aLDEAS: a Language to Define Epiphytic Assistance Systems

Blandine Ginon, Stéphanie Jean-Daubias, Pierre-Antoine Champin, Marie Lefevre

aLDEAS: a Language to Define Epiphytic Assistance Systems

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

[1] Université de Lyon, CNRS,
[2] INSA-Lyon, LIRIS, UMR5205, F-69621, France
[3] Université Lyon 1, LIRIS, UMR5205, F-69622, France
{name}.{surname}@liris.cnrs.fr

Abstract. We propose a graphical language that enables the specification of assistance systems for a given application, by means of a set of rules. This language is completed by several assistance actions patterns. We implemented these propositions through an assistance editor aimed at assistance designers, and a generic assistance engine able to execute the specified assistance for the target-application end users, without a need to modify this application. We performed several experimentations both with assistance designers and with target-applications end users.

Keywords: Language, user assistance, rule-based system, epiphytic tools.

1 Introduction

User assistance is one solution to overcome the difficulties of handling and using software applications. It helps preventing the user from under-exploiting or even rejecting those applications.

We define assistance as the set of means facilitating the handling and use of an application, in a way suitable to the user and to the use context. The assistance aims at enabling the user to exploit fully all the possibilities of an application, and it facilitates the appropriation of the knowledge and competencies required to use this application. It includes the four assistance types defined by [3]: supplementation, support, assistance and substitution.

The development of an assistance system suitable to an application is a complex and expensive task, often neglected by applications designers. A person other than the application designer may then wish to plug an assistance system to an application that has no assistance or an incomplete one. For instance, in the context of a user community, an expert may wish to design an assistance system to make novice users benefit from his/her experience. In the case where the assistance designer is not the target-application designer, the source code of the target-application is most of the time not available; it is then not possible to integrate an assistance system directly in the application. What's more, as in our example, the potential assistance designer is not always

a programmer. An alternative to the classical approach of development of an assistance system integrated into the application consists in adopting an epiphytic approach to make possible the *a posteriori* specification and execution of an assistance system in an existing application without the need to modify it. An epiphytic assistance, that we call an *epi-assistant*, is an assistant able to perform actions in an external target-application, without disturbing its functioning [6].

The work presented in this paper takes place in the context of the AGATE project that aims to propose generic models and unified tools to make possible the setup of assistance systems in any existing application, that we call the target-application. For this purpose, we proposed an adjunction process of epi-assistance system in a target-application in two phases (cf. Fig. 1). The first phase involves an assistance designer: an expert of the target-application who wishes to design an assistance system. This phase enables the assistance designer to specify the assistance that he/she wishes for the target-application, by defining a set of assistance rules. The second phase involves the target-application end users. It consists in the execution of the assistance specified by the designer. This phase occurs at each use of the target-application by an end user; it is composed of three processes. The monitoring of the target-application exploits a set of epi-detectors [4] that enable the continuous observation and the tracing of all interactions between the user and the target-application interface. In parallel, the process identifying assistance needs exploits the assistance rules defined by the designer and triggers the process elaborating an answer to the identified assistance need. The answer is executed as one or several assistance actions, performed by an epi-assistant in the target-application.

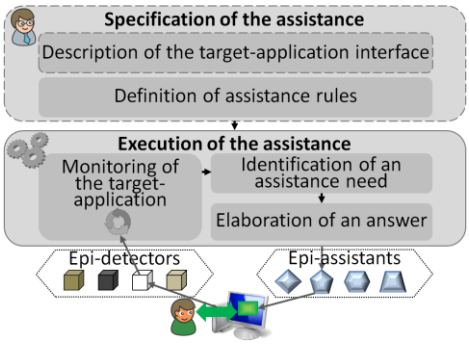

Fig. 1. Adjunction process of an epi-assistance system in a target-application

During the specification phase, the assistance designers need to elicit their knowledge about how to use the target application, in order to specify the assistance system. This phase must be accessible to an assistance designer that can be neither non-computer scientist nor knowledge engineering expert. Then the specified assistance is executed automatically during the execution phase. As a consequence, the assistance needs to be formalized enough. A solution to make these two points possible is to use a pivot language: we proposed aLDEAS, a graphical language to help assistance designers designing assistance rules. Moreover, we propose a set of patterns to guide them in defining complex assistance actions. We then present the im-

plementation of these patterns in the SEPIA system. Finally, we describe the experimentations that we performed in order to evaluate the usability of aLDEAS and its implementation in the SEPIA system.

2 Related work

Several authors have studied the *a posteriori* specification of assistance systems for existing target-applications. The approaches of [7] and [1] allow to plug an advisor system in a scenario from the environments Telos and ExploraGraph. These advisor systems are defined by an assistance designer through a set of rules of the form <trigger event, trigger condition, assistance action, end event>. The trigger condition can include a consultation of the user profile and of the assistance history in order to personalize and contextualize the assistance. The proposed assistance actions are textual messages displayed in a pop-up for Telos, and animations or messages conveyed by an animated agent for ExploraGraph. The approach proposed by [8] allows to plug an advisor system in a Web application, in order to trigger assistance actions when the end user clicks on a link. The proposed actions are textual messages displayed in a pop-up, which can contain links to a Web page or to resources related to the assistance. The provided assistance can be personalized according to a browsing history. The assistance provided by these advisor systems can also be expressed by rules with the form <trigger event: click on a link, trigger condition: browsing history, assistance action>. The CAMELEON model [2] allows to plug an animated agent able to move, perform animations and display messages in a Web application, thanks to tags inserted in an epiphytic way in the page.

Assistance rules seem to be suited to represent epiphytic assistance systems. Nevertheless, these different approaches cannot be used in any application. Indeed, they are specific to a given environment or to the Web. We are interested in the *a posteriori* plugging of assistance systems to very diverse existing target-applications. What's more, we would like to allow a fine-grained personalization of the assistance, according to the user profile and to the assistance history, like in Telos and ExploraGraph, but also according to the user's past actions, not only the browsing history as in [8], and according to the state of the target-application. Finally, to make possible a wider personalization of the assistance, we would like to propose a large choice of assistance actions: if the same assistance action can be performed in different ways, like pop-up or animated agent, it will be easier for the assistance designer to specify assistance suited to the user's specificities and to the context.

3 The aLDEAS language

We propose aLDEAS (a Language to Define Epi-Assistance Systems), a graphical rule language aimed at assistance designers for defining assistance systems. We choose to propose a graphical language, very suitable to the representation by a set of simple rules of assistance systems, in particular epiphytic assistance systems. Its different components (cf. Fig. 2) are presented in detail in this section. We will show

thereafter how these components can be combined to create assistance actions addressing assistance needs that will then be executed by our system SEPIA on top of the target-application. An aLDEAS block is a labelled direct graph, with nodes and edges taken from Fig. **2**. It must have exactly one source, which must be a "block start", and one or several sinks, which must all be "block end". It is executed by walking the graph; the effect of walking each type of node is described in the following.

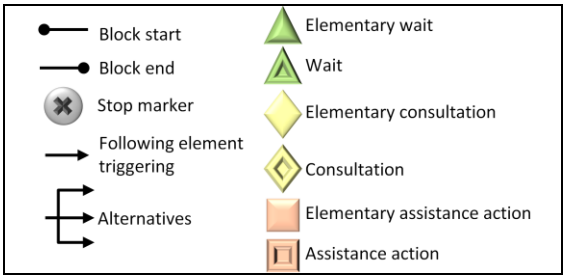

Fig. 2. Components of aLDEAS

3.1 Waits

Waits are nodes that cause the execution to pause until the occurrence of a given event. Elementary waits, represented in the language by the symbol , can expect a specific user's action, like a click on a given button, they can expect an event of the assistance system, like the triggering of a given assistance action, or they can wait for the end of a timer. A timer is associated to a duration and can be triggered after any event linked to the user's action or to the assistance system. For instance, a timer can be used to trigger an assistance action after two minutes without any user's action.

Several elementary waits can be combined to create a complex wait, represented by . Thus, such elements can wait for a given succession of elementary events making up a higher level event, for instance an action of red eyes correction on a photo.

3.2 Consultations

Consultations are nodes that fetch information, in order to personalize and contextualize the assistance. The aLDEAS language proposes several types of elementary consultations, represented in the language by . It makes possible the direct consultation of the user, to make him choose between several options for instance. It is also possible to consult the target-application state, in order for instance to know the content of a text field, or which item is selected in a combo box. Finally, aLDEAS makes possible the consultation of resources linked to the assistance, such as the user's profile (that contains notably information on the user's preferences regarding assistance, or his skills in the target-application), the assistance history (that contains information relative to the rules and actions triggered for the user), and the user's traces (that contains information about all the interactions between the user and the target-application interface). Elementary consultations can be combined by a logical formula to create a complex consultation represented by . A consultation, either elementary or com-

plex, returns a value which can have different types: Boolean, text or number. For instance, a consultation of the assistance history can return a number indicating how many times an assistance action has been trigger.

Consultation nodes typically have several outgoing edges, represented as ⋮ (alternative branching), where each edge is labeled with a Boolean expression. The execution of the block will continue only along the edge(s) with the expression of which is satisfied by the returned value. If several edges are satisfied, the corresponding paths will be executed in parallel. An example of alternative is given in the rule R0 in Fig. 7.

3.3 Assistance actions

Our language proposes two categories of elementary assistance actions, represented by ▭: integrated actions and actions external to the target-application interface. The grayed actions on Fig. 3 and Fig. 4 that are the only aLDEAS actions that are not implemented in the current version of SEPIA (cf. section 5).

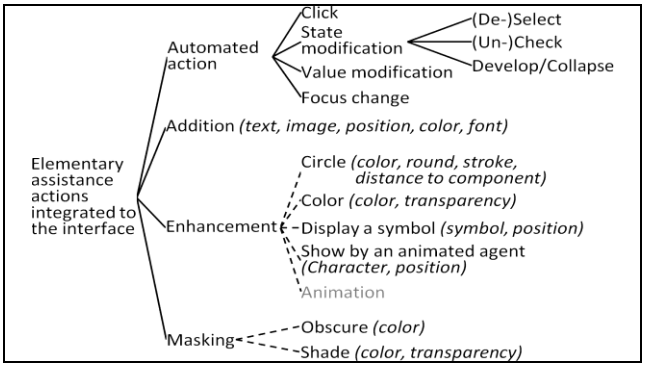

Fig. 3. Elementary assistance actions integrated to the target-application interface

The *integrated actions* act directly on a given component of the target-application interface, like a button or a text field. aLDEAS proposes four types of such actions on a component (cf. Fig. 3): automated actions, to act on behalf of the user; addition of component, to enrich the target-application interface (for instance by a new button enabling the user to ask for help); enhancement, to guide the user and attract his attention on a component; and masking, to simplify the target-application interface to the user's eyes. An action integrated to the target-application interface can be associated to several optional parameters, indicated by dotted line in Fig. 3. For instance, for an enhancement, a component of the target-application interface can be shown by an animated agent or circled by a stroke with a given color and size.

On the other hand, *actions external* to the target-application interface are not specifically linked to a component. aLDEAS proposes three types of such actions (cf. Fig. 4): messages, associated to a text that can be displayed and/or read; animations, for instance an animated agent applauding; and resources that can be proposed to the user, for instance a demonstration video, a forum or an application like the calculator.

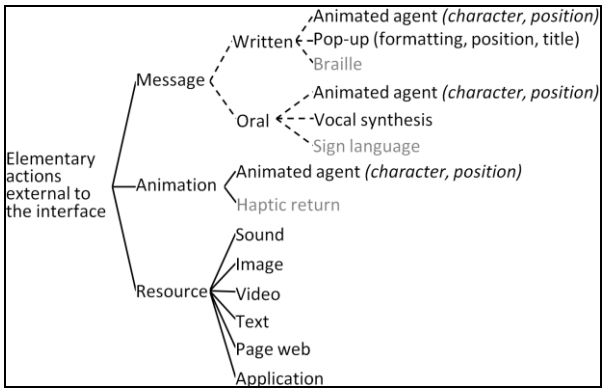

Fig. 4. Elementary assistance actions external to the target-application interface

Whenever an elementary action node is encountered, the corresponding action is launched, and the execution walks immediately to the next node; this allows several actions to be active at the same time (such as a message and an enhancement). Actions can stop on their own (e.g. playing a sound), be stopped by the user (e.g. by closing a message box), but some actions need to be explicitly stopped. This can be specified with the stop marker : this node causes all the actions started in the current block to stop. Fig. 5 gives the example of a block composed of two elementary assistance actions: a message and an enhancement. The action also contains an end event: the message and the enhancement will disappear after 30 seconds.

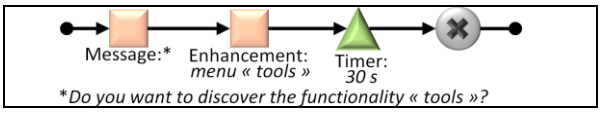

Fig. 5. Example of a block with a stop marker

3.4 Complex assistance actions

Any aLDEAS block can be invoked in another block as a complex assistance action, represented by the symbol . This node causes the execution of the invoking block to stall until the execution of the invoked block finishes.

A complex assistance action can return a value, which must be indicated under the corresponding end marker (cf. examples Fig. 9 and Fig. 10). In the blocks invoking this action, the corresponding node will be followed by an alternative branching, as described in the end of Section 3.2.

4 aLDEAS patterns

In order to facilitate the definition in aLDEAS of complex assistance actions, we propose a set of patterns. For that purpose, we enrich our language with two structures: “or” branching and optional elements preceded by a “?”. These structures do

not describe the assistance execution; they represent a choice to be made by the assistance designer at the time of the pattern instantiation.

4.1 Assistance rules pattern

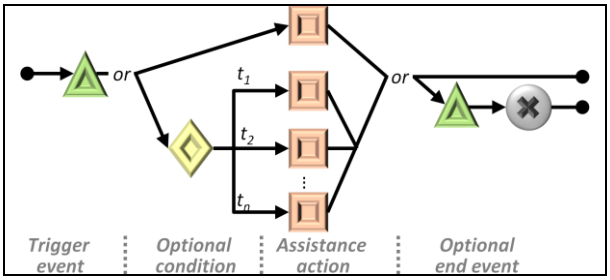

Fig. 6. Assistance rules pattern

The aim of our language is to enable assistance designers to specify the assistance that they wish for a target-application, by means of a set of rules. In aLDEAS, we define an assistance rule as a complex action that complies with the pattern given in Fig. 6. A rule starts with a trigger event that corresponds to the waiting of an event. A rule then contains an assistance action or a consultation with an alternative branching followed by as many assistance actions. A rule can be associated with an end event that corresponds to a wait for that event followed by the stop marker. If a rule has no end event, it will leave all its actions to stop on their own (or be stopped by the user). Fig. 7 gives the example of two assistance rules created for the target-application PhotoScape, a freeware for photo correction. Rule R0 contains a trigger event (the assistance launch) and a consultation of the user asking if he/she wishes help. R1 will be triggered if the user chooses the option "yes, I want help". The rule R7 is triggered by a click on component 228 (the "tools" menu of PhotoScape). R7 triggers an assistance action made up of two elementary assistance actions: the enhancement of component 210 (the "red eyes" button of PhotoScape), and a message suggesting the user to click on this button. The click on the "red eyes" button will put an end to the rule R7 and thus delete both the enhancement and the message.

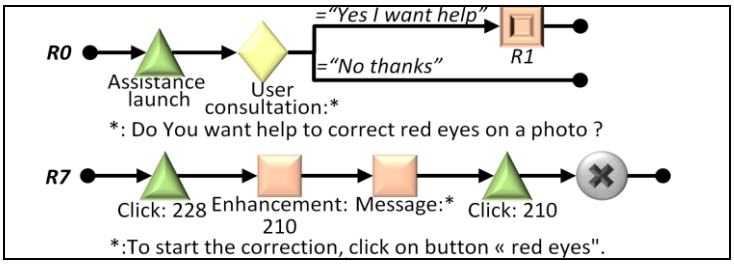

Fig. 7. Example of the assistance rules R0 and R7 for PhotoScape

4.2 Complex assistance actions pattern

The aLDEAS language makes possible the definition of complex assistance actions, combining several elementary elements. The definition of such actions can be difficult. For this reason, we defined complex assistance actions patterns associated with our language, in order to facilitate the definition of some complex actions frequently encountered in existing assistance systems: animated agents actions, guided presentation and step-by-step. In this section, only the patterns relative to step-by-step actions are given.

An **animated agent action** makes possible the combination of several elementary actions for a given character: messages, animation (show a component, applaud, greet…), and moving on the screen. For instance: the animated agent places itself next to the "e-mail" field, it displays the message "don't forget to fill your e-mail address", and it shows the field until the user modifies the value of this field.

A **guided presentation** is made of several steps, in which a component is enhanced and possibly presented by a message. We frequently find this kind of assistance actions in existing applications, notably when an application is launched for the first time, or after an update. The instantiation of the guided presentation pattern is a way for the assistance designer to representation his knowledge on the target-application functionalities.

A **step-by-step** aims at facilitating the achievement of a given task by detailing it in several elementary steps. Each step corresponds to an action to perform on a component of the target-application interface. We call **automated step-by-step** a step-by-step in which the assistance system will perform the actions on behalf of the user. We call **guided step-by-step** a step-by-step in which the assistance system will ask the user to perform the required actions. The patterns of a step for automated step-by-step and for guided step-by-step are given respectively in Fig. 9 and Fig. 10: these two step patterns are used by the step-by-step pattern (cf. Fig. 8), on which the step are represented by . The instantiation of the step-to-step pattern is a way for the assistance designer to representation his knowledge on the different actions to do in order to perform a given task in the target-application.

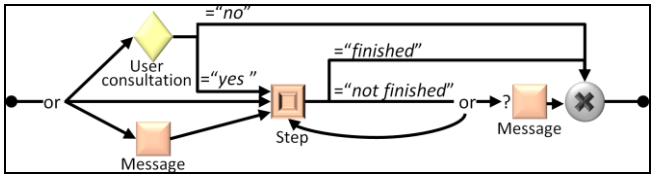

Fig. 8. Step-by-step pattern

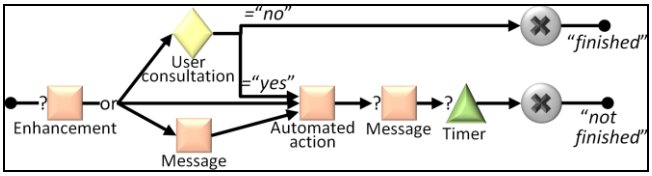

Fig. 9. Step for automated step-by-step pattern

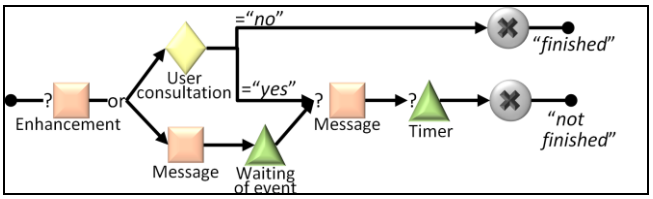

Fig. 10. Step for guided step-by-step pattern

5 IMPLEMENTATION IN SEPIA

We implemented aLDEAS and the patterns that complete it in the SEPIA environment (Specification and Execution of Personalized Intelligent Assistance).

The **assistance editor** is a tool aimed at assistance designers. It implements aLDEAS and makes possible the specification, for any given target-application, of an assistance system described by a set of assistance rules complying with the rule pattern (cf. Fig. 6). The assistance editor provides an interface for the definition the elementary assistance action presented in Section 3.3 (except the grayed actions on Fig. 3 and Fig. 4), and for the definition of action that instantiate the pattern proposed in Section 4.2. No knowledge on aLDEAS is required to use the assistance editor.

We developed a set of **epi-assistants**, able to perform in a target-application the elementary assistance actions proposed by aLDEAS and defined thanks to our assistance editor. The *automated actions* are available on any web application, on Windows native and Java applications, and on Linux GKT and Qt applications. Our epi-assistants performing these actions use a user script for web applications and different accessibility libraries for desktop applications: UIAutomation, JavaAccessibility and ATK-SPI [4]. *Enhancement* and *masking* actions are available for the same categories of applications as they need the same technology to find the bounding rectangle of the component that must be enhanced or masked. *Animated agent actions* are available on any Windows web or desktop application. *Messages* and *resource launch* are available for any application.

We developed a **generic assistant engine** able to execute the assistance specified by the assistance designer in the editor. To perform elementary assistance actions, the engine invokes one of our epi-assistants. Regarding complex assistance actions instantiating one of our patterns, the engine ensures their management and invokes an epi-assistant when necessary.

6 EVALUATIONS

The propositions presented in this paper have already been evaluated in several ways. Regarding the feasibility of our approach, it is demonstrated by its implementation in SEPIA, through the editor and the engine, completed by our epi-detectors and epi-assistants. In this section, we present the studies lead to evaluate the usability of

aLDEAS. We also performed several studies to evaluate the utility and acceptation of the assistance provided to end users that are not presented here [5].

6.1 Usability of aLDEAS

We made two experiments that aimed at evaluating the usability of aLDEAS language by assistance designers, with both computer-scientists and non-computer scientists.

Use of aLDEAS by computer scientists

In the context of a computer science master degree, students have developed in pairs simple ILE (Interactive Learning Environment), without any assistance system. We asked to the 29 students the create assistance system for the 14 ILE developed in order to demonstrate that aLDEAS can help assistance designers to specify an assistance system. First, each pair has been separated in two rooms, during 90 minutes. In the first rooms, 14 assistance designers have to define "on paper" an assistance system for their ILE, without any formalism. In the other room, 15 assistance designers have to define an assistance system in aLDEAS. They previously received explanations on aLDEAS. Each pair was then gathered during 90 minutes in order to share their work, and to define their final assistance system in aLDEAS. In a second time, the assistance designers had to use the SEPIA system during 3 hours, in order to define their assistance system, and to test it in their ILE.

Each pair succeeded in creating an assistance system with aLDEAS. A satisfaction questionnaire showed that 93% of the assistance designers found aLDEAS easy to use, and 76% thought that aLDEAS helped them to define their assistance system. 63% of the assistance designers thought that the use of aLDEAS facilitates the specification of the assistance with the SEPIA system and 67% found it easy to use to define assistance rules. Several assistance designers also noticed in their comment to the satisfaction questionnaire that aLDEAS helped them to work in pair and to formalize their ideas.

Use of aLDEAS by non-computer scientists

We ask five non-computer scientists to use aLDEAS in order to represent an assistance system for the red eyes correction on a photo with PhotoScape. These five assistance designers were between 11 to 68 years old. The assistance designers received explanations on aLDEAS during 5 minutes, and they had examples of aLDEAS rules, coming from another assistance system.

First, the assistance designer used successfully aLDEAS language to define "on paper" between 7 to 9 assistance rules complying with aLDEAS pattern. This work lasted between 30 to 45 minutes. In a second time, they used the SEPIA system to create and test their assistance system, after a short presentation of the SEPIA assistance editor. This work lasted between 55 to 95 minutes.

Four assistance designers on five found aLDEAS easy to use, and all of them declared that aLDEAS helped them a lot to create their assistance system. What's more, they found that the definition of rules with the SEPIA editor is easy once the rules are

defined in aLDEAS. These encouraging results seem to show that the aLDEAS language and its implementation in the SEPIA system can be used by assistance designers without any knowledge in programming. It has to be confirmed by a wider experiment with non-computer scientists, with other target-applications than PhotoScape.

7 DISCUSSION

In order to evaluate the coverage of the language that we propose, we used it to model various existing assistance systems, representative of the assistance types that are the most frequently found. Thus, aLDEAS makes possible the modeling of assistance systems frequently used in Web applications that contain a form: for instance, theses assistance systems explain to the user where to find information to fill a field, or advise the user when a field is not filled. Guided presentation of a software or functionality, a typical assistance system, can also be modeled in our language. Otherwise, there are many tutorials proposed by expert users in order to guide a user step-by-step to achieve a task, thanks to annotated screen shots and messages. Our language allows to model such assistance systems, with the advantage that they will be integrated in the target-application. Thus, instead of annotated screen shots, the assistance system can directly enhance the target-application user interface. Nevertheless, some assistance systems cannot be modeled with our language: it is the case for assistance systems very specific to an application and that require information not available from outside the application. For instance, some recommender systems integrated in online sale applications use information relative to all the articles consulted or bought by all users. aLDEAS does not make possible the consultation of this kind of information.

We also evaluated aLDEAS language by comparing it to existing approach for *a posteriori* specification of assistance for a target-application. Compared to the approach proposed by [8], aLDEAS can model such assistance action, as a wait for a click event on a given link, optionally followed by a consultation of the browsing history, then by an assistance action of type message (that can contain link towards another web page or resource). aLDEAS also proposes elementary assistance action to launch resources directly without a need use a link in a message. Compared to the approach proposed by [2], aLDEAS allows to define assistance actions involving animated agents able to move themselves in a web page, to express themselves orally or textually, and by gestures and animations. We also facilitate the definition of such actions by proposing a pattern of animated agent actions. Finally, our language allows to define rules of the form <trigger event, trigger condition, assistance action, end event> (cf. Section 4.1) equivalent to the rules used in the approaches of [7] and [1]. What's more aLDEAS allows to define more complex and varied assistance rules. aLDEAS also proposes a larger choice of elementary assistance actions, and patterns facilitating the definition of assistance actions composed of several elements.

8 CONCLUSION AND PERSPECTIVES

The aLDEAS language and the complex assistance action patterns that we presented allow to define assistance systems suitable to their target-application. By adopting a fully epiphytic approach, we make possible the *a posteriori* plugging of assistance systems in existing applications. These applications have not to be specifically designed for the integration of assistance, any access to its source code is necessary and any programming skills are required of the assistance designer. The experimentation that we performed showed that aLDEAS can be used to define assistance systems "on paper", by assistance designers with or without programming skills. Moreover, aLDEAS seem to help assistance designers to represents the assistance that they wish for the target-application.

The aLDEAS language is implemented in the SEPIA system. The experimentation that we performed showed that SEPIA can be used by assistance designers, with or without programming skills. What's more, SEPIA can be used to provide end users with efficient assistance, in order to help them to achieve a given task, in particular in the context of the discovery or occasional use of the target-application.

We now work on a method to help the assistance designer to identify the assistance needs of an application, in order to help him/her to define an efficient assistance system. This method will also make possible the end users to provide the assistance designer with feedback, in order to help him/her to improve the assistance system.

References

1. Dufresne, A., Paquette, G.: ExploraGraph: a flexible and adaptive interface to support distance learning. In: Ed-Media, pp. 304-309, Victoria, Canada (2000).
2. Carlier, F., Renault, F.: Educational webportals augmented by mobile devices with iFrimousse architecture. In: ICALT, pp. 236-240, Sousse, Tunisia (2010).
3. Gapenne, O., Lenay, C., Boullier, D.: Defining categories of the human/technology coupling: theoretical and methodological issues. In: ERCIM Workshop on User Interface for All, Paris, France, pp. 197-198, (2002).
4. Ginon, B., Champin, P.-A., Jean-Daubias, S.: Collecting fine-grained use traces in any application without modifying it. In: Workshop EXPPORT of ICCBR, New-York, USA (2013).
5. Ginon, B., Thai, L. V., Jean-Daubias, S., Lefevre, M., Champin, P.-A.: Adding epiphytic assistance systems in learning applications using the SEPIA system. In: Ec-Tel, Graz, Austria (2014).
6. Paquette, G., Pachet, F., Giroux, S., Girard, J.: EpiTalk, a generic tool for the development of advisor systems. In: IJAIED, pp. 349-370, (1996).
7. Paquette, G., Rosca, I., Mihaila, S., Masmoudi, A.: TELOS: A Service-Oriented Framework to Support Learning and Knowledge Management. In: E-Learning Networked Environments and Architectures, Pierre, S. (eds.), pp. 79-109, (2007).
8. Richard, B., Tchounikine, P.: Enhancing the adaptivity of an existing Website with an epiphyte recommender system. In: New review of hypermedia and multimedia, vol. 10, pp. 31-52, (2004).