant improvement over the Form interface (Mann-Whitney U test: $p < 0.01$). Note that Confidence and Hallucination absence metrics were only applicable to the Chat interface.

Metric	Form	Chat
Task duration (min) ↓	6.71 ± 6.53	**2.70 ± 5.25***
Actions (clicks/inputs) ↓	14.48 ± 9.93	**3.30 ± 0.85***
Overall Usability ↑	3.09 ± 1.04	**4.64 ± 0.67***
Ease of Use ↑	3.06 ± 1.20	**4.85 ± 0.36***
Satisfaction ↑	3.55 ± 1.18	**4.67 ± 0.69***
Perceived Timeliness ↑	3.27 ± 1.35	**4.06 ± 0.90**
Use case familiarity barrier ↓	4.10 ± 1.20	**1.60 ± 0.84***
Platform familiarity barrier ↓	3.70 ± 1.06	**1.40 ± 0.70***
Confidence ↑	-	**4.30 ± 1.07***
Hallucination absence ↑	-	**4.76 ± 0.76***

interface reduced cognitive load and execution time compared to the Form baseline. Across all tasks, the mean duration dropped from 6.71 ± 6.53 min in the Form to 2.70 ± 5.25 min in the Chat. For the three questions, the chat interface execution was approximately 5, 2, and 3 times faster, respectively. In addition, we tracked the actions executed for each interface. Actions were considered any inputs (clicks) from the participants and displayed results. For the Chat, the mean number of actions per question was 3.30 ± 0.85, while for the Form was 14.48 ± 9.93, indicating a 77.20 % reduction.

Regarding the participants' answers, high accuracy was achieved in both interfaces. Through the Chat, all responses were correct, while four incorrect responses were given through the Form interface and specifically to Question 2, which required more complex configurations. Furthermore, questionnaire responses also indicate the dominance of the chat interface over the Form. In terms of usability, ease of use, satisfaction and timeliness, the Chat interface was better by at least one point. One of the study's most important findings is the reduction of the "Familiarity Barrier". Questions in the overall questionnaire measured how the lack of knowledge about the Use Case or the Analytics Platform contributed to the task difficulty. The participants' responses showcase a considerable difference between the interfaces and particularly for the Use case, suggesting that the chat interface enacts as an abstraction layer, decoupling the ability to extract analyses results from requiring deep use case and platform knowledge. Moreover, despite the known challenge of LLM hallucinations, participants reported high confidence in the generated results and absence of hallucinations.

To complement the quantitative metrics, we analysed the open-ended feedback of participants at the end of the survey. Three primary themes were identified from their responses:

- **Accessibility.** Consistent with the Liker scale questionnaires, participants highlighted the chat interface's lower technical barriers. While the form interface was perceived as more demanding, requiring expertise, the chat was viewed as universally accessible. One participant wrote: *"I strongly believe that the chat interface is superior as it doesn't require any deep technical knowledge and can be used by everyone."*. This sentiment was expressed by others who described the chat interface as *"crystal clear even for a user with no technical background"*.
- **Explainability and Trust.** Participants responded positively to the survey's decision to display the agents' executed queries alongside the chat response. *"I liked that the chat interface shows the sequence of queries used … it is easy to understand the steps …. [to] get the result."*. However, some users also expressed for even deeper verification capabilities to mitigate trust issues. They suggested linking parts of the answer to documentation, more verbose explanations in the response, or *"re-run the queries dynamically to verify their content"*.
- **Learning curve and cognitive load.** Validating the quantitative findings, participants also commented on the learning curve and cognitive load of the interfaces. One user stated that *"As the survey went on, I was getting more familiar with the form, so I could answer the questions faster"*. Some participants noted that they used the Chat first, which helped them verify the results in the Form interface

5. Conclusions and future work

The enhancement of systems with AI and the augmentation of AI systems with external resources derives from the ability of LLMs to effectively use tools via function calling. Providing tools to LLMs (and LLM-based agents) can yield multiple benefits such as increased accuracy, domain-adaptation, knowledge expansion and robustness (Niketan and Batatia, 2025; Patil et al., n.d.; Qu et al., 2025), as well as the capacity to complete and automate long and complex tasks within workflows (Qiao et al., 2025; Shen et al., 2024). The above will play a crucial role towards autonomous data analytics, where AI agents are going to be utilised in order to automate the analytics lifecycle using relevant tools. With our work, we showcased how an existing analytics platform with traditional API functionalities can be transformed into a tool for LLMs. Leveraging existing standards such as the MCP, the platform can be integrated with compatible AI systems and gain access to analytic functionalities. The experiments conducted showed promising results between the integration of a standalone LLM and the analytics platform, especially for larger models such as the Qwen. More importantly, the proposed framework, which incorporates an agentic architecture, indicates superiority in terms of overall performance, consistency and model independence. Firstly, in the domain of analytics, the reliability of insights is crucial as it provides the basis of data-driven decisions, thus performance and consistency are desired properties. Secondly, model independence can allow wider adoption with the usage of smaller open-source models that require less computational power and limit

concerns about security and data privacy. Thirdly, the execution of the Flow required more execution time, agent execution and tool calls. Especially for the smaller model, Mistral, where its improved performance resulted in the increase of those metrics.

Another advantage of the proposed framework is the ability to split a monolithic workflow into several states that provide modularity and adaptability. It enabled us to create explicit workflows that touch analytics lifecycle steps, such as retrieval of entities and algorithms during designing and developing, execution of analyses during deployment and result retrieval during interpretation. Having multiple agents and tasks provides fine-grained control for adjustments and configurations to improve the agents' performance. Empirically, during the Flow implementation, we were able to pinpoint parts that underperformed, enabling us to easily adjust specific agents and tasks. For example, at first the manager agent did not have access to the tools, which decreased its ability to provide concrete instructions to the other agents. Adding to the prompt a short description of the tools improved the instructions and the performance of subsequent agents. This iterative refinement approach was also applied to detect and mitigate hallucination patterns that emerged during experimentation. While the agents performed adequately on general tasks (e.g., tool usage, information processing, and generating structured outputs), their performance occasionally deteriorated due to platform-specific requirements or inconsistencies. Two prominent examples of this issue were related to the intents and the result filtering mechanism. Within the analytics platform, the term "intent" is used interchangeably with "analytics goal". This inconsistency confused the agents, particularly when the user requested an intent, while the tools required as input the parameter of the analytics goal, resulting in tool usage errors. Similarly, the filtering mechanism posed challenges due to its complexity, as it required the specification of multiple attributes with distinct functionalities. For example, certain attributes were optional, whereas others demanded specific values depending on the analysis and data types. These challenges were addressed by enriching the agents' prompts with relevant context and refining the description of the tools to include additional information.

The results from our user study provide evidence that integrating LLM-based agents into the analytics platform through a chat interface significantly enhances the operational efficiency and user accessibility. Quantitative analysis revealed that the chat interface, compared to the form interface, reduced task completion times and interaction costs. Crucially, the participants expressed their preference towards the chat interface indicating that the lack of use case and analytics platform familiarity was not a barrier. These findings suggest that natural language interfaces do not just simplify interaction but could shift the analytics workflow from manual query construction to high-level question answering.

Lastly, both the transformation of analytics functionalities into MCP tools and the flexibility in agentic workflow designing contribute to the generalizability of the proposed framework. The former provides interoperability with any services that support the MCP protocol, allowing agents to easily access and utilise tools from other sources. For example, the current analytics platform could be replaced with another that supports MCP and exposes the same (or similar) services and still the Flow could run in the same manner. The latter allows the configuration of workflows to meet platform or case specific requirements, facilitating adoption. For instance, additional states and agents could be configured to address additional functionalities, while agents' context can be adapted to incorporate specific instructions. Thus, giving the proposed framework the possibility to be platform and use-case-agnostic.

5.1. Limitations

Platform functionalities and process complexity. Even though the proposed framework could implement a variety of complex workflows, the tested architecture can be seen as simplistic. While it shows some useful agentic patterns like communication between the agents, a manager that instructs other agents, and a final step to evaluate the response, it follows a straightforward execution with only three expert agents. The available functionalities exposed by the analytics platform are a significant factor for the design of such architectures. Currently, the platform mainly provides tools to extract information about analytics (i.e. results, predictions, entities) with only one actionable functionality, which is to build an analysis. Expanding the exposed functionalities can greatly increase the agents' abilities and enable the implementation of a more complex architecture.

Scalability. Our experiments showed that the execution of the agentic architecture came at a cost of both time and resources. This finding may result in noticeable drawbacks when scaling the workflow to more states and agents. Another aspect we did not consider in our current work is related to scalability and concurrent multi-user accessibility. Applying the proposed framework into large scale example with simultaneous users may result in unseen challenges. Developments in the field of AI, such as the MCP or improvements to the size and capabilities of the models (Belck et al., 2025) and routing (Ong et al., 2024) could be used to resolve scalability issues.

Evaluation. For the evaluation we relied to one library, Deepeval, which provides a limited spectrum of metrics, that mainly use the LLM-as-a-judge approach. Expanding the evaluation of the proposed framework with additional benchmarks such as WorfBench (Qiao et al., 2025) or TaskBench (Shen et al., 2024) can provide more comprehensive results. In addition, we did not consider the evaluation of our work in terms of AI trustworthiness, which has become an important aspect in the field of AI. Previously, we noted that our approach could contribute to security and privacy concerns, but we did not evaluate or examine these aspects in our current work. While the reasoning that with our framework open-source models to increase security and data privacy is logical, it needs to be reviewed systematically. In addition, these perspectives must also be examined through the tool wrapping and protocol used, as recent studies have shown safety concerns (Jing et al., 2025; Radosevich and Halloran, 2025).

5.2. Future work

For the future, our goal is to significantly expand and improve the proposed framework. Firstly, we aim to increase the functionalities of the analytics platform to enable the agents to contribute to the whole analytics lifecycle with the ability to take actions and produce analytics. This will also enable us to design more complex workflows to better test and adapt our approach. Secondly, we plan to incorporate more GenAI models, which are also multimodal, as we believe that with voice or vision the abilities of the agents will increase. Lastly, regarding evaluation we would like to enlarge our experiments with additional cases, AI models and metrics or benchmarks; and evaluate our approach to real-world scenarios and conduct further usability studies.

CRediT authorship contribution statement

Mattheos Fikardos: Conceptualization, Methodology, Software, Investigation, Writing – original draft. **Katerina Lepenioti:** Conceptualization, Methodology, Writing – original draft. **Alexandros Bousdekis:** Conceptualization, Writing – original draft. **Dimitris Apostolou:** Conceptualization, Writing – review & editing, Supervision. **Gregoris Mentzas:** Conceptualization, Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was funded by the European Union's Horizon 2020 project WASABI "White-label shop for digital intelligent assistance and human-AI collaboration in manufacturing" (Grant agreement No 101092176). The work presented here reflects only the authors' view and the European Commission is not responsible for any use that may be made of the information it contains.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.iswa.2026.200626.

Data availability

The data that has been used is confidential.

References

Anthropic, 2024. Model context protocol by anthropic, URL https://www.anthropic.com/news/model-context-protocol. Accessed July 22, 2025.

Awasthi, R., Mishra, S., Mahapatra, D., Khanna, A., Maheshwari, K., Cywinski, J., & Mathur, P. (2023). *HumanELY: Human evaluation of LLM yield, using a novel web-based evaluation tool*. MedRXIV, 2023-12.

Belcak, P., Heinrich, G., Diao, S., Fu, Y., Dong, X., Muralidharan, S., & Molchanov, P. (2025). Small language models are the future of agentic AI. arXiv preprint arXiv: 2506.02153.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., & Amodei, D. (2020). Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in Neural Information Processing Systems* (pp. 1877–1901). Curran Associates, Inc.

Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y.T., Li, Y., Lundberg, S., Nori, H., Palangi, H., Ribeiro, M.T., Zhang, Y., 2023. Sparks of artificial general intelligence: Early experiments with GPT-4. *arXiv preprint arXiv: 2303.12712*.

Cai, T., Wang, X., Ma, T., Chen, X., Zhou, D., 2023. Large language models as tool makers. *arXiv preprint arXiv:2305.17126*.

Cao, L. (2018). Data science: A comprehensive overview. *ACM Computing Surveys, 50*, 1–42. https://doi.org/10.1145/3076253

Chen, G., Dong, S., Shu, Y., Zhang, G., Sesay, J., Karlsson, B.F., Fu, J., Shi, Y., 2023. AutoAgents: A framework for automatic agent generation. *arXiv preprint arXiv: 2309.17288*.

Chopra, B., Singha, A., Fariha, A., Gulwani, S., Parnin, C., Tiwari, A., Henley, A.Z., 2023. Conversational challenges in AI-powered data science: Obstacles, needs, and design opportunities. https://doi.org/10.48550/ARXIV.2310.16164.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A. M., Pillai, T. S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Diaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S., & Fiedel, N. (2022). PaLM: Scaling language modeling with pathways. *Journal of Machine Learning Research, 24*(240), 1–113.

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., Webson, A., Gu, S. S., Dai, Z., Suzgun, M., Chen, X., Chowdhery, A., Castro-Ros, A., Pellat, M., Robinson, K., Valter, D., Narang, S., Mishra, G., Yu, A., Zhao, V., Huang, Y., Dai, A., Yu, H., Petrov, S., Chi, E. H., Dean, J., Devlin, J., Roberts, A., Zhou, D., Le, Q. V., & Wei, J. (2024). Scaling instruction-finetuned language models. *Journal of Machine Learning Research, 25*(70), 1–53.

CrewAI Inc., 2025. CrewAI. *[Computer software]* https://github.com/crewAIInc/crewAI.

Databricks, 2023. Introducing Databricks Assistant, a context-aware AI assistant. URL https://www.databricks.com/blog/introducing-databricks-assistant. Accessed July 22, 2025.

DataChat Inc, 2025. DataChat. URL https://datachat.ai. Accessed July 22, 2025.

De Bie, T., De Raedt, L., Hernández-Orallo, J., Hoos, H.H., Smyth, P., Williams, C.K.I., 2021. Automating data science: Prospects and challenges. 65 *(3)*, 76–87 https://doi.org/10.48550/ARXIV.2105.05699.

Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2023). QLoRA: Efficient finetuning of quantized LLMs. *Advances in neural information processing systems, 36*, 10088–10115.

Diaz-de-Arcaya, J., Torre-Bastida, A. I., Zárate, G., Miñón, R., & Almeida, A. (2024). A joint study of the challenges, opportunities, and roadmap of MLOps and AIOps: A systematic survey. *ACM Computing Surveys, 56*, 1–30. https://doi.org/10.1145/3625289

Ding, J., Ma, S., Dong, L., Zhang, X., Huang, S., Wang, W., Zheng, N., Wei, F., 2023. LongNet: Scaling transformers to 1,000,000,000 Tokens. *arXiv preprint arXiv: 2307.02486*.

Ehtesham, A., Singh, A., Gupta, G.K., Kumar, S., 2025. A survey of agent interoperability protocols: Model Context Protocol (MCP), Agent Communication Protocol (ACP), Agent-to-Agent Protocol (A2A), and Agent Network Protocol (ANP). https://doi.org/10.48550/arXiv.2505.02279.

Farhi, F., Jeljeli, R., Aburezeq, I., Dweikat, F. F., Al-shami, S. A., & Slamene, R. (2023). Analyzing the students' views, concerns, and perceived ethics about chat GPT usage. *Computers and Education: Artificial Intelligence, 5*, Article 100180.

Gao, L., Madaan, A., Zhou, S., Alon, U., Liu, P., Yang, Y., Callan, J., & Neubig, G. (2023). PAL: Program-aided language models. In *International Conference on Machine Learning* (pp. 10764–10799). PMLR.

Goertzel, B., & Pennachin, C. (2007). *Artificial General Intelligence, Cognitive Technologies*. Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-68677-4

Gu, J., Jiang, X., Shi, Z., Tan, H., Zhai, X., Xu, C., Li, W., Shen, Y., Ma, S., Liu, H., Wang, S., Zhang, K., Wang, Y., Gao, W., Ni, L., Guo, J., 2025. A survey on LLM-as-a-judge. arXiv preprint arXiv:2411.15594.

Hadi, M. U., Tashi, Q. A., Qureshi, R., Shah, A., Muneer, A., Irfan, M., Zafar, A., Shaikh, M. B., Akhtar, N., Wu, J., & Mirjalili, S. (2023). Large language models: A comprehensive survey of its applications, challenges, limitations, and future prospects. *Authorea preprints, 1*(3), 1–26. https://doi.org/10.36227/techrxiv.23589741.v4, 2023.

He, V.F., Li, S., Puranam, P., & Lin, F. (2025). Tool or Tutor? Experimental evidence from AI deployment in cancer diagnosis. arXiv preprint arXiv:2502.16411.

Hollmann, N., Müller, S., & Hutter, F. (2023). Large language models for automated data science: Introducing CAAFE for context-aware automated feature engineering. *Advances in Neural Information Processing Systems, 36*, 44753–44775.

Hou, X., Zhao, Y., Liu, Y., Yang, Z., Wang, K., Li, L., Luo, X., Lo, D., Grundy, J., & Wang, H. (2023). Large language models for software engineering: A systematic literature review. *ACM Transactions on Software Engineering and Methodology, 33*(8), 1–79.

Hsieh, C.-Y., Chen, S.-A., Li, C.-L., Fujii, Y., Ratner, A., Lee, C.-Y., Krishna, R., Pfister, T., 2023. Tool documentation enables zero-shot tool-usage with large language models. arXiv preprint arXiv:2308.00675.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2021). LoRA: Low-Rank adaptation of large language models. *ICLR, 1*(2), 3.

Hugging Face, 2024. Transformers agents. URL: https://huggingface.co/blog/agents. Accessed July 22, 2025.

Inkpen, K., Chappidi, S., Mallari, K., Nushi, B., Ramesh, D., Michelucci, P., & Quinn, G. (2023). Advancing human-AI complementarity: The impact of user expertise and algorithmic tuning on joint decision making. *ACM Transactions on Computer-Human Interaction, 30*(5), 1–29.

Jing, H., Li, H., Hu, W., Hu, Q., Xu, H., Chu, T., Hu, P., Song, Y., 2025. MCIP: Protecting MCP safety via model contextual integrity protocol. arXiv preprint arXiv: 2505.14590.

John, R. J. L., Bacon, D., Chen, J., Ramesh, U., Li, J., Das, D., Claus, R., Kendall, A., & Patel, J. M. (2023). DataChat: An intuitive and collaborative data analytics platform. In *Companion of the 2023 International Conference on Management of Data. Presented at the SIGMOD/PODS '23: International Conference on Management of Data* (pp. 203–215). Seattle WA USA: ACM. https://doi.org/10.1145/3555041.3589678.

Kaddour, J., Harris, J., Mozes, M., Bradley, H., Raileanu, R., McHardy, R., 2023. Challenges and applications of large language models. arXiv preprint arXiv: 2307.10169.

Kasneci, E., Sessler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., Krusche, S., Kutyniok, G., Michaeli, T., Nerdel, C., Pfeffer, J., Poquet, O., Sailer, M., Schmidt, A., Seidel, T., Stadler, M., Weller, J., Kuhn, J., & Kasneci, G. (2023). ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and Individual Differences, 103*, Article 102274. https://doi.org/10.1016/j.lindif.2023.102274

Kotaru, M. (2023). Adapting foundation models for operator data analytics. In *Proceedings of the 22nd ACM Workshop on Hot Topics in Networks. Presented at the HotNets '23: The 22nd ACM Workshop on Hot Topics in Networks* (pp. 172–179). Cambridge MA USA: ACM. https://doi.org/10.1145/3626111.3628191.

LangChain, 2025. LangChain agents. [Computer software] https://python.langchain.com/docs/tutorials/agents/.

Lepenioti, K., Fikardos, M., Bousdekis, A., Apostolou, D., Mentzas, G., n.d. Autonomous analytics as a service: From vision to action., in: Enterprise interoperability XI: Enterprise interoperability through data, AI, and robotics. Proceedings, accepted.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-Augmented generation for knowledge-intensive NLP tasks. *Advances in neural information processing systems, 33*, 9459–9474.

Li, H., Jiang, H., Zhang, T., Yu, Z., Yin, A., Cheng, H., Fu, S., Zhang, Y., He, W., 2023. TrainerAgent: Customizable and efficient model training through LLM-powered multi-agent system. arXiv preprint arXiv:2311.06622.

Liu, H., Tam, D., Muqeeth, M., Mohta, J., Huang, T., Bansal, M., & Raffel, C. (2022). Few-Shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems, 35*, 1950–1965.

Liu, N.F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., Liang, P., 2023. Lost in the middle: How language models use long contexts. arXiv preprint arXiv: 2307.03172.

Liu, Y., Iter, D., Xu, Y., Wang, S., Xu, R., Zhu, C., 2023. G-Eval: NLG evaluation using GPT-4 with better human alignment. arXiv preprint arXiv:2303.16634, 12.

Maynez, J., Narayan, S., Bohnet, B., McDonald, R., 2020. On faithfulness and factuality in abstractive summarization. arXiv preprint arXiv:2005.00661.

Microsoft, 2023. Data Wrangler. URL: https://devblogs.microsoft.com/python/data-wrangler-release/. Accessed July 22, 2025.

Moncada-Ramirez, J., Matez-Bandera, J. L., Gonzalez-Jimenez, J., & Ruiz-Sarmiento, J. R. (2025). Agentic workflows for improving large language model reasoning in robotic object-centered planning. *Robotics, 14*(3), 24.

Morris, M.R., Sohl-dickstein, J., Fiedel, N., Warkentin, T., Dafoe, A., Faust, A., Farabet, C., Legg, S., 2023. Levels of AGI: Operationalizing progress on the path to AGI. arXiv preprint arXiv:2311.02462.

Mukherjee, S., Mitra, A., Jawahar, G., Agarwal, S., Palangi, H., Awadallah, A., 2023. Orca: Progressive learning from complex explanation traces of GPT-4. arXiv preprint arXiv:2306.02707.

Niketan, N., & Batatia, H. (2025). Integrating external tools with large language models to improve accuracy. In *International Conference on Information Technology and Applications* (pp. 409–421). Singapore: Springer Nature Singapore. https://doi.org/10.1007/978-981-96-1758-6_34.

Ong, I., Almahairi, A., Wu, V., Chiang, W.L., Wu, T., Gonzalez, J. E., & Stoica, I. (2024). Routellm: Learning to route llms with preference data. arXiv preprint arXiv: 2406.18665.

OpenAI, 2023. Introducing ChatGPT enterprise URL https://openai.com/blog/introducing-chatgpt-enterprise Accessed July 22, 2025.

OpenAI, 2024. Function calling and other API updates. URL https://openai.com/index/function-calling-and-other-api-updates/Accessed July 22, 2025.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., & Schulman, J. (2022). Training language models to follow instructions with human feedback. *Advances in neural information processing systems, 35*, 27730–27744.

Packer, C., Fang, V., Patil, S.G., Lin, K., Wooders, S., Gonzalez, J.E., 2023. MemGPT: Towards LLMs as operating systems. https://doi.org/10.48550/ARXIV.2310.08560.

Pan, S., Luo, L., Wang, Y., Chen, C., Wang, J., & Wu, X. (2023). Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering, 36*(7), 3580–3599.

Patil, S.G., Mao, H., Yan, F., Ji, C.C.-J., Suresh, V., Stoica, I., Gonzalez, J.E., n.d. The Berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In Forty-second International Conference on Machine Learning.

Peng, B., Li, C., He, P., Galley, M., Gao, J., 2023. Instruction tuning with GPT-4. arXiv preprint arXiv:2304.03277.

Polymer Search, 2024. Polymer. URL: https://www.polymersearch.com. Accessed July 22, 2025.

Project Jupyter, 2025. Jupyter AI. https://jupyter-ai.readthedocs.io/en/latest/ Accessed July 22, 2025.

Qiao, S., Fang, R., Qiu, Z., Wang, X., Zhang, N., Jiang, Y., Xie, P., Huang, F., Chen, H., 2025. Benchmarking agentic workflow generation. arXiv preprint arXiv: 2410.07869.

Qlik, 2025. Qlik sense. URL: https://www.qlik.com/us/products/qlik-sense Accessed July 22, 2025.

Qu, C., Dai, S., Wei, X., Cai, H., Wang, S., Yin, D., Xu, J., & Wen, J.-R. (2025). Tool learning with large language models: A survey. *Frontiers of Computer Science, 19*(8), Article 198343. https://doi.org/10.1007/s11704-024-40678-2

Radosevich, B., Halloran, J., 2025. MCP safety audit: LLMs with the model context protocol allow major security exploits. arXiv preprint arXiv:2504.03767.

Ray, P. P. (2023). ChatGPT: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope. *Internet of Things and Cyber-Physical Systems, 3*, 121–154. https://doi.org/10.1016/j.iotcps.2023.04.003

Salesforce, 2025. Tableau AI. URL: https://www.tableau.com/products/artificial-intelligence Accessed July 22, 2025.

Schäfer, M., Nadi, S., Eghbali, A., & Tip, F. (2023). An empirical evaluation of using large language models for automated unit test generation. *IEEE Transactions on Software Engineering, 50*(1), 85–105.

Shen, Y., Song, K., Tan, X., Zhang, W., Ren, K., Yuan, S., Lu, W., Li, D., & Zhuang, Y. (2024). TaskBench: Benchmarking large language models for task automation. *Advances in Neural Information Processing Systems, 37*, 4540–4574.

Significant Gravitas, 2023. AutoGPT. *[Computer software]* https://github.com/Significant-Gravitas/AutoGPT.

Siregar, F. H., Hasmayni, B., & Lubis, A. H. (2023). The analysis of Chat GPT usage impact on learning motivation among scout students. *International Journal of Research and Review, 10*(7), 632–638.

Sun, K., Xu, Y. E., Zha, H., Liu, Y., & Dong, X. L. (2023). Head-to-Tail: How knowledgeable are large language models (LLM)? A.K.A. *Will LLMs Replace Knowledge Graphs?*. arXiv preprint arXiv:2308.10168.

Thirunavukarasu, A. J., Ting, D. S. J., Elangovan, K., Gutierrez, L., Tan, T. F., & Ting, D. S. W. (2023). Large language models in medicine. *Nature medicine, 29*(8), 1930–1940. https://doi.org/10.1038/s41591-023-02448-8

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., & Zhou, D. (2022). Chain-of-Thought prompting elicits reasoning in large language models. *Advances in neural information processing systems, 35*, 24824–24837.

White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., Schmidt, D.C., 2023. A prompt pattern catalog to enhance prompt engineering with ChatGPT. arXiv preprint arXiv:2302.11382.

Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., Jiang, L., Zhang, X., Zhang, S., Liu, J., Awadallah, A. H., White, R. W., Burger, D., & Wang, C. (2023). AutoGen: Enabling Next-Gen LLM applications via multi-agent conversation. In *First Conference on Language Modeling*.

Xi, Z., Chen, W., Guo, X., He, W., Ding, Y., Hong, B., Zhang, M., Wang, J., Jin, S., Zhou, E., Zheng, R., Fan, X., Wang, X., Xiong, L., Zhou, Y., Wang, W., Jiang, C., Zou, Y., Liu, X., Yin, Z., Dou, S., Weng, R., Cheng, W., Zhang, Q., Qin, W., Zheng, Y., Qiu, X., Huang, X., & Gui, T. (2023). The rise and potential of large language model based agents: A survey. *Science China Information Sciences, 68*(2), Article 121101.

Yan, B., Zhou, Z., Zhang, L., Zhang, L., Zhou, Z., Miao, D., Li, Z., Li, C., Zhang, X., 2025. Beyond self-talk: A communication-centric survey of LLM-based multi-agent systems. arXiv preprint arXiv:2502.14321.

Yang, Y., Chai, H., Song, Y., Qi, S., Wen, M., Li, N., Liao, J., Hu, H., Lin, J., Chang, G., Liu, W., Wen, Y., Yu, Y., Zhang, W., 2025. A survey of AI agent protocols. arXiv preprint arXiv:2504.16736.

Yang, Z., Li, L., Wang, J., Lin, K., Azarnasab, E., Ahmed, F., Liu, Z., Liu, C., Zeng, M., Wang, L., 2023. MM-REACT: Prompting ChatGPT for multimodal reasoning and action. arXiv preprint arXiv:2303.11381.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., & Narasimhan, K. (2023a). Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems, 36*, 11809–11822.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023b). ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Zhang, H., Dong, Y., Xiao, C., Oyamada, M., 2023. Large language models as data preprocessors. arXiv preprint https://doi.org/10.48550/arXiv.2308.16361.

Zhang, Y., Li, Y., Cui, L., Cai, D., Liu, L., Fu, T., Huang, X., Zhao, E., Zhang, Y., Chen, Y., & Wang, L. (2025). Siren's song in the AI ocean: A survey on hallucination in large language models. *Computational Linguistics*, 1–46.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., & Stoica, I. (2023). Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. *Advances in neural information processing systems, 36*, 46595–46623.