arXiv:2605.02384v3 [cs.SE] 24 Jun 2026

This is a preprint.

The peer-reviewed paper has been published in the proceedings of ICWE 2026.

Please reference the published version.

DOI: https://doi.org/10.1007/978-3-032-29372-5_4

A Low-Code Approach for the Automatic Personalization of Conversational Agents

Aaron Conrardy
Luxembourg Institute of Science and Technology, Luxembourg
University of Luxembourg, Luxembourg
aaron.conrardy@list.lu

Alfredo Capozucca
University of Luxembourg, Luxembourg
alfredo.capozucca@uni.lu

Jordi Cabot
Luxembourg Institute of Science and Technology, Luxembourg
University of Luxembourg, Luxembourg
jordi.cabot@list.lu

Abstract

The rise of Large Language Models (LLMs) has increased the demand for Conversational Agents (CAs) capable of understanding human conversations as part of web applications. While traditional CAs consist of deterministic states, LLMs enhance their capabilities to handle open conversations, handling arbitrary requests. Numerous tools exist that allow non-technical users to create such CAs. Yet, the creation of personalized CAs able to adapt to the profile of end-users to offer an optimal user experience remains in the hands of experienced developers implementing ad-hoc personalizations. In this work, we propose a pipeline that follows a low-code/no-code approach to facilitate the modeling and generation of personalized CAs. A pilot user study was performed to get preliminary results on perceived usability and usefulness and the full pipeline has been implemented on top of an open-source low-code platform.

***K*eywords** Conversational agents · Low-code · Model-driven · Personalization

1 Introduction

Conversational Agents (CAs) are software applications that interact autonomously with end-users in natural language, supporting task completion, conversational web, service delivery, as well as social engagement and entertainment [4, 21, 5]. Known under other names such as chatbots or digital assistants, they have become a ubiquitous component of modern platforms and services. While their adoption was already increasing before the widespread use of Large Language Models (LLMs), recent advances in LLMs have further established CAs as a common additional interaction modality in web applications and more. CAs may support different types of conversations, ranging from predefined, task-oriented dialogues based on deterministic rules to open-ended interactions enabled by LLMs, or a combination of both.

Personalized CAs (PCAs) aim to further elevate the users' experience by personalizing their interaction with the CA. Personalization refers to adapting systems to better fit an individual user's needs, preferences, and context [13] offering an optimal user experience.

At a high level, personalization typically consists of two processes: user profiling and system adaptation [13] where the latter takes care of personalizing the interaction to the specific attributes of the user. For instance, for extroverted users, aligning CAs to act extroverted by adapting communication styles has shown to increase engagement and usage intention [3]. While these processes are well-studied [22], their implementation in CAs remains complex and requires deep technical expertise [24].

In this sense, this paper aims to facilitate the personalization of CAs, enabling non-technical people to define the type of users they want to support, the adaptations to be done for each user type and the generation of the corresponding personalized agents. To do so, we propose a new low-code pipeline to model and automatically generate PCAs. A style

of model-driven development [8, 12], low-code development reduces the need for manual programming by relying on higher-level, graphical models to describe system behavior. Such approaches have already been applied to the development of CAs [5] and, partially, to the personalization of user interfaces [25, 26] but, to the best of our knowledge, not to the challenge of creating PCAs.

The rest of this paper is structured as follows. Section 2 presents an overview of the proposed pipeline. Section 3 presents the modeling of PCAs, followed by Section 4 that describes how these PCA models are transformed into the actual agents. Tooling support is discussed in Section 5. Section 6 discusses a user study to validate our approach. Section 7 presents the related work. The paper is concluded in Section 8.

2 Overview of our low-code pipeline for the development of PCAs

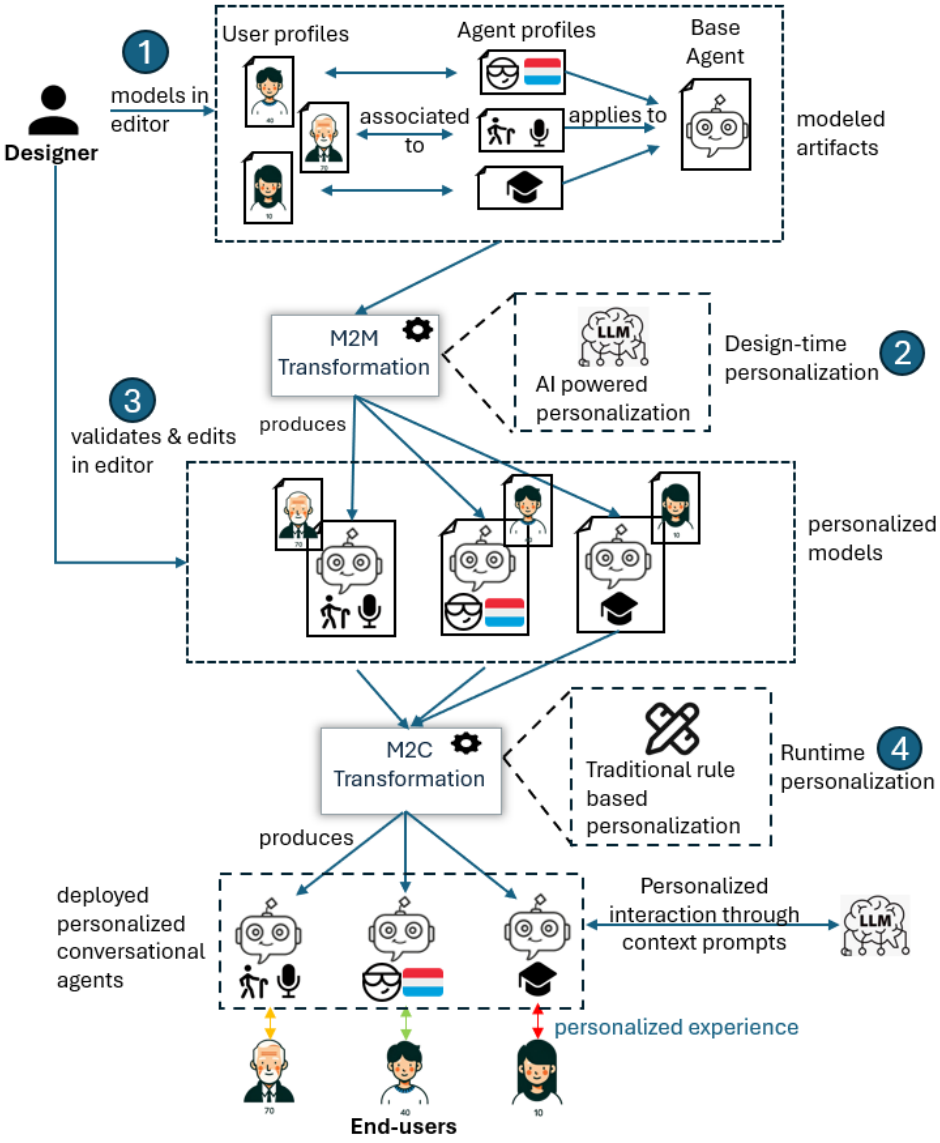

Figure 1: Overview of steps of low-code personalized agent creation pipeline

In this section, we present our low-code pipeline for designing and generating PCAs. Figure 1 illustrates the four main activities:

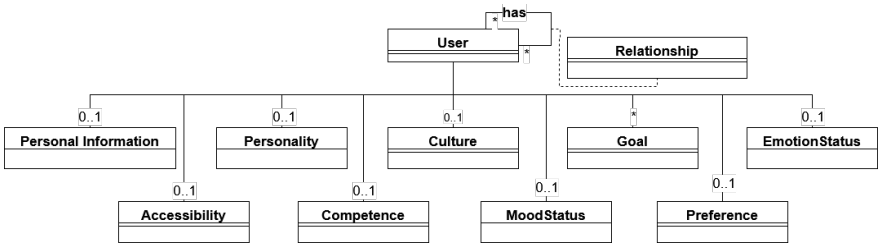

Figure 2: High-level view of user metamodel from [9]

1. **Modeling the user and agent profiles**: The PCA designer models the user profiles (representing the types of users we target in our CA), the base agent model (with the common functionality) and the agent profiles (representing the different agent adaptations to be mapped to the corresponding user profiles). Separating the agent profile from the base agent model makes the agent profiles agent-agnostic, as they can be applied to any agent. Similarly, separating user profiles from agent profiles also enables the mapping of the same agent profile to different users, fostering reusability.
2. **Design time personalization**: Based on the user and agent profiles, the agent model is enriched via a model-to-model (M2M) transformation that adapts any agent answer hardcoded in the model. At this time, adaptations can include translation [27], complexity variation [20] and integration of cultural values [19]) among others. An LLM is used to automate these adaptations. Note that design-time adaptation is always preferred when possible as it improves runtime performance (removing the need for additional calls at every agent request) and it enables the agent validation before putting it into production.
3. **Manual model validation and update**: At this stage, designers can validate the personalized texts in each model and apply manual modifications if needed (e.g., improving the adapted text or adding unique states to one of the personalized agent models for a deeper adaptation). At the end of this phase, we have a set of personalized agent models linked to specific user profiles. The next step will complete the adaptation process.
4. **Runtime personalization**: As part of a final model-to-code (M2C) transformation and, based on the specified customization in the agent profile, appropriate code excerpts are included by the deterministic code-generation templates. This code extends the agent behavior with additional adaptations that will be dynamically triggered when the corresponding user interacts with the agent.

3 Modeling users, agents, and agent profiles

To support the modeling of the previously discussed artifacts, we define a set of interrelated domain-specific languages (DSLs). Each DSL is presented by describing both their abstract (using a metamodel expressed as a UML class diagram) and concrete syntax (using a graphical or form-based notation).

To illustrate our approach, we use as a running example a personalized gym assistant designed to provide accessible and adapted advice to different user profiles. The agent's core task is to provide pre-defined gym-related advice on nutrition and exercises, and be able to handle any other query via generative AI as a fallback. Two user profiles are considered to receive a personalized experience: (1) an elderly user that only speaks their native language, and (2) a paraplegic user who cannot use their legs. The former prefers oral interaction, simple and formal text with a conversation in their mother tongue. The latter must not receive exercise recommendations involving the lower body, guaranteeing that adequate advice is given.

3.1 User modeling

User profile models have the goal to describe the end-user of an application [10]. They can either describe individual users or user categories (e.g., elderly users in our running example) to which then real users are mapped to.

Abstract syntax For this metamodel, we reuse the user metamodel we proposed in [9], with Figure 2 showing a high-level view. The metamodel consolidates results of a Systematic Literature Review (SLR) on user modeling that we performed [10], providing a complete and detailed overview of users via the proposed formalization, allowing the definition of user profiles with various attributes.

Concrete syntax Conforming to the rules of the abstract syntax, as part of this work we have developed a graphical notation to model user profiles.

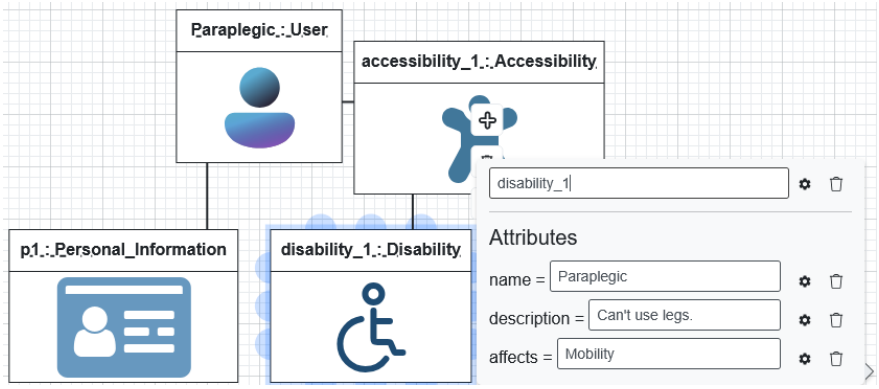

Figure 3: Paraplegic user profile model in graphical notation

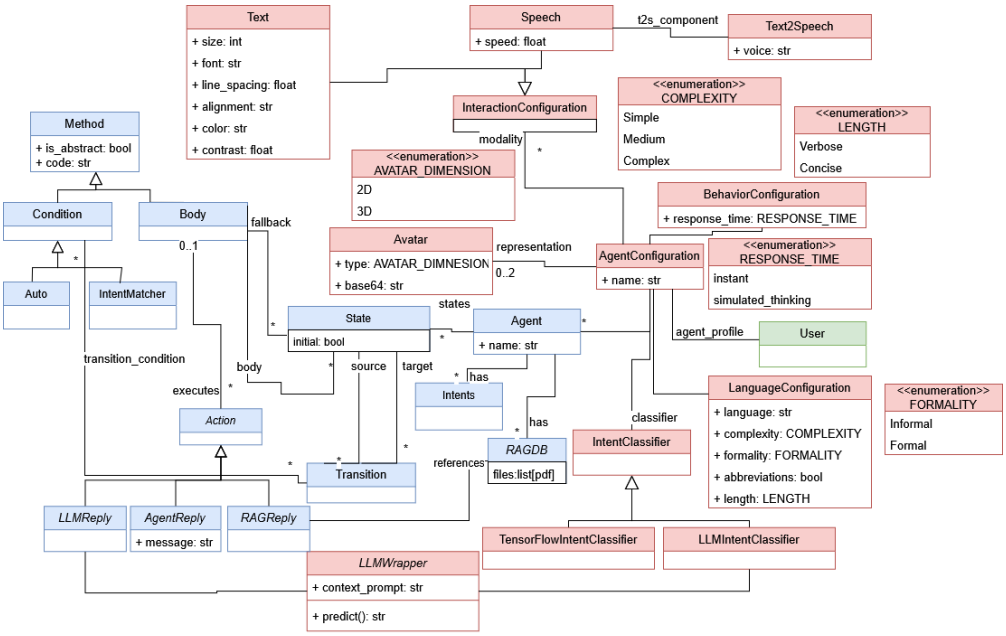

Figure 4: Agent and agent profile metamodel

Figure 3 shows a partial user profile model for our running example, representing the paraplegic user profile, containing the information on the disability and what it affects, as it is relevant for the personalization. The full list of icons of our graphical notation is available in our online tool (see Section 5).

3.2 Agent modeling

The base agent model reflects the conversational states of the agent, thus, its core functionality.

Abstract syntax The metamodel to describe agents is shown in Figure 4 (classes in blue). Note that this is a simplified version, omitting attributes and (sub)classes not relevant to showcase the personalization pipeline, as it follows a formal-state-machine-like syntax commonly used when defining CAs [16].

At its core, the metamodel includes the *State*, *Body* of states, executed *Actions* in a given body, *Transitions* that define when to jump from one state to another once specific *Conditions* are fulfilled, and *Intents* that represent the possible messages an end-user might pose when interacting the agent. Note that for the actions, one can choose between sending a pre-defined response, an AI generated response via an LLM, or an AI generated response using retrieval augmented generation (RAG) based on an available database.

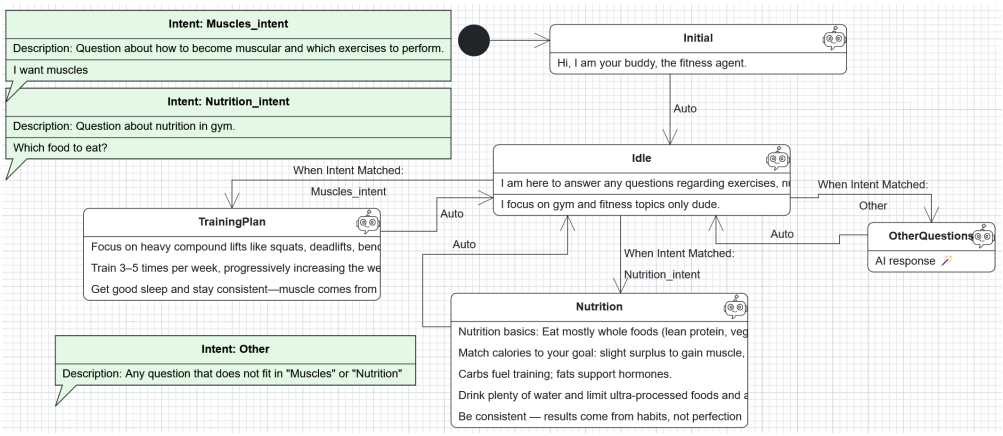

Figure 5: Graphical agent model representing the gym assistant agent

Concrete syntax We propose a graphical notation that strongly follows the UML state machine formalism[1].

Figure 5 displays our running agent example using this graphical syntax. For simplicity, the agent answers only one question before returning to an idle state. The model consists of four states: an initial greeting state that automatically transitions to an idle state, two predefined response states (*TrainingPlan* and *Nutrition*), and an *OtherQuestions* state handled by an LLM. Three intents (*Muscles_intent*, *Nutrition_intent*, and *Other*) control transitions to the corresponding states. After each response, the agent returns to the idle state.

3.3 Modeling agent personalizations

The key element is to model the personalization options we want our PCA to offer to all or some of the user profiles we want to support.

But before deciding which of the adaptations we want to include in our agent, we need to establish the set of possible adaptations available for agents and what benefits each adaptation could potentially bring.

Unfortunately, there is not a general taxonomy of personalizable components suited for CAs. Therefore, and based on the individual personalization ideas published in the literature (e.g, see [14, 22, 1, 17]), we propose in Table 1 a non-exhaustive list of personalizable components in CAs, consolidated into four distinct categories: (1) Content, (2) Presentation, (3) Behavior, and (4) Modality. These four categories are inspired by an assessment scheme that was used in an SLR on PCAs in health care [17], where we renamed *User Interface* to *Presentation*, *Delivery Channel* to *Modality* and *Functionality* to *Behavior*, and broadening their definition in what they capture. A takeaway here is that both the conversation of the agent and the UI it will take place on need to be personalized.

Abstract syntax Based on the taxonomy in Table 1, we have defined the metamodel for modeling agent configurations, visible in Figure 4 (red classes). Related to Modality, *InteractionConfiguration* defines the used modalities. For Presentation, *Text* and *Speech* allow for the specification of stylistic changes related to the chosen channel of communication. *Avatar* defines the visual representation of the CA. *LanguageConfiguration* relates to stylistic changes in the used language. Related to Behavior, *BehaviorConfiguration* lets users change if the agent pretends to type out a message or instantly responds. For Content, it is enough that the information of the user profile is provided, as the semantics will be changed based on that information, which is represented by the association between the agent configuration and the user. Additionally, the components *IntentClassifier* (necessary for natural language understanding), *LLMWrapper*, *RagDB*, *Platform* and *Text2Speech* define technological configurations of the agent, focusing on used tools and methods to run the agent.

Concrete syntax To specify the agent profile, we decided to implement it via a form-based syntax, given the configurational and feature-like nature of the agent configuration. Figure 6 contains the agent configuration in the form-based syntax created based on the requirements for the Elderly profile, where the four personalization categories are represented. The configurable content in each pillar is straightforward, such as setting Language Complexity to *Simple* in Presentation, selecting *Speech* as modalities for both input and output in Modality, or setting the boolean to adapt content to the user profile to true.

[1] https://www.omg.org/spec/UML/2.5.1/About-UML

Category (definition)	Aspect	Example (with reference)
Content (adapts *what* is shown or done, i.e., the semantics of the interaction)	Selection / structure	Adaptive conversation paths [17]; paragraph length adapted to cognitive capabilities [22]
	Enrichment	Response tailored to fit user constraints [17]
Presentation (adapts *how* content is delivered without changing its meaning)	Language and style	Code-switching [22]; simplified text [14]; text formality [1, 22]
	Visual / auditory form	Larger text for elderly users [25]; emotional tone of voice [22]; avatar appearance [14]
Behavior (adapts *how the interaction unfolds* over time)	Interaction management	Response timing, turn-taking, and proactiveness [14, 22]
	Social and repair strategies	Clarification questions [22]; small talk or humor [14]
Modality (adapts the *input/output channels* used for interaction)	Channels	Text, speech, images/video, gestures, or haptics [22, 17]

Table 1: Overview of personalization dimensions with embedded definitions and representative examples.

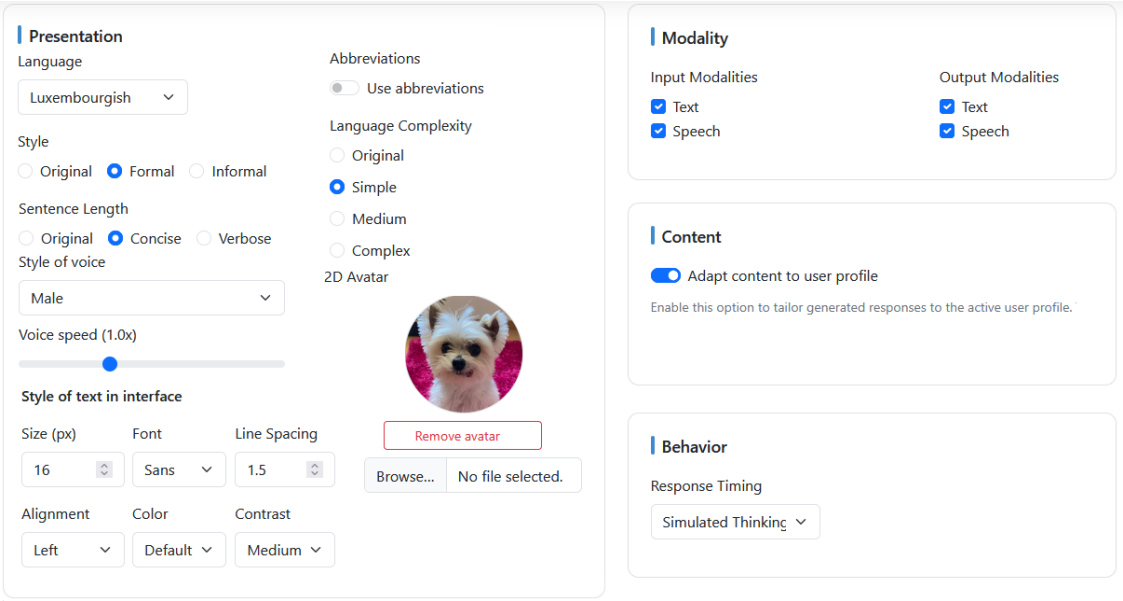

Figure 6: Form-based configuration specification

3.4 Linking the models

In a final step, the personalization model needs to be mapped to the affected base agent model and target user profile, illustrated in Figure 4 in the metamodel with the association starting from the class *Agent*, respectively *User* to *Agent Configuration*. This straightforward connection is implemented via forms, where the designer defines the mapping that is then stored on top of the defined models.

4 Generating the personalized conversational agents

We will now explain the two-step personalization process of our pipeline, depending on whether the personalization can be completely done at design time or it is part of the runtime execution of the agent.

This process is repeated for every agent profile we want to add to our base agent model, resulting in a family of related agents that share the same core behavior but present it on different shapes and styles depending on the personalization applied to generate each of them.

```
You are an editing engine focused on brevity. Rewrite each numbered text to be
    concise while preserving the original meaning. Remove redundancy, trim filler,
    and keep sentences short. Return the rewritten texts as a numbered list in the
    same order.
```

Figure 7: Prompt for concise sentence generation.

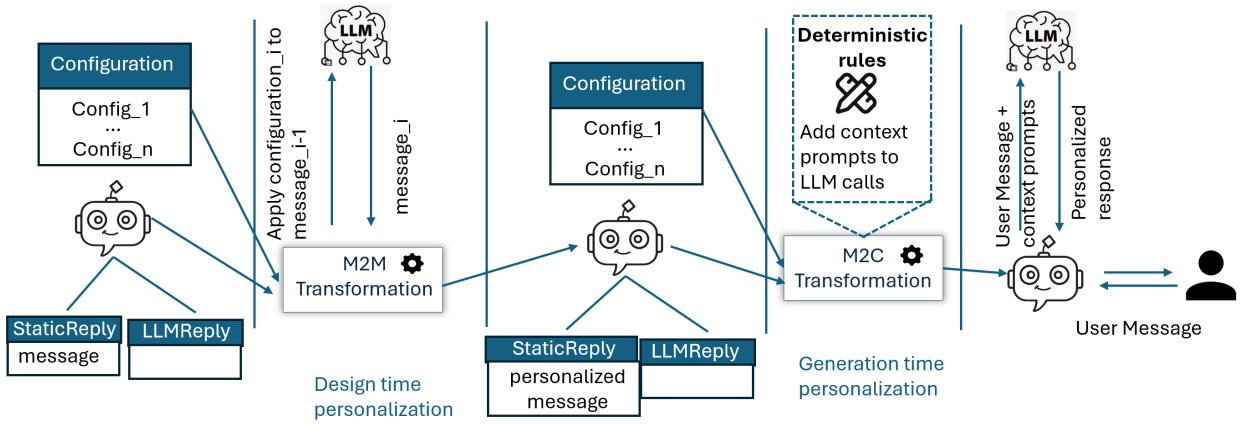

Figure 8: Process for nondeterministic personalization aspects

4.1 Design time personalization

Design time personalization in the model-to-model transformation is only applied to the predefined textual responses in the agent model, and only if the agent profile specifies changes related to nondeterministic personalization aspects. These include changes in *Language*, *Style*, *Sentence Length*, *Abbreviations*, *Language Complexity* and adapting the *Content* to fit the user profile. For each aspect, an external LLM is prompted to adapt all the predefined messages to fit the specific requested adaptation for the task.

Personalization aspects are applied sequentially: after each step, the resulting output replaces the previous version and is used as input to the next aspect. This sequential design constrains each LLM call to a single aspect, reducing prompt complexity and limiting cross-interference between aspects. Listing 7 reflects the used prompt when *Sentence Length* is set to concise, with other prompts following a similar structure, available in the replication package[2]. This process outputs the base agent model with the personalized textual responses for a given agent configuration.

4.2 Runtime personalization

Runtime personalization requires generating code that guarantees a personalized runtime interaction. For each personalization aspect, deterministic rules are applied, adding the necessary bits of code (de)activating or modifying certain agent features or wrapping agent calls to external services to make sure the call response is adapted to the personalization requirement.

Among the former we have *Style of Voice*, *Voice Speed*, *Style of text in interface*, *Avatar*, *Input and output Modality* and *Response Timing*. For the latter, a context prompt will be added describing the expected personalization aspects the LLM should adhere to when preparing the response. If necessary, a call to a second LLM to either confirm the adaptation is correct (e.g. checking the content is suitable for the target age) or forcing a reiteration of the adaptation (e.g. if the second LLM detects the adaptation is insufficient) can be also added.

Figure 8 illustrates some personalization aspects for our running example.

[2]`https://github.com/Aran30/LowCodeConversationalAgentPersonalization_replication`

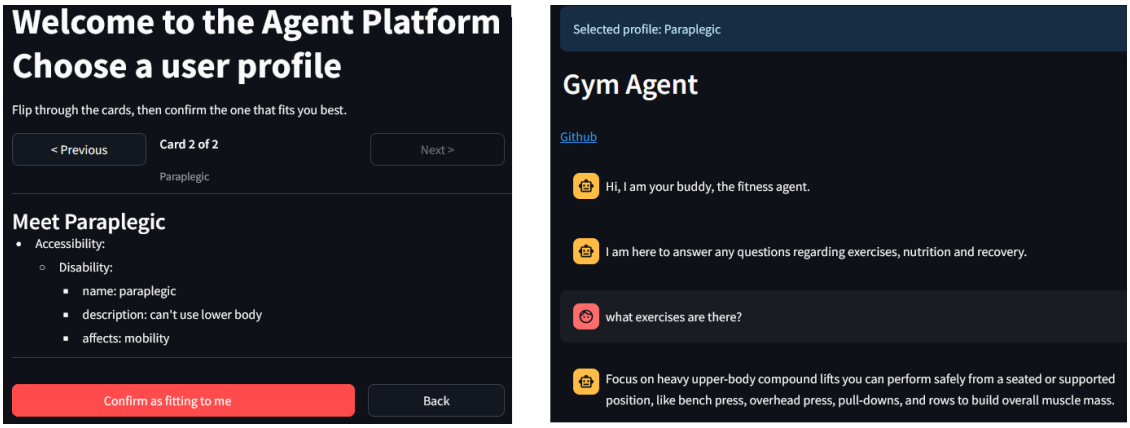

Figure 9: Profile picker and chat with personalized agent for paraplegic user

5 Tool support

The complete pipeline has been implemented by extending the open source BESSER framework [2], which focuses on the low-code/no-code development of smart software. Note that the screenshots shown in Section 3 were taken directly from the tool. More precisely, we extended three components of the BESSER suite[3], which are all grouped in the replication repository[4].

First, we extended the BESSER Agentic Framework (BAF). In itself, the framework acts as a Python-based library to create CAs following a state-machine-like formalism. We extended it to be able to, at design time, specify agent variations of state actions and user profile, and at runtime, allow end-users to choose a profile to trigger the corresponding personalized interactions, visible in Figure 9.

Second, we extended the core low-code BESSER platform. In particular, we added new transformations and generators to implement the personalizations as described in the Section 4.

Finally, we added new modeling perspectives to the BESSER Web Modeling editor to enable the graphical modeling of the profiles and the form-based configuration for the personalization.

6 Pilot Study

A pilot user study was conducted to evaluate our CA personalization tool regarding the following two research questions (RQs): perceived usability (RQ1) and perceived usefulness (RQ2). The study further aimed to explore whether users with different technical backgrounds could understand and complete the personalization workflow. The previously shared replication package also contains the study handout and questionnaire data.

Participants Eight participants were recruited for the exploratory study through direct contact to obtain the required profiles. Four participants had a technical background with prior experience in software development and modeling, while the other four had no programming experience (with the exception of one who reported limited experience) and no prior exposure to software modeling. This diversity was intentional, as the study aimed to explore early perceptions across the two target user profiles of the tool: expert (technical) and layman (non-technical).

Procedure After signing a consent form, participants received a document describing the running example and the tasks required to complete one iteration of the CA personalization pipeline. Upon opening the web-based modeling tool, they were presented with a pre-modeled gym agent, which they inspected and explained to the study supervisor to confirm their understanding of the model's syntax and semantics. Next, based on task descriptions, participants created two predefined user profiles (*Elderly* and *Paraplegic*), defined two agent profiles, mapped user profile to agent profile, and triggered the M2M transformation to generate personalized models. Participants also compared the adapted and original models to identify the differences that occurred through the design time adaptations. Finally, they generated and interacted with the personalized agent. The tasks were designed to capture participants' initial understanding and impressions of the personalization process. Afterward, participants completed a questionnaire including the System

[3] `https://github.com/BESSER-PEARL`

[4] `https://github.com/Aran30/LowCodeConversationalAgentPersonalization_replication`

Table 2: Comparison of expert and layman evaluation results

Metric	Experts	Laymen
SUS Score	92.50	78.75
Perceived Usefulness (PU)	95.00	85.00
Perceived Ease of Use (PEOU)	96.60	71.60
Average completion time (min)	39.75	43.25
Average number of interventions (number)	2.25	3.75

Usability Scale (SUS) [7] and perceived usefulness and ease-of-use items adapted from the Technology Acceptance Model (TAM) [11], both widely used and validated instruments for measuring usability and technology acceptance. Responses were collected on a five-point Likert scale ranging from 1 ("Strongly Disagree") to 5 ("Strongly Agree").

Results Table 2 contains the result of the questionnaire, the completion time, and the intervention count as averages based on the participants' profile. The answers from the questionnaires are normalized to 100 to match the standardized SUS score.

RQ1 (Usability) The high average scores that reflect usability (SUS and PEOU) for both experts (SUS: 92.50, PEOU: 96.60) and laymen (SUS: 78.75, PEOU: 71.60) indicate good usability overall. Nonetheless, a clear difference in perception between experts and laymen (ΔSUS: 13.75, ΔPEOU: 25.00) is observed. A possible explanation would be the lack of familiarity with modeling tools for the laymen compared to the experts. In contrast, the delta between experts and laymen for completion time (Δ3.50) or interventions (Δ1.50) is very low, showcasing a similar performance. Independent of the participant profile, minor usability concerns have been noted. Most notably, we observed that it was unclear sometimes that elements in the user profile editor needed to be linked with each other, and the agent configuration page was perceived as overloaded, which increased the likelihood of overlooking elements. Addressing these and adding better documentation would increase usability.

RQ2 (Usefulness) Participants of both profiles perceived the pipeline to have a high usefulness, as evident by the high scores in the questionnaire (PU: 95.00 for experts and 85.00 for laymen). A small difference is observable between the profiles (ΔPU: 10), which possibly stems from modeling experts understanding and appreciating the advantages of the approach more than laymen.

Threats to validity Several factors may threaten the validity of the claims made in this study. Regarding internal validity, as participants were acquaintances of the study supervisor, there is a risk of response bias towards more positive evaluations, although they were explicitly encouraged to provide honest and critical feedback. Concerning construct validity, the instruction sheet may have provided substantial guidance, potentially reducing the perceived complexity. Yet, we argue this reflects the documentation typically provided in production-ready tools. For external and conclusion validity, the small sample size limits the generalizability of the results and the statistical power, meaning observed trends should be interpreted with caution and confirmed in larger replications.

7 Related Work

We comment on two types of related works: those focusing on a model-driven approach for the generation of CA and those proposing ad-hoc adaptation techniques for CAs. As we discuss, there are no model-driven approaches that target the automatic personalization of CAs that could be embedded in a web application.

Low-code and model-driven approaches for developing CAs Model-driven approaches have been applied in the past to the development of web and mobile UIs, see for instance, tools and languages such as IFML [6], OOWS [15] or UWE [18] among many others. Yet, these do not include a CA perspective, their support for specifying user profiles is limited to key-value pairs and the offered personalization is limited. Newer approaches focus more on personalization [26, 25], albeit not targeting either CAs and only providing textual syntaxes, therefore not catered towards non-technical users.

There are also a few approaches targeting low-code development for CAs (see this SLR conducted by Ouaddi et al. [5]), though, similar to the approaches above, they do not cover personalization aspects. As an example, a recent no-code approach was proposed to enable the creation of voice CAs on top of existing web-pages as a plug-in [23]. The approach does not focus on personalization or catering to diverse user profiles.

Personalization in agents Personalizing CAs is not a novel topic in itself, as evident by the existence of SLRs covering this topic (e.g., [1, 17] to name a few) that provide insights about techniques that exist or types of personalization and their benefits. Yet, tools that facilitate the creation of such personalized agents combining various existing

personalization options are lacking. Our pipeline acts as an initial attempt to facilitate the creation of personalized CAs by leveraging technologies that showed positive results in the past in a way that non-technical users could benefit from.

8 Conclusions

This paper presented a pipeline for designing and generating personalized conversational agents using low-code and no-code principles. Our proposal enables the specification of user-specific agent profiles, linking users with customized agents derived from a core agent specification via a combination of deterministic transformations and LLM customizations.

As further work, we plan to increase the types of personalizations available and automatically propose personalization templates for the most common types of users. We also plan to extend the personalization to other types of UIs (e.g., for extended reality scenarios) and add some quality checks to the configuration process to detect potentially wrong or inconsistent personalization options. Finally, larger scale user studies shall be conducted to further validate the pipeline's usability, but also scalability and quality of generated personalization.

Acknowledgments

This work is supported by the Luxembourg National Research Fund (FNR) PEARL program, grant agreement 16544475.

References

[1] Tarek Ait Baha, Mohamed El Hajji, Youssef Es-Saady, and Hammou Fadili. The power of personalization: A systematic review of personality-adaptive chatbots. *SN Computer Science*, 4(5), 2023. doi:10.1007/s42979-023-02092-6.

[2] Iván Alfonso, Aaron Conrardy, Armen Sulejmani, Atefeh Nirumand, Fitash Ul Haq, Marcos Gomez-Vazquez, et al. Building besser: an open-source low-code platform. In *International Conference on Business Process Modeling, Development and Support*, pages 203–212, 2024. doi:10.1007/978-3-031-61007-3_16.

[3] Mohammad Amin Kuhail, Mohamed Bahja, Ons Al-Shamaileh, Justin Thomas, Amina Alkazemi, and Joao Negreiros. Assessing the impact of chatbot-human personality congruence on user behavior: A chatbot-based advising system case. *IEEE Access*, 12:71761–71782, 2024. doi:10.1109/ACCESS.2024.3402977.

[4] Marcos Baez, Cinzia Cappiello, Claudia M. Cutrupi, Maristella Matera, Isabella Possaghi, Emanuele Pucci, et al. Supporting natural language interaction with the web. In *Web Engineering: 22nd International Conference, ICWE 2022, Bari, Italy, July 5–8, 2022, Proceedings*, page 383–390, 2022. doi:10.1007/978-3-031-09917-5_26.

[5] Lamya Benaddi, Charaf Ouaddi, Abdeslam Jakimi, Hasna Chaibi, Abdellah Chehri, Gwanggil Jeon, et al. A survey on model-driven engineering and domain-specific languages for chatbot development: Requirements, challenges and solutions. *Expert Systems*, 42(10):e70124, 2025. doi:10.1111/exsy.70124.

[6] Marco Brambilla and Piero Fraternali. *Interaction flow modeling language: Model-driven UI engineering of web and mobile apps with IFML*. Morgan Kaufmann, 2014.

[7] John Brooke. Sus: A "quick and dirty" usability scale. In *Usability Evaluation in Industry*, pages 189–194. 1996.

[8] Jordi Cabot and Robert Clarisó. Low code for smart software development. *IEEE Software*, 40(1):89–93, 2023. doi:10.1109/MS.2022.3211352.

[9] Aaron Conrardy, Alfredo Capozucca, and Jordi Cabot. Towards a unified user modeling language for engineering human centered ai systems, 2025. URL: https://arxiv.org/abs/2505.24697.

[10] Aaron Conrardy, Alfredo Capozucca, and Jordi Cabot. User modeling in model-driven engineering: A systematic literature review. *Journal of Object Technology*, 24(2):2:1–14, 2025. doi:10.5381/jot.2025.24.2.a12.

[11] Fred D. Davis. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly*, 13(3):319–340, 1989. doi:10.2307/249008.

[12] Davide Di Ruscio, Dimitris Kolovos, Juan Lara, Alfonso Pierantonio, Massimo Tisi, and Manuel Wimmer. Low-code development and model-driven engineering: Two sides of the same coin? *Software and Systems Modeling*, 21:437–446, 2022. doi:10.1007/s10270-021-00970-2.

[13] Haiyan Fan and Marshall Scott Poole. What is personalization? perspectives on the design and implementation of personalization in information systems. *Journal of Organizational Computing and Electronic Commerce*, 16(3-4):179–202, 2006. doi:10.1080/10919392.2006.9681199.

[14] Jasper Feine, Ulrich Gnewuch, Stefan Morana, and Alexander Maedche. A taxonomy of social cues for conversational agents. *International Journal of Human-Computer Studies*, 132:138–161, 2019. doi:10.1016/j.ijhcs.2019.07.009.

[15] Joan Fons, Vicente Pelechano, Oscar Pastor, Pedro Valderas, and Victoria Torres. *Applying the Oows Model-Driven Approach for Developing Web Applications. The Internet Movie Database Case Study*, pages 65–108. Springer London, 2008. doi:10.1007/978-1-84628-923-1_5.

[16] Syed Mahmudul Huq, Rytis Maskeliūnas, and Robertas Damaševičius. Dialogue agents for artificial intelligence-based conversational systems for cognitively disabled: a systematic review. *Disability and Rehabilitation: Assistive Technology*, 19(3):1059–1078, 2024. doi:10.1080/17483107.2022.2146768.

[17] Ahmet Baki Kocaballi, Shlomo Berkovsky, Juan C Quiroz, Liliana Laranjo, Huong Ly Tong, Dana Rezazadegan, et al. The personalization of conversational agents in health care: Systematic review. *J Med Internet Res*, 21(11):e15360, 2019. doi:10.2196/15360.

[18] Nora Koch, Alexander Knapp, Gefei Zhang, and Hubert Baumeister. *Uml-Based Web Engineering*, pages 157–191. Springer London, 2008. doi:10.1007/978-1-84628-923-1_7.

[19] Cheng Li, Mengzhuo Chen, Jindong Wang, Sunayana Sitaram, and Xing Xie. Culturellm: Incorporating cultural differences into large language models. In *Advances in Neural Information Processing Systems*, volume 37, pages 84799–84838, 2024. doi:10.52202/079017-2693.

[20] Emiliana Murgia, Maria Soledad Pera, Monica Landoni, and Theo Huibers. Children on chatgpt readability in an educational context: Myth or opportunity? In *Adjunct Proceedings of the 31st ACM Conference on User Modeling, Adaptation and Personalization*, page 311–316, 2023. doi:10.1145/3563359.3596996.

[21] Tatwadarshi P. Nagarhalli, Vinod Vaze, and N. K. Rana. A review of current trends in the development of chatbot systems. In *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pages 706–710, 2020. doi:10.1109/ICACCS48705.2020.9074420.

[22] Mohammad (Matt) Namvarpour and Afsaneh Razi. The art of talking machines: A comprehensive literature review of conversational user interfaces. In *Proceedings of the 7th ACM Conference on Conversational User Interfaces*, 2025. doi:10.1145/3719160.3736621.

[23] Gonzalo Ripa, Manuel Torre, Matias Urbieta, Gustavo Rossi, Alejandro Fernandez, Alex Tacuri, and Sergio Firmenich. Generating voice user interfaces from web sites. *Behaviour & Information Technology*, 43(13):3129–3152, 2024. doi:10.1080/0144929X.2023.2272192.

[24] Alexander Skuridin and Martin Wynn. Chatbot design and implementation: Towards an operational model for chatbots. *Information*, 15(4), 2024. doi:10.3390/info15040226.

[25] Shavindra Wickramathilaka, John Grundy, Kashumi Madampe, and Omar Haggag. Adaptive and accessible user interfaces for seniors through model-driven engineering. *Automated Software Engineering*, 32(2):74, 2025. doi:10.1007/s10515-025-00547-z.

[26] Enes Yigitbas, Ivan Jovanovikj, Kai Biermeier, Stefan Sauer, and Gregor Engels. Integrated model-driven development of self-adaptive user interfaces. *Softw. Syst. Model.*, 19(5):1057–1081, 2020. doi:10.1007/s10270-020-00777-7.

[27] Wenhao Zhu, Hongyi Liu, Qingxiu Dong, Jingjing Xu, Shujian Huang, Lingpeng Kong, et al. Multilingual machine translation with large language models: Empirical results and analysis. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2765–2781, 2024. doi:10.18653/v1/2024.findings-naacl.176.