Who Holds the Agency? Meta-Design in the Age of Generative AI

Daniela Fogli[1], Luigi Gargioni[1] and Davide Guizzardi[1]

[1]University of Brescia - Department of Information Engineering, Via Branze 38, Brescia, 25123, Italy

Abstract

The increasing adoption of Large Language Models (LLMs) in healthcare opens new opportunities for supporting caregivers and patients, but also raises important challenges related to reliability, transparency, and human control. In this work, we explore how a meta-design perspective can guide the development of AI-assisted End-User Development (EUD) environments that enable non-technical users to define and evolve system behaviors through natural language interaction. We present a conversational system that allows caregivers to create and modify therapy plans for patients through interaction with an LLM-based assistant. The system combines the semantic reasoning capabilities of LLMs with deterministic validation mechanisms and structured knowledge management to ensure safety, consistency, and traceability. Based on this design experience, the paper outlines a set of design implications for AI-assisted EUD systems, highlighting how meta-design principles can help balance the flexibility of generative AI with the need to preserve transparency, accountability, and meaningful user agency in safety-critical domains.

Keywords

End-User Development, AI-Assisted Development, Meta-Design, Large Language Models

1. Introduction

With the rapid growth of the aging population, the development of technologies for medical support and ambient assisted living has become increasingly urgent. In this context, the adoption of Large Language Models (LLMs) to promote healthy aging and support elderly patient care is gaining growing attention in the literature [1, 2, 3, 4]. Kaliappan et al. [1] examine how interaction with LLMs can enhance healthcare delivery and improve older adults' quality of life. The study highlights how conversational interfaces can lower technological access barriers for older users, but also raises concerns related to privacy and data security. At the same time, LLMs may assist healthcare professionals, such as clinicians and caregivers, in tasks including diagnosis, dietary monitoring, and recommendation of physical exercises. However, risks of misdiagnosis and misinformation remain, due to issues such as non-reproducible outputs, hallucinations, and the intrinsic opacity of LLMs. The review by Busch et al. [3] explores LLM applications in the medical domain, focusing on their ability to answer patient questions, summarize or translate medical documents, and support clinical documentation workflows. The authors identify a research gap in the use of LLMs to directly support patients and caregivers in disease management. They also underline important limitations of current approaches, many of which rely on proprietary models that may introduce inconsistencies or unreliable information when updated. To mitigate this issue, they advocate the adoption of open-source LLMs. Another important finding of this review is that nearly half of the surveyed studies reported concerns related to LLM safety, a critical issue in healthcare, thereby reinforcing the need for human oversight to prevent adverse outcomes. LLM-based agents for healthy aging are also discussed by Li et al. [4], who emphasize their potential to integrate health data with decision-making knowledge to support patient care. For instance, Parmanto et al. [2] developed a Caregiving Language Model (CaLM) that leverages Retrieval-Augmented Generation (RAG) to improve the reliability of LLM-generated responses. Their project aims to support family caregivers

Proceedings of the 10th International Workshop on Cultures of Participation in the Digital Age (CoPDA 2026): Exploring the Relationship between EUD, AI-Assisted Development, and Meta-Design, June 2026, Venice, Italy.

daniela.fogli@unibs.it (D. Fogli); luigi.gargioni@unibs.it (L. Gargioni); davide.guizzardi@unibs.it (D. Guizzardi)

0000-0003-1479-2240 (D. Fogli); 0000-0003-4354-916X (L. Gargioni); 0009-0000-7761-7103 (D. Guizzardi)

of individuals with Alzheimer's disease and related dementias by answering questions about disease implications and providing guidance on managing their relatives' daily care.

Building on similar premises, we investigate the potential of LLMs to support caregivers in managing therapies for different patients by framing therapy creation and updating as End-User Development (EUD) activities [5, 6]. In the considered scenario, caregivers, who are typically non-programmers, interact with a conversational assistant to define, modify, and maintain patient therapy plans that include activities such as medication intake, physical exercises, and daily routines. Therapies must comply with medical prescriptions while also addressing patients' contingent needs and health conditions, such as allergies, potential drug side effects, and comorbidities. In this sense, the therapy acts as high-level executable instructions (i.e., programs): it encodes a sequence of actions and constraints that can later be parsed and executed by assistive technologies such as reminder systems, conversational assistants, or social robots [7]. The conversational assistant, therefore, serves as a development environment in which users incrementally construct and adapt these behavioral specifications through natural language interaction.

Given the aforementioned limitations of LLMs, a socio-technical system supporting LLM-assisted EUD of therapies must incorporate well-defined boundaries and verification checkpoints. On the one hand, these mechanisms should ensure correct and predictable system behavior; on the other hand, they must preserve user control over decisions and provide transparent explanations of the system's underlying reasoning processes. We argue that meta-design can offer a valuable framework for integrating EUD with LLM-based systems. Meta-design is a socio-technical approach that aims to empower end users in the continuous evolution of their systems [8, 9, 10]. In this position paper, we claim that, in the AI era, meta-design plays a pivotal role in steering the evolutionary trajectory of LLM-based systems. Therefore, beyond describing the system and its architecture, the paper reflects on the underlying design decisions and distills a set of design implications for AI-assisted EUD from a meta-design perspective. Specifically, it identifies principles for creating systems that enable non-technical users to define and evolve structured behaviors through natural language interaction, while ensuring transparency, human oversight, and meaningful user agency. These design implications contribute to the broader research agenda on meta-design and AI-assisted EUD by outlining strategies to balance automation with human control in complex and evolving socio-technical environments.

2. Aims and Assumptions

This work aims to explore how meta-design can shape AI-assisted EUD in complex and sensitive domains such as healthcare. Within this perspective, the work adopts a meta-design approach in which the system is not intended to autonomously manage therapies, but rather to provide an environment that empowers users to iteratively shape and refine them. The system architecture, therefore, combines the flexibility of LLMs with deterministic tools, structured data representations, and validation mechanisms that constrain the model's behavior and avoid hallucinations. In particular, tailored system prompts, tool-calling interfaces, and structured knowledge management enable a form of configurable transparency in which system actions, detected inconsistencies between existing therapy and the user's requested updates (simply called "conflicts" in the following), and proposed modifications are explicitly forwarded to the user.

A central assumption of the proposed design is that reliable AI-assisted EUD requires maintaining the caregiver in the loop. All modifications to therapies are staged and require explicit confirmation, while conflict detection and semantic checks provide explanations that support informed decision-making. This human-in-the-loop validation process mitigates risks of overreliance while preserving user agency over the therapy design process. Furthermore, knowledge extracted from interactions is stored in structured representations and retrieved through a RAG pipeline, enabling both the user and the system to operate on a consistent, auditable knowledge base.

By combining conversational interaction, structured knowledge management, and explicit validation workflows, this work investigates how LLMs can serve as cognitive partners that augment, rather

than replace, human decision-making. In this sense, the project contributes to ongoing discussions on meta-design and cultures of participation in the AI era, illustrating how AI-assisted EUD environments can be designed to support transparency, accountability, and human agency in socio-technical systems.

3. The Proposed System

The proposed system is designed as a conversational assistant that allows caregivers to perform a variety of actions on patients' therapies, such as adding new activities, modifying existing ones, or removing activities that are no longer needed.

A central design principle of the system is the combination of the contextual reasoning capabilities of LLMs with deterministic functions. When evaluating the safety of user requests with respect to patients' conditions and existing activities, the system leverages the LLM for tasks that require semantic interpretation (e.g., assessing whether a medication is compatible with a patient's medical conditions). Conversely, deterministic functions are used when algorithmic and structured checks are required (e.g., verifying whether a scheduled activity temporally overlaps with existing ones).

3.1. Therapy Structure

In the system, a therapy consists of a structured set of patient-specific information and a schedule of activities designed to support the patient's daily routine. The patient data includes demographic information (i.e., name, birth date, and gender) and the list of their medical condition (e.g., diabetes, insomnia, renal failure). Each activity is defined as a structured JSON object having various fields, including a name, a description, a time, and a set of days during which the activity takes place. Activities in the system recur on a weekly basis; therefore, the "days" field specifies weekdays rather than specific calendar dates. Additionally, an activity can have validity dates (i.e., valid from and valid until) that define the period in which the activity must be considered part of the patient's therapy. Activities may also have dependencies, indicating that certain tasks must occur before others (e.g., having a meal before taking a medicine). Listing 1 presents an example of a "Light walk after lunch" activity that takes place from Monday to Friday at 15:15 and that depends on a successful blood glucose measurement.

Listing 1: Example of activity representation

```json
{
    "activity_id": "lb_001",
    "day_of_week": ["Mon","Tue","Wed","Thu","Fri"],
    "dependencies": ["Blood Glucose Measurement"],
    "description": "Light walk after glucose check",
    "duration": 60,
    "name": "Light walk after lunch",
    "time": "15:15",
    "valid_from": "10-03-2026",
    "valid_until": "24-03-2026"
}
```

3.2. Architecture

The system is implemented entirely in Python and integrates several components that support conversational interaction, knowledge management, and therapy validation. At the core of the architecture is an LLM hosted locally through the *Ollama*[1] framework and accessed via its API. OpenAI *gpt-oss-20b*[2] is the chosen model for the implementation. This choice is motivated by two main factors. First, the model is open source, allowing for full local development and data privacy. Since the system may

[1]Ollama - https://ollama.com
[2]gpt-oss-20b - https://openai.com/index/introducing-gpt-oss

process sensitive patient information, it is essential to ensure that all data remains within the local infrastructure. Running the model locally avoids sending patient data to external cloud services and allows full control over how the model is deployed and managed. Second, the model provides a suitable trade-off between size and performance: it is sufficiently capable of generating reliable responses and interpreting user requests while maintaining relatively short response times when deployed locally.

To support system operation, the architecture includes a knowledge management module responsible for storing and retrieving structured and unstructured information used during interactions. Then, the system includes several deterministic functions implemented in Python. These functions provide algorithmic capabilities that complement the contextual reasoning abilities of the LLM. Some examples of these deterministic functions include:

- the detection of temporal overlaps between the scheduled time of an activity the user wants to add and an already existing activity;
- the suggestion of alternative time slots for an activity when overlaps are detected;
- the detection of situations in which an activity is scheduled before another activity that is defined as its dependency;
- the detection of the removal of an activity that is required as a dependency for another activity.

Lastly, the application orchestrates the interaction among all system components. It receives user messages, forwards them to the LLM, exposes the available tools that can be invoked during request processing, and returns the generated responses and results back to the user interface.

3.3. User Interface

User interaction with the system occurs through a Single-Page Application (SPA) that serves as the main access point for caregivers. The page is implemented using *Stremlit*[3], a Python framework that enables the development of interactive web applications using pre-configured graphic components. The interface (presented in Figure 1) is organized into two main sections. On the left side, accessible through a collapsible drawer, contextual information about the currently selected patient is displayed, including the patient's name, age, medical conditions, and a list of activities that compose their therapy. Caregivers can change the patient by using the dropdown element at the top of the collapsible drawer. On the right side, a conversational chat panel allows caregivers to interact directly with the application, sending requests and receiving responses in natural language. This layout allows caregivers to access relevant information while interacting with the conversational assistant, avoiding the need to prompt the model for patient data and the state of the therapy. Moreover, data updates in real time provide visual feedback on the action the caregiver performed.

3.4. Knowledge Management

The proposed system implements a dual-store knowledge architecture that explicitly separates *(i)* structured, transactional data from *(ii)* semantic, retrieval-oriented knowledge.

Patient demographics and therapy plans are persisted in a relational SQL database (i.e., *PostgreSQL*[4]) because they require strong consistency, explicit schema constraints, and auditable tracking. By contrast, a vector database (i.e., *ChromaDB*[5]) is used for knowledge that is naturally expressed as free text and is primarily consumed through similarity search rather than exact key queries, namely medicines descriptions, history of critical events, preference statements, and previously therapy management resolved conflicts. This vector database enables approximate matching, which is essential to retrieve relevant context even when the caregiver's wording varies.

The vector database uses the ChromaDB default embedding model (i.e., *Sentence Transformers all-MiniLM-L6-v2*[6]) and cosine distance indexing; each record is stored as a document enriched with

[3]Streamlit - https://streamlit.io
[4]PostgreSQL - https://www.postgresql.org
[5]ChromaDB - https://www.trychroma.com
[6]Sentence Transformers all-MiniLM-L6-v2 - https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

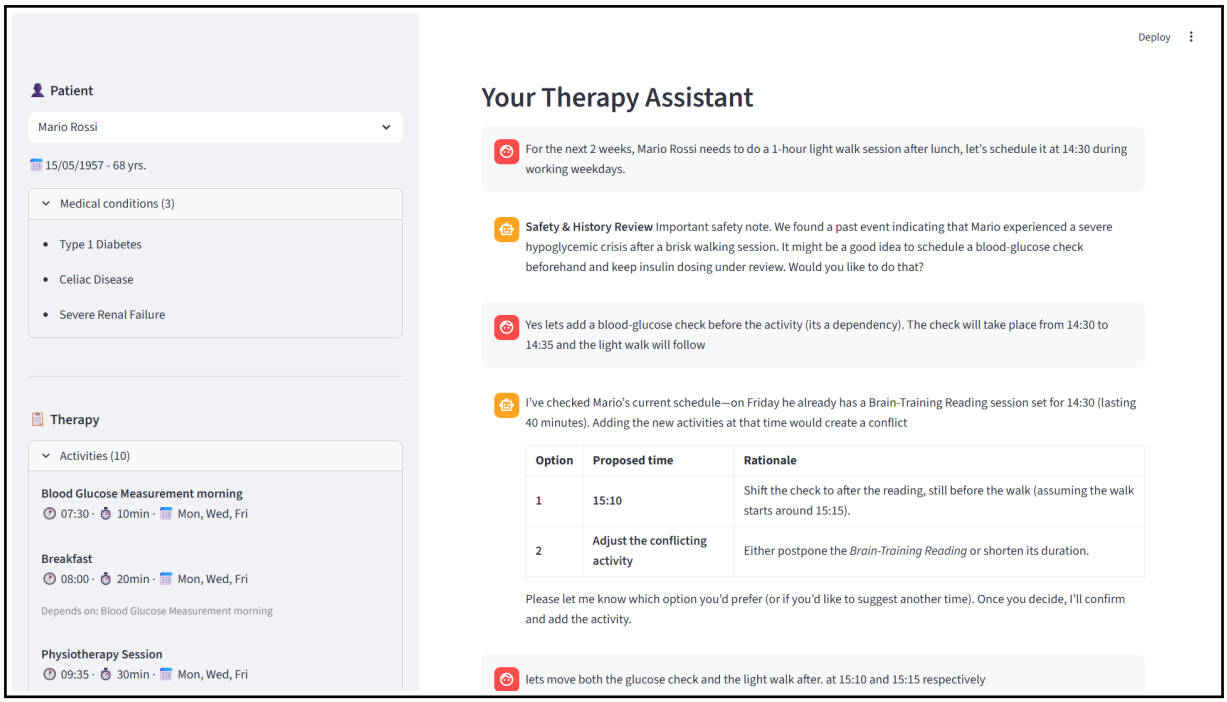

Figure 1: User interface of the system

metadata (e.g., patient identifier, category, date) to support patient-scoped retrieval and downstream filtering. The update policy distinguishes between static and evolving knowledge. Medicine descriptions and the patient event history are treated as read-only during runtime because they originate from curated or externally validated sources. Allowing automatic updates to these records could introduce knowledge drift and reduce traceability. Conversely, patients' preferences and conflict-resolution patterns are read–write records because they represent evolving, patient and caregiver-specific operational knowledge that should adapt over time.

Retrieval follows a RAG pattern: when the assistant needs background knowledge, it calls the specific Python tool to perform semantic queries on the appropriate collection[7] of the vector database and injects the retrieved documents into the LLM context. The vector database is organized into four collections, each representing a distinct knowledge source for retrieval. The *medicines* collection stores pharmacological monographs to support medication-related queries. The *patient_history* collection contains patient-specific historical safety events described in natural language and associated with the relevant activity. The *conflict_resolutions* collection stores textual descriptions of previously encountered therapy-management conflicts along with their resolutions, enabling retrieval of similar past cases. Finally, *patient_preferences* contains statements capturing patient habits and preferences, which can be retrieved to support personalized suggestions.

When retrieving medicine information, the system first attempts an exact name-based match. If no exact match is found, the system accepts only results within a predefined similarity distance threshold in order to avoid attributing properties to a medicine that has not been explicitly documented in the knowledge base.

Retrieval from the *patient_history* collection applies a stricter similarity threshold than other collections, so that only events closely related to the current query are included in the LLM context.

Similarly, when the caregiver proposes an action potentially conflicting with the existing therapy plan, the system queries the *conflict_resolutions* collection, constrained to the current patient, to surface previously accepted modifications, and queries the preference collection via semantic search to personalize suggestions and be proactive.

[7]In ChromaDB, data is organized into *collections*, the database-specific term for named groups of documents that can be queried independently.

Knowledge updates are performed only at session end as a controlled consolidation step. The system analyses the full conversation to extract two knowledge types, patient preferences and conflict resolutions, and stores them in the vector database as documents used for future retrieval. To avoid redundant or contradictory knowledge from accumulating, before inserting a new record, the system searches for near-duplicates within the same patient scope; if a new item is essentially related to the same concept as an existing one, the old record is replaced.

In contrast, the SQL database uses an append-only policy: the current therapy plan is saved as a new versioned snapshot to preserve an auditable history of what was scheduled.

This separation between the relational database for structured transactional data and the vector database for semantic knowledge keeps static knowledge stable, allows experiential knowledge to evolve consistently, and preserves transactional state with full traceability.

3.5. Interaction Flow

Figure 2 presents a UML sequence diagram of the operations needed to complete a user request. To further improve clarity, the analysis of the diagram is integrated with a real example from a conversation that one of the authors had on the 10th of March. The result was the creation of the activity in Listing 1. A portion of the conversation can also be found in Figure 1.

The flow begins when the caregiver sends a request to the system (*Step 1 - User Request*). For example, they may ask the system:

> *"For the next 2 weeks, Mario Rossi needs to do a 1-hour light walk session after lunch, let's schedule it at 14:30 during working weekdays."*

Once the request is received, the LLM interprets the request to understand the user's intent and determine what information or actions may be required. To obtain relevant context, the LLM calls Python tools to retrieve data from the vector database (*Step 2 - Extraction of data from RAG*), which returns contextual information that can support the request interpretation. In the example interaction, the system calls a retrieval tool to search for relevant patient history events related to walking activity:

```
get_patient_history_events({"query": "light walk session after lunch
14:30 1 hour"})
```

The retrieved information is used during the semantic conflict check (*Step 3 - Semantic conflicts check*) to determine whether the user's request is safe for the patient in light of their medical condition and the retrieved contextual information. If a semantic conflict is detected, the system initiates an interactive conflict resolution procedure with the user (*Step 3.a - Interactive conflict resolution*), allowing the user to clarify or adjust the request before proceeding. In this case, the system finds a previous hypoglycemic crisis associated with physical exercise. As a result, it warns the caregiver and requests confirmation before proceeding:

> *"Important safety note. We found a past event indicating that Mario experienced a severe hypoglycemic crisis after a brisk walking session. It might be a good idea to schedule a blood-glucose check beforehand and keep insulin dosing under review. Would you like to do that?"*

The caregiver decides to follow the assistant's suggestion and creates a second activity:

> *"Yes, let's add a blood-glucose check before the activity. The check will take place from 14:30 to 14:35, and the light walk will follow."*

If no semantic conflict is found (or it was solved by the user, as in this case), the process continues with a deterministic check (*Step 3.b - Deterministic conflicts check*). This check evaluates potential overlaps among activities, guarantees dependency consistency, and, overall, ensures that caregivers' requests

remain within system constraints. The decision to perform a semantic check first and a deterministic check afterwards is motivated by a desire to prioritize patients' safety and caregivers' time. An activity that is not safe for a patient according to their medical conditions would never be accepted by the system; therefore, performing additional checks on its scheduling would be unnecessary and could introduce additional steps that may be frustrating for caregivers who need to address the resulting issues.

If a deterministic conflict is detected, the system once again triggers interactive conflict resolution (*Step 4.a - Interactive conflict resolution*), providing, if possible, suggestions that can guide the user in resolving the conflict. In our example, the deterministic check identifies that an activity already exists at the requested time. In particular, the patient has a "Brain-Training Reading" session scheduled on Friday at 14:30. As a result, the system informs the caregiver about the overlap and proposes possible ways to resolve it:

> *"I've checked Mario's current schedule, on Friday he already has a Brain-Training Reading session set for 14:30 (lasting 40 minutes). Adding the new activities at that time would create a conflict. How would you like to handle this overlap?*
>
> *1. Shift the check and the Light Walk after the reading. The proposed time is 15:10 for the check and 15:15 for the light walk.*
>
> *2. Adjust the conflicting activity. Either postpone the Brain-Training Reading or shorten its duration."*

Due to the impossibility of changing the *Brain-Training Reading* schedule, given the specialist's limited availability, the caregiver decides to adjust the activities according to the first suggestions, solving the detected overlap.

Once the deterministic conflict is solved (or if it never occurred), the workflow proceeds to action execution (*Step 4.b - Action execution*), where the appropriate tool performs the requested operation. To add the requested activities to Mario Rossi's schedule, the assistant invokes the *add_therapy_activity* tool:

```
add_therapy_activity(..., "name": "Blood Glucose Measurement", time:
"15:10", "valid_until": "24-03-2026")
add_therapy_activity(..., "name": "Light walk", time: "15:15",
"valid_until": "24-03-2026", "dependencies": ["Blood Glucose
Measurement"])
```

Once the action is completed, the tools return an action success message, which the LLM then interprets (*Step 5 - Results interpretation*) and reports back to the user (*Step 6 - Confirmation to user*) as follows:

> *"Both activities have been scheduled for the next two weeks. They respect the patient's medical history – a blood glucose check precedes any walk, helping to avoid the previously recorded hypoglycemic event after late-afternoon exercise."*

This confirmation concludes the request-processing workflow.

4. Design Implications

A meta-design perspective has been adopted in the design of the proposed system, considering the need for empowering end users without technical knowledge in creating and updating therapy plans, but also the necessity of providing them with mechanisms for easily managing the evolution of these plans over time while coping with the limitations of the generative AI component.

We claim that meta-design of AI-assisted EUD environments must incorporate appropriate "fences" to mitigate challenges such as non-determinism, inaccuracy, and hallucinations, while preserving human

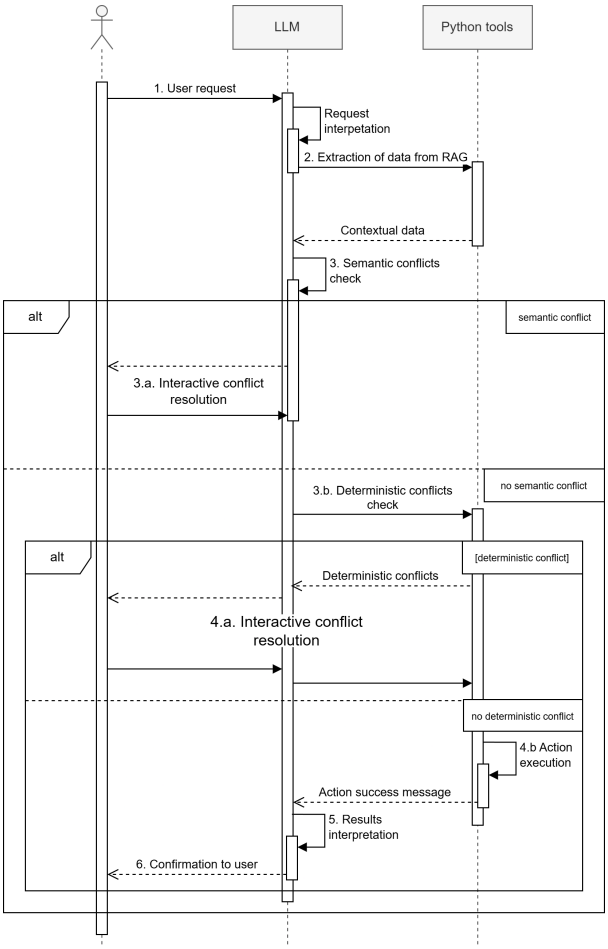

Figure 2: UML Sequence diagram representing the interpretation and execution of a user request

agency, particularly in safety-critical domains such as healthcare. This fosters user participation in EUD activities, with the reassurance that system behavior is always programmatically controlled and that its decisions are never made automatically without user evaluation.

This design experience led us to reflect on the design of AI-assisted EUD environments and preliminarily identify the following design implications.

Combining generative AI with deterministic mechanisms. While LLMs are effective at interpreting natural language and reasoning over contextual information, their non-deterministic nature introduces risks such as hallucinations and inconsistent outputs. Systems that support EUD in sensitive domains should therefore integrate generative models with deterministic mechanisms that enforce structural constraints and validation rules. In our architecture, the LLM is responsible for semantic interpretation and contextual reasoning, while deterministic functions handle verifiable checks such as temporal overlaps, activity dependencies, and system constraints. This hybrid approach limits unreliable behaviors while preserving the flexibility of conversational interaction.

Designing transparent and human-controlled validation workflows. AI-assisted EUD systems should explicitly support human oversight by making system reasoning and potential conflicts visible to users. Rather than autonomously executing requested changes, the system stages modifications and presents them for confirmation. Detected conflicts are explained and, when possible, accompanied by suggested alternatives that help users resolve them. Such validation workflows maintain caregivers in the loop and support informed decision-making, preventing over-reliance on automated outputs.

Supporting structured and evolving knowledge. Systems should combine stable representations of structured data with mechanisms that allow experiential knowledge to evolve through interaction. In the proposed design, therapy plans and patient information are stored in a relational database to ensure

consistency and traceability, while a vector database supports the retrieval of contextual knowledge such as patient preferences and previously resolved conflicts. This separation enables the system to maintain reliable core data while gradually incorporating knowledge emerging from user interactions.

Leveraging conversational interaction to support easier end-user development. Conversational interaction could be exploited to create EUD environments for defining and evolving a system behavior in an easy and intuitive manner. For example, through natural language interaction, caregivers can progressively construct and adapt therapy plans that act as executable specifications for assistive technologies. Conversational interfaces should therefore provide contextual information, real-time feedback, and clear representations of system state to support users in understanding and controlling the effects of their actions.

5. Conclusion

This paper discussed how a meta-design perspective can inform the development of AI-assisted EUD environments based on LLMs. Through the case study of therapy management for elderly patients, we illustrated how conversational AI can support caregivers in defining and evolving therapy plans while maintaining human oversight and system reliability. The proposed architecture shows how the integration of LLMs with deterministic validation mechanisms and structured knowledge management can help balance the flexibility of natural language interaction with the need for safety and transparency. Future work will focus on evaluating the proposed approach in real-world scenarios and exploring further design strategies to improve transparency, explainability, and user control in AI-assisted EUD environments.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT and Grammarly in order to: grammar and spelling check, paraphrase, and reword. After using these services, the authors reviewed and edited the content as needed; thus, they take full responsibility for the publication's content.

References

[1] S. Kaliappan, A. S. Anand, K. Saha, R. Karkar, Exploring the role of llms for supporting older adults: Opportunities and concerns, 2024. URL: https://arxiv.org/abs/2411.08123. arXiv:2411.08123.

[2] B. Parmanto, B. Aryoyudanta, T. W. Soekinto, I. M. A. Setiawan, Y. Wang, H. Hu, A. Saptono, Y. K. Choi, A reliable and accessible caregiving language model (calm) to support tools for caregivers: Development and evaluation study, JMIR Form Res 8 (2024) e54633. doi:10.2196/54633.

[3] F. Busch, L. Hoffmann, C. Rueger, E. H. C. van Dijk, R. Kader, E. Ortiz-Prado, M. R. Makowski, L. Saba, M. Hadamitzky, J. N. Kather, D. Truhn, R. Cuocolo, L. C. Adams, K. K. Bressem, Current applications and challenges in large language models for patient care: a systematic review, Communications Medicine 5 (2025) 26. doi:10.1038/s43856-024-00717-2.

[4] P. Li, J. Wu, S. Shang, Q. Zhan, Harnessing large language model agents for healthy aging, Medicine Plus 2 (2025) 100084. doi:https://doi.org/10.1016/j.medp.2025.100084.

[5] H. Lieberman, F. Paternò, M. Klann, V. Wulf, End-User Development: An Emerging Paradigm, in: H. Lieberman, F. Paternò, V. Wulf (Eds.), End User Development, volume 9, Springer Netherlands, Dordrecht, 2006, pp. 1–8. doi:10.1007/1-4020-5386-X_1, series Title: Human-Computer Interaction Series.

[6] B. R. Barricelli, F. Cassano, D. Fogli, A. Piccinno, End-user development, end-user programming and end-user software engineering: A systematic mapping study, Journal of Systems and Software 149 (2019) 101–137. doi:10.1016/j.jss.2018.11.041.

[7] L. Gargioni, R. Alami, D. Fogli, Towards a hybrid llm/model-based architecture for robot coaching: An instance of human-machine collaboration, in: CEUR Workshop Proceedings, volume 4101 of

CEUR Workshop Proceedings, CEUR-WS.org, 2025, p. 8. URL: https://ceur-ws.org/Vol-4101/paper6.pdf.

[8] G. Fischer, E. Giaccardi, Meta-design: A Framework for the Future of End-User Development, in: H. Lieberman, F. Paternò, V. Wulf (Eds.), End User Development, Springer Netherlands, Dordrecht, 2006, pp. 427–457. doi:10.1007/1-4020-5386-X_19.

[9] G. Fischer, D. Fogli, A. Piccinno, Revisiting and broadening the meta-design framework for end-user development, in: New Perspectives in End-User Development, Springer, Cham, Switzerland, 2017, pp. 61–97.

[10] B. R. Barricelli, G. Fischer, D. Fogli, A. Morch, A. Piccinno, S. Valtolina, Advancing the integration of artificial intelligence in meta-design, in: Proceedings of the 2024 International Conference on Advanced Visual Interfaces, ACM Press, New York, USA, 2024, pp. 1–5.