Graph of Goal-Oriented Thoughts: Design and Implementation of LLM Agents

Dario Badagliacca[1,†], Gabriele Caruso[2,†], Agnese Augello[3,†] and Luca Sabatucci[4,†]

[1,2,3,4] *National Research Council of Italy - Institute for High Performance Computing and Networking (ICAR-CNR), Via Ugo La Malfa, 153, 90145, Palermo*

Abstract

This paper presents a novel framework for managing complex user interactions in conversational agents through a goal-oriented model that integrates both proactive and reactive querying of Large Language Models (LLMs). Drawing inspiration from goal-based agents and the "Graph of Thoughts" (GoT) paradigm, we propose a structured approach to goal management that tracks and reasons about goals and sub-goals. Goals are represented as explicit "thoughts" within the GoT framework, enabling transparent and observable reasoning processes. Through an illustrative example, the paper shows the flexibility and effectiveness of this approach in adapting to dynamic and complex contexts. Moreover, future developments could extend the applicability of this paradigm to more intricate domains.

Keywords

Large Language Model (LLM), Goal Model, Graph of Thoughts, Conversational Agents

1. Introduction

The rapid and extensive development of *Large Language Models* (LLMs) has facilitated their integration into mainstream applications, marking a significant shift in the landscape of artificial intelligence. These models, trained on vast text corpora, have demonstrated remarkable capabilities in natural language understanding, generation, and reasoning. This breakthrough has raised the question of whether this advanced technology can be used effectively to address individual needs in a manner that is more intuitive, efficient, and user-friendly.

One of the most promising applications of LLMs in this sense is in the development of *goal-oriented conversational agents*, designed to interact with users in natural language, with the potential to enhance user experiences and automate complex tasks. In this context, a key area of research involves the enhancement of conversational agents to go *beyond* simple response generation and adopt proactive and adaptive strategies. This includes understanding user intents, anticipating needs, and guiding conversations toward successful outcomes [1, 2]. Such advancements are crucial in domains where interactions involve multiple goals, conflicting priorities, or non-collaborative behaviors, which are situations where the interaction between the conversational agent and the user may not be mutually cooperative or productive. For instance, the user might be hesitant to accept suggestions, may omit necessary information, or may act in ways that hinder the agent's ability to achieve the desired outcome [1, 2]. In these cases, the agent must adapt by understanding and navigating these behaviors, ultimately guiding the conversation toward a resolution despite the lack of collaboration from the user [1, 2].

Designing and implementing such *conversational agents* introduces new challenges primarily revolving around aligning LLM-generated dialogues with specific *user goals*, adhering to task completion metrics and navigating domain-specific constraints [2].

Our previous experience with autonomous and proactive intelligent agents [3, 4] has suggested leveraging the use of goal models together with LLMs for designing proactive conversational agents [5].

Joint Proceedings of the ACM IUI Workshops 2025, March 24-27, 2025, Cagliari, Italy

[†]These authors contributed equally.

dario.badagliacca@gmail.com (D. Badagliacca); gabricaru01@gmail.com (G. Caruso); agnese.augello@icar.cnr.it (A. Augello); luca.sabatucci@icar.cnr.it (L. Sabatucci)
0000-0001-6463-9151 (A. Augello); 0000-0003-2852-9355 (L. Sabatucci)

Therefore, this paper relies on the broad literature on goal-oriented requirement engineering to formalize the expectation with respect to the conversation. Specifically, Goals can be a profitable instrument to model how the agent will behave during the conversation with the final user. Our idea is to implement proactivity as the agent's ability to reason and respond according to some design-time goals.

To this aim, we believe that the Graph of Thoughts (GoT) paradigm [6] can facilitate the integration of goals in an LLM-based conversational agent. Among the emerging prompt engineering approaches [7, 8, 9], GoT can offer a promising framework for implementing advanced reasoning capabilities through large language models. The main idea of GoT is to explicitly model the reasoning process as a flexible graph structure where units of information are referred to as *"thoughts"*. This enables advanced operations such as aggregating multiple reasoning paths into cohesive solutions and dynamic reorganization of thought processes. In addition, the framework also shows reduced operational costs with respect to other prompting engineering techniques [6].

In this paper, we show a preliminary architecture for a general-purpose goal-oriented conversational agent that adopts goals and a graph of thoughts. This experience also resulted in a learned lesson, i.e., from a software engineering point of view, these elements could be the key to accommodating complex task decomposition and decision-making in a manner that aligns algorithmic structures more closely with human cognitive patterns [5].

The rest of this paper is structured as follows. First, Section 2 reviews the state of the art in areas such as *Large Language Models*, *Graph of Thoughts*, and other relevant topics, highlighting key developments and discussing their limitations. Section 3 introduces our framework, explaining its components and the principles that guide its design. Following that, in Section 4, we expose the structure of our solution, leveraging on the *Graph of Operations*, one of the key components of our work. In Section 5, we briefly present an illustrative example, whereas some conclusions are sketched up in 6.

2. Background

2.1. Large Language Models and Prompt Engineering

LLMs are large-scale, pre-trained models that leverage statistical methods and neural network architectures to process and generate natural language [10].

In this context, *Prompt Engineering* has emerged as a critical technique for optimizing the performance of LLMs. Central to this approach is the concept of *prompting*, which involves providing structured and specific input prompts to guide the model's behavior. Rather than modifying the underlying model or relying solely on external data, prompt engineering refines the process of crafting these inputs to achieve accurate, coherent, and task-specific outputs. However, as models transition from basic retrieval-based approaches, such as *Retrieval Augmented Generation* [11], to more sophisticated conversational agents, the need for *orchestration* becomes crucial.

Orchestration frameworks enable the integration and coordination of various components, allowing the system to move beyond simple retrieval to a dynamic, contextually aware conversational flow. This transition empowers the model to not only retrieve information but also engage in interactive, multi-turn conversations, making the system capable of responding intelligently and coherently over extended dialogues. Among prompting techniques, we can mention: 1) **Input-Output (IO) Approach**, the simplest method, involves using an LLM to directly convert an input sequence x into an output y, without incorporating intermediate reasoning steps; 2) **Chain of Thought (CoT)** [7], where intermediate steps are introduced between the input and the output to improve performances significantly; 3) **Multiple CoTs** that uses multiple chains. The output is selected from the chain that performs best (according to a given evaluation metric); 4) **Tree of Thoughts (ToT)** [8, 9], that shapes the reasoning process as a tree structure, where each node represents a partial solution; and 5) **Graph of Thoughts (GoT)** [6], where the reasoning process is shaped as a *directed graph*.

2.2. Graph of Thoughts

Introduced by *Besta et al.* [6], the *Graph of Thoughts* adopts a *directed graph* where **vertices** represent the thoughts or partial solutions and the **edges** indicate the relationship where one thought is constructed from another, with each edge pointing from the thought that serves as input to the thought that results from it. This graph structure facilitates the representation of intricate reasoning processes, enabling the application of *thought transformations* to evolve and improve the LLM's reasoning. The GoT approach, as stated by the authors, improves prompting engineering by offering a more sophisticated model of **human-like reasoning**. When working on a novel idea, a human would not only follow a chain of thoughts (as in CoT) or try different separate ones (as in ToT), but would actually form a more complex *network of thoughts*. This richer structure allows for deeper, more dynamic interactions between ideas, leading to enhanced problem-solving capabilities.

GoT introduces these innovative thought transformations, referred to as *graph-enabled transformations*, through the unique structure of the reasoning model. The authors expose three types of transformations: i) *Aggregation Transformations* – Aggregate thoughts into new ones; ii) *Refining Transformations* – Refine the current thought; and iii) *Generation Transformations* – Generate one or more new thoughts based on an existing one.

The thoughts can also be scored to determine whether the current solution is good enough and, finally, the best thought is selected through the *KeepBest* operation.

2.3. Goal Models and Epistemic Goals

The notion of ***Goal*** has been extensively employed in Artificial Intelligence (AI) where rational agents act to achieve functional objectives [12, 13] and in Goal-Oriented Requirement Engineering (GORE) as a means to elicit, understand and specify functional and non-functional aspects of a complex systems [14, 15, 16].

In literature, goals are also classified as Functional, Quality and Epistemic. Functional goals are those the agent wants to operationalize through its actions. Quality goals have been largely analyzed and discussed in the literature as instruments to depict non-functional requirements (such as usability, maintainability, security and performance) [17, 18, 19]. The agent may use them to self-limit its actions. An Epistemic goal is an objective to know something. Epistemic goals were initially proposed in [20, 21] as awareness requirements for implementing a self-adaptive system. In that case, epistemic goals are addressed by monitoring the agent's environment in search of obstacles/opportunities that may lead to corrective action if an obstacle is encountered or an opportunity arises.

In this paper, we adopt a mixed vision in which goals derive from the requirement analysis but are also run-time elements. There is a broad area of research that shares this vision to leverage flexibility into self-adaptive systems [22, 23], wherein behaviour adapts to dynamic context changes to ensure desired outcomes [24].

The goal-oriented requirements engineering perspective allows for the introduction of a higher abstraction point of view on the system's behaviour. In this view, a goal model is a standard instrument to decompose a root goal into subgoals to explore why, what and how dimensions [25, 26, 27]. On the other side, the agent has some goals, where each goal conveys a desired state of affairs, and it knows to own them. In other words, it has a representation of its goals, which are causally connected to behavior, thereby ensuring proactivity, self-adaptation and dependability.

Epistemic goals are particularly important in the context of a conversational agent. In this case, the desired state of affairs is an epistemic state, i.e., the need to acquire some information. It is similar to the concept of epistemic goal expressed in [5] related to the concept of awareness and, in particular, with self-awareness: "awareness is a state of mind of an agent who wants to know things about its goals" [28, 29].

3. The Proposed Solution

This work aims to design an efficient management of complex user interactions using a goal-oriented model, leveraging both proactive and reactive querying of Large Language Models (LLMs). The integration of both **proactive** and **reactive** behaviour ensures that the system will: (i) reactively analyze and respond to user inputs in real-time, and (ii) proactively generate relevant follow-up queries to clarify or expand upon unresolved goals.

For what concerns the proactivity, we take inspiration from goal-based agents. Our conversation agent will be aware of its objectives. Specifically, it will own a hierarchical decomposition of a *root goal*, known as *goal model*. Therefore, a **Proactive Controller** is responsible for generating proactive responses by tracking and reasoning on the state of each goal and subgoals (e.g., to establish if they are *satisfied*, *partially satisfied*, or *unsatisfied*), and taking a decision on the next (conversational) action to perform. To accomplish this kind of *awareness*, we have drawn inspiration from the *Graph of Thoughts* paradigm [6], to create an architecture for making agent's *thoughts* explicit elements. In this way, goal reasoning may be implemented via GoT's *thoughts*, providing a novel and dynamic approach to goal management.

3.1. Implementing Goals through Graph of Thoughts

The cornerstone of our framework is to **implement the goals of our goal model as if they were the thoughts of the conversational agent**. To achieve this, we first defined *awareness* through a goal model structure specifically aligned with this purpose, providing a hierarchical tree-like representation designed to organize and track goals by breaking them down into smaller, actionable sub-goals. This tree structure facilitates the management of dependencies between goals, ensuring a clear and systematic approach to their execution. It is worth noting that in the rest of the paper when we talk about goals, we mainly focus on epistemic goals because they rule out the need for knowledge of a conversational agent. However, the principles discussed here remain quite generic to all kinds of goals.

Within the *goal model*, relationships between goals are defined through **goal links**, which represent logical dependencies. These links are categorized as:

- **AND Link** indicating that a main goal is satisfied when all the sub-goals are satisfied;
- **OR Link** indicating that a main goal is satisfied when at least one of the sub-goals is satisfied, offering flexibility in how objectives are accomplished.

This mechanism mirrors principles of mathematical logic, where AND corresponds to conjunction and OR to inclusive disjunction, providing a structured and systematic approach to task management.

Therefore, we implemented goal reasoning by implementing two basic mechanisms:

- **Goal Checker** – Tracks the state of each goal, classifying them as *Satisfied*, *Partially Satisfied* (when some sub-goals have been achieved, but additional tasks are required; and *Unsatisfied.*
- **Task Checker** – Monitors the progress of sub-goals, recording completed tasks and prioritizing pending ones.

This structure offers several key advantages:

- Enables a systematic breakdown of high-level objectives, reducing complexity;
- Establishes clear prioritization of tasks based on logical dependencies;
- Enhances the efficiency and precision of the system's responses by maintaining a comprehensive overview of user objectives.

In essence, the goal model structure forms the backbone of the proactive controller, enabling the agent to dynamically address user needs while maintaining a clear trajectory toward achieving the root goal.

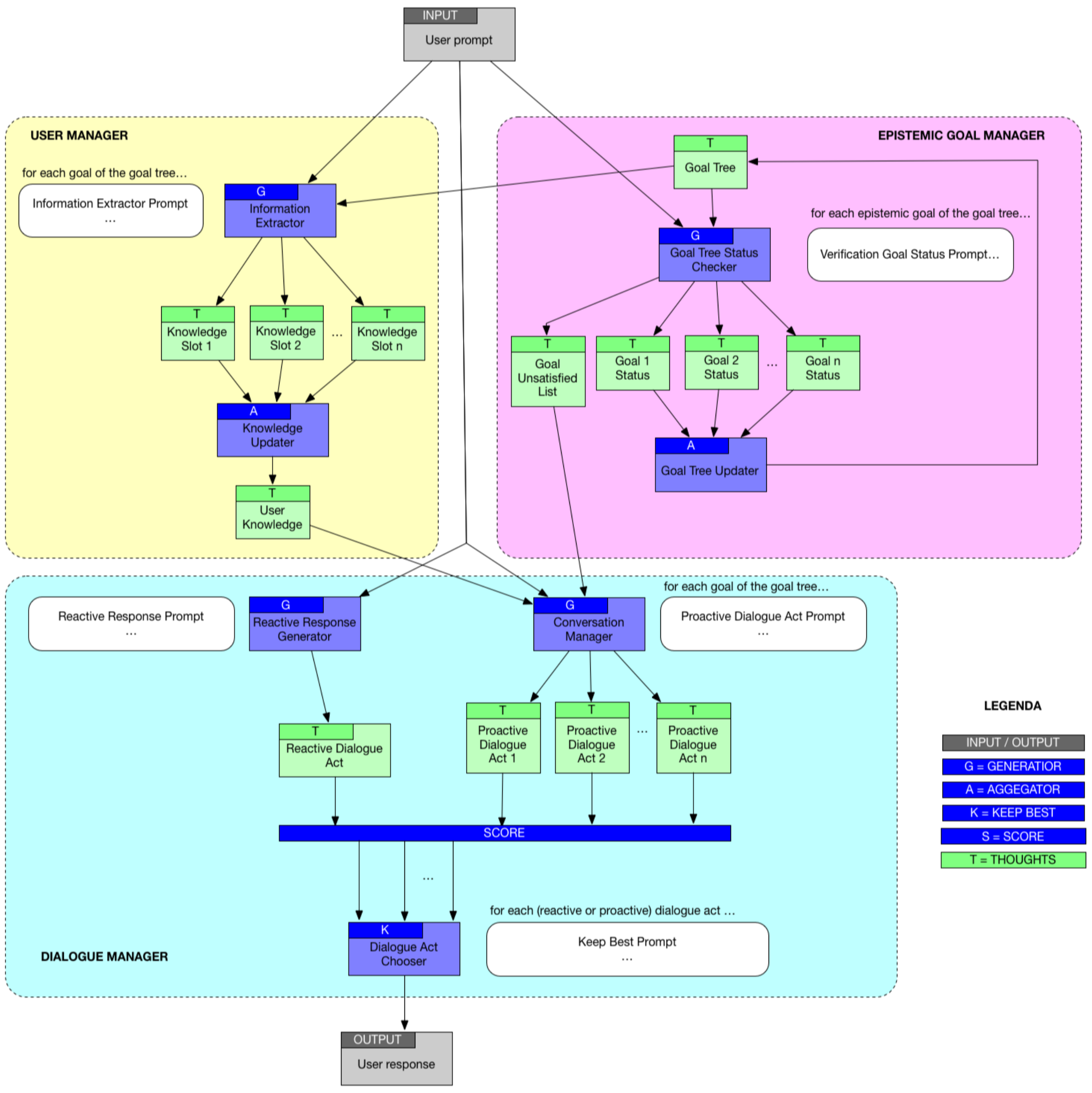


Figure 1: Detailed overview of the Graph of Operations and its three main functional areas

4. Conversational Agent Architecture

Figure 1 presents the *Graph of Operations* which serves as a template for the goal-based proactive conversational agent. It consists of three pivotal and interconnected areas: ***User Manager***, ***Epistemic Goal Manager***, and ***Dialogue Manager***. These are *functional areas* of the graph. Overall, the graph consists of interconnected nodes representing *Operations* that work on the agent's *Thoughts*. The ultimate objective is to process user prompts and produce a response. To do this, the graph tracks goals' status, extracts information and chooses the appropriate responses by evaluating both reactive or proactive behaviour, according to previous user interactions and the context.

4.1. User Manager

The **User Manager** area plays a crucial role in the dialogue system by extracting relevant information from the user prompt. This extracted data is essential for driving the operations of subsequent nodes, ensuring that the dialogue remains aligned with user input and system goals. To perform this task effectively, the *Information Extractor* node is a *Generator* that uses User Prompt and Goal Tree to query the LLM. The query employs the 'Information Extractor Prompt'[1], specifically designed for analyzing and extracting the key information conveyed by the user[2]:

Information Extractor Prompt

Your task is to identify text in the [*user prompt*] that mentions or refers to the [*goal name*].
Instructions:
1. Consider if the goal appears exactly as [*goal name*], or if its concept is expressed through synonyms, definitions, or equivalent descriptions.
2. Ignore content unrelated to [*goal name*]. Do not consider other goals unless explicitly related.

The extracted data are then organized into distinct *Knowledge Slots* (e.g., Knowledge Slot 1, Slot 2, ..., Slot n), each representing a discrete piece of information relevant to the system's goals.

Once the *Knowledge Slots* are populated, they are aggregated and updated in the *User Knowledge repository* via the *Knowledge Updater*. This is a persistent thought acting as a central knowledge base, tracking user-specific data to support reasoning and decision-making and the execution of tasks throughout the system.

4.2. Epistemic Goal Manager

The **Epistemic Goal Manager** area operates independently from the *User Manager*, yet complements it. It's objective is to evaluate user prompt against the goals in goal model, to determine whether any of them have been fulfilled on the basis of the information extracted by the *User Manager*. To perform this task, the *Goal Tree Status Checker* node, which is the *Generator* of this area, queries the LLM using the User Prompt and the Goal Tree. The query uses the 'Verification Goal Status Prompt', specifically designed for this purpose:

Verification Goal Status Prompt

Your task is to evaluate whether the [*user prompt*] explicitly mentions, refers to, or negates the goal named [*goal name*].
Instructions:
1. Mark as 'completed' only if all of the following conditions are met:
- The [*goal name*] is explicitly mentioned in [*user prompt*] using the exact term, precise synonyms, or clearly equivalent and unambiguous descriptions, OR if the goal is explicitly negated (e.g., 'I don't want [*goal name*]) with clear and direct language.
- The reference to the goal must be specific and leave no room for interpretation.
- There is no ambiguity, vagueness, or partial mention of the goal.
2. Mark as 'uncompleted' if any of the following are true:
- The [*user prompt*] does not clearly and explicitly mention, describe, or negate the goal [*goal name*].
- The [*user prompt*] only partially mentions or vaguely references the goal, leaving room for interpretation.
- The [*user prompt*] is unrelated, ambiguous, or generic, even if it hints at the goal without being specific.
- The [*user prompt*] includes broad or generic statements without sufficient detail to confirm alignment with [*goal name*].

[1]These prompts are intentionally generic to ensure their functionality in multiple scenarios. When we want to instantiate them for specific use, the parameters within square brackets are replaced with actual values.
[2]For the sake of space, the paper reports short versions of the various prompts. Their full versions are reported in the GitHub repository at https://github.com/ls-cnr/MIRA.

The current status of each goal is then updated by the *Goal Tree Updater* node, an *Aggregator* that ensures that the goal model remains accurate and reflective of the agent's progress.

Any goal that remains unmet after this evaluation is compiled into the *Goal Unsatisfied List*, a thought that serves as a repository of unfulfilled goals, allowing the *Dialogue Manager* to prioritize them when generating proactive responses.

4.3. Dialogue Manager

The **Dialogue Manager** area contains the core component of the dialogue agent's architecture, synthesizing information provided by the *User Manager* and *Epistemic Goal Manager* areas to generate system responses.

Its primary function is to produce two types of responses: *Reactive Responses*, which directly address the user's prompt; *Proactive Responses*, which proactively address unmet goals in the goal list.

To achieve this, the *Dialogue Manager* utilizes two *Generators* to orchestrate the creation of both types of responses: the *Reactive Response Generator* and the *Conversation Manager*, which leverage two specialized prompts to query the LLM.

The 'Reactive Response Prompt' generates responses that directly address the user's input, while the 'Proactive Dialogue Act Prompt' produces proactive responses for each unmet goal in the goal list, ensuring the system continues to make progress toward its objectives.

Reactive Response Prompt

Your name is [*agent name*]. You are [*agent occupation*].
Your task is to analyze the user's requests, understand them, and provide the best response.
Follow these personality and response style expectations: [*personality*] and [*social practices*].
Always respond in [*response language*], regardless of input language.

Proactive Dialogue Act Prompt

Your task is to carefully analyze the [*user prompt*] and focus exclusively on identifying, using your related [*goal information*], the specific request related to [*goal name*]. Your primary objective is to address or clarify any aspects of the [*goal name*] in detail.
Based on the content of the [*user prompt*], you must take one of the following approaches:
1) Provide a clear and accurate response to the user's potential question related to [*goal name*].
2) If the input is unclear, ask a specific clarifying question to better understand the user's intent regarding [*goal name*].
3) When appropriate, provide a hybrid response that combines an answer with a question to guide the user in revealing more details about [*goal name*].

Each proactive response corresponds to a specific goal from the *Goal Unsatisfied List*, resulting in a set of *Proactive Dialogue Acts* (Proactive Dialogue Act 1, Proactive Dialogue Act 2, ..., Proactive Dialogue Act n).

Once all responses are generated, they are evaluated using a scoring mechanism that determines the priority of each response. This scoring process leverages the *'Scoring Prompt'*, which helps the *Dialogue Act Chooser* to identify the most appropriate response based on predefined criteria:

Scoring Prompt

Evaluate the relevance of the [*response*] to the [*user prompt*] in relation to the goal [*goal name*] and its [*goal type*].
Evaluation Rules:
1. Focus on how the [*response*] aligns with the goal [*goal name*].
2. Assign with an annotation one of the following score levels based on relevance and depth:
- **high**: The response directly addresses the goal, provides detailed alignment, and is clearly relevant.
- **medium**: The response moderately addresses the goal, with some alignment, but lacking full detail.
- **low**: The response has minimal or no alignment with the goal, or is vague.

The *Dialogue Act Chooser* node is a *KeepBest* operation that simply selects the highest-ranked response to return to the user, ensuring the system provides the most relevant and effective output.

5. Illustrative Example: Travel Agent

The travel industry serves as a compelling example of how goal-oriented conversational agents, built on the previously discussed framework, can significantly improve operational efficiency and elevate customer satisfaction. This illustrative example highlights how such a structured approach can seamlessly integrate into a travel agent's workflow, effectively managing complex, goal-driven interactions.

Problem Description – Travel agents are often required to manage multiple concurrent tasks, such as creating personalized itineraries, responding to customer inquiries, and adapting to changing preferences in real time. Traditional systems struggle with:

- **Personalization**: Difficulty in tailoring recommendations to individual preferences.
- **Adaptability**: Challenges in dynamically responding to unforeseen changes in travel plans.

Solution Framework – By leveraging a goal-oriented approach integrated with GoT, the travel agent system is designed to address customer needs proactively and reactively. Each customer request is treated as a hierarchical set of goals, broken down into sub-goals such as:

- Identifying travel preferences (e.g., destinations, budgets).
- Recommending suitable itineraries.

The GoT paradigm enables dynamic reasoning, allowing the system to evaluate multiple paths and refine its responses based on the knowledge gathered by user input.

Implementation – The travel agent system consists of the following components:

- **Dialogue Manager**: Generates natural, conversational responses for both reactive (answering direct questions) and proactive (suggesting alternatives) interactions.
- **Goal Model Analyzer**: Tracks the state of goals
- **Reasoning Module**: Applies the GoT framework to generate, refine, and select the best course of action, such as suggesting alternative travel routes when disruptions occur.

Figure 2 shows a section of the goal model used by the system. The goals are organized in a tree structure where the completion of the goals connected by an AND link allows the achievement of the root goal.

Each goal is labeled as 'completed', 'partially completed' or 'not completed', determining the overall progress of the system.

The conversation with the customer concludes when all sub-goals connected by the AND link are marked as completed, leading to the completion of the main goal.

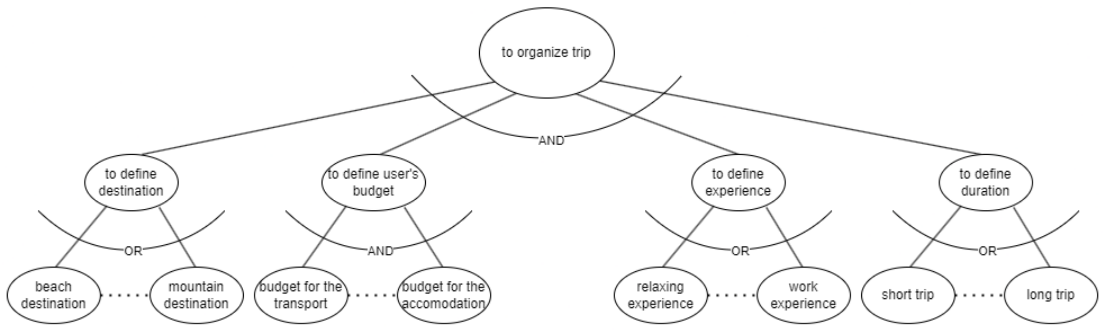
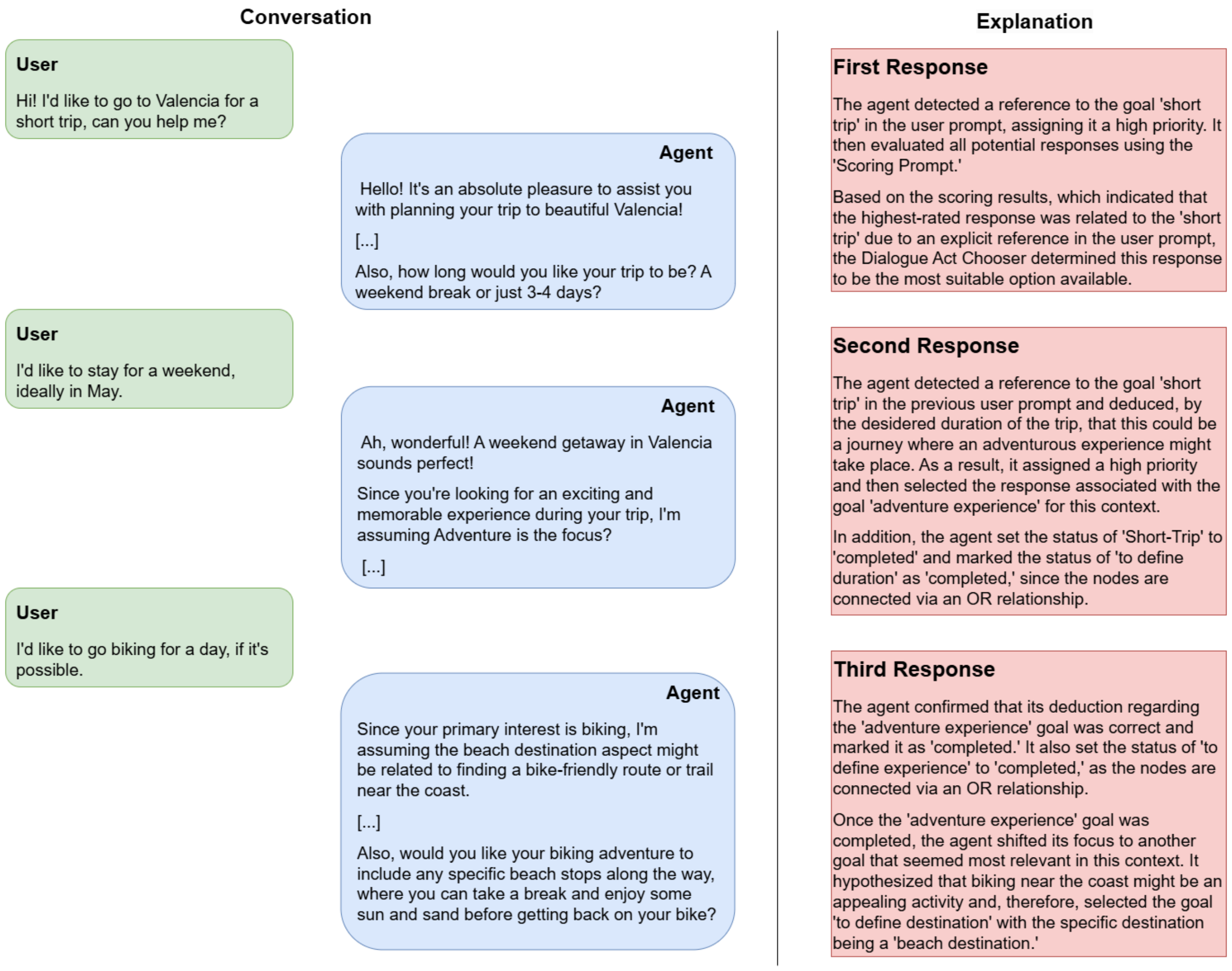


Figure 2: A Portion of the travel agent goal model



Conversation

User

Hi! I'd like to go to Valencia for a short trip, can you help me?

Agent

Hello! It's an absolute pleasure to assist you with planning your trip to beautiful Valencia!

[...]

Also, how long would you like your trip to be? A weekend break or just 3-4 days?

User

I'd like to stay for a weekend, ideally in May.

Agent

Ah, wonderful! A weekend getaway in Valencia sounds perfect!

Since you're looking for an exciting and memorable experience during your trip, I'm assuming Adventure is the focus?

[...]

User

I'd like to go biking for a day, if it's possible.

Agent

Since your primary interest is biking, I'm assuming the beach destination aspect might be related to finding a bike-friendly route or trail near the coast.

[...]

Also, would you like your biking adventure to include any specific beach stops along the way, where you can take a break and enjoy some sun and sand before getting back on your bike?

Explanation

First Response

The agent detected a reference to the goal 'short trip' in the user prompt, assigning it a high priority. It then evaluated all potential responses using the 'Scoring Prompt.'

Based on the scoring results, which indicated that the highest-rated response was related to the 'short trip' due to an explicit reference in the user prompt, the Dialogue Act Chooser determined this response to be the most suitable option available.

Second Response

The agent detected a reference to the goal 'short trip' in the previous user prompt and deduced, by the desidered duration of the trip, that this could be a journey where an adventurous experience might take place. As a result, it assigned a high priority and then selected the response associated with the goal 'adventure experience' for this context.

In addition, the agent set the status of 'Short-Trip' to 'completed' and marked the status of 'to define duration' as 'completed,' since the nodes are connected via an OR relationship.

Third Response

The agent confirmed that its deduction regarding the 'adventure experience' goal was correct and marked it as 'completed.' It also set the status of 'to define experience' to 'completed,' as the nodes are connected via an OR relationship.

Once the 'adventure experience' goal was completed, the agent shifted its focus to another goal that seemed most relevant in this context. It hypothesized that biking near the coast might be an appealing activity and, therefore, selected the goal 'to define destination' with the specific destination being a 'beach destination.'

Figure 3: Extract from a conversation between the user and the dialogue agent

Figure 3 provides an example of a dialogue between the agent and a user. The user begins by requesting an itinerary for a trip to Valencia. The agent leverages the *Dialogue Manager* functional area to provide a reactive response, aiming to gather as much information as possible to determine the desired duration of the trip.

Once the duration goal is satisfied based on the information provided by the user's subsequent prompt, the agent identifies the next goal to address and requests specific information related to it.

Finally, the conversation concludes when all sub-goals connected by the AND link are marked as 'completed', allowing the root goal to be considered achieved.

6. Conclusions

Our experience in implementing a dialogue agent using Graph of Thoughts and epistemic goals has revealed promising insights into a potential design methodology for dialogue systems, offering a structured approach to designing observable reasoning patterns.

Our key insight is that the Graph of Thoughts can serve as an explicit design tool for modelling the agent's reasoning process by explicitly defining and controlling how the agent processes information and makes decisions. This observable-by-design approach means that the reasoning flow is deliberately engineered during the design phase.

On the one hand, the integration of epistemic goals within this framework provides a natural way to specify what the agent needs to understand about the user and the conversation context. By modeling these knowledge requirements as explicit goals, designers can systematically identify the information-gathering points in the dialogue flow. On the other hand, the Graph of Thoughts architecture enables designers to implement specific verification points in the reasoning process where the agent's understanding and decision-making can be validated before proceeding with the dialogue.

Our experience suggests that this design approach could be generalized into a methodology centred around three key principles: 1) *Ensuring* that the decision-making process is transparent and controllable; 2) *Providing* a systematic way to manage the agent's knowledge acquisition process; 3) *Integrating* verification points to enable quality control of the dialogue flow.

This represents the first step in the comprehensive development of this methodology. Questions remain about the optimal granularity of reasoning steps, the balance between structured reasoning and natural conversation flow, and the scalability of this approach to more complex dialogue scenarios. Additionally, while our travel agent implementation demonstrates the feasibility of this approach, its application to other domains still requires further investigation.

Acknowledgments: this study was Funded by the European Union (Multilingual and Cross-cultural interactions for context-aware, and bias-controlled dialogue systems for safety-critical applications (ELOQUENCE) project, Grant agreement No. 101135916). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Commission-EU. Neither the European Union nor the granting authority can be held responsible for them.

Disclosure of Interests: the authors have no competing interests to declare that are relevant to the content of this article.

References

[1] Y. Deng, L. Liao, L. Chen, H. Wang, W. Lei, T.-S. Chua, *Prompting and evaluating large language models for proactive dialogues: Clarification, target-guided, and non-collaboration*, arXiv:2305.13626 (2023).

[2] Y. Deng, W. Lei, M. Huang, T.-S. Chua, *Rethinking Conversational Agents in the Era of LLMs: Proactivity, Non-collaborativity, and Beyond*, Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region (2023) 298–301.

[3] L. Sabatucci, M. Cossentino, Supporting dynamic workflows with automatic extraction of goals from bpmn, ACM Transactions on Autonomous and Adaptive Systems (TAAS) 14 (2019) 1–38.
[4] L. Sabatucci, M. Cossentino, From means-end analysis to proactive means-end reasoning, in: 2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, IEEE, 2015, pp. 2–12.
[5] M. Cossentino, L. Sabatucci, J. Mylopoulos, Consciousness as a trigger to adaptation, Journal of Artificial Intelligence and Consciousness 10 (2023) 27–47.
[6] M. Besta, N. Blach, A. Kubicek, R. Gerstenberger, M. Podstawski, L. Gianinazzi, J. Gajda, T. Lehmann, H. Niewiadomski, P. Nyczyk, T. Hoefler, *Graph of thoughts: Solving elaborate problems with large language models*, Proceedings of the AAAI Conference on Artificial Intelligence 38 (2024) 17682–17690.
[7] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. V. Le, D. Zhou, *Chain-of-thought prompting elicits reasoning in large language models*, Advances in neural information processing systems 35 (2022) 24824–24837.
[8] J. Long, *Large language model guided tree-of-thought.*, arXiv:2305.08291 (2023).
[9] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, K. Narasimhan, *Tree of thoughts: Deliberate problem solving with large language models*, Advances in Neural Information Processing Systems 36 (2024).
[10] S. Minaee, T. Mikolov, N. Nikzad, M. Chenaghlu, R. Socher, X. Amatriain, J. Gao, *Large Language Models: A Survey*, arXiv:2402.06196 (2024).
[11] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al., Retrieval-augmented generation for knowledge-intensive nlp tasks, Advances in Neural Information Processing Systems 33 (2020) 9459–9474.
[12] S. K. Venero, B. Schmerl, L. Montecchi, J. C. Dos Reis, C. M. F. Rubira, Automated planning for supporting knowledge-intensive processes, in: Enterprise, Business-Process and Information Systems Modeling, Springer, 2020, pp. 101–116.
[13] M. Pistore, A. Marconi, P. Bertoli, P. Traverso, Automated composition of web services by planning at the knowledge level, in: IJCAI, 2005, pp. 1252–1259.
[14] P. Giorgini, M. Kolp, J. Mylopoulos, M. Pistore, The tropos methodology, Methodologies and Software Engineering for Agent Systems (2004) 89–106.
[15] J. Mylopoulos, L. Chung, E. Yu, From object-oriented to goal-oriented requirements analysis, Communications of the ACM 42 (1999) 31–37.
[16] S. Liaskos, S. A. McIlraith, S. Sohrabi, J. Mylopoulos, Representing and reasoning about preferences in requirements engineering, Requirements Engineering 16 (2011) 227–249. doi:10.1007/s00766-011-0129-9.
[17] L. Feng-Lin, J. Horkoff, J. Mylopoulos, A. Borgida, G. Guizzardi, R. Guizzardi, L. Lin, Non-functional requirements as qualities, with a spice of ontology, in: 22nd IEEE International Requirements Engineering Conference (RE'14), 2014.
[18] L. Chung, B. Nixon, E. Yu, J. Mylopoulos, Non-Functional Requirements in Software Engineering, Kluwer Publishing, 2000.
[19] J. Mylopoulos, L. Chung, B. Nixon, Representing and using non-functional requirements: A process-oriented approach, IEEE Transactions on Software Engineering 18(6) (1992) 483–497.
[20] V. E. S. Souza, A. Lapouchnian, W. N. Robinson, J. Mylopoulos, Awareness requirements for adaptive systems, in: Proceedings of the 6th international symposium on Software engineering for adaptive and self-managing systems, 2011, pp. 60–69.
[21] V. E. S. Souza, A. Lapouchnian, J. Mylopoulos, (requirement) evolution requirements for adaptive systems, in: 2012 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2012, pp. 155–164.
[22] J. Whittle, P. Sawyer, N. Bencomo, B. H. C. Cheng, J.-M. Bruel, RELAX: a language to address uncertainty in self-adaptive systems requirement, Requirements Engineering 15 (2010) 177–196.
[23] M. Salehie, L. Tahvildari, Self-adaptive software: Landscape and research challenges, ACM transactions on autonomous and adaptive systems (TAAS) 4 (2009) 1–42.

[24] L. Sabatucci, M. Cossentino, From means-end analysis to proactive means-end reasoning, in: Proceedings of 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, Florence, Italy, 2015.

[25] E. S. Yu, Modeling organizations for information systems requirements engineering, in: [1993] Proceedings of the IEEE International Symposium on Requirements Engineering, IEEE, 1993, pp. 34–41.

[26] E. S. Yu, J. Mylopoulos, Understanding" why" in software process modelling, analysis, and design, in: Proceedings of 16th international conference on software engineering, IEEE, 1994, pp. 159–168.

[27] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, J. Mylopoulos, Tropos: An agent-oriented software development methodology, Autonomous Agents and Multi-Agent Systems 8 (2004) 203–236.

[28] D. Rosenthal, Consciousness and confidence, Neuropsychologia 128 (2019) 255–265. Neural Routes to Awareness in Vision, Emotion and Action: A tribute to Larry Weiskrantz.

[29] D. Smithies, The epistemic role of consciousness, Philosophy of Mind, 2019.