Decision Support Based on Large Language Models: Ontology-based Architecture and Generalized Scenario

Alexander Smirnov, Tatiana Levashova, Andrew Ponomarev
St. Petersburg Federal Research Center of the Russian Academy of Sciences
St. Petersburg, RF
{smir, tatiana.levashova, ponomarev}@iias.spb.su

***Abstract*—The paper presents initial results on models and methods of LLM-assisted augmented intelligence in context-aware decision support. It investigates a range of LLM-based decision support scenarios and reveals that they face common challenges, including problem clarification, user preference consideration, domain knowledge usage, generating alternatives and making decisions. Based on these findings, a set of requirements for LLM-based decision support is specified, an architecture for an LLM-based decision support system is proposed, and a generalized LLM-based decision support scenario is developed. The architecture combines agent-based interaction, conceptual modeling, and external solvers to support flexible and explainable decision-making. It enables the dynamic adaptation of recommended decisions to user preferences and evolving contexts. The generalized scenario incorporates the common challenges into a unified workflow. It integrates ontology-driven knowledge representation, conversational problem modeling, and problem solving by computational components. Within the framework of this scenario, the specific scenarios of LLM-assisted ontology development and decision support are discussed in detail. The contribution provides a foundation for developing LLM-based decision support systems enabling effective and informed decision-making.**

I. Introduction

In recent years, Large Language Models (LLMs) have gained significant attention in decision support, as they enable more personalized user experiences by adapting to individual preferences and behaviors [1]–[3]. An analysis of studies on LLM-based decision support has shown that these studies deal with similar (typical) tasks that are solved during decision support process. This motivated the development of an LLM-based Decision Support System (DSS) that integrates these tasks into a generalized scenario.

In the paper, different LLM-based decision support scenarios are investigated. Based on these scenarios, requirements for LLM-based decision support are specified, architecture of an LLM-based DSS is proposed, and a generalized scenario of such DSS is developed. Main features of the DSS offered are agent-based DSS architecture, combination of LLM capabilities and conceptual modeling and of methods supported by various solvers and computational modules when processing a user request. These features enable effective and informed decision-making.

The remainder of this paper is organized as follows. The remainder of this paper is organized as follows. Section II presents an overview of LLM-based decision support scenarios and specifies requirements for LLM-based decision support. Section III introduces the architecture of an LLM-based DSS. Section IV describes decision support scenarios enabled by this architecture.

II. Large Language Models in Decision Support Scenarios: Overview

The analysis of a number of decision support scenarios that use LLM has revealed that similar (typical) activities are present across different scenarios. The scenarios considered span several application domains: medicine, emergency management, logistics, scheduling and planning, smart city management, and public administration. The majority of these scenarios are drawn from the medical domain because there they have proven to be the only tool capable of effectively working with the basis of healthcare – medical text and data – freeing up physicians' time for truly complex clinical tasks and improving the accessibility of knowledge.

In this Section, the overviewed scenarios are grouped in accordance with the typical activities. A typical activity is a recurring process that manifests across multiple scenarios. Four categories of typical activities have been identified. They are problem clarification, user preferences consideration, domain knowledge usage, and alternatives consideration.

A. Problem clarification

In the analyzed scenarios, request or problem statement clarification may be required when the user is not satisfied with the DSS's response or when the DSS lacks the information needed to process the request.

An request clarification scenario proceeds as follows. First, the LLM proposes a decision and then initiates an interaction with the user to confirm their satisfaction. If the user is unsatisfied or requires clarification, they may ask follow-up questions. Should the LLM's responses reveal that its initial output was inaccurate due to a misinterpretation of the original request, the request is accordingly revised (i.e., the problem statement is updated). This mechanism is exemplified in the fire evacuation plan development scenario, particularly during the phase where the LLM's proposed plan is clarified [4].

A discussion between the user and the LLM to obtain missing information occurs in the decision-making scenario where the LLM diagnoses and recommends a treatment plan for a patient [5]. In this scenario, requesting missing information is an iterative process. When, during the clinical decision-making process, the LLM determines that it lacks the information needed to propose a decision, it requests the necessary information from the user (doctors, medical professionals, etc.). Once the LLM determines that sufficient information has been gathered, it outputs the final diagnosis and treatment plan, ending the clinical decision-making task for the patient.

B. User preferences consideration

The analysis of the studied decision support scenarios has shown that explicit and tacit user preferences in them are identified through the examining text discussions, user-proposed alternatives, and user comments on the decisions (recommendations) offered by the DSS.

An example of a scenario wherein the DSS cannot fully take into account the user's preferences is the semi-structured decision-making in last-mile logistics based on a combination of traditional vans and autonomous delivery robots [6]. Here, the decision-making process involves balancing the preferences of multiple stakeholders and addressing conflicting objectives. The DSS aims at determining optimal locations for parking slots (which serve as transshipment points) and establishing efficient vehicle routes and schedules. When a solution is found, it is presented to stakeholders (e.g., customers, company managers, representatives of drivers' unions, local communities, etc.), who provide qualitative feedback on this solution in text form. This feedback, among other things, represents tacit and explicit stakeholder preferences. Based on the feedback, the LLM updates the problem statement considering these preferences. In this scenario, the LLM serves to interpret user text and update the problem statement; i.e., it facilitates the revealing preferences without identifying them itself.

Revealing user preferences from their text messages is applied in the meeting organization scenario, where the problem of organizing an online meeting is solved as a constraint-based scheduling problem [7]. In this scenario, the online meeting organizer uses a chat to set the time and duration of the meeting and specify the meeting participants. The LLM checks the participants' availability in their electronic calendars, forms a set of constraints, encodes them, and passes them to a constraint solver. Then, the LLM converts the result received from the solver into a human-readable view and publishes the recommended meeting time in the chat. The participants respond via the chat, indicating whether they are satisfied and, if not, what exactly they would like to change. In accordance with these preferences, the LLM adjusts the set of constraints, solves the scheduling problem again, and recommends a new meeting time. The discussion continues until a recommendation that suits all the participants is found, or until the solver returns information that no solution exists. The LLM then enters the agreed meeting time into the electronic calendars of the participants.

Revealing user preferences based on the analysis of alternatives proposed by users during a chat is one of the tasks addressed by the scenario of adapting an LLM to the decision-making process [8]. The scenario is implemented as an interaction of LLM agents that model participants in a discussion when choosing a destination for a joint trip. The scenario involves participant agents (the travelers) and a constructive agent that helps the participants reach a consensus. The participants' suggestions for the trip destination reflect their preferences and are considered alternatives. During the discussion in the chat, the participants and the constructive agent propose new alternatives to accommodate everyone's preferences. The emergence of a new alternative denotes changes in the participants' preferences. The constructive agent supports the participant agents during the discussion, seeks compromises, and builds upon the participants' alternatives with new ones emerging during the discussion. The discussion continues until a decision that satisfies all the trip participants is found.

In both scenarios above [7], [8], the LLM processes text messages. However, in the meeting organization scenario, these messages explicitly or implicitly indicate the preferences of individual users (e.g., preferred or non-preferred meeting times) rather than proposing a group-level decision. In contrast, in the trip planning scenario, each message is considered an alternative, as it contains a concrete proposal for the trip destination.

C. Domain knowledge usage

This section examines scenarios that leverage domain-specific knowledge repositories, which fall into the following categories:

- *Domain ontology together with a knowledge graph, or a knowledge graph alone*. When both an ontology and a knowledge graph are used, the ontology provides a framework that defines the concepts, relationships, and entities within the graph. The knowledge graph is a specific instantiation of the ontology, representing real-world data in a graph structure where nodes are entities and edges are relationships [4]. In scenarios that use a knowledge graph as an independent repository, this graph organizes and links knowledge from various sources [9] and ensures its comprehensibility and coherence. Specifically,
  - in the decision support scenario for fire safety planning [4], the ontology and knowledge graph represent knowledge about scenarios for which evacuation plans have been developed and discussed with the user;
  - a prompt-managed LLM builds an OWL domain ontology in the decision support scenario aimed at optimizing intermodal freight transportation [10]. To this end, it uses a set of documents provided by the user. The ontology is modular and contains modules that thematically group: (i) domain knowledge,

(ii) knowledge for describing the problem statement, and (iii) optimization models extracted from various academic articles. A knowledge graph is derived from the ontology to describe a specific scenario;
  - in the clinical decision support scenario based on explainable AI [9], the knowledge graph serves as a repository for constantly updated medical recommendations. This graph is enriched when a user request for a recommendation cannot be processed due to a knowledge gap. New knowledge is then retrieved from external documents by searching for request terms and identifying semantically relevant content.

- *Domain knowledge classifiers*. An example here is the scenario aimed at improving a physician alert system (e.g., for vaccinations or tests) wherein a classifier is used to categorize the reasons why physicians override recommendations/alerts [11]. Reasons are added to this classifier by integrating and generalizing physicians' comments accompanying the overrides. Beyond assigning reasons to existing classes, the system also is able to identify new reason categories and expands the classifier accordingly.
- *Public knowledge bases that support a specific standard for knowledge representation*. An example of public knowledge is openEHR architecture [12] that is used in many scenarios from the medical domain. This architecture is an open standard for managing, storing, and sharing electronic health records. In addition to patient health records, the base contains archetypes (standardized templates for representing a specific piece of medical knowledge, expressed using the same formal apparatus). The archetypes offer knowledge that physicians can reuse for their own purposes. Among the scenarios analyzed, the scenario of clinical decision support in prostate cancer [13] deals with openEHR knowledge base enriching it. This is done by extracting information from medical information systems and storing it in the base in the form of archetypes.
- *Internal knowledge repositories of a DSS.* The structure of such repositories is determined by the specific DSS. For example, in the scenario of cooperative driving automation [14], a memory module accumulates domain knowledge. This module stores the scenarios processed by the system, the solutions recommended for them, and the decisions made. Knowledge is then structured using a vector database.

D. Alternatives consideration

In the scenarios examined, the following views on alternatives in a DSS can be identified: (1) an alternative as a single DSS recommendation; (2) multiple alternatives as recommendations from different LLMs; (3) alternatives that are dynamically produced as the situation changes; and (4) alternatives aimed at effective or optimal decision-making.

1) An alternative as a single DSS recommendation. The recommendation proposed by an LLM-based DSS is considered an alternative when the recommendation is a basis for the final decision-maker's decision. The decision-maker can either accept the recommendation as proposed; adjust it, thus generating a new alternative; or reject it and make his/her own decision, different from the one recommended. The scenarios falling in this category are overviewed below.

In the decision support scenario for diagnosing and developing a patient treatment plan recommendation [5], when diagnosing and developing the recommendation, the LLM uses the patient's description, their medical history, completed tests, and medical guidelines for diagnosis and treatment. The decision-maker can ask to recheck, reject, or accept the diagnosis and recommendation proposed by the LLM.

In the administrative decision-making scenario [15], the applicant files an application for a building permit with the competent authority and hands in all required documents electronically in a standardized format. The LLM, trained on legislative documents and examples of decisions, makes a determination for a positive or negative decision, prepares a draft decision with explanations, and the DSS transmits this decision to the competent person (decision-maker). The decision-maker can choose the recommended decision (accordingly, the draft becomes the final version of the determination), adjust the recommended decision and modify the draft decision, or propose their own decision, different from the one recommended.

In the multi-agent DSS for triage and treatment planning in emergency departments [16], the alternative is the initial patient treatment recommendation proposed by the LLM agents during the collection and analysis of primary patient data. This process utilizes domain knowledge in the form of treatment guidelines and medication standards. The alternative contains a primary diagnosis, a list of recommended medications, and a proposed treatment plan. This alternative, along with the supporting information and explanations, is passed to another LLM agent, which analyzes all the received information and makes the final decision regarding the treatment plan and the need for hospitalization. This agent evaluates the alternative using patient information gathered by the LLM agents, medication standards, and data from web resources. Based on this evaluation, the agent decides either to follow the recommendation or to propose a different suggestion. The agent provides explanations to support its decision.

2) Multiple alternatives as recommendations from different LLMs. Several different LLMs provide recommendations to the same request, each LLM's recommendation is considered an alternative.

Examples of scenarios with several LLMs are intraoperative decision support in plastic surgery and decision support in personalized oncology.

The plastic surgery scenario [17] is a scenario for the comparative evaluation of LLMs. Two LLMs are used to generate answers to natural language questions concerning the planning of a surgical operation, taking into account possible complications. In case of complications, the LLMs must

propose alternative solutions for performing the operation. The LLMs are evaluated on a 5-point scale regarding surgical planning, knowledge of general anatomy, surgical procedures, and the ability to find solutions and alternatives in case of possible complications.

In the personalized oncology scenario [18], four LLMs are used to generate a response to a request formulated using a typed prompt. This scenario is used for cases where there are no standardized recommendations for responding to the request, but rather multiple sources with different recommendations. Solutions proposed by multiple LLMs are referred to an expert panel for consideration, while those suggested by only one LLM are discarded.

3) Dynamically produced alternatives. A DSS generates alternatives as the current situation evolves.

Examples of scenarios with dynamically produced alternatives include the decision support scenario for fire safety planning [4] and the two discussed above: the scenarios of meeting organization [7] and joint trip planning [8].

In the fire safety planning scenario [4], the user sends the LLM a plan and a description of the structural features of the room where signs of fire have appeared. Based on this information, the LLM generates a step-by-step evacuation plan with explanations and proposes an alternative plan for emergency evacuation (for example, by breaking windows). Although the DSS provides for interaction with the user, who can ask the LLM clarifying questions as the fire spreads and the evacuation process proceeds, such interaction in an extreme situation is not always possible. Therefore, without a request from the user, the LLM gives additional recommendations based on general domain knowledge about fire safety (for example, if the fire has spread across the ceiling, everyone should move closer to the floor, as the air at floor level will be fresher and cooler). All recommendations are accompanied by explanations.

In the meeting organization scenario [7], an alternative is a meeting time that reflects the participants' preferences. Through negotiation, the DSS produces new alternatives until a mutually agreeable time is found.

In the joint trip planning scenario [8], alternatives are dynamically proposed by the LLM agents during their chat discussion about the trip destination. Each new alternative is an attempt to accommodate the preferences of all the participants regarding the destination. These preferences change as alternatives are proposed.

4) Alternatives for effective or optimal decision-making. A decision is made by evaluating alternatives against a set of criteria.

Scenarios that involve searching for an optimal or effective decision by evaluating a set of alternatives include: the intermodal freight transportation optimization scenario [10], the multi-agent DSS for smart city management [19], and the surgical decision support scenario [20].

In the intermodal freight transportation optimization scenario [10] alternatives are generated by the LLM. Here, the alternatives are transportation routes from the departure point to the destination. Each route is associated with values for the optimization criteria defined in the problem statement—namely, maximizing fuel efficiency, minimizing greenhouse gas emissions, and minimizing total operating costs. A knowledge graph is used to generate the set of routes. To evaluate the alternatives and select the most efficient transportation options, optimization algorithms based on a relational lookup table filled with decision metrics are used. These algorithms are represented in the modular domain ontology.

The scenario of multi-agent DSS for smart city management [19] is structured as a sequence of requests that implement a decision-making process—namely, information collection, alternative development, and decision selection. In this scenario, the LLM generates a set of alternatives in response to a request that implies multiple possible decisions. For example, in city improvement requests, alternatives may include locations for new parks or schools. To generate these alternatives, the system uses city development documents and city service APIs. The choice of an alternative—i.e., making a decision—results from processing the decision selection request according to predefined criteria.

The surgical decision support scenario [20] is prompt-driven. Four RAG prompts are used: 1) to check patient records for missing clinical investigation data, 2) to identify and flagging investigation results outside of normal ranges; 3) to develop recommendations for next management steps based on national surgical guidelines; 4) to prepare structured operative notes based upon recommended management steps. The scenario uses databases with reference values for clinical test indicators, a knowledge base of required clinical tests, and clinical guidelines. Using RAG prompts, the LLM develops patient management recommendations considering possible alternatives and selects one preferred alternative as a recommendation. If the alternatives are equivalent, the preferred alternative is the treatment method not involving surgical intervention. The LLM's reasoning is accompanied by explanations.

E. Specification of requirements for LLM-based decision support

Based on the scenarios analyzed above, requirements for LLM-based decision support are specified. These fall into three categories, as follows.

1) Requirements for LLM-based DSS:

- the DSS must enable user-system interaction through both chat and graphical interface;
- the DSS must enable the LLM to proactively initiate interaction with the user;
- the DSS must provide the user with recommended decisions in a clear form suitable for expert analysis and evaluation;

- DSS-proposed recommendations must consider preferences and personal qualities of the users (decision-makers);
- the DSS must be context-aware regarding dynamically changing problem statements, user preferences, and the current situation;
- the DSS must incorporate the latest advances by integrating modern technologies for alternative generation, including augmented intelligence, machine learning, neural networks, and others;
- the DSS must support the development of a domain knowledge repository via an LLM or alternative approaches;
- the DSS must have a "memory" capability, enabling it to accumulate knowledge about executed scenarios, proposed recommendations, and decisions made – all in a form suitable for analysis and reuse.

2) Requirements for LLM (some of the requirements, e.g. ability to searching for information, suppose that the LLM is integrated with external tools)*:*

- in multi-agent DSSs, LLM agents must be able to interact in a chat;
- the LLM must be able to interpret user requests – whether online or offline – in natural language, graphics, or in mixed formats;
- when responsible for the knowledge repository, the LLM must be able to: (i) extract relevant knowledge from heterogeneous sources; (ii) represent it in a compatible format; and (iii) enrich the repository with new knowledge gained through search, problem-solving, or scenario execution;
- LLM must be capable of searching, retrieving, processing, and integrating relevant information (knowledge) from heterogeneous external sources (including the Web and the user); presenting it in the required format; and, in certain cases, evaluating it;
- the LLM must be able to propose recommendations and alternatives based on: the user's problem formulation; knowledge from heterogeneous sources (e.g., instructions, guidelines, legislative documents); domain knowledge repository; and an assessment of gathered information;
- if other DSS components besides the LLM participate in processing the user's request, the LLM must supply the necessary information in a format suitable for those components;
- all outputs intended for the user – whether responses from the LLM or calculation results from other DSS components – must be presented in a form that is both user-understandable and suitable for expert analysis and evaluation;
- the LLM must provide explanations alongside its answers, recommendations, and the results of the development, evaluation, and selection of alternatives. These explanations should, where appropriate, include references to the knowledge sources that informed the LLM's outputs;
- the LLM must be able to dynamically update the problem statement in accordance with changes in the current situation;
- when responsible for generating recommendations and alternatives, the LLM must incorporate the latest advances available from web resources;
- when responsible for alternative generation, the LLM must support both refinement of existing alternatives and dynamic creation of new ones in response to evolving circumstances;
- when responsible for evaluating and selecting alternatives, the LLM must be able to base its decisions on multiple inputs: explicitly defined criteria; criteria drawn from domain knowledge sources (including web resources); and its own "general" knowledge;
- the LLM must be able to represent the problem stated in the user request in a format compatible with the DSS's domain knowledge repository;
- when responsible for identifying user preferences, the LLM must be able to extract explicit and implicit them based on information from the user (for example, user's text messages) during the development of alternatives and decision-making;
- the LLM must be able to identify decision-makers' behavioral patterns during their evaluation of alternatives;
- the LLM must be able to compose formal documents to document the decision made, based on available templates.

3) Requirements for organizing the decision support process:

- if necessary, the DSS should support pre-prompting, as well as prompting techniques for expanding context (RAG) and managing reasoning (Chain-of-Thought) in decision support.

4) Requirements for domain experts:

- expert decision-makers must be competent to evaluate the recommended decisions proposed by the DSS;
- RAG and prompt engineers must satisfy the required qualifications.

III. Archtecture of LLM-based DSS

The proposed architecture is based on the idea that a DSS implementing LLM-based decision support should be able to address the following tasks: (1) clarifying the problem formulation; (2) generating context-aware alternative recommendations; and (3) evaluating these alternatives – including providing explanations and recommending a decision – to support the decision-maker in selecting the final decision. Implementing these tasks requires: (i) a conversation

with the decision-maker to clarify the problem statement; (ii) generating feasible solutions (alternatives); and (iii) evaluating the alternatives while fostering trust between the decision-maker and the LLM. Trust is achieved, in particular, through the explainability of the LLM's reasoning. Here, a problem is defined as a decision problem that is formulated in the user request to the DSS.

Key architectural components of the LLM-based DSS are data sources, domain ontology, the LLM, LLM-agents, user interface modules, and repository of solvers and generative models (Fig. 1).

Data sources serve as suppliers of data, information, and knowledge that are incorporated into the domain ontology or can be consulted when responding to user requests.

Domain ontology provides structured knowledge about the problem domain and possible decision support tasks, shaping the decision support process. It is used in several ways: to align output with user-expected concepts, to formally represent the problem, user preferences and context, and to select relevant solvers. Ontologies for knowledge representation appear to be the most promising approach. This is because, although LLMs can deal with both textual and structured knowledge, the latter generally yields better results when incorporated into prompts.

The architecture includes two LLM agents: the Modeling agent and the Decision Support agent (DS-agent).

The Modeling agent assists the domain expert in building domain ontology. It interacts with this expert to obtain an ontology that matches the experts' perspective and needs. The DS-agent further leverages this ontology in the decision support process.

The DS-agent plays a central role during DSS functioning. It is responsible for managing the conversation with the decision-maker. During this conversation, the agent performs three tasks: 1) clarifies the problem and identifies the decision-maker's preferences; 2) builds a problem model based on the domain ontology; and 3) selects solvers and/or generative models from the repository, possibly arranging them into a decision support workflow, suitable for the decision problem. The domain ontology and data sources offer additional information that supports and enriches the conversation.

User interface modules (one for each agent) manage interaction with the users and help to represent information in most suitable way (not necessary only plain text). User interface support multiple modalities: free text, document uploads (e.g., text files, images, databases) with associated metadata, and configurable settings via specialized interface elements.

The LLM supports the development of the domain ontology, interpretation of the decision-maker's texts and mapping them to the problem model. It can also serve as a fallback solver, but specialized solvers from the repository are preferred, as they guarantee strict (or at least controlled) satisfaction of user requirements and offer greater explainability. In addition, the LLM can be used to unify data representation and/or form a consolidated representation of the solution.

The repository of solvers and generative models houses computational models for problem-solving, from which a specific model is selected for a given problem. Modules from this repository can be wrapped as "tools" (for LLMs supporting the use of tools), but other solver discovery protocols can be used as well.

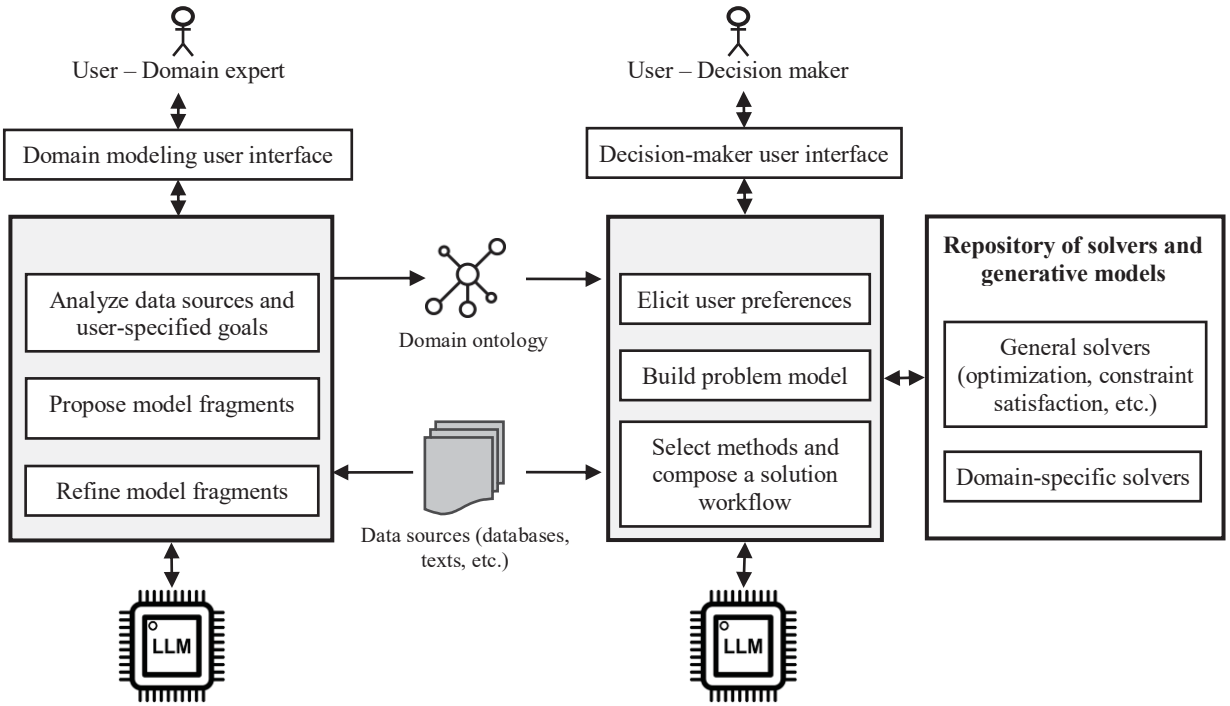

Fig. 1. Ontology-driven architecture of LLM-based DSS

IV. LLM-BASED DECISION SUPPORT SCENARIOS

The generalized scenario for LLM-based decision support (Fig. 2) starts with building the domain ontology and a formal model of the problem. Because the probability of obtaining a fully consistent and detailed ontology directly from an LLM is low [21], the ontology development process is iterative involving the Modeling agent and the domain expert.

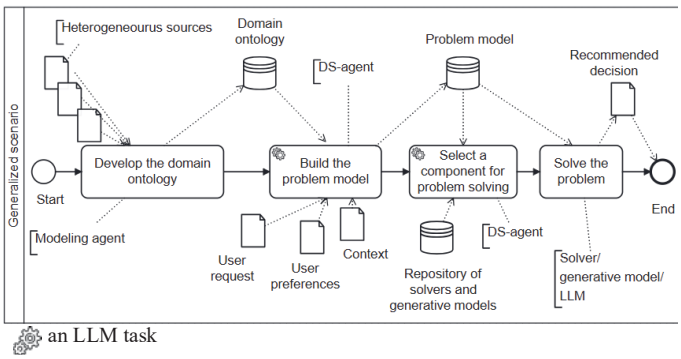

Fig. 2. Generalized LLM-based decision support scenario

The DS-agent, using the domain ontology, creates the problem model from the current user request and instantiates it with context information. This model captures ontology knowledge relevant to the user request and user preferences.

Then, the DS-agent selects a component (a solver or a model) from the repository of solvers and generative models for problem solving. This component solves the problem and provides a recommended decision.

The ontology development process is implemented using RAG prompting (Fig. 3). First, relevant structured and unstructured data sources (databases, documents, knowledge models, etc.) are identified. If these sources include diagrams or other graphical content, they are converted to text using specialized tools. A RAG prompt then feeds this content to the LLM.

Some more prompts are used to describe the ontology development scheme according to the chosen methodology (e.g., METHONTOLOGY [22], Protégé [23], On-to-Knowledge [24], etc.), specify the ontology requirements and the representation format, formulate competency questions, and may contain examples of expected results when few-shot prompting is applied.

The Modeling agent develops an initial draft ontology and begins interacting with the domain expert to refine it. In some cases, the draft ontology may be represented simply as a set of related concept pairs, similar to RDF triples. The ontology is completed when the expert is satisfied with the result.

Fig. 4 shows the workflow from the user request to the final recommendation. The DS-agent supports interactions with the user, receives the request, extracts context, builds the problem model, selects a computational component for problem-solving, converts information and knowledge, recommends a decision in a user-readable view, and provides necessary explanations.

The selected problem-solving component receives the problem model in a compatible format. It processes the model as a decision support problem and outputs a recommended decision. The DS-agent then presents this recommendation to the decision-maker for further evaluation and final decision-making.

The LLM solves the problem if no component that can solve it is found in the repository of solvers and generative models.

VII. CONCLUSION

This paper introduces an ontology-driven architecture for an LLM-based DSS and a generalized decision support scenario that incorporates the typical problems encountered in such systems.

The architecture combines agent-based interaction, conceptual modeling, and external solvers to support effective and explainable decision-making.

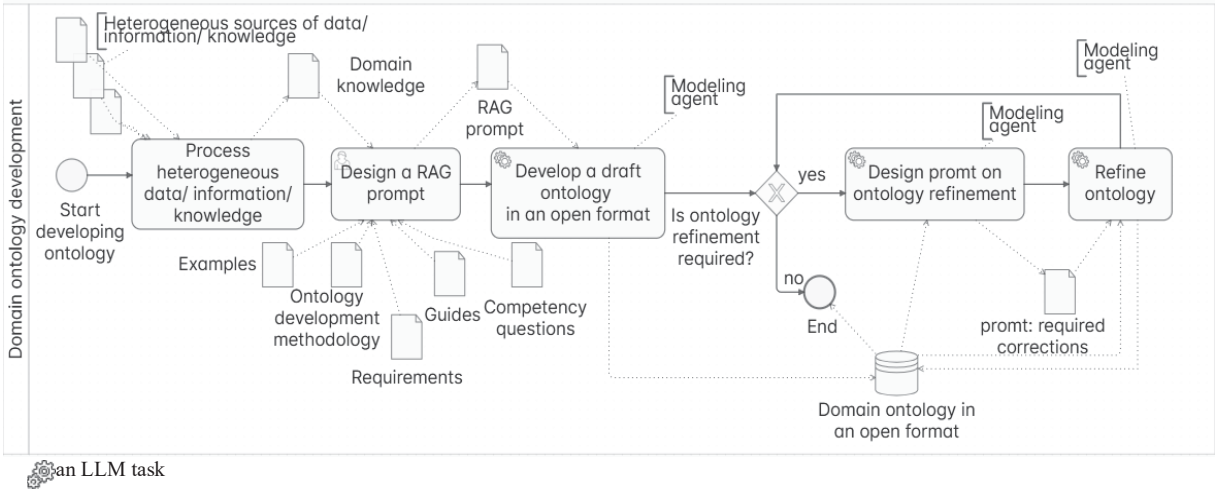

Fig. 3. Domain ontology development

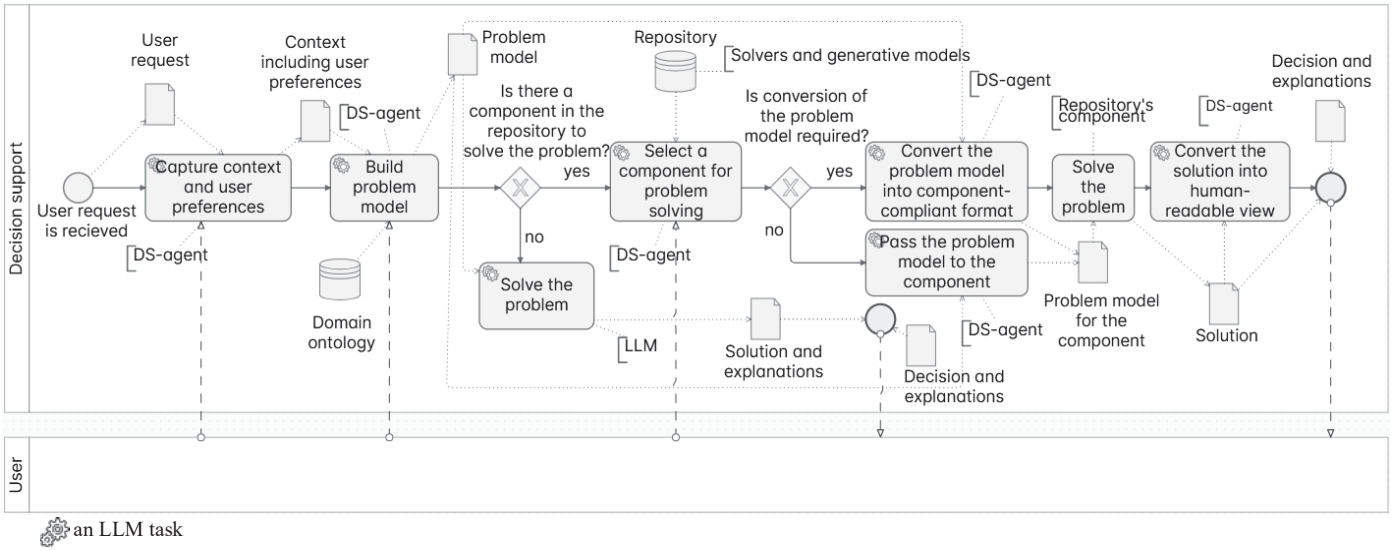

Fig. 4. Decision support

The proposed scenario is the result of an investigation of several LLM-based decision support applications, which reveal that they face a common set of challenges. The scenario integrates these challenges into a unified workflow.

The research contributes to the field of LLM-based decision support by demonstrating how LLMs can enable effective and informed decision-making through adaptation to individual preferences and behaviors, conversational problem modelling, integration with specialized solvers and generative models, and usage of structural knowledge.

The paper presents the initial results of a study on models and methods of LLM-assisted augmented intelligence in context-aware decision support. This phase focuses on developing conceptual models and does not yet include experimental validation. Future work will focus on implementation issues and practical validation.

ACKNOWLEDGMENT

The research is s funded by the Russian Science Foundation (project 25-11-00127).

REFERENCES

[1] M. Abolnejadian, S. Amirshahi, M. Brehmer, and A. Crisan, “AInsight: Augmenting Expert Decision-Making with On-the-Fly Insights Grounded in Historical Data,” Jul. 2025, doi: 10.1145/3719160.3737633.

[2] A. Handler, K. R. Larsen, and R. Hackathorn, “Large language models present new questions for decision support,” *Int. J. Inf. Manage.*, vol. 79, p. 102811, Dec. 2024, doi: 10.1016/j.ijinfomgt.2024.102811.

[3] R. Deng *et al.*, “CPGPrompt: translating clinical guidelines into large language model-executable decision support,” *J. Am. Med. Informatics Assoc.*, Feb. 2026, doi: 10.1093/jamia/ocag026.

[4] D. Durmus, A. Giretti, O. Ashkenazi, A. Carbonari, and S. Isaac, “The Role of Large Language Models for Decision Support in Fire Safety Planning,” in *41st International Symposium on Automation and Robotics in Construction (ISARC 2024)*, Jun. 2024, pp. 339–346, doi: 10.22260/ISARC2024/0045.

[5] P. Hager *et al.*, “Evaluation and mitigation of the limitations of large language models in clinical decision-making,” *Nat. Med.*, vol. 30, no. 9, pp. 2613–2622, Sep. 2024, doi: 10.1038/s41591-024-03097-1.

[6] G. Ghiani, G. Solazzo, and G. Elia, “Integrating Large Language Models and Optimization in Semi- Structured Decision Making: Methodology and a Case Study,” *Algorithms*, vol. 17, no. 12, p. 582, Dec. 2024, doi: 10.3390/a17120582.

[7] C. Lawless *et al.*, “‘I Want It That Way’: Enabling Interactive Decision Support Using Large Language Models and Constraint Programming,” *ACM Trans. Interact. Intell. Syst.*, vol. 14, no. 3, pp. 1–33, Sep. 2024, doi: 10.1145/3685053.

[8] S. Shiiku and Y. Takeuchi, “Exploring the Mutual Adaptation of Large Language Models and Emergent Decision-Making in Simulated Small Group Interactions,” in *Proceedings of the 12th International Conference on Human-Agent Interaction*, Nov. 2024, pp. 270–277, doi: 10.1145/3687272.3688317.

[9] C. Park, H. Lee, S. Lee, and O. Jeong, “Synergistic Joint Model of Knowledge Graph and LLM for Enhancing XAI-Based Clinical Decision Support Systems,” *Mathematics*, vol. 13, no. 6, p. 949, Mar. 2025, doi: 10.3390/math13060949.

[10] J. Tupayachi, H. Xu, O. A. Omitaomu, M. C. Camur, A. Sharmin, and X. Li, “Towards Next-Generation Urban Decision Support Systems through AI-Powered Construction of Scientific Ontology Using Large Language Models—A Case in Optimizing Intermodal Freight Transportation,” *Smart Cities*, vol. 7, no. 5, pp. 2392–2421, Aug. 2024, doi: 10.3390/smartcities7050094.

[11] S. Liu *et al.*, “Why do users override alerts? Utilizing large language model to summarize comments and optimize clinical decision support,” *J. Am. Med. Informatics Assoc.*, vol. 31, no. 6, pp. 1388–1396, May 2024, doi: 10.1093/jamia/ocae041.

[12] “openEHR Architecture Overview.” https://specifications.openehr.org/releases/BASE/latest/architecture_overview.html#_architecture_overview.

[13] P. Kaiser *et al.*, “The interaction of structured data using openEHR and large Language models for clinical decision support in prostate cancer,” *World J. Urol.*, vol. 43, no. 1, p. 67, Jan. 2025, doi: 10.1007/s00345-024-05423-1.

[14] S. Fang *et al.*, “Towards Interactive and Learnable Cooperative Driving Automation: a Large Language Model-Driven Decision-Making Framework,” *IEEE Trans. Veh. Technol.*, pp. 1–12, 2025, doi: 10.1109/TVT.2025.3552922.

[15] P. J. Pesch, “Potentials and Challenges of Large Language Models (LLMs) in the Context of Administrative Decision-Making,” *Eur. J. Risk Regul.*, vol. 16, no. 1, pp. 76–95, Mar. 2025, doi: 10.1017/err.2024.99.

[16] S. Han and W. Choi, “Development of a Large Language Model-based Multi-Agent Clinical Decision Support System for Korean Triage and Acuity Scale (KTAS)-Based Triage and Treatment Planning in Emergency Departments,” *Adv. Artif. Intell. Mach. Learn.*, vol. 05, no. 01, pp. 3261–3275, 2025, doi: 10.54364/AAIML.2025.51187.

[17] C. A. Gomez-Cabello, S. Borna, S. M. Pressman, S. A. Haider, and A. J. Forte, "Large Language Models for Intraoperative Decision Support in Plastic Surgery: A Comparison between ChatGPT-4 and Gemini," *Medicina (B. Aires).*, vol. 60, no. 6, p. 957, Jun. 2024, doi: 10.3390/medicina60060957.

[18] M. Benary *et al.*, "Leveraging Large Language Models for Decision Support in Personalized Oncology," *JAMA Netw. Open*, vol. 6, no. 11, p. e2343689, Nov. 2023, doi: 10.1001/jamanetworkopen.2023.43689.

[19] A. Kalyuzhnaya *et al.*, "LLM Agents for Smart City Management: Enhancing Decision Support Through Multi-Agent AI Systems," *Smart Cities*, vol. 8, no. 1, p. 19, Jan. 2025, doi: 10.3390/smartcities8010019.

[20] C. S. Ong, N. T. Obey, Y. Zheng, A. Cohan, and E. B. Schneider, "SurgeryLLM: a retrieval-augmented generation large language model framework for surgical decision support and workflow enhancement," *npj Digit. Med.*, vol. 7, no. 1, p. 364, Dec. 2024, doi: 10.1038/s41746-024-01391-3.

[21] F. Neuhaus, "Ontologies in the era of large language models – a perspective," *Appl. Ontol.*, vol. 18, no. 4, pp. 399–407, Dec. 2023, doi: 10.3233/AO-230072.

[22] M. Fernández-López, A. Gómez-Pérez, N. Juristo, M. Fernandez, A. Gomes-Perez, and N. Juristo, "METHONTOLOGY: From Ontological Art Towards Ontological Engineering," in *AAAI Proceedings of the Symposium on Ontological Engineering*, 1997, pp. 33–40.

[23] N. F. Noy and D. L. McGuinness, "Ontology Development 101: A Guide to Creating Your First Ontology," Stanford, 2001. [Online]. Available: http://www.ksl.stanford.edu/people/dlm/papers/ontology-tutorial-noy-mcguinness-abstract.html.

[24] Y. Sure, S. Staab, and R. Studer, "On-To-Knowledge Methodology (OTKM)," in *Handbook on Ontologies*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 117–132.