Adding epiphytic assistance systems in learning applications using the SEPIA system

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

Adding epiphytic assistance systems in learning applications using the SEPIA system

Blandine Ginon[1, 2], Le Vinh Thai[1, 3], Stéphanie Jean-Daubias [1, 3], Marie Lefevre[1, 3] and Pierre-Antoine Champin[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. In this paper, we present how the SEPIA system can be used to plug pedagogical and technical assistance systems in applications used by learners in an educational context. The SEPIA system consists in two main tools: an assistance editor that enables assistance designers to specify the assistance they wish for existing applications, and a generic assistance engine that executes the specified assistance in order to provide the application end-users with personalized assistance. We also present an experimentation of an assistance system setup with SEPIA in the context of a bachelor degree.

Keywords: User assistance, epiphytic approach, generic models.

1 Introduction

In the educational context, more and more applications are used, whether they are specifically dedicated to learning, the ILEs (Interactive Learning Environment), or other software used as material of a learning activity. The acquisition of knowledge by learners can nevertheless be compromised by difficulties of handling and use of the application. Adding an assistance system, suitable to both the application and the activity pedagogical goals, is a solution to face these difficulties and prevent the user from abandoning or under-exploiting the application and losing motivation [8].

The work presented in this paper is in the context of the AGATE project in which we have developed the SEPIA system (cf. Section 2.2). With the SEPIA system, assistance designers can add assistance systems to existing applications. This paper aims to present how the SEPIA system can be used to setup assistance systems for applications used in an educational context.

First, we present the AGATE project, its theoretical propositions and their implementation in the SEPIA system. Then, we present the actors and applications concerned by assistance in an educational context, before identifying the needs to which such assistance should answer. Finally, we explain how SEPIA can be used in the educational context and we detail the experimentation that we performed. We conclude by discussing the strengths and weaknesses of our propositions for assistance to learners, comparing them with other approaches.

2 Epi-assistance

This research work takes place in the context of the AGATE project (Approach for Genericity in Assistance To complEx tasks). It aims at proposing generic models and unified tools to make possible the setup of assistance systems in various existing applications, that we call *target-applications*. This project adopts a fully generic and epiphytic approach [14]. An epiphytic application, that we call *epi-application*, is an application able to perform actions in a third-party application without requiring any change to it. Thus, the functioning of an epiphytic assistance system doesn't disturb the functioning of the target-application. The models and tools proposed in the AGATE project are not specific to an application or to a domain, but on the contrary they can be used to setup assistance in a wide range of applications, without a need for these target-applications to have been designed specifically to enable the plug of assistance.

2.1 State of the art of epi-assistance approaches

Several authors have studied the *a posteriori* specification of assistance systems for existing target-applications. The approaches of [15] and [7] make possible the plugging of an advisor system in a scenario from the Telos and ExploraGraph environments respectively. 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 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 [16] and by the CAMELEON model [4] make possible the plugging of 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 for [16], or an animated agent able to move, perform animations and display messages for [4].

Nevertheless, these different approaches cannot be used in any application. Indeed, they are specific to a given environment or to Web applications. In the context of the AGATE project, we are interested in the *a posteriori* plugging of assistance systems to very diverse existing target-applications. What's more, we would like to make possible 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 [16], 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.

2.2 Propositions: process, language and system to add epi-assistance

The AGATE project leads to several propositions that we present in this section: the adjunction process of epi-assistance systems in existing target-applications, the aLDEAS language that enables the definition of various assistance systems and the SEPIA system that implements these theoretical propositions.

Adjunction process of epi-assistance systems

Figure 1 shows our adjunction process of epi-assistance systems organized in two phases: the assistance specification and the assistance execution in an epiphytic way.

The **assistance specification** is performed by an expert of the target-application, called the *assistance designer*. This preparatory phase enables the designer to specify the assistance that he wishes for a given target-application, expressed as a set of assistance rules. For this purpose, he must provide a description of the target-application interface in order to make possible the setup of an assistance that seems integrated in this application but executed in an epiphytic way. Then, the designer has to define the assistance rules that describe the behavior that the assistance system will have during the assistance execution.

The **assistance execution** concerns the target-application end-users. It consists in the execution of the assistance wished by the designer; it occurs at any use of the target-application by an end-user. This phase is composed of three processes: the monitoring of the target-application, the identification of an assistance need and the elaboration of an answer suitable to this need.

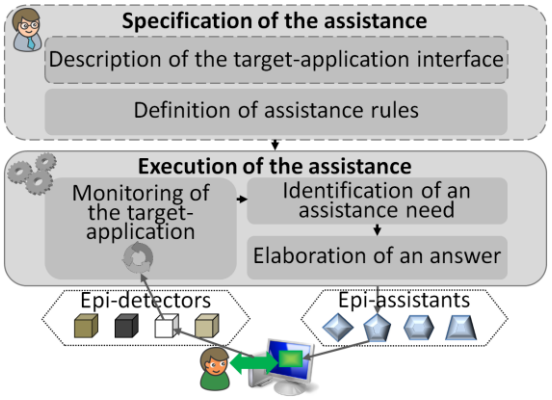

Figure 1 : Adjunction process of epi-assistance systems

The **monitoring of the target-application** is performed continuously while the user uses the target-application. It uses a set of epi-detectors (which are described in more detail in [9]) based on accessibility libraries that enable the access to all components of an application and the subscription to different kinds of events on these components. This process enables the epi-assistance system to get all information related to the interactions between the user, the target-application and the assistance system without disturbing the functioning of the target-application. The assistance system is

aware of any user's low level action (like clicks, mouse movements, menu openings or item selections). This process uses the description of the target-application interface, built during the specification of the assistance, in order to identify the source component of each low level event. Indeed, knowing only the type (click, keystroke) of the user's action is not enough: it is much more informative for the assistance system to know precisely on which button the user clicked, over which image he moved the mouse pointer, etc., as in our proposition. While it is not the topic of this paper, let us stress the fact that the assistance designer could couple SEPIA to a trace management system [20] for combining low-level actions into higher level actions (*e.g.* filling a form or correcting red-eyes on a photo) that could in turn be used to trigger assistance rules.

The **identification of an assistance need** is performed when the monitoring process detects an event: it uses the assistance description made by the designer. More precisely, an assistance need is identified by a rule whose trigger event has just been detected by the monitoring process and whose trigger conditions are satisfied. In this case, the elaboration of an answer is launched.

The **elaboration of an answer** to the user's need uses the assistance description to trigger one or several assistance actions in the target-application with an epiphytic approach. This process also implements the personalization of the assistance according to the assistance designers' wishes. Indeed, the launched assistance actions can differ from a user to another according to his specificities, like his preferences for instance. The assistance actions are performed in the target-application by epi-assistants.

The aLDEAS language

aLDEAS (a Language to Define Epi-Assistance Systems) is a language that enables the definition of assistance systems as a set of rules. This language is the core of the adjunction process of epi-assistance systems described above: it enables the designer to define the wished assistance during the assistance specification phase, and it is used during the assistance execution through its implementation. aLDEAS is constituted of different types of components that can be combined to create assistance rules and actions, in order to answer the most various needs. The main elements of aLDEAS are event waits, consultations and assistance actions.

An **event wait** causes the execution to pause until the occurrence of the given event. An event can concern a user's action (like a click on a given button), an action of the assistance system (like the triggering of a given assistance rule), an absence of event (like an absence of click during a given duration), or the performing of a "high-level" task (like the red-eyes correction on a photo).

Consultations fetch information in order to personalize and contextualize the assistance. Thus, the triggering of an assistance action can depend on information relative to the user's past actions (his traces), to his choices, to the state of the target-application (in order for instance to know the content of a text field, or which item is selected in a combo box) and to his profile. The user profiles used in the AGATE project can contain any information found relevant by the assistance designer in order

to personalize the assistance. It can contain in particular information on the user's preferences regarding assistance, or his skills in the target-application. For space reasons, how the user profile is constructed is not discussed in this paper.

Finally, aLDEAS provides a large choice of **assistance actions**: messages, component enhancements, automated actions on the target-application, proposition of external resources (like demonstration videos, forums or course materials), etc.

The SEPIA system

The SEPIA system implements these theoretical propositions through two main tools: an assistance editor and a generic assistance engine.

The **assistance editor** implements the assistance specification phase of the adjunction process of epi-assistance systems (cf. Figure 1). It provides assistance designers with a graphical interface for defining the assistance that they wish for a target-application, through a set of rules complying with the aLDEAS language.

The **generic assistance engine** implements the assistance execution phase of adjunction process of epi-assistance systems. It provides end-users with personalized assistance according to the rules defined by the designer.

The generic engine is completed by a set of **epi-detectors** that enables the target-application monitoring [9]. Currently, three epi-detectors have been developed: for Windows native applications, for Java applications and for Web applications.

Finally, the generic engine manages a set of **epi-assistants** that can perform assistance actions in the target-application at the request of the engine. The epi-assistants that we have developed can perform the assistance actions proposed by aLDEAS. In particular, they can display or read messages, enhance a component of the target-application interface, display an animated agent able to move on the screen, to express itself textually, orally, or by gestures and animations (for instance, the Merlin companion can applause and say "Well done, we succeeded!"). Some epi-assistants are also able to perform actions on the target-application interface, for instance to automatically click on a button, set text in a text field or select an item in a combo box.

aLDEAS also provides a set of *patterns* in order to facilitate the combination of aLDEAS elements, especially a pattern for assistance rules with the form <trigger event, optional condition, assistance actions, optional end-event>.

3 Assistance in an educational context

We wish to use the SEPIA system to setup epi-assistance systems in an educational context. More particularly, we would like to enable teachers to specify an assistance system for each target-application used by learners.

3.1 Which actors are concerned?

In an educational context, the assistance designer is a pedagogical designer that wishes to plug an assistance system in an application that has no assistance or an assistance which is incomplete or unsuitable to his needs. The *pedagogical designer* can

be a teacher or a pedagogical team, eventually assisted by a computer scientist or an expert of the target-application. The concerned end-users are the learners.

Another use of the SEPIA in the educational context could be the plugging of an assistance system by an assistance designer in a tool aimed at teachers as end-users, for example an authoring tool or a teaching assistant, but it is out of the scope of this paper.

3.2 Which applications are concerned?

There are two main categories of applications used in an educational context: the ILEs and the other software used with a pedagogical goal. Both can be concerned by the adjunction of an epi-assistance system.

An **ILE** is an application that is specifically intended for learning. Otherwise, a "**non-pedagogical" application**, *i.e.* an application not specifically intended for learning, can also be used in the educational context. First, a pedagogical activity can aim at the discovery of an application. For instance, a company can organize a course to train its employees to use an application that they need in their work, like an Enterprise Resource Planning or Computer-Aided Design application. Some courses for the general public may also aim at teaching how to use a tool, like image editing applications or office applications.

Furthermore, learning how to use a "non-pedagogical" application is not the only pedagogical activity that may require the use of that application. A teacher can ask his students to use Word in order to learn how to present official letters in an office course, or to use Excel in order to learn algebra [17]. A tutor in a foreign language can also ask his course participants to use a sound recording application in order to improve their pronunciation. In these contexts, the non-mastery of the software used as course material can slow down the acquisition of the involved knowledge.

Finally, in some cases, the goals of a pedagogical activity can be about both application-related knowledge and domain knowledge. For instance, in a computer science course, a teacher can require his students to use a given IDE (Integrated Development Environment), such as NetBeans or Visual Studio, to perform exercises in order to learn how to use the environment *and* to acquire programming skills.

3.3 What are the assistance needs?

The identification of the assistance needs is a key task for the specification of an assistance system. Thanks to a bibliographic study and to a study of existing assistance systems, we identified the main assistance needs of learners that use applications in an educational context (cf. Table 1). These needs are of two kinds: technical (T1 and T2) or pedagogical (P1 to P10). In this section, we illustrate these assistance needs with examples coming from research works.

Learners' assistance needs in an educational context	Description
T1. Handling	To help the first use of the application
T2. Use	To help the current use of the application
P1. Choice of the activity	To suggest activities suitable for the learner
P2. Learning prerequisites	To help the learner to acquire not mastered prerequisites
P3. Explanations on steps	To split up the different steps to perform for the activity
P4. Clues	To give clues helping to find the solution
P5. Examples	To give examples of similar problems or situations
P6. Transitional diagnostic	To add diagnosis as a complement to the application functionalities
P7. Explanations on errors	To explain his errors to the learner
P8. Sub-task automation	To perform a part of a task instead of the learner
P9. Summary, monitoring	To provide a monitoring during or at the end of the activity
P10.Pedagogical guidance	To propose a pedagogical scenario integrated in the application

Table 1: Assistance needs in learning

Technical assistance needs

Learners who use an application for the first time can face difficulties in handling (cf. T1-Table 1). They can also face difficulties in later uses of an application (T2), particularly in case of occasional use, or discovery of a new functionality. In general, these technical difficulties don't directly concern the knowledge to be taught and can also occur in a non-educational context. Nevertheless, they can prevent or slow down the acquisition of the involved knowledge or cause the learner to give up the pedagogical activity.

The setup of an assistance system to answer these technical assistance needs is one solution to face these difficulties. More particularly, in the educational context, such assistance systems prevent learners from losing time or motivation because of technical difficulties. What's more, in classrooms, a technical assistance system can make the learners more autonomous and thus enable the teacher to reduce his technical interventions in order to concentrate himself on the pedagogical aspects.

Pedagogical assistance needs

In addition to the technical assistance needs, learners that use an application in an educational context can face difficulties to which a pedagogical assistance could answer. The goal of a pedagogical assistance is not necessarily to enable a learner to finish correctly and quickly a pedagogical activity. To provide the solution to a learner seems not pertinent. However, some kinds of pedagogical assistance can be considered as pertinent by the teacher in order to facilitate the acquisition of the involved knowledge.

It is the case of the assistance for the choice of a pedagogical activity (P1), in particular when this choice is done by the pedagogical designer in a personalized way for each learner [11]. The help to acquire prerequisites (P2) can be useful to suggest to a learner to perform other activities (like exercises or courses) in order to help him in an activity for which he doesn't master all the prerequisites. For instance, the system

ELM-ART [3] presents to the learner links towards prerequisite concepts if he has never learnt or could not master them. Explanations on the steps to follow (P3) to perform an activity can be interesting to guide a learner and make him acquire methodology. For instance, in training mode, Aplusix [1] explains the steps of algebra problems solving upon request of the learner. Providing a learner with clues (P4, like in Aplusix) or examples (P5, like in Ambre-add [13]) can also facilitate the solving of the activity, the acquisition of the involved knowledge, as well as motivate the learner. The transitional diagnosis (P6) is useful to confirm to the learner that what he did is correct or to inform him that he did mistakes. Explanations on errors (P7) can be also given in order to help the learner to understand his mistakes. For instance, while Aplusix can only indicate that there is an error, Intelligent Tutoring Systems in Mobile Author [19] can give explanations on errors. In some cases, it can be relevant to automate a part of the task (P8) without compromising the acquisition of the involved knowledge. For instance, in the case of a difficulty in problem solving, Aplusix allows the sub-tasks automation upon request of the learner. A summary or a monitoring of the activity (P9) shows to the learner what he did and what he still has to do, in order to motivate him. For instance, the Explor@ Advisor Agent [12] displays the student's progression by indicating what has been achieved or completed. Finally, an assistance system can integrate a pedagogical scenario (P10) defined by the teacher and followed by the learner. For instance, OASIS [10] allows creating scenarios where learners are assisted step by step in the training mode, or where they try to achieve the final goal in a limited time without any help or feedback, in the evaluation mode.

As a conclusion, an assistance system used in an educational context may, if well designed, answer both technical and pedagogical assistance needs. The assistance provided to a learner must facilitate the acquisition of knowledge, complying with the designer's pedagogical strategy. In the next section, we present how the SEPIA system can be used to setup such assistance systems.

4 Use of SEPIA in an educational context

The SEPIA system has been designed mainly to enable the setup of assistance systems able to answer the technical assistance needs for the users of various target-applications. Nevertheless, we show in this section that SEPIA can also be used to setup assistance systems suitable to teachers' pedagogical strategies. First, we present examples of assistance systems specified with SEPIA for various applications used in educational contexts. Then, we detail the experimentation that we performed with SEPIA in the context of a bachelor course.

4.1 Use of SEPIA for various applications used in an educational context

In French schools, 8 year-old pupils have to acquire skills in using a computer for the B2I diploma [2]. For this reason, primary teachers make them use different tools: some are specifically intended for learning, like the web site "Je Revise", or with a

different purpose, like Paint. We have specified an assistance system for “Je Revise” that provides learners with pedagogical assistance and an assistance system for Paint in order to teach learners how to handle an image, which is one of the skills required for the B2I diploma.

Andes [18] is an Intelligent Tutoring System in physics problem solving. We have created an assistance system for Andes that guides students to choose activities in order to revise for an exam.

In the context of an ILE course in a master of University of Lyon, 20 assistance systems have been specified with SEPIA for a rudimentary ILE created by past students. These assistance systems aimed at helping 7-8 year-old children to discover the solar system using the ILE. Indeed this ILE didn’t provide any technical or pedagogical assistance.

We have created an assistance system for CodeBlocks that is used by students in the context of the first algorithmic course of University of Lyon. This assistance system guides students through all the steps of the first practical course.

We have also specified an assistance system to handle the forge (a platform allowing collaborative development) that is used by master students, and to teach them how to use SVN tools in order for instance to manage a project, to create a branch and to solve a conflict.

4.2 Experimentation of SEPIA for a HCI course

We have also performed a deeper experimentation of the SEPIA system in an educational context: the setup of an assistance system in the framework of a bachelor degree course in HCI (Human-Computer Interaction), at University of Lyon. During this course, that is not a programming course; students have to design graphical applications, mainly in Java using the NetBeans IDE. However, many students have little experience with NetBeans and in Java programming and teachers notice each semester that students face difficulties that slow down their work, especially for the first course.

For this reason, teachers created demonstration videos available on the course website [5]. They also decided to define a tutorial integrated to NetBeans with SEPIA [6], in order to help volunteer students to discover the IDE and to practice basic exercises during the first course of the semester.

Assistance specification by teachers

In a first step, four assistance designers used the SEPIA assistance editor to specify an assistance system for the target-application NetBeans. This assistance system was a tutorial constituted of five independent parts corresponding to the main students’ assistance needs identified by the teachers during the first courses of past semesters: creating a NetBeans project, editing the properties of a NetBeans project, adding and using a button in a form, creating a menu bar and handling GUI events.

Each part of the tutorial contains around 14 assistance rules and 23 assistance actions, mainly of the type *assistance messages* with explanations or instructions, asso-

ciated with an enhancement of the related components in the NetBeans interface. The creation of each part of the tutorial took about 3 hours. Although assistance specification represents a substantial work for teachers, the specified assistance system can be reused in future semesters.

As an example, Figure 2 shows a screen-shot of two assistance actions of the tutorial: a message and an enhancement of the text field “Arguments” in the screen “Project properties” of NetBeans. In this part of the tutorial, students learn how to retrieve and use these arguments in the parameter *args[]* of the *Main* function of the project. In this basic exercise, students have to use a *string array*, a *for* loop, and they have to cast a *string* to an *integer* and display a message in the *output*.

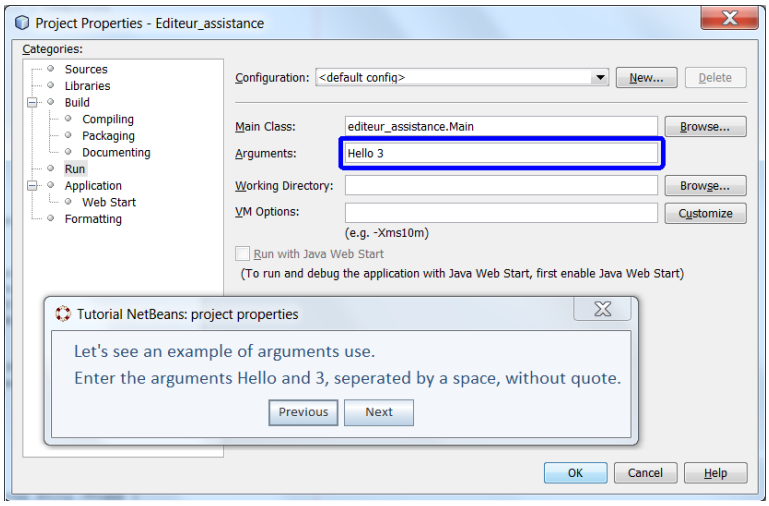

Figure 2 : Screen-shot of one SEPIA tutorial for NetBeans

Assistance execution for students

In a second time, teachers proposed to their volunteer students to use the tutorial at the beginning of the first course. On the 85 students, 64 volunteered to use the tutorial. Because of a lack of available computers with SEPIA assistance, only 52 students were able to participate. The tutorial sessions lasted between 30 and 90 minutes, depending on students. Indeed, students were free to work as they wanted with the tutorial: they could skip a part they deemed uninteresting and spend as much time as they needed on each part.

During the tutorial, students where free to exchange with other students and ask complementary information to teachers. Thus, we observed that some students who decided to skip a part of the tutorial changed their mind and do it after an exchange with another student who did it and found it useful. We also noticed that some students compared what they learned after each part of the tutorial. What’s more, some of the students who had already used NetBeans compared with each other the method learned with the tutorial to the method that they previously used.

Results of the experiment with NetBeans

The experiment was preceded by a questionnaire and a pre-test, in order to know the students' level in Java programing and their skill level in using NetBeans. After the definitive close of the assistance system, a final questionnaire and a post-test aimed at assessing their satisfaction and evaluating their progression concerning the target skills.

Students were very satisfied by the tutorial: 90.4% declared that they appreciated it (cf. left chart on Figure 3) and 62% wished to follow other similar tutorials integrated to NetBeans. In particular, in their comments on the final questionnaire, many students asked for tutorials integrated to NetBeans, but addressing more advanced Java programming skills, some students also asked for tutorials integrated to other tools and IDEs used in other courses. Moreover, most students found the tutorials more efficient than a demonstration video or a demonstration by the teacher (cf. Figure 4).

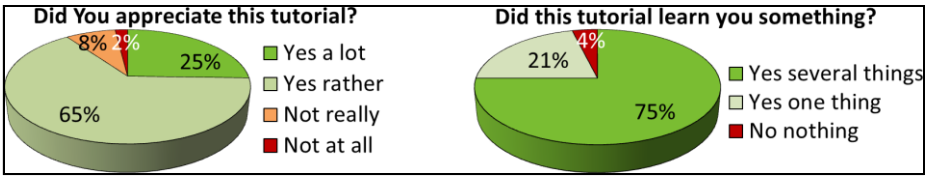

Figure 3 : Final questionnaire: students' satisfaction

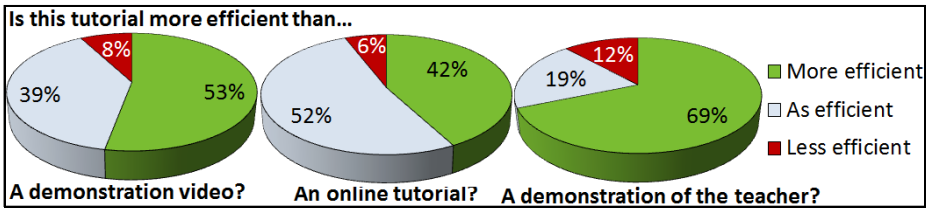

Figure 4 : Final questionnaire: students' opinion on the tutorial efficiency

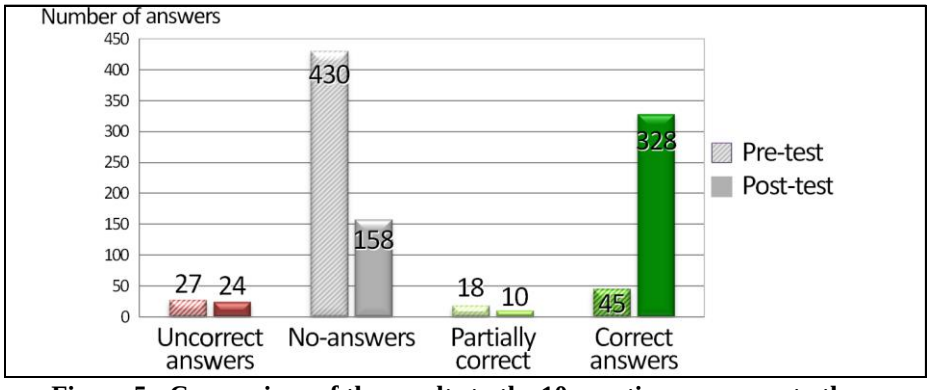

Figure 5 : Comparison of the results to the 10 questions common to the pre-test and the post-test, for the 52 participants

The tutorials seem to have facilitated the acquisition of knowledge and skills involved in the HCI course: 96.2% of the students declared that the tutorial taught them something (cf. right chart on Figure 3). This excellent result is confirmed by the improvement of their knowledge in NetBeans and Java programming measured by the comparison between the results of the pre-test and post-test (cf. Figure 5). Indeed, these tests contained notably 10 common questions with free answers: Figure 5 and Figure 6 concern these questions. We notice that the number of questions without answer has strongly decreased after the tutorial (- 63%), and that the number of good answer has very strongly increased (+ 629%). The post-test also contained 3 questions that wasn't in the pre-test, in order to see if students answered more correctly to the post-test only because they were more concentrate on the part of the tutorial that deals with the pre-test questions. For the post-test, the average success rate to the 10 common questions is 64%, and the average result to the 3 questions found only in the post-test is 62%, which is equivalent.

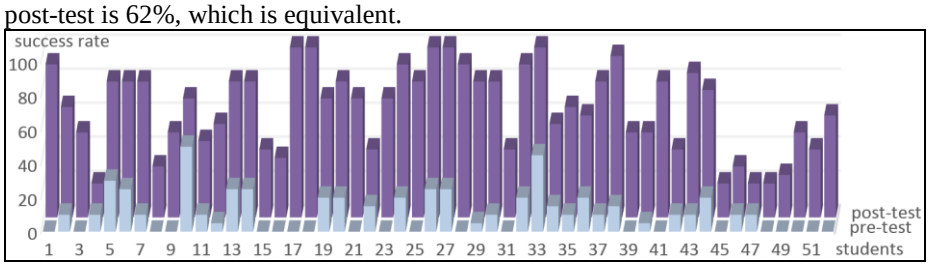

Figure 6 : Students' success rates to the 10 questions common to both tests

Figure 6 shows the success rate of each student to the pre-test (in the foreground) and to the post-test (in the background). We notice that all students improved their success rate after the tutorial: the smallest progression is 10%, the higher one is 100% (for students who had no correct answer to the pre-test and all correct answers to the post-test), and the average progression is 53.65%.

Furthermore, teachers were particularly satisfied by the use of the tutorial and plan to use it again for the next semesters. Teachers also would like to complete the tutorial with missing parts required by students in their comments. These parts, addressing more advanced Java programming skills, are not necessary for the first HCI course. However they can be useful for the rest of the semester, in the HCI course as well as for other courses that also require Java programming skills.

5 Discussion

This first experiment showed that the SEPIA system can actually be used to setup assistance systems in target-applications used by learners. Such assistance systems can meet the pedagogical goals of teachers and be well accepted by learners, like the tutorial for NetBeans used in a HCI course. Indeed, if the SEPIA system is particularly intended to setup technical assistance systems, it can also be used to setup pedagogical assistance system. However, there are some limitations to the use of SEPIA in an educational context that we discuss in this section.

First, for the moment, SEPIA cannot take domain knowledge into account. Such knowledge is yet necessary for most advanced assistance. For example, in a target-application that deals with algebra exercises, if the assistance designer wants to provide learners with an assistance that proposes transitional diagnosis, he has to specify all the possible correct answers (integrating for instance that "X-8" is equivalent to "-8+X"). It can be very demanding for the assistance designer and it becomes impossible in the case of an ILE that randomly generates the exercises. What's more, domain knowledge could be useful to generate examples and clues. Currently, the assistance designer specifies himself all the examples and clues necessary for his assistance system. With domain knowledge, the assistance system could automatically generate such examples and clues, contextualized with the current learner's production.

Then, knowledge relative to the activities proposed by an ILE would make possible an automation of some parts of the assistance designer's work. Indeed, for the moment, if an assistance designer wants the assistance system to guide learners through several activities, he has to specify the order of those activities. With the appropriate knowledge, the assistance designer would only have to specify that he wants the learners to perform the activities in the ascending order of difficulty. Moreover, with knowledge on the difficulty level of the ILE activities, the assistance designer could only specify that if a learner makes too many errors on an activity, then the assistance system should propose him for instance to move to an easier activity.

Finally, some improvements should be done to make the SEPIA assistance editor more accessible to assistance designers. Indeed, in the case of the HCI course, teachers were also computer scientists. However, in the general case, teachers may not be familiar with the concepts of rules, conditions, actions, and events that are at the core of the SEPIA system. Teachers could also face difficulties in understanding the description of the target-application interface that is an essential step to make a link between the epiphytic assistance system and the target-application. For the moment, SEPIA can be used by a teacher to setup an assistance system in an ILE, but with the help of a computer scientist, an expert of the ILE or a SEPIA expert.

6 Conclusion and future work

In this paper, we have presented the SEPIA system and its use to setup assistance systems in applications used by learners in an educational context. The assistance systems defined with SEPIA are epiphytic: they can be grafted on an existing application without a need to modify this application and without a need that the application has been designed specifically to enable the plugging of assistance.

We have shown how SEPIA can be used in an educational context to setup an assistance system suitable to teachers' pedagogical goals. This assistance system has been appreciated by learners and teachers that found it nice and useful. What's more this assistance system seems to have facilitated learning. We now plan to work on proposing patterns in order to simplify the work of assistance designers. In addition, the integration of domain knowledge to the SEPIA system would be an improvement to design assistance for ILEs involving complex problems solving.

References

1. Aplusix: http://www.aplusix.com/en/
2. B2I diploma: http://eduscol.education.fr/cid46073/b2i.html
3. Brusilovsky, P., Schwarz, E., Weber, G.: ELM-ART: An intelligent tutoring system on World Wide Web. Intelligent tutoring systems. pp. 261–269 (1996).
4. Carlier, F., Renault, V.: Educational webportals augmented by mobile devices with iFrimousse architecture. In: ICALT, Sousse, Tunisia (2010)
5. Demonstration videos for the HCI course: http://liris.cnrs.fr/~fduchate/ens/LIF14
6. Demonstration videos for the tutorial NetBeans http://liris.cnrs.fr/blandine.ginon/PhDWork.html
7. Dufresne, A., Paquette, G.: ExploraGraph: A Flexible and Adaptive Interface to Support Distance Learning. In: Ed-Media, pp 304-309. Victoria, Canada (2000)
8. Gapenne, O., Lenay, C., Boulier, D.: Defining categories of the human/technology coupling: theroretical and methodological issues, workshop ERCIM on User Interface for All, France, pp 187-198 (2002)
9. Ginon, B., Champin, P.-A., Jean-Daubias, S.: Collecting traces in a Windows application , workshop EXPPORT, ICCBR, New York, USA (2013)
10. Guéraud, V., Pernin, J.-P.: Developing pedagogical simulations: generic and specific authoring approaches. Artificial Intelligence in Education (AIED 99), Eds SP Lajoie and M. Vivet, IOS Press. (1999).
11. Lefevre, M., Jean-Daubias, S., Guin, N.: An approach for unified personalization of learning. UMAP Workshops (2012).
12. Lundgren-Cayrol, K., Paquette, G., Miara, A., Bergeron, F., Rivard, J., Rosca, I.: Explor@ Advisory Agent: Tracing the Student's Trail. WebNet. pp. 802–808 (2001).
13. Nogry, S., Guin, N., Jean-Daubias, S.: AMBRE-add: An ITS to Teach Solving Arithmetic Word Problems. Technology, Instruction, Cognition & Learning. 6, (2008).
14. Paquette, G., Pachet, F., Giroux, S., Girard, J.: Epitalk, a generic tool for the development of advisor systems. In: IJAIED, pp 349-370 (1996)
15. Paquette, G., Rosca, I., Mihaila, S., Masmoudi, A.: TELOS: A Service-Oriented Framework to Support Learning and Knowledge Management (2007)
16. Richard, B., Tchounikine, P.: Enhancing the adaptivity of an existing Website with an epiphyte recommender system. In: New review of hypermedia and multimedia, pp 31-52. (2004)
17. Ritter, S., Koedinger, K.: Towards lightweight tutoring agents. In AIED, pp 16-19 (1995)
18. Vanlehn, K., Lynch, C., Schulze, K., Shapiro, J., Shelby, R., Taylor, L., Treacy, D., Weinstein, A., and Wintersgill, M.: The Andes Physics Tutoring System: Five Years of Evaluations. In AIED, Amsterdam, Netherland (2005)
19. Virvou, M., Alepis, E.: Mobile educational features in authoring tools for personalised tutoring. Computers & Education. 44, 53–68 (2005).
20. Zarka, R., Champin, P.-A., Cordier, A., Egyed-zsigmond, E., Lamontagne, L., Mille, A.: TStore: A Trace-Base Management System using Finite-State Transducer Approach for Trace Transformation. In: MODELSWARD (2013)