# Taking into account users' knowledge, abilities and preferences to personalize animated assistant agents

Blandine Ginon[1,2], Pierre-Antoine Champin[1,3], Stéphanie Jean-Daubias[1,3]

1 Université de Lyon, CNRS,
2 INSA-Lyon, LIRIS, UMR5205, F-69621, France
3 Université Lyon 1, LIRIS, UMR5205, F-69622, France

**Abstract.** In this paper, we present our approach to personalize existing and various animated agents, taking into account the context and the users' knowledge, abilities and preferences. For this purpose, we propose a grammar to describe animated agents in a common formalism, and specify their characteristics and abilities. According to this description, it is then possible to define actions with parameters that can be used and personalized in a description of the required assistance for a given application, which is used by a generic assistant to provide each user with personalized assistance.

**Keywords.** User assistance, animated assistant agents, personalization.

## 1 Introduction

Animated agents are met in many applications and in many fields, like commercial web-sites, institutional web-sites or recreational applications. In particular, more and more ILE (Interactive Learning Environments) implement pedagogical animated agents [1] to facilitate learning. These agents differ a lot in their appearance: it can be human-like, animal, robot or imaginary (genius, object…). These agents can assist learners by means of demonstrations or explanations; they can guide learners and interact with them through verbal and non-verbal communication [1]. Some studies had shown the interest of these pedagogic agents: they motivate learners and help them in their learning process [2]; they also produce greater reported satisfaction and enjoyment by learners [3]. Furthermore, personalization is an important issue. It allows proposing more relevant content, possibly in a more appropriate form, taking into account the users' knowledge, preferences and abilities [4][5]. In this paper, we present our approach to personalize the actions of animated agents, taking into account the context and the user's knowledge, abilities and preferences. In section 2, we present the context of this work, with an overview of a framework allowing personalized user assistance by means of formally describing the assistants, the application to assist, and user's profiles and preferences. Then in section 3 we focus on the description of animated agents. This description details every characteristic of an existing animated agent and the values that these characteristics can take. The description also includes actions with parameters, complying with the description of the animated agent and showing the agent's abilities. An example of using actions with parameters is given. Finally section 4 concludes and describes the next steps of our work.

## 2 Context

The research presented in this paper is part of a research project on user assistance [6]. We have first proposed a typology of user assistance, which presents the user needs and

compares the different techniques and approaches of assistance that can meet these needs [7]. Among the approaches of assistance, we find counselor systems, adaptive interfaces, recommender systems, tutorials… In this paper, we focus on one particular approach of assistance: animated agents. We have then proposed a model of generic assistant that will provide personalized assistance to each user [6], complying with the description of the required assistance for a given application and the user model. For this purpose, the generic assistant will use a set of epiphyte assistants that can be grafted onto an application to perform the required assistance actions.

An epiphyte assistant is an assistant that can be grafted onto a given application, without perturbing its work [8]. For instance, Microsoft proposes an epiphyte assistant than can be grafted onto any application compatible with Windows [9]. This assistant provides a set of animated agents (like Merlin or Robby the robot), able to communicate with the user by means of gestures (like pointing at an interface component), facial expressions (like frowning), or messages, textual or read by a speech synthesis. Our generic assistant model aims at allowing the personalization of assistants already integrated in an application. For this reason, it is necessary to describe in a unified formalism an existing assistant, epiphyte or integrated in an application, in order to be able to run it, or personalize its actions with respect to respect the description of the required assistance.

In order to adapt the assistance to users, their knowledge, preferences, capacities, goals and experiences, our model required a user model (also called user profile) that can contains any information about a user that can be useful in this context. For that purpose, we chose to use the profile modeling language PMDLe [10][11] that allows the definition of reusable user profiles. PMDLe is fully implemented in an environment that makes it possible to define and use of such profiles [11]. An extension to PMDLe, cPMDLe [11], allows to define constraints on user profile respecting PMDLe formalism. These constraints are used to personalize assistance according to the user specificities. For instance, an assistance action could be performed only for users that are novice in the application, or only for users whose hearing level is between 50% and 80%.

The formalism we use to describe the assistance required for a given application is based on a model proposed by [12]. The required assistance is described by rules of the form "If assistance condition Then assistance action". The assistance conditions contain a start condition (like a click on a component of the application's interface) and any cPMDLe constraint on the user's profile, in order to allow the personalization according to both the context and the user specificities. What's more, the assistance action involved an assistant, epiphyte or integrated into the application, and must respect the description of this assistant.

## 3 Propositions

We propose here an approach to describe the possible actions of animated assistant agents in order to personalize their behaviour. This approach is based on a grammar for describing animated assistant agents that presents the characteristics of an animated agent, and proposes actions with parameters that can be perform by this animated agent to provide personalized assistance to a user. These actions can then be used in the description of the required assistance for a given application. Contrary to existing approaches to describe animated agents [14], our aim is not to define new animated agents but to describe in a common formalism existing animated agents and their abilities, whatever their origin and intern formalism. A description in a common formalism of varied animated agents makes it possible to reuse them in various contexts. Let's suppose that MSAgents, WebLéa and Cantoche animated agents are describe in a common formalism, with for each a description of an action called "greet" that depends on their individual abilities. Then, the designer of an ILE can specify that an animated agent must *greet* learners when starting. The choice of

the agent will depend on the learners' preferences for instance, provided that the chose agent provides an action called "greet".

### 3.1 Grammar for animated agent description

We propose an abstract grammar for animated assistant agents' description. This grammar is used to describe any animated agent, epiphyte or integrated in an application. Our description of an animated agent consists in five parts, graphically represented on Fig. 1. In this graphical representation, the symbol + means at least one, the symbol ? means 0 or 1, the symbol * means zero, one or more, and the presence of "…" after an element means that it contains others elements (described elsewhere). The five parts of an animated agent description are: information about the involved animated agent (like its name and identifier), information about the description itself (like its author and date of creation), a *characteristics_list*, a *message_channels_list*, and an *action_definition*.

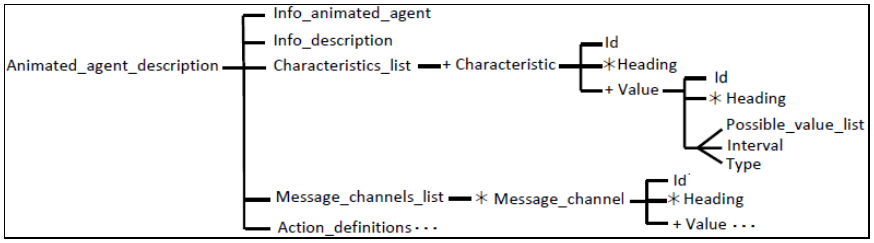


Fig. 1 Grammar for animated agents' description

Many elements in our grammar contain an *Id* and optionally one or several *Heading*s. The former is used to unambiguously identify the element for programs, while the latter is used to display the element to a user. Headings can be provided in different languages, hence the possibility to have several of them. The aim of the *characteristics_list* of an animated agent is to list all the characteristics (or attributes) of this animated agent and the values they can take; an example will be given in Fig. 3. For instance, an animated agent can have a characteristic "avatar", which can take several values specifying the appearance of the assistant. The *characteristics_list* contains at least one *characteristic*, each associated with an *Id*, some *heading*, and at least one *value*. A *value* is associated with an *Id*, some *heading* and either a *possible_value_list*, an *interval* (associated with a max, a min, and a scaling) or a *type* (like color, font…). The aim of the *message_channels_list* is to describe all the ways that the animated agent can pass a message to the user. For instance, an animated agent can pass a given message by means of a cartoon balloon or orally. The *message_channels_list* contains a set of *message_channel*, each associated with an *Id*, some *heading* and at least one *value* acting as parameters for that channel (*i.e.* font size or voice gender). The *action_definition* (cf. Fig. 2 ) contains a set of *posture*s and a set of *action_with_parameters*. A *posture* is associated with an *Id*, some *heading* and at least one *constraint* that must be chosen among the *characteristics* of the animated agent. A *posture* aims at being used in an *action_with_parameters*. In a *posture*, all *constraints* refer to a *characteristic* of the animated agent, but the possible values associated with this characteristic are reduced. For example, if the characteristic is associated with an interval, the constraint can be associated with a subset of this interval; if the characteristic is associated with a set of 12 possible values, the constraint can be associated with a subset of these possible values, or even with only one value among them. An example of postures and their use in an action with parameters is given in Fig. 4 and Fig. 5.

An *action_with_parameters* is associated with an *Id*, some *heading*, an *animation* and an optional *message*. The aim of an *animation* is to describe the behaviour of the animated agent during this action. An *animation* consists of a *postures_sequence* and a repetitions number or repetition duration of this sequence. A *message* consists of a *text* and at least one

*message_channel* that describe how this text will be presented to the user. Similarly to characterstics in postures, the provided message_channel must correspond to one of the available channels of the assistant, and its *value*s, if provided, must restrict the available values of that channel. All the *value*s that are still under-constrained (*i.e.* for which there are more than one possible value left) are called the *parameters* of the action.

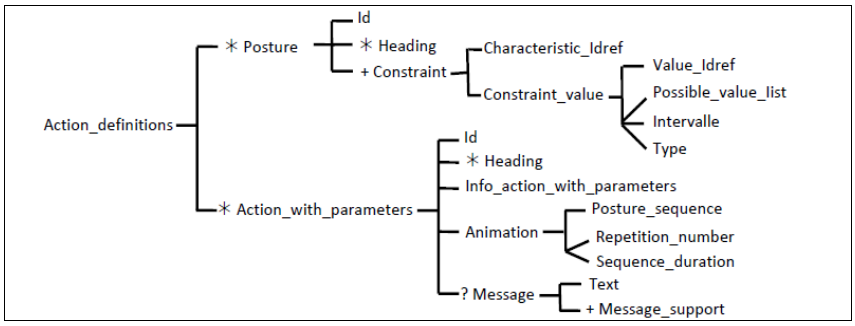


Fig. 2 Action_definition for animated agents

Let's take the example of WebLéa [13], an animated assistant agent than can be integrated in any web application. A graphical representation of an extract of the characteristics list from the description of WebLéa complying with our grammar is given in Fig. 3. We can see that six characterstics of this animated agent, each associated with a list of possible values. For instance, the characteristic "C1", with heading "Avatar", can take the values "Léa", "Marco", "Julien" or "Djinn". Similarly, Fig. 4 and Fig. 5 illustrate some part of the *action_definition* for WebLéa that will be discussed in the next section.

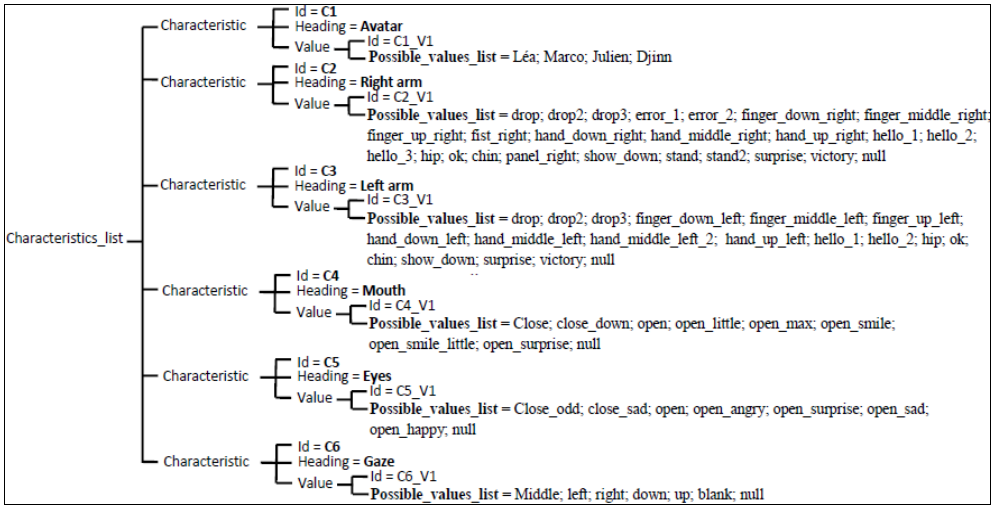


Fig. 3 Characteristics list from the description of the WebLéa animated agent.

### 3.2 Use of actions with parameters

When describing the required assistance for a given application, we will use the actions with parameters in assistance rules. These actions with parameters must comply with the ones defined in the description of the assistant that will perform them. The use of parameters will enable the personalization according to the user's profile. Let's take an example of an action with parameters for the animated agent WebLéa. The left part of Fig. 4 represents a posture that describes a smiling agent with the right arm at right up and the left arm drop. We give examples of WebLéa agents taking this posture on the right part of

Fig. 4, with the four possible avatars. We can see that the Id of this posture is P1, it contains a constraint on five of the six characteristics of WebLéa (cf. Fig. 33). For these five characteristics, the posture definition reduces the possible value list. For instance, the characteristic mouth (with Id C4) can only take one value in this posture: "open_smile"; the characteristic left_arm (with Id C3) can only take three values: "drop", "drop2" and "drop3". One characteristic of WebLéa, avatar, is not present in the description of the P1: its means that in this posture, the characteristic avatar can take any value from its possible value list.

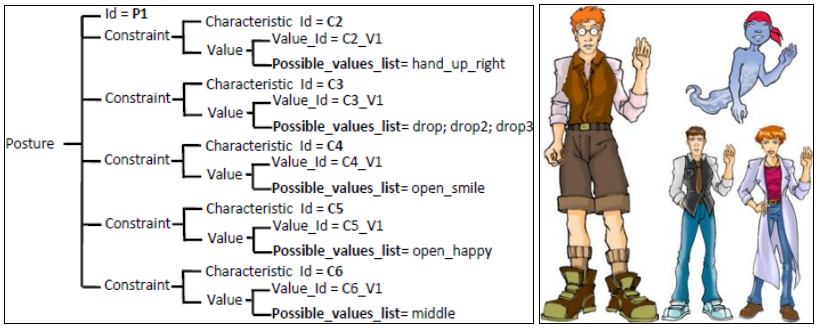


Fig. 4 Description of the posture P1 for WebLéa and illustration.

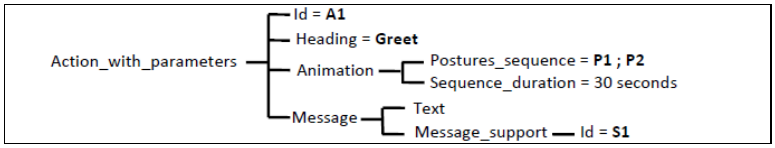


Fig. 5 Description of an action with parameters for WebLéa animated agent.

Fig.5 shows the description of an action with parameters for WebLéa animated agent, with Id A1 and heading Greet. This action uses postures P1 and P2 (P2 is another posture that describes an agent with the right arm at hand up left). In action A1, the agent switches between postures P1 and P2 during 30 seconds, causing it to wave its right hand to greet the user. The agent also communicates with the user by means of a message using message_channel S1 (not described because of space limitation; we will assume it displays the text in a baloon). The parameters of action A1 are the text of the message, that is not defined in the action, and all characteristics of the agent that are not associated with a single possible value in the action, *i.e.* the choice of the avatar (that is free), and the choice of the value for left_arm (reduced to three possible values). The action with parameters can now be used in the rules describing the required assistance for a given application. The action parameters will allow the personalization. For instance, an assistance rule could be "If {application start and user_level="novice"}, then {WebLéa performs (action_id=A1; text='Welcome! To discover this application, please clic on the button "Start with this application"'; C1=(preferences); C4=(random)}. This action will make a WebLéa animated agent appear and greet a novice user with its right hand, while saying the text in a baloon. According to this assistance rule, the choice of the avatar (corresponding to characteristic C1) will be done according to the user's preferences (express in the user profile) among the four possibilities ("Léa", "Marco", "Julien" and "Djinn"). The choice of the left_arm value (corresponding to characteristic C4) will be done randomly among the three possibilities defined in the action description ("drop", "drop2" and "drop3").

## 4 Conclusion

Assistance to users is a very important issue, in order to help users with both the handling and the common use of an application. One popular assistance approach is the use

of animated agents that interact with users. The context of this work is to define a generic approach to personalize the action of animated agents (as well as other kinds of assistants). For this purpose, we have proposed in this paper a grammar for describing animated agents, their characteristic and their abilities. The grammar also allows the definition of actions with parameters that rely on the animated agent's characteristics' description. Assistance rules using actions with parameters will make allow the personalization of the assistance according to the users' profile (that can contain information on their knowledge, capacities, preferences, goals, experiences etc.). On the one hand, we can personalize assistance rules thanks to cPMDLe constraints on the user's profile. Indeed, an assistance action could be performed only for some users, whose profiles meet some cPMDLe constraints. On the other hand, we can personalize assistance thanks to constraints on the assistance actions. Indeed, the parameters of an action can depend on the user profile.

Finally, a generic assistant will use the description of the required assistance of an application, containing assistance rules, to provide each user with personalized assistance complying with this description. It will determine which assistance action should be performed and by which assistant. As a consequence, an action should comply with the description of animated agents involved in the assistance action, and the generic assistant should be able to pilot the assistant according to its description, in order to make it perform the correct action. We are currently working on implementing such a generic assistant, based on the assistant descriptions described in this paper. We are also working on extending the definition of our grammars to describe others kinds of assistant, like *e.g.* counselor systems.

## References


[1] Johnson, W.L., Rickel, J.W. and Lester, J.C.: Animated pedagogical agents: face-to-face interaction in interactive learning environments. In: IJAIED, pp 47-78. (2000)

[2] Nunes, M., Dihl, L., Fraga, L., Woszezenki, C., Oliveira, L., Francisco, D., Machado, G., Nogueira, C. and Notargiacomo, M.: Animated Pedagogical Agent in the Intelligent Virtual Teaching Environment. In: IEM, pp 53-61. (2002)

[3] Yan, J. and Agada, R.: Life-Like Animated Virtual Pedagogical Agent Enhanced Learning. In: Journal of educational multimedia and hypermedia, pp 4-12. (2009)

[4] Wenger, E.: Artificial Intelligence and Tutoring Systems (1987)

[5] Brusilovsky, P.: Adaptive Hypermedia. In: UMAI'01, pp 87-110. (2001)

[6] Ginon, B.: Towards a generic model for user assistance. In: Doctoral consortuim of UMAP, pp Montréal, Canada (2012)

[7] Ginon, B., Jean-Daubias, S., Champin, P.-A. and Vinh, T.L.: Assistance à l'utilisateur : typologie et outils épiphytes. In: TICE (2012, submitted)

[8] Paquette, G., Pachet, F. and Giroux, S.: Épitalk, un outil générique pour la construction de systèmes conseillers. (1994)

[9] Microsoft: MSAgents. http://www.microsoft.com/products/msagent/main.aspx (2012)

[10] Eyssautier-Bavay, C. and Jean-Daubias, S.: PMDL: a modeling language to harmonize heterogeneous learners profiles. In: ED-MEDIA (2011)

[11] Ginon, B., Jean-Daubias, S. and Lefevre, M.: Evolutive learners profiles. In: ED-MEDIA (2011)

[12] Dufresne, A., Basque, J., Paquette, G., Léonard, M., Lundgren-Cayrol, K. and PromTep, S.: Vers un modèle conceptuel générique de système d'assistance pour le téléapprentissage. In: STICEF, pp 57-88. (2003)

[13] Martin, J.-C. and Sansonnet, J.-P.: WebLéa. http://perso.limsi.fr/jps/online/weblea/leaexamples/leawebsite/index.html (2012)

[14] Mancini, M., Pelachaud, C.: Dynamic behavior qualifiers for conversational agents. Intelligent Virtual Agents, Paris, France (2007)