Langage et outils pour la spécification et l'exécution d'assistance à l'utilisateur dans des applications existantes

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

Université de Lyon, CNRS,
Université Lyon 1, LIRIS, UMR5205, F-69622, France
{prénom}.{nom}@liris.cnrs.fr

RÉSUMÉ. *Afin de permettre la spécification et l'exécution automatique de systèmes d'assistance, nous proposons aLDEAS, un langage graphique. Un système d'assistance* aLDEAS est défini par un ensemble de règles *d'assistance respectant un patron de règles. Ce* lang*age est complété par plusieurs patrons qui facilitent la définition d'action*s *d'assistance* complexes fréquemment utilisées. aLDEAS est implémenté dans le système SEPIA, qui consiste principalement en un *éditeur d'assistance et un moteur générique d'assistance*. La plupart des applications existantes peuvent être surveillées par SEPIA afin de fournir aux utilisateurs finaux une assistance contextualisée et personnalisée. Nous avons réalisé plusieurs expéri*mentations avec d'une part*, *des concepteurs d'assistance et*, *d'autre part*, des *utilisateurs finaux afin d'évaluer nos propositions.*

ABSTRACT. To enables the specification of assistance systems, we propose aLDEAS, a graphical language. An aLDEAS assistance system is defined by a set of assistance rules complying with a rule pattern. This language is enriched by several patterns that facilitate the definition of complex frequently used actions. aLDEAS is implemented in the SEPIA system, that mainly consists in an assistance editor and a generic assistance engine. Most existing applications can be monitored by SEPIA in order to provide end-users with contextualized and personalized assistance. We performed several experimentations both with assistance designers and end-users in order to assess our propositions.

MOTS-CLÉS : l*angage, représentation de connaissances, assistance à l'utilisateur, système à* base de règles, outils épiphytes.

KEYWORDS: language, knowledge representation, user assistance, rule-based system, epiphytic tools.

DOI:10.3166/RIA.30.705-733 © Lavoisier 2016

1. Introduction

De nos jours, les applications informatiques sont omniprésentes et peuvent s'adresser à chacun d'entre nous, que nous soyons jeunes ou séniors, utilisateurs novices ou informaticiens confirmés. Qu'il s'agisse d'applications web ou d'applications bureau, ces applications peuvent être utilisées dans un contexte professionnel, éducatif ou personnel. Ainsi, dans le cadre de la vie privée, l'utilisation d'applications informatiques autrefois réservées aux professionnels s'est largement développée. C'est le cas, par exemple, pour des applications de bureautique, pour des outils de retouche d'images, de montage vidéo ou pour les éditeurs d'albums de photos.

De nombreuses applications informatiques proposent ainsi à leurs utilisateurs des fonctionnalités très variées et utiles, dans différents contextes et sur différents supports, que ce soit sur ordinateur, smartphone, tablette ou borne interactive. Pourtant, beaucoup d'utilisateurs renoncent à utiliser certains logiciels en raison de difficultés de prise en main et d'utilisation (Cordier et al., 2010 ; Ginon et al., 2012). De même, certaines applications informatiques sont sous-exploitées par leurs utilisateurs qui ignorent que l'application propose une fonctionnalité qui les intéresse ou qui sont effrayés par sa complexité réelle ou supposée. Une interface très chargée, une tâche perçue comme trop longue ou complexe, l'usage d'une fonctionnalité inconnue, l'implication dans une tâche de connaissances ignorées ou oubliées sont autant d'écueils potentiels pour les utilisateurs d'applications informatiques.

L'assistance à l'utilisateur est une solution pour pallier les difficultés de prise en main et d'utilisation d'une application, et ainsi réduire le risque d'abandon et de sous-exploitation des applications informatiques. Nous définissons l'assistance comme l'ensemble des moyens qui facilitent la prise en main et l'utilisation d'une application, d'une manière adaptée à l'utilisateur et au contexte d'utilisation. L'assistance permet à l'utilisateur qui en bénéficie d'exploiter pleinement les possibilités offertes par une application, et de s'approprier des connaissances et compétences requises pour l'utiliser. Ceci inclut les quatre types d'assistance définis par Gapenne et al., (2002) : substitution, suppléance, assistance et aide.

Le développement d'un système d'assistance pour une application est une tâche complexe et coûteuse, souvent négligée par les concepteurs d'application. Une personne autre que le développeur d'une application pourrait alors souhaiter greffer un système d'assistance dans une application existante qui en est dépourvue ou qui propose une assistance incomplète ou inadaptée. Ainsi, dans le cadre d'une communauté d'utilisateurs, un expert pourrait souhaiter concevoir un système d'assistance afin de faire bénéficier les autres utilisateurs de son expérience.

Dans le cas où le concepteur de l'assistance n'est pas le développeur de l'application-cible, le code source de l'application-cible est souvent indisponible. Il est alors impossible d'intégrer directement l'assistance dans l'application-cible. De plus, le concepteur de l'assistance n'est pas toujours un programmeur. Une alternative à l'approche classique de développement d'un système d'assistance

intégré à une application consiste à adopter une approche épiphyte[1] afin de rendre possible a posteriori la spécification et l'exécution d'un système d'assistance dans une application existante, sans la modifier. Un système d'assistance épiphyte est un système capable de réaliser des actions d'assistance dans une application-cible existante, sans perturber son fonctionnement (Paquette et al., 1996).

Dans cet article, nous proposons un processus d'adjonction d'un système d'assistance épiphyte à une application existante, constitué de deux phases : la spécification de l'assistance par un concepteur d'assistance, et l'exécution automatique de cette assistance dans l'application-cible. Comme pivot entre ces deux phases, nous proposons en section 5 le formalisme graphique aLDEAS. Celui-ci peut être utilisé par un concepteur d'assistance afin de spécifier l'assistance qu'il souhaite greffer à une application-cible donnée. En aLDEAS, l'assistance est spécifiée sous la forme d'un ensemble de règles, exécutables par un moteur d'assistance générique sans avoir à modifier l'application-cible. Aucune connaissance en programmation n'est nécessaire pour spécifier de l'assistance avec ce formalisme. Les patrons qui complètent aLDEAS et facilitent la spécification de l'assistance sont présentés en section 6. Nous présentons ensuite en section 7 le système SEPIA qui met en œuvre nos propositions. Dans SEPIA, deux types de connaissances sont manipulées : des connaissances sur l'application-cible et des connaissances relatives à l'assistance à lui greffer. Les connaissances sur l'application-cible consistent en une description technique de son interface, associée à des informations relatives à ses composants. Par exemple, la description d'une application peut préciser les valeurs minimale et maximale qu'un composant de type « spinner » accepte, et contenir une description de l'action associée à un bouton. De telles connaissances peuvent être utilisées pour adapter et contextualiser l'assistance fournie aux utilisateurs finaux de l'application-cible. SEPIA permet la découverte automatique de ces connaissances techniques, qui peuvent être complétées par un expert de l'application-cible. Les connaissances relatives à l'assistance sont spécifiées en aLDEAS par le concepteur de l'assistance, elles ne sont ni identifiées, ni intégrées automatiquement. Cet article n'aborde pas la nécessaire phase d'étude des besoins d'assistance des utilisateurs, intégrant experts de l'application-cible et ergonomes, qui est un préalable à la phase de conception. Nous terminons cet article, en section 8, en présentant les différentes expérimentations que nous avons réalisées afin d'évaluer nos propositions.

2. Scénarios d'usage

Dans cette section, nous présentons deux scénarios d'usage sur lesquels nous nous appuierons tout au long de cet article.

1. Le terme épiphyte est emprunté à la biologie. Un logiciel épiphyte, par analogie avec les plantes épiphytes qui vivent sur d'autres plantes, se greffe sur un logiciel existant.

Le premier scénario concerne un usage en contexte personnel de l'application de retouche d'images PhotoScape[2]. Marie aime faire des photos de ses petits enfants : pour corriger les yeux rouges sur ses photos, elle utilise PhotoScape que son fils lui a fait découvrir. Cependant, elle oublie régulièrement comment l'utiliser et doit téléphoner à son fils pour qu'il essaie de la guider à distance. Une solution au problème de Marie consisterait à permettre à son fils de mettre en place un système d'assistance dans PhotoScape, adapté aux besoins de Marie. En effet, son fils a une bonne connaissance de PhotoScape et maîtrise en particulier les outils nécessaires à la correction des yeux rouges sur une photo.

Le second scénario d'usage se situe dans un contexte éducatif. Jean est enseignant à l'université. Dans le cadre de son cours d'ergonomie, il demande à ses étudiants d'utiliser l'environnement de développement intégré NetBeans, afin de développer des applications graphiques en langage Java. Cependant, au début de chaque semestre, la plupart des étudiants n'ont jamais utilisé NetBeans, et certains n'ont aucune expérience de la programmation Java. En conséquence, Jean passe beaucoup de temps lors des premiers TP à aider les étudiants à prendre en main NetBeans et à débuter en programmation Java, ce qui lui laisse peu de temps pour transmettre les concepts d'ergonomie qui sont pourtant au centre de son enseignement. Une solution consisterait à permettre à Jean de mettre en place un tutoriel à l'intention de ses étudiants, afin de les guider dans la découverte de NetBeans et de leur faire pratiquer des exercices de programmation Java dans le but de les rendre plus autonomes. Un tel système d'assistance s'appuierait sur les connaissances de Jean relatives à NetBeans et ses différentes fonctionnalités, sur ses connaissances en programmation Java, ainsi que sur ses connaissances pédagogiques.

Ces deux scenarios d'usage mettent en valeur des besoins d'assistance dans des contextes différents et dans des applications différentes. Dans les deux cas, le concepteur potentiel de l'assistance n'est pas le développeur de l'application et n'a pas accès au code source de l'application-cible. De plus, dans le premier scénario d'usage, le concepteur potentiel de l'assistance n'est pas informaticien. En conséquence, dans ces deux cas il est nécessaire de permettre à un concepteur d'assistance, potentiellement sans connaissance en programmation, de mettre en place de l'assistance dans une application existante, sans avoir à la redévelopper ou à la modifier, et même sans accès à son code source. Pour cette raison, dans la section suivante, nous nous intéressons aux travaux relatifs à l'assistance épiphyte.

3. Travaux connexes

Plusieurs travaux concernent la mise en place a posteriori d'assistance dans une application existante, avec une approche épiphyte.

Les approches de Paquette et al., (2007) et de Dufresne et Paquette, (2000) permettent de greffer un système conseiller dans les environnements Telos et

2. http://www.photoscape.org

respectivement ExploraGraph. Ces systèmes conseillers sont définis par un concepteur d'assistance qui peut n'avoir aucune connaissance en programmation. Ces systèmes conseillers sont définis par un ensemble de règles d'assistance de la forme < événement déclencheur, condition de déclenchement optionnelle, action d'assistance, événement de fin optionnel >. La condition de déclenchement d'une règle peut inclure une consultation du profil de l'utilisateur ou de l'historique de l'assistance, afin de personnaliser et contextualiser l'assistance. Les actions d'assistance proposées sont des messages textuels affichés dans une fenêtre pop-up pour Telos, et des animations, des messages textuels ou vocaux communiqués par des agents animés dans ExploraGraph.

L'approche proposée par Richard et Tchounikine (2004) permet de greffer un système conseiller dans une application web. Dans cette approche, l'assistance est déclenchée par un clic de l'utilisateur sur un lien. Les actions d'assistance proposées sont des messages textuels affichés dans une fenêtre pop-up, les messages peuvent contenir des liens vers des pages web ou des ressources relatives à l'assistance. L'assistance fournie peut être personnalisée en fonction de l'historique de la navigation. Cette assistance peut être exprimée par un ensemble de règles d'assistance de la forme < événement déclencheur, condition de déclenchement optionnelle, action d'assistance >.

Le modèle CAMELEON (Carlier et Renault, 2010) permet quant à lui de greffer dans une page web un agent animé, capable de se déplacer, de réaliser des animations et d'afficher des messages grâce à des tags insérés de manière épiphyte.

Les règles d'assistance telles que nous les avons présentées semblent être adaptées pour représenter un système d'assistance épiphyte. La définition de règles d'assistance semble de plus accessible à des concepteurs d'assistance non-informaticiens. En revanche, dans les différentes approches présentées dans cette section, seules certaines applications sont concernées, puisque ces approches sont spécifiques à un environnement donné ou au web. Nous nous intéressons en revanche à la mise en place d'assistance a posteriori dans des applications existantes très variées. De plus, nous souhaitons permettre une personnalisation très fine de l'assistance, en fonction du profil de l'utilisateur et de l'historique de l'assistance, comme dans Telos et ExploraGraph, mais également en fonction des actions passées de l'utilisateur, non restreintes à un historique de la navigation comme (Richard et Tchounikine, 2004), ainsi qu'en fonction de l'état de l'application-cible. Finalement, pour rendre possible une plus grande variété dans l'assistance, nous souhaitons proposer un très large choix d'actions d'assistance. De plus, si une même action d'assistance peut être réalisée de différentes manières, une plus grande personnalisation de l'assistance est possible, notamment en fonction des préférences de l'utilisateur. Ainsi, un message d'assistance peut être affiché dans une fenêtre pop-up, transmis par un agent animé, ou lu par une synthèse vocale.

4. Processus d'adjonction d'un système d'assistance épiphyte à une application-cible

Le travail présenté dans cet article se situe dans le contexte du projet AGATE, qui vise à proposer des modèles génériques et des outils unifiés pour permettre la mise en place de systèmes d'assistance dans des applications existantes quelconques. Pour ajouter un système épiphyte à une application existante, nous proposons un processus d'adjonction constitué de deux phases (cf. figure 1). La première phase, la spécification de l'assistance, concerne un concepteur d'assistance, qui est expert de l'application-cible. Cette phase permet au concepteur de l'assistance de spécifier l'assistance qu'il souhaite mettre en place dans l'application-cible sous la forme d'un ensemble de règles d'assistance.

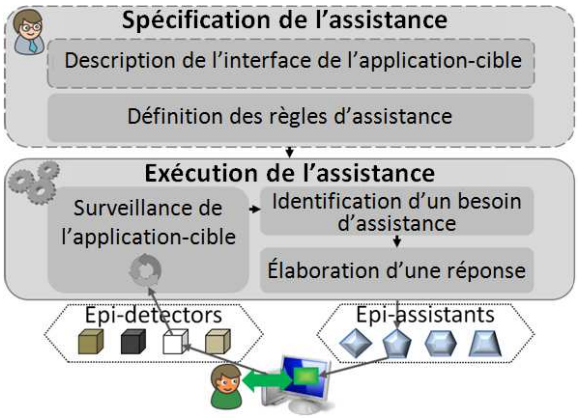

Figure 1. *Processus d'adjonction d'un système d'assistance* épiphyte à une application existante

La seconde phase, l'exécution de l'assistance, concerne les utilisateurs finaux de l'application-cible. Elle consiste en l'exécution automatique de l'assistance spécifiée par un moteur générique d'assistance. Cette phase a lieu à chaque utilisation de l'application-cible par un utilisateur final. Elle est constituée de trois processus. La surveillance de l'application-cible s'appuie sur un ensemble de détecteurs épiphytes, que nous appelons épi-détecteurs, permettant d'observer en continu et de tracer les interactions entre l'utilisateur final et l'interface de l'application-cible (Ginon et al., 2013). En parallèle, le processus d'identification d'un besoin d'assistance exploite les règles d'assistance définies par le concepteur de l'assistance, et déclenche le processus d'élaboration d'une réponse à un besoin d'assistance qui a été identifié. La réponse consiste en une ou plusieurs actions d'assistance exécutées dans l'application-cible par des assistants épiphytes, que nous appelons épi-assistants. Pendant la phase de spécification, le concepteur de l'assistance doit expliciter ses connaissances relatives à l'utilisation de l'application-cible. Cette phase doit être accessible à un concepteur d'assistance non informaticien et non expert en

ingénierie des connaissances, et ne doit donc pas requérir de connaissances en programmation. Par ailleurs, l'assistance spécifiée doit pouvoir être exécutée automatiquement par un moteur générique d'assistance durant la phase d'exécution de l'assistance. En conséquence, l'assistance doit être spécifiée de manière suffisamment formelle. Une solution pour rendre la définition formelle de l'assistance accessible au plus grand nombre de concepteurs d'assistance consiste à utiliser un formalisme pivot. Dans la section suivante, nous présentons donc le formalisme que nous proposons, aLDEAS.

5. Le langage aLDEAS

aLDEAS (a Language to Define Epi-Assistance Systems) est un formalisme graphique sans sémantique formelle que nous désignons par le terme langage pour plus de commodité. aLDEAS permet la définition d'un système d'assistance par un concepteur d'assistance. Nous avons choisi de proposer un langage graphique, très adapté à la représentation de systèmes d'assistance sous la forme d'un ensemble de règles simples. Les composants du langage aLDEAS (cf. figure 2) sont présentés en détails dans cette section. Nous montrerons par la suite comment ces composants peuvent être combinés pour créer des règles permettant de répondre à des besoins d'assistance très variés.

Un bloc aLDEAS est un graphe orienté, dont les nœuds et les arêtes sont des composants aLDEAS. Un bloc doit avoir exactement une source, qui doit être de type « début de bloc », et un ou plusieurs puits, qui doivent être de type « fin de bloc ». L'exécution d'un bloc aLDEAS consiste à parcourir le graphe; les effets de chaque type de nœud sont décrits par la suite.

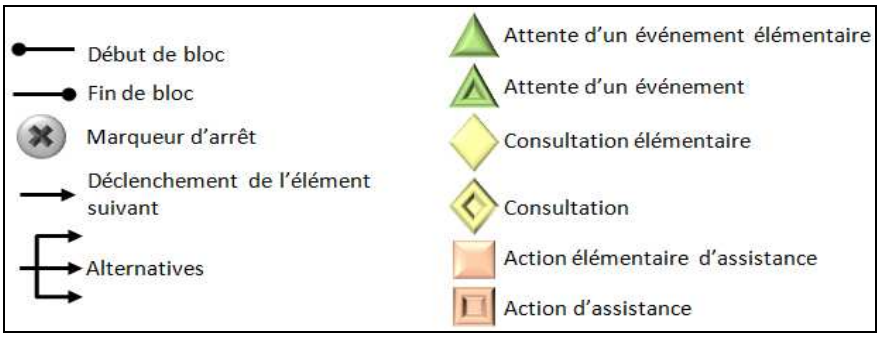

Figure 2. Composants *d'*aLDEAS

5.1. *Attentes d'événements élémentaires*

Les attentes d'événements sont des nœuds qui provoquent la suspension de l'exécution du bloc jusqu'à ce que l'événement attendu ait eu lieu.

Les attentes d'événements élémentaires, représentées dans le langage par le symbole (cf. figure 3), peuvent concerner une action de l'utilisateur, comme un clic sur un bouton donné ; elles peuvent concerner un événement lié au système d'assistance, comme le déclenchement d'une action d'assistance donnée ; ou la fin d'un timer donné. Un timer est associé à une durée et peut être déclenché à la suite de n'importe quel événement lié aux actions de l'utilisateur ou du système d'assistance. Par exemple, un timer peut être utilisé pour déclencher de l'assistance lorsque l'utilisateur est resté inactif pendant deux minutes.

Plusieurs attentes d'événements élémentaires peuvent être combinées pour former une attente d'événement composé, représenté par le symbole . Ainsi, un événement composé d'une succession d'événements élémentaires, comme des clics, peut représenter une action de plus haut niveau, comme la correction des yeux rouges sur une photo ou l'envoi d'un mail.

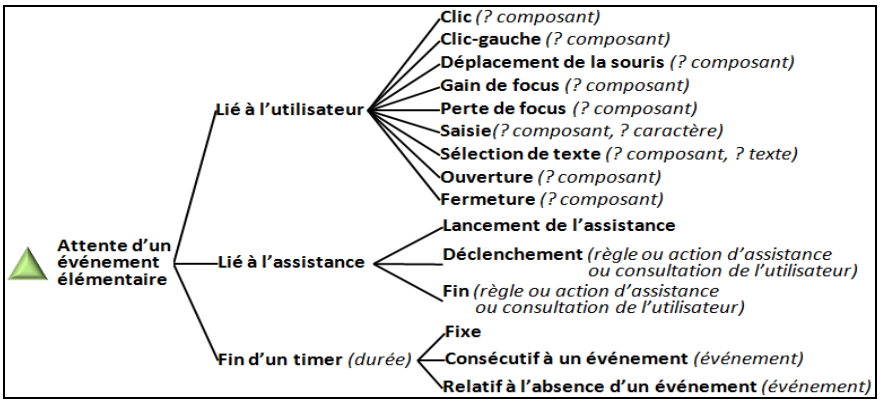

Figure 3. aLDEAS : attente*s d'événement*s élémentaires

5.2. Consultations

Les consultations sont des nœuds qui permettent d'aller chercher de l'information, afin de personnaliser et contextualiser l'assistance. Le langage aLDEAS propose plusieurs types de consultations élémentaires, représentées dans le langage par le symbole (cf. figure 4). Une consultation élémentaire permet la consultation directe de l'utilisateur final, afin de lui proposer plusieurs options par exemple. Une consultation élémentaire permet également de consulter l'état de l'application-cible, afin de connaître le contenu d'un champ de saisie ou de savoir quel item est sélectionné dans une liste déroulante par exemple. Enfin, aLDEAS permet la consultation de ressources liées à l'assistance, telles que le profil de l'utilisateur (qui peut notamment contenir des informations relatives à ses préférences vis-à-vis de l'assistance, ou son niveau de maîtrise des outils de l'application-cible), l'historique de l'assistance (qui contient toutes les informations relatives à l'assistance qui a été fournie), et les traces de l'utilisateur (qui

contiennent les informations relatives aux interactions passées entre l'utilisateur, l'interface de l'application-cible et le système d'assistance).

Les consultations élémentaires peuvent être combinées par une formule logique afin de créer une consultation complexe, représentée par  . Une consultation, qu'elle soit élémentaire ou non, retourne une valeur qui peut avoir différents types : booléen, texte ou nombre. Par exemple, une consultation de l'historique de l'assistance peut retourner un nombre indiquant combien de fois une action d'assistance a été déclenchée ; une consultation de l'utilisateur retourne un texte qui correspond à l'option choisie par l'utilisateur.

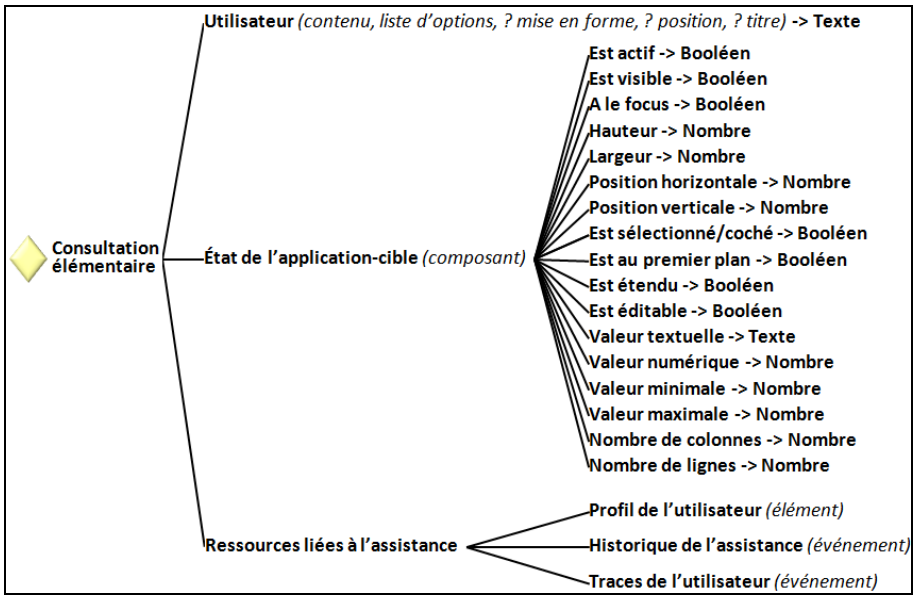

Figure 4. aLDEAS : consultations élémentaires

Les nœuds de type consultation sont généralement suivis par des alternatives à n branches, représentées par  , dans lesquelles chaque branche est associée à un test. L'exécution du bloc se poursuit en parallèle dans toutes les branches pour lesquelles la valeur retournée par la consultation vérifie le test correspondant. Un exemple d'alternative sera donné dans la règle R0 dans la section 6.1.

5.3. Actions *d'assistance*

aLDEAS propose deux catégories d'actions élémentaires d'assistance, représentées par le symbole  : les actions intégrées et les actions extérieures à l'interface de l'application-cible. Toutes les actions d'assistance aLDEAS sont mises

en œuvre dans le système SEPIA (cf. section 7), à l'exception de celles grisées sur les figures 5 et 6.

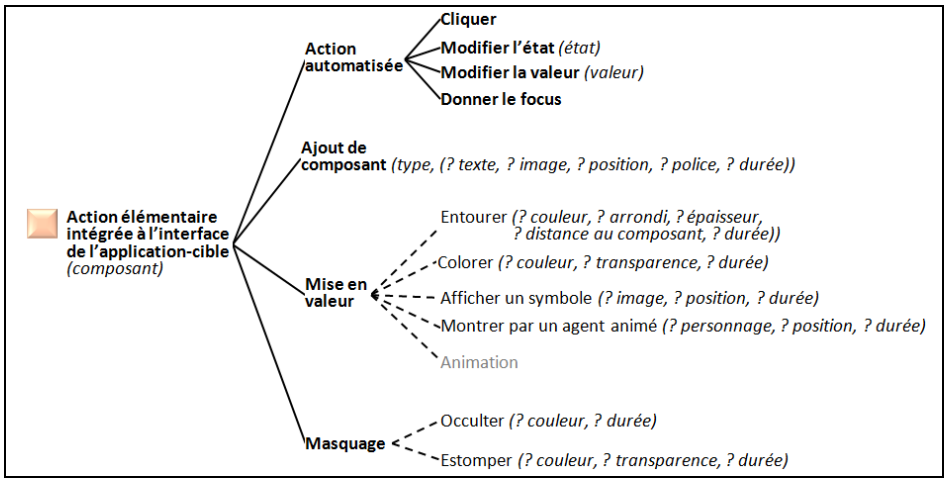

Figure 5. aLDEAS : *actions d'assistance élémentaires intégrées à l'application*-cible

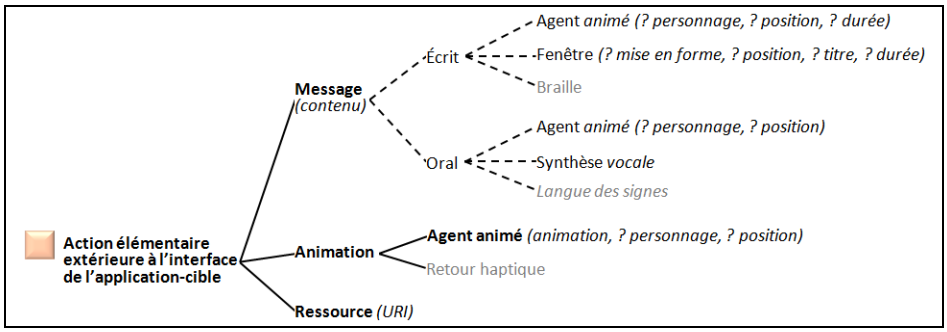

Figure 6. aLDEAS : *actions d'assistance élémentaires extérieures à l'application*-cible

Les actions intégrées agissent directement sur un composant de l'interface de l'application-cible, comme un bouton ou un champ de saisie. aLDEAS propose quatre types d'actions intégrées (cf. figure 5) : les actions automatisées permettent d'agir à la place de l'utilisateur, par exemple pour remplir à sa place un formulaire ; l'ajout de composants permet d'enrichir l'interface de l'application-cible, par exemple pour ajouter un bouton permettant de demander de l'aide ; les mises en valeur permettent de guider l'utilisateur et d'attirer son attention sur un composant ; et les masquages permettent de simplifier l'interface de l'application aux yeux de l'utilisateur. Les actions intégrées à l'interface de l'application-cible peuvent être

associées à plusieurs paramètres, indiqués entre parenthèses ou par des pointillés sur la figure 5. Par exemple, un composant peut être mis en valeur par un agent animé ou être entouré par un cadre d'une couleur et d'une épaisseur paramétrable.

Les actions extérieures à l'interface de l'application ne sont pas spécifiquement liées à un composant. aLDEAS propose trois types d'actions extérieures à l'application-cible (cf. figure 6) : les messages, associés à un texte qui peut être affiché et/ou lu ; les animations, par exemple un agent animé qui applaudit ; et des ressources qui peuvent être proposées à l'utilisateur, telles qu'une vidéo de démonstration, un forum ou une application comme la calculatrice.

Lors de l'exécution d'un bloc aLDEAS, lorsqu'un nœud de type action d'assistance est rencontré, l'action correspondante est déclenchée et l'exécution du bloc se poursuit immédiatement avec le prochain nœud sans attendre la fin de l'action déclenchée ; cela permet l'exécution simultanée de plusieurs actions d'assistance, par exemple un message et une mise en valeur. Les actions d'assistance peuvent prendre fin par elles-mêmes, par exemple un message lu par synthèse vocale prend fin lorsque tout le texte a été lu ; elles peuvent parfois également être stoppées par l'utilisateur, l'utilisateur peut par exemple mettre fin à un message affiché dans une fenêtre pop-up en la fermant. Le marqueur d'arrêt d'aLDEAS, représenté par le symbole  , permet également de mettre fin à toutes les actions élémentaires d'assistance déclenchées depuis le début du bloc. La figure 7 donne l'exemple d'un bloc composé de deux actions élémentaires d'assistance : un message et une mise en valeur. Ce bloc contient également un événement de fin, c'est-à-dire une attente d'événement suivie du marqueur d'arrêt. Dans cet exemple, le message et la mise en valeur s'effaceront au bout de 30 secondes.

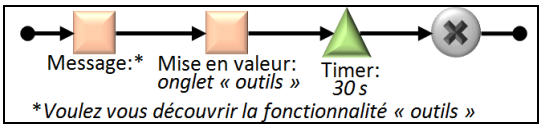

Figure 7. Exe*mple d'un bloc aLDEAS contenant un événement de fin*

5.4. *Actions d'assistance complexes*

Un bloc aLDEAS est représenté par le symbole  et peut être déclenché par un autre bloc, avec un fonctionnement identique à celui du déclenchement d'une action élémentaire d'assistance. Cependant, une action d'assistance complexe peut retourner une valeur, qui est indiquée le cas échéant sous le marqueur de fin de bloc (cf. exemples des figures 11 et 12). Si une action d'assistance complexe retourne une valeur, elle peut être suivie par une alternative à n branches, avec un fonctionnement identique à celui d'une consultation, décrit à la fin de la section 5.2.

6. Patrons aLDEAS

Afin de faciliter la définition en aLDEAS d'actions d'assistance complexes par le concepteur de l'assistance, nous avons proposé un ensemble de patrons. Pour cela, nous avons enrichi notre langage avec deux structures : l'embranchement « ou » et les éléments optionnels précédés d'un « ? ». Ces structures ne décrivent pas l'assistance telle qu'elle sera exécutée ; elles représentent un choix que le concepteur de l'assistance doit faire lorsqu'il instancie un patron lors de la phase de spécification de l'assistance.

6.1. P*atron de règles d'assistance*

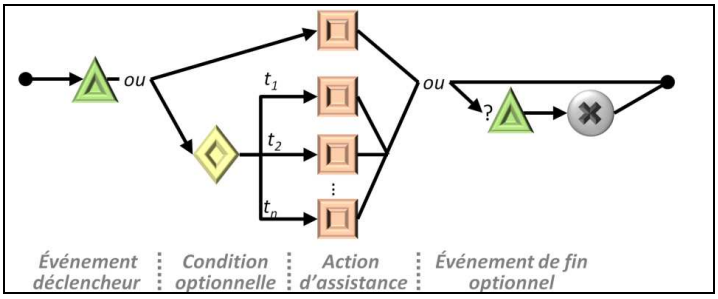

Figure 8. *Patron de règles d'assistance*

Le but d'aLDEAS est de permettre aux concepteurs d'assistance de spécifier l'assistance qu'ils souhaitent pour une application-cible, sous la forme d'un ensemble de règles qui seront exécutées dans un second temps pour fournir l'assistance à l'utilisateur final sur cette application. Une règle est définie comme une action d'assistance complexe qui respecte le patron donné en figure 8. Une règle d'assistance débute par un événement déclencheur qui correspond à une attente d'événement, qu'il soit élémentaire ou non. Une règle contient ensuite soit une action d'assistance, soit une condition de déclenchement qui correspond à une consultation suivie d'une alternative à n branches, chacune associée à une action d'assistance. Enfin, une règle peut se terminer par un événement de fin, c'est-à-dire une attente d'événement suivie du marqueur d'arrêt. La figure 9 donne l'exemple de deux règles d'assistance créées pour l'application-cible PhotoScape. La règle R0 contient un événement déclencheur (le lancement de l'assistance) et une consultation de l'utilisateur lui permettant de choisir s'il souhaite ou non recevoir de l'aide. Si l'utilisateur choisit l'option « Oui, je veux de l'aide », la règle R1 sera déclenchée. La règle R7 est déclenchée par un clic de l'utilisateur sur le composant 228 (le menu « outils » de PhotoScape). R7 déclenche une action d'assistance composées de deux actions élémentaires : une mise en valeur du composant 210 (le bouton « yeux rouges » de PhotoScape) et un message suggérant à l'utilisateur de cliquer sur le

bouton « yeux rouges ». Le clic de l'utilisateur sur ce bouton entraînera l'effacement de la mise en valeur et du message.

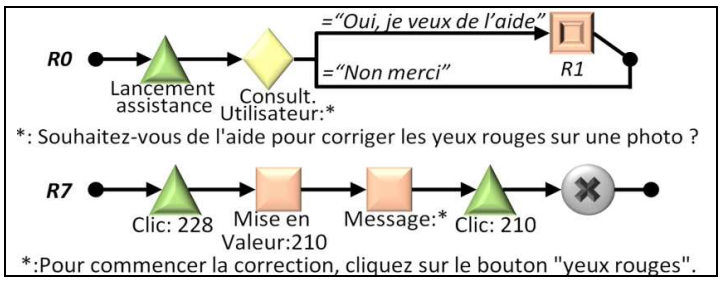

Figure 9. Exemple *des règles d'assistance* R0 et R7 pour PhotoScape

6.2. *Patrons d'actions d'assistance complexes*

Le langage aLDEAS permet la définition d'actions d'assistance complexes, combinant plusieurs actions d'assistance élémentaires. La définition de telles actions peut être difficile pour le concepteur de l'assistance. Pour cette raison, nous avons défini des patrons d'actions d'assistance complexes, qui facilitent la définition d'actions complexes fréquemment rencontrées dans les systèmes d'assistance existants : les pas-à-pas, les présentations guidées et les actions d'agents animés.

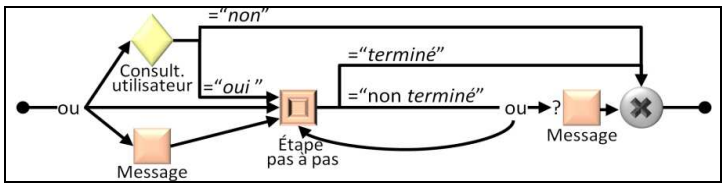

Figure 10. aLDEAS : patron de pas-à-pas

Un pas-à-pas vise à faciliter la réalisation d'une tâche donnée en la détaillant en plusieurs étapes. Chaque étape correspond à une action que l'utilisateur final doit réaliser sur un composant de l'interface de l'application-cible. Nous appelons « pas-à-pas automatisé » un pas-à-pas dans lequel le système d'assistance réalise automatiquement ces actions à la place de l'utilisateur final. Nous appelons « pas-à-pas guidé » un pas-à-pas dans lequel le système d'assistance attendra que l'utilisateur ait réalisé lui-même les actions requises. Les patrons d'étapes de pas-à-pas automatisé et guidé sont donnés respectivement en figures 11 et 12 : ces deux patrons sont exploités par le patron de pas-à-pas (cf. figure 10), dans lequel les étapes sont représentées par le symbole . L'instanciation du patron de pas-à-pas

est une manière pour le concepteur de l'assistance de représenter ses connaissances relatives à la réalisation d'une tâche donnée dans l'application-cible.

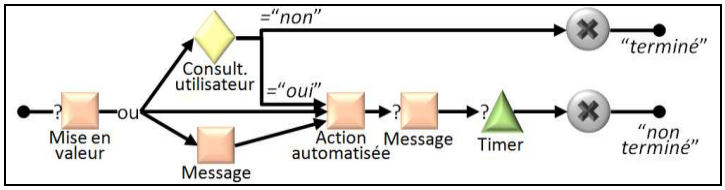

Figure 11. aLDEAS *: patron d'étape de pas*-à-pas automatisé

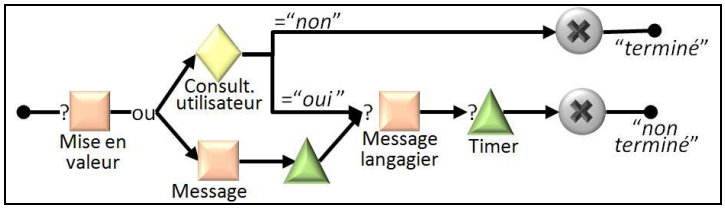

Figure 12. aLDEAS *: patron d'étape de pas*-à-pas guidé

Un pas-à-pas peut commencer par un message (qui peut par exemple indiquer à l'utilisateur que le pas-à-pas va commencer), ou par une consultation de l'utilisateur (afin de lui permettre de choisir s'il souhaite que le pas-à-pas ait lieu ou non). Dans le cas où le pas-à-pas commence par une consultation de l'utilisateur, nous observons que la branche correspondant à l'option « non » est directement reliée au marqueur de fin du bloc, ce qui signifie qu'aucune étape n'aura lieu. En effet, un pas-à-pas est principalement constitué d'une ou plusieurs étapes qui peuvent être soit automatisées soit guidées. Enfin, un pas-à-pas peut se terminer par un message.

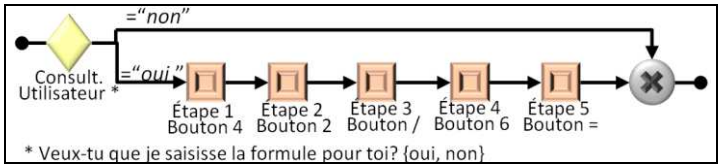

Figure 13. Exemple de pas-à-pas automatisé pour la calculatrice

Prenons comme exemple d'instanciation de pas-à-pas automatisé, une action d'assistance complexe qui concerne l'outil calculatrice et permet la saisie automatique de la formule « 42/6= ». Une telle action d'assistance peut par exemple être proposée dans le contexte d'une assistance pédagogique dans un logiciel de

mathématiques. En effet, si l'objectif pédagogique ne concerne pas directement le calcul, mais par exemple la résolution de problèmes, un enseignant peut souhaiter fournir à un enfant en difficulté une assistance qui lance la calculatrice et saisit pour lui une formule. Ainsi, l'enfant ne sera pas bloqué dans son activité pédagogique.

La définition en aLDEAS de cette action d'assistance est donnée en figure 13. Cette action débute par une consultation de l'utilisateur et contient cinq étapes automatisées, qui correspondent aux cinq boutons de la calculatrice sur lesquels il faut cliquer pour saisir la formule "42/6=". Chaque étape instancie le patron d'étape de pas-à-pas automatisé (cf. figure 11). À titre d'exemple, la définition en aLDEAS de l'étape concernant le bouton 4 est donnée en figure 14. Pendant cette étape, le bouton 4 de la calculatrice est mis en valeur et un clic est réalisé automatiquement par le système d'assistance. Cette étape dure 3 secondes, puis la mise en valeur est effacée et la prochaine étape commence.

Figure 14. Exemple *d'une étape de pas*-à-pas automatise pour la calculatrice

Une présentation guidée est constituée de plusieurs étapes, dans lesquelles un composant est mis en valeur et éventuellement présenté par un message. On retrouve fréquemment ce type d'actions d'assistance dans les applications existantes, notamment quand une application est lancée pour la première fois, ou après une mise à jour. L'instanciation du patron de présentation guidée est une manière pour le concepteur de l'assistance de représenter ses connaissances relatives aux fonctionnalités proposées par l'application-cible.

Une présentation guidée en aLDEAS (cf. figure 15) peut débuter par un message. Elle contient une ou plusieurs étapes qui instancient le patron d'étape de présentation guidée (cf. figure 16). Une présentation guidée peut se terminer par un message. Chaque étape de présentation guidée concerne un composant de l'interface de l'application-cible qui est présenté. Une étape peut contenir un message spécifique au composant et contient également une mise en valeur de ce composant afin d'attirer l'attention de l'utilisateur pendant un temps donné.

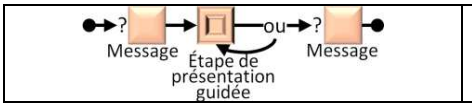

Figure 15. Patron de présentation guidée

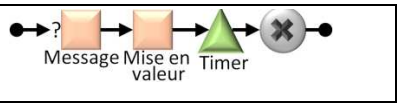

Figure 16. *Patron d'étape* de présentation guidée

Prenons l'exemple d'une présentation guidée existante dans le jeu « Dora l'exploratrice : le spectacle de fin d'année[3] », destiné aux jeunes enfants. Ce jeu consiste à créer un spectacle qui sera réalisé ensuite par un ou deux personnages, dans un décor donné. Au lancement du jeu, ou après une période d'inaction de 5 secondes, une présentation guidée est déclenchée. Une flèche bleue met en valeur un à un les principaux composants de l'interface pendant qu'une synthèse vocale explique leurs fonctions. Cette action d'assistance peut être représentée en aLDEAS par une présentation guidée (cf. figure 16). Cette action d'assistance débute par un message qui sera lu par une synthèse vocale pendant toute la présentation guidée, qui contient 26 étapes (cf. figure 17) : 8 mouvements de danse disponibles (4 boutons de chaque côté de l'écran), 10 cases correspondant aux mouvements de danse à choisir par l'utilisateur final (au sommet de l'écran), et 7 décors possibles (en bas de l'écran).

Figure 17. Présentation guidée existante dans le jeu *"Dora l'exploratrice* : *le spectacle de fin d'année"*

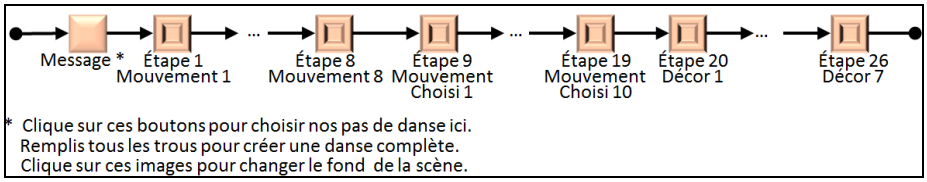

Figure 18. Exemple de définition *en aLDEAS d'une* présentation guidée pour le jeu *"Dora l'exploratrice* : le *spectacle de fin d'année"*

À titre d'exemple, la définition en aLDEAS de la première étape de cette présentation guidée est donnée en figure 19. Cette étape concerne le bouton qui

3. http://www.tfou.fr/dora-l-exploratrice/jeux/le-spectacle-de-fin-d-annee-7817399-739.html

permet de choisir le premier mouvement de danse possible, ce qui correspond à la copie d'écran de la figure 17. Ce bouton est mis en valeur par une flèche bleue pendant 2 secondes.

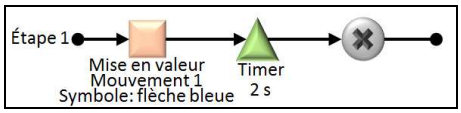

Figure 19. Ex*emple d'une étape de* présentation guidée pour le jeu *"Dora l'exploratrice : le spectacle de fin d'année"*

Une action d'agent animé permet la combinaison de plusieurs actions élémentaires d'assistance à réaliser par un même agent animé : messages, animations (montrer un composant, applaudir, saluer…), et déplacements à l'écran. Une action d'agent animé en aLDEAS (cf. figure 20) concerne un personnage et consiste en une ou plusieurs étapes. Chaque étape (cf. figure 21) contient une action élémentaire d'assistance de type message, mise en valeur, animation ou déplacement. Une étape peut contenir un événement de fin, tel qu'un timer à durée fixe ou l'attente d'une action de l'utilisateur.

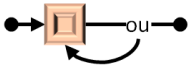

Figure 20. aLDEAS : *patron d'action d'agent* animé

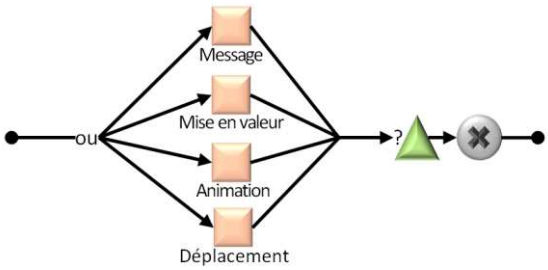

Figure 21. aLDEAS : *patron d'étape d'action d'agent animé*

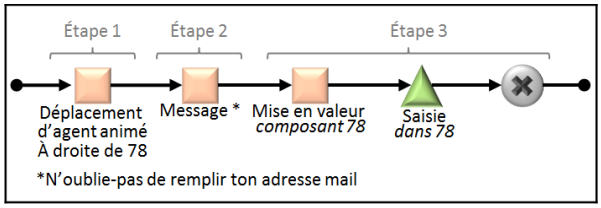

Figure 22. Exemple *d'action d'agent animé pour un formulaire d'inscription*

Prenons l'exemple d'une assistance qui est fréquemment proposée dans les formulaires d'inscription de sites web. Cette assistance permet de vérifier que l'utilisateur a complété les champs de saisie du formulaire avant de la valider, notamment le champ correspondant à l'adresse mail de l'utilisateur. Dans certaines applications web, spécialement celles s'adressant à des enfants, cette assistance peut prendre la forme d'une action d'agent animé. Ainsi, lorsque l'utilisateur valide le formulaire sans avoir complété son adresse mail, un agent animé se place à côté de ce champ de saisie et affiche ou lit un message tel que « n'oublie pas de compléter ton adresse mail », tout en désignant le champ jusqu'à ce que l'utilisateur l'ait rempli. La définition en aLDEAS d'une telle action est donnée en figure 22. Elle consiste en 3 étapes : un déplacement, un message et une mise en valeur.

7. Implémentation dans sepia

Nous avons implémenté aLDEAS et les patrons qui le complètent dans le système SEPIA (Specification and Execution of Personalized Intelligent Assistance). SEPIA est principalement composé de deux outils (cf. figure 23) : l'éditeur d'assistance concerne la phase de spécification de l'assistance et le moteur générique d'assistance concerne la phase d'exécution de l'assistance dans l'application-cible. L'éditeur d'assistance de SEPIA (cf. Ⓐ figure 23) est un outil destiné aux concepteurs d'assistance. Il met en œuvre aLDEAS et permet, pour une application-cible donnée, la spécification de l'assistance sous la forme d'un ensemble de règles aLDEAS (cf. figure 8). Aucune connaissance préalable d'aLDEAS n'est requise pour utiliser l'éditeur d'assistance.

Pour définir un système d'assistance avec l'éditeur de SEPIA, la première étape consiste à indiquer l'application-cible concernée. Si cela n'a pas déjà été fait, le concepteur de l'assistance doit lancer la génération par SEPIA d'une description de l'interface de cette application-cible. Ce fichier de description contient des connaissances sur l'application-cible et attribue notamment un identifiant à chacun de ses composants. Ces connaissances sont obtenues automatiquement par SEPIA grâce à des bibliothèques d'accessibilité initialement conçues pour permettre l'utilisation d'outils tels que les lecteurs d'écran et les barrettes Braille, afin de rendre accessibles des applications informatiques aux personnes en situation de handicap. Elles fournissent un accès à la hiérarchie des composants d'une application. Pour chaque composant, ces bibliothèques fournissent également un accès aux propriétés du composant, par exemple pour savoir si un champ de saisie doit contenir une valeur de type textuel ou numérique, avec une valeur minimale et maximale le cas échéant. De telles connaissances peuvent être utiles pour fournir une assistance contextualisée. L'étape de description de l'application-cible est principalement automatisée et dure quelques secondes. L'épi-descripteur (Ginon et al., 2013) (cf. Ⓑ figure 23) approprié est appelé par l'éditeur de SEPIA pour générer le fichier de description. Le choix de l'épi-descripteur est fait en fonction du type de l'application-cible (application Windows native, application Java, application web…), renseigné par le concepteur de l'assistance. En effet, nous avons

développé plusieurs épi-descripteurs, chacun exploitant une bibliothèque d'accessibilité, car chaque type d'application est ouvert à une bibliothèque d'accessibilité donné. Ainsi, UIAutomation4 (Haverty, 2005) concerne les applications natives Windows ; JavaAccessibility5 (Harper et al., 2005) concerne les applications Java, ATK/T-SPI6 concerne les applications GTK et Qt ; et Apple API Accessibility7 (Malacria et al., 2013) concerne les applications natives Mac OS.

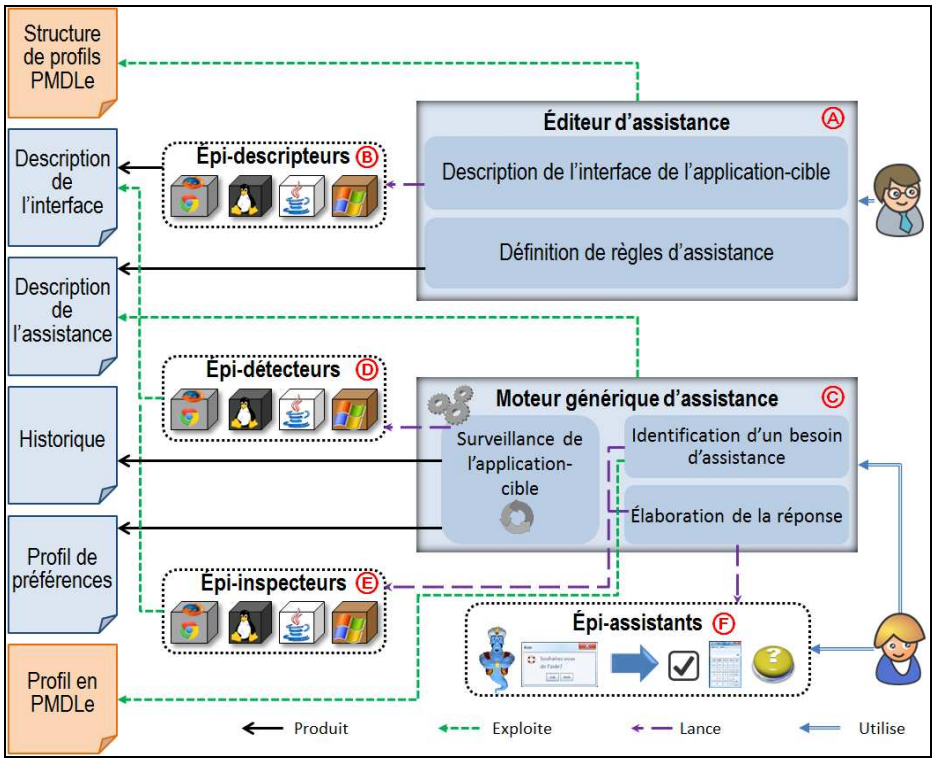

Figure 23. Architecture de SEPIA

La principale étape de la phase de spécification de l'assistance consiste en la définition des règles d'assistance par le concepteur de l'assistance. Pour cela, l'éditeur de SEPIA fournit au concepteur de l'assistance une interface pour la définition de toutes les actions élémentaires d'assistance présentées en section 5.3 (sauf les actions grisées sur les figures 5 et 6), et pour la définition des actions

4. http://msdn.microsoft.com/fr-fr/library/ms747327%28v=vs.110%29.aspx
5. http://docs.oracle.com/javase/8/docs/technotes/guides/access/ index.html
6. http://www.linuxfoundation.org/collaborate/ workgroups/accessibility/atk/at-spi
7. https://developer.apple.com/library/mac/documentation/Accessibility/Conceptual/AccessibilityMacOSX/OSXAXIntro/OSXAXintro.html

d'assistance complexes qui instancient l'un des patrons présentés en section 6. Si le concepteur de l'assistance souhaite utiliser des consultations du profil de l'utilisateur afin de personnaliser l'assistance en fonction des spécificités de l'utilisateur, il doit indiquer la structure de profils PMDLe qu'il souhaite utiliser. PMDLe (Profile MoDeling Language - Evolutive) est un modèle qui permet la définition de structures de profils très variées (Ginon et al., 2011). Un profil d'utilisateur peut ainsi rassembler les informations que le concepteur d'assistance juge pertinentes pour personnaliser l'assistance. Par exemple, un profil d'utilisateur peut contenir les informations relatives aux préférences de l'utilisateur vis-à-vis de l'assistance, ces connaissances et compétences en lien avec l'utilisation de l'application-cible, ses éventuels handicaps et son niveau d'expérience de l'application-cible.

Enfin, l'éditeur de SEPIA génère le fichier de description de l'assistance qui contient les règles définies par le concepteur, ainsi que certaines métadonnées (le fichier de description de l'interface de l'application-cible utilisé, le nom du concepteur de l'assistance, la date de création…).

Le moteur générique de SEPIA (cf. © figure 23) permet quant à lui d'exécuter automatiquement un système d'assistance préalablement spécifié par un ensemble de règles aLDEAS. Pour exécuter les actions d'assistance élémentaires, le moteur fait appel à l'un des épi-assistants. Le moteur assure également l'exécution des actions d'assistance complexes qui instancient l'un des patrons aLDEAS, en faisant appel si besoin à des épi-assistants. La figure 24 montre à titre d'exemple l'exécution de deux actions d'assistance élémentaires : la mise en valeur d'un bouton par une flèche et un message affiché dans une fenêtre pop-up. Elles sont déclenchées dans l'application-cible PhotoScape par la règle R7 (cf. figure 9 en section 6.1)

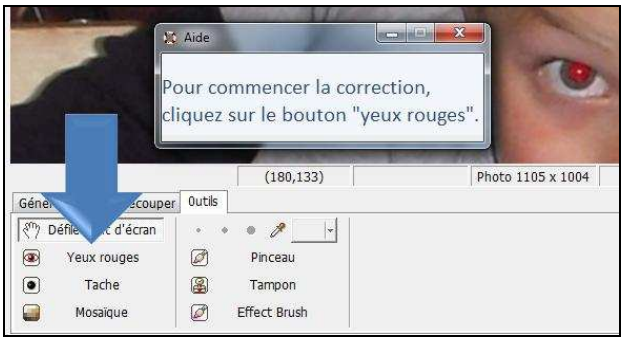

Figure 24. Exemple *de deux actions d'assistance élémentaires* déclenchées dans PhotoScape

Afin d'être informé de ce qui se passe dans l'application-cible et ainsi de pouvoir déclencher l'assistance définie par le concepteur d'assistance, le moteur de SEPIA exploite l'un de nos épi-détecteurs (Ginon et al., 2013) (cf. ©figure 23). Tout

comme les épi-descripteurs qui génèrent le fichier de description de l'application-cible, les épi-détecteurs s'appuient sur une bibliothèque d'accessibilité. Le choix de l'épi-détecteur est donc fait en fonction du type de l'application-cible. Les épi-détecteurs surveillent en continu l'application-cible et informent le moteur de chaque événement qui survient à l'interface (comme un clic, la saisie d'un texte, la sélection d'un item dans une liste déroulante…). Grâce au fichier de description de l'application-cible, les épi-détecteurs peuvent identifier précisément la source d'un événement, par exemple le bouton à l'origine d'un clic.

Les épi-assistants que nous avons développés pour exécuter les actions d'assistance élémentaires d'aLDEAS sont présentés en section 5.3. Les actions automatisées, les mises en valeur et les masquages sont disponibles pour les applications Windows natives et Java, pour les applications web (hors Flash), pour les applications Linux GTK et Qt, ainsi que pour les applications Mac OS natives. Les actions d'agents animés sont disponibles sous Windows ; les messages et le lancement de ressources (telles que des pages web ou des vidéos de démonstration) sont disponibles pour toutes les applications-cibles.

Tableau 1. Actions d'assistance élémentaires aLDEAS disponibles selon le type de l'application-cible

		Action automatisée	Mise en valeur masquage	Message	Agent animé	Lancement de ressource
Windows	Native	✓	✓	✓	✓	✓
	Java	✓	✓	✓	✓	✓
	GTK / Qt			✓	✓	✓
	Web	✓	✓	✓	✓	✓
Linux	Java			✓		✓
	GTK / Qt	✓	✓	✓		✓
	Web	✓	✓	✓		✓
Mac OS	Native	✓	✓	✓		✓
	Java			✓		✓
	GTK / Qt			✓		✓
	Web	✓	✓	✓		✓

8. Évaluations

Les propositions présentées dans cet article ont été évaluées de plusieurs manières. Concernant la faisabilité de l'approche de mise en place a posteriori d'assistance épiphyte dans des applications existantes, elle est démontrée par son implémentation pleinement opérationnelle dans le système SEPIA. Dans cette section, nous présentons les études menées afin d'évaluer la couverture du langage aLDEAS, l'utilisabilité d'aLDEAS et de SEPIA, ainsi que l'utilité et l'acceptation de l'assistance fournie aux utilisateurs finaux.

8.1. *Couverture d'*aLDEAS

Afin d'évaluer la couverture du langage que nous proposons, nous l'avons utilisé avec succès pour représenter des systèmes d'assistance existants variés, représentatifs des assistances les plus fréquentes dans les applications existantes. Ainsi, aLDEAS permet de représenter les systèmes d'assistance fréquemment utilisés dans les applications web, par exemple l'aide à la saisie et à la vérification de formulaires. Les présentations guidées d'une application nouvelle ou récemment mise à jour peuvent également être représentées à l'aide d'aLDEAS. Par ailleurs, il existe de nombreux tutoriels qui permettent de guider un utilisateur dans la réalisation d'une tâche donnée, grâce à des copies d'écrans annotées et des messages. Notre langage permet la représentation de tels systèmes d'assistance, en présentant l'avantage de paraître plus intégrés à l'application-cible aux yeux de l'utilisateur final. En effet, les copies d'écrans annotées peuvent être remplacées par des mises en valeur directement exécutées dans l'application-cible. Cependant, certains systèmes d'assistance existants ne peuvent pas être représentés en aLDEAS : c'est le cas de certains systèmes d'assistance très spécifiques à une application et qui requièrent des informations non disponibles depuis l'extérieur de l'application. Par exemple, certains systèmes de recommandation intégrés à des applications de vente en ligne exploitent des informations relatives aux achats effectués par tous les utilisateurs du site. aLDEAS ne permet pas la consultation de telles informations puisque celles-ci ne sont pas accessibles directement à l'utilisateur final du site web.

Nous avons également évalué le langage aLDEAS en le comparant aux autres approches de spécification d'assistance a posteriori. Concernant l'approche de (Richard et Tchounikine, 2004), aLDEAS permet la représentation de tels systèmes conseillers, puisqu'il permet la consultation de l'historique de la navigation (inclus dans les traces de l'utilisateur), l'attente d'événements de type clic sur un lien donné, et les actions d'assistance de type message (contenant éventuellement des liens vers d'autres pages web). aLDEAS permet par ailleurs de lancer directement une page web ou toute autre ressource sans avoir à passer par un lien contenu dans un message. Concernant l'approche proposée par (Carlier et Renault, 2010), aLDEAS permet la proposition d'actions d'assistance impliquant des agents animés capables de se déplacer, de s'exprimer par des animations, des gestes et par des messages textuels ou oraux. La proposition du patron d'action d'agents animés facilite la définition de telles actions pour le concepteur de l'assistance. Enfin, notre langage permet la définition de règles d'assistance de la forme <événement déclencheur, condition de déclenchement optionnelle, action d'assistance, événement de fin optionnel> (cf. section 6.1) équivalentes aux règles d'assistance utilisées dans les approches de Paquette et al. (2007) et de Dufresne et Paquette (2000). De plus, aLDEAS permet la définition d'actions d'assistance plus complexes et variées que celles proposées dans ces différentes approches.

8.2. Utilisabilité *d'aLDEAS et de* SEPIA

Dans un premier temps, nous avons utilisé aLDEAS et SEPIA pour spécifier et exécuter des systèmes d'assistance dans de nombreuses applications variées[8]. Dans un deuxième temps, nous avons réalisé deux expérimentations présentées dans cette section. Elles visent à évaluer l'utilisabilité du langage aLDEAS et du système SEPIA par des concepteurs d'assistance, qu'ils soient ou non informaticiens.

8.2.1. U*tilisation d'*aLDEAS et de SEPIA par des informaticiens

Dans le cadre d'un cours de master en informatique à l'Université de Lyon, les étudiants doivent développer en binômes des logiciels pédagogiques simples. Afin de démontrer qu'aLDEAS permet de faciliter la spécification d'un système d'assistance, nous avons demandé à 29 étudiants de spécifier un système d'assistance pour leur logiciel pédagogique. Dans un premier temps, chaque binôme était séparé dans deux salles, pendant 90 minutes. Dans la première salle, 14 étudiants jouant le rôle de concepteurs d'assistance ont défini « sur papier » un système d'assistance pour leur logiciel pédagogique, sans utiliser de formalisme particulier. Dans la seconde salle, 15 autres concepteurs d'assistance ont défini un système d'assistance pour leur logiciel pédagogique, en utilisant « sur papier » aLDEAS. Ils avaient pour cela reçu pendant 5 minutes des explications sur aLDEAS, en particulier sur le patron de règles d'assistance. Chaque binôme a ensuite été réuni pendant 90 minutes afin de mettre en commun ses idées et de définir en aLDEAS un système d'assistance commun. Les concepteurs d'assistance ont ensuite utilisé SEPIA durant une autre séance de travail de 3 heures, afin de spécifier et tester leur système d'assistance sur leur logiciel pédagogique.

Chaque binôme a réussi à définir en aLDEAS le système d'assistance qu'il désirait, et à le mettre en place dans son logiciel pédagogique avec SEPIA. Un questionnaire de satisfaction a montré que 93 % des concepteurs d'assistance impliqué dans cette étude ont trouvé aLDEAS facile ou très facile à utiliser, et 76 % pensent qu'aLDEAS les a aidés à définir leur système d'assistance (cf. figure 25). 63 % d'entre eux ont également pensé que l'utilisation préalable d'aLDEAS « sur papier » facilitait ensuite l'utilisation de SEPIA et 67 % ont trouvé SEPIA facile ou très facile pour définir les règles d'assistance (cf. figure 26). Plusieurs participants ont par ailleurs noté dans la partie commentaires libres du questionnaire de satisfaction qu'aLDEAS les avait aidés à travailler à deux et à formaliser leurs idées.

8.2.2. *Utilisation d'aLDEAS et de SEPIA par des non*-informaticiens

En complément de l'expérimentation présentée dans la section précédente, nous avons demandé à 5 non-informaticiens d'utiliser aLDEAS afin de représenter un système d'assistance pour la correction des yeux rouges sur une photo avec PhotoScape. Ceci correspond au scénario d'usage présenté en section 2. Ces 5 concepteurs d'assistance sans connaissance en programmation avaient entre 11 et

8. Liste des applications disponible à http://liris.cnrs.fr/blandine.ginon/detection.html

68 ans. Les systèmes d'assistance qu'ils ont créés sont équivalents à celui expérimenté auprès de 200 utilisateurs finaux de PhotoScape (cf. section 8.3.1). Ils avaient préalablement reçu pendant 5 minutes des explications sur aLDEAS, en particulier sur le patron de règles d'assistance, et ils avaient à leur disposition quelques exemples de règles d'assistance instanciant ce patron.

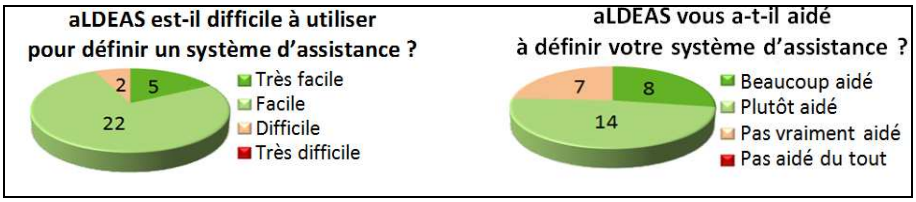

Figure 25. Résultats des questions relatives
à l'utilisation d'aLDEAS « sur papier »

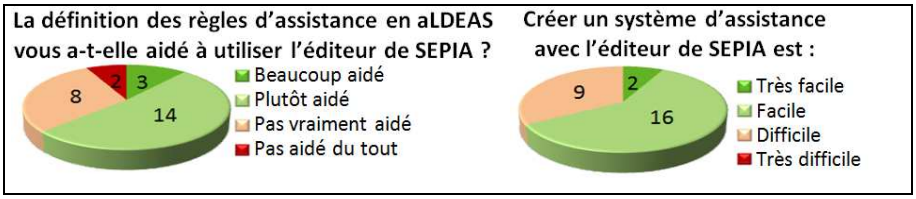

Figure 26. Résultats des questions relatives à la spécification
de l'assistance avec SEPIA

Dans un premier temps, chaque participant a utilisé avec succès aLDEAS pour définir « sur papier » entre 7 et 9 règles d'assistance. Ce travail a duré entre 30 et 45 minutes. Dans un deuxième temps, ils ont utilisé le système SEPIA pour spécifier et tester leur système d'assistance pour PhotoScape, après avoir reçu quelques explications sur le fonctionnement de l'éditeur d'assistance de SEPIA. Ce travail a duré entre 55 et 95 minutes.

4 des 5 participants ont trouvé aLDEAS facile ou très facile à utiliser, et tous ont déclaré qu'aLDEAS les avait aidés à définir leur système d'assistance. De plus, ils ont trouvé que l'utilisation préalable d'aLDEAS « sur papier » facilitait l'utilisation de l'éditeur d'assistance de SEPIA. Ces résultats encourageants semblent montrer que le langage aLDEAS et son implémentation dans le système SEPIA peuvent être utilisés par des concepteurs d'assistance sans connaissance en programmation. Ces premiers résultats doivent cependant être confirmés par une expérimentation plus vaste avec des concepteurs d'assistance non-informaticiens, et avec des applications-cibles variées.

8.3. Utilité *et acceptation de l'assistance*

Nous avons réalisé deux expérimentations afin d'évaluer la phase d'exécution de l'assistance. Tout d'abord, ces expérimentations visaient à montrer que le langage aLDEAS et son implémentation dans le système SEPIA peuvent être utilisés pour fournir aux utilisateurs d'une application-cible une assistance pertinente et utile. Ensuite, elles visaient à montrer qu'une assistance ainsi définie peut être très bien acceptée par les utilisateurs. Pour cela, nous avons défini en aLDEAS deux systèmes d'assistance correspondant aux scénarios d'usage présentés en section 2, que nous avons ensuite exécutés avec SEPIA pour des utilisateurs finaux de PhotoScape et de NetBeans. Le premier système d'assistance était conçu pour un usage personnel de PhotoScape, et le second pour un usage de NetBeans en contexte éducatif.

8.3.1. Correction des yeux rouges avec PhotoScape

Nous avons demandé à 200 personnes de corriger les yeux rouges avec PhotoScape sur une photo donnée, sans aucune assistance pour le groupe A de 100 personnes, et avec notre système d'assistance pour le groupe B de 100 personnes également. L'expérimentation était précédée d'un questionnaire permettant de connaître certaines informations sur les participants : âge, genre, connaissance de PhotoScape ou d'outils similaires, etc. L'expérimentation était également suivie par un questionnaire de satisfaction dont les principaux résultats sont donnés en figures 28 et 29. Pour le groupe A, on distingue 2 sous-groupes : A1 pour les 49 utilisateurs qui ont réussi à corriger les yeux rouges sans assistance, et A2 pour les 51 utilisateurs qui ont échoué à réaliser cette tâche sans assistance et à qui nous avons demandé d'essayer à nouveau avec notre système d'assistance.

Tableau 2. *Extrait des résultats relatifs à l'expérimentation avec PhotoScape*

	Effectif	**Sans assistance**		**Avec assistance**	
		Taux de succès	Durée moyenne	Taux de succès	Durée moyenne
Groupe A	100	49%	-	-	-
A1	49	100%	146,1 s	-	-
A2	51	0%	-	100%	72,15 s
Groupe B	100	-	-	100%	65,33 s

On note que l'assistance proposée réduit de moitié en moyenne le temps de réalisation de la tâche. Dans le groupe A1, constitué des utilisateurs qui ont réussi à réaliser la tâche demandée, la moitié aurait malgré tout apprécié de recevoir de l'aide, ce qui montre l'utilité de l'assistance dans ce contexte. Dans le groupe A2, constitué des utilisateurs qui n'avaient pas réussi à réaliser la tâche sans assistance, tous ont ensuite réussi avec l'assistance proposée et tous l'ont jugée utile, ce qui montre l'efficacité de l'assistance dans ce contexte. Dans le groupe B, 97 % des utilisateurs ont jugé l'assistance utile. De plus, cette assistance a été très appréciée

par les participants : 92 % des utilisateurs du groupe A2 et 87 % des utilisateurs du groupe B l'ont appréciée. Ces résultats très positifs montrent la pertinence de l'assistance proposée dans ce contexte, et sa très bonne acceptation par les utilisateurs finaux.

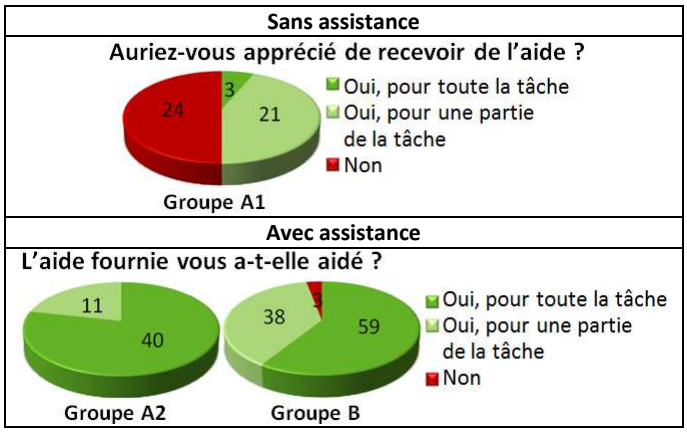

Figure 27. Extrait des résultats du questionnaire de satisfaction *consécutif à l'expérimentation avec* PhotoScape

8.3.2. Développement *d'applications graphiques avec* NetBeans

Cette seconde expérimentation avec des utilisateurs finaux se situe dans un contexte éducatif (Ginon et al., 2014). Nous avons travaillé avec des enseignants d'un cours de licence sur les IHM (Interfaces Homme-Machine) à l'Université de Lyon. Les enseignants ont défini avec SEPIA un tutoriel pour NetBeans, afin d'aider les étudiants à découvrir et prendre en main NetBeans, ainsi qu'à pratiquer des exercices simples de programmation Java.

Dans un premier temps, quatre concepteurs d'assistance ont utilisé l'éditeur d'assistance de SEPIA pour spécifier un système d'assistance pour NetBeans[9], composé de 5 parties indépendantes. Dans un deuxième temps, ce système d'assistance a été utilisé par 52 étudiants volontaires durant le premier TP du semestre. L'utilisation du tutoriel a duré entre 30 et 90 minutes selon les étudiants. Cette expérimentation était précédée par un pré-test et suivie par un post-test, afin de mesurer la progression des connaissances des étudiants relative à l'utilisation de NetBeans et à la programmation Java. L'expérimentation était également suivie par un questionnaire de satisfaction.

Les étudiants ont été très satisfaits de ce tutoriel : 90,4 % d'entre eux ont déclaré l'avoir apprécié. Le tutoriel semble avoir facilité l'acquisition des connaissances et

9. Vidéo de démonstration disponible à http://liris.cnrs.fr/blandine.ginon/PhDWork.html

compétences visées par le cours d'IHM : 96,2 % des étudiants ont déclaré que le tutoriel leur avait appris quelque chose. Cet excellent résultat est confirmé par la progression moyenne des étudiants ayant suivi le tutoriel (cf. tableau 3) : la comparaison entre le pré-test et le post-test montre une progression moyenne de 56 %, avec une progression minimale de 10 % et maximale de 100 % pour les étudiants n'ayant eu aucune réponse correcte au pré-test et uniquement des réponses correctes au post-test. Pour les 45 étudiants du groupe témoin qui ont réalisé le même TP et rempli les mêmes pré-test et post-test mais sans avoir suivi le tutoriel, la progression moyenne était seulement de 8 %.

Tableau 3. Extrait des résultats de l'expérimentation avec NetBeans

	Groupe témoin	**Groupe ayant suivi le tutoriel**
Taux de succès moyen au **pré-test**	15%	11%
Taux de succès moyen au **post-test**	23%	64%
Progression moyenne	+8%	+56%
Progression minimale	-25%	+10%
Progression maximale	+50%	+100%

Ces résultats très positifs semblent montrer que le tutoriel a facilité l'apprentissage des étudiants. De plus, les enseignants ont été particulièrement satisfaits par ce tutoriel et projettent de l'utiliser à nouveau pour les prochains semestres.

9. Conclusion et perspectives

Le langage aLDEAS et les patrons d'actions d'assistance complexes que nous avons présentés dans cet article permettent la définition de systèmes d'assistance adaptés à des applications-cibles très variées. En adoptant une approche épiphyte, nous avons rendu possible l'adjonction a posteriori de systèmes d'assistance dans des applications existantes sans avoir à les modifier. Ces applications n'ont pas à avoir été conçues spécifiquement pour permettre l'ajout d'assistance, et aucun accès à leur code source n'est nécessaire. De plus, cette approche ne requiert pas de connaissance en programmation et est accessible à des concepteurs d'assistance non-informaticiens.

aLDEAS et les patrons qui le complètent permettent la représentation des connaissances du concepteur de l'assistance relatives à l'application-cible et à l'assistance souhaitée. Les expérimentations que nous avons réalisées montrent qu'aLDEAS peut être utilisé « sur papier » pour définir des systèmes d'assistance, par des concepteurs d'assistance informaticiens ou non. Le langage aLDEAS est mis en œuvre de façon opérationnelle dans le système SEPIA. SEPIA exploite d'une part des connaissances représentées en aLDEAS par le concepteur de l'assistance et

d'autre part, des connaissances sur l'application-cible automatiquement découvertes par les épi-descripteurs de SEPIA et éventuellement complétées par le concepteur de l'assistance. Les expérimentations que nous avons réalisées montrent que SEPIA peut être utilisé par des concepteurs d'assistance, avec ou sans connaissance en programmation. De plus, SEPIA peut fournir aux utilisateurs finaux de l'application-cible une assistance efficace pour répondre à leurs besoins et les aider à réaliser une tâche donnée, en particulier dans un contexte de découverte ou d'utilisation occasionnelle de l'application-cible.

L'utilisation de SEPIA ne requiert aucune connaissance préalable d'aLDEAS, ni de connaissance en programmation. Cependant, elle demande des connaissances sur le fonctionnement de l'application-cible. De plus, spécifier un système d'assistance pertinent et efficace nécessite une analyse préalable des besoins des utilisateurs finaux par des experts de l'application-cible, des ergonomes et le concepteur de l'assistance, ce qui est une tâche complexe. Pour cette raison, nous travaillons maintenant sur une méthode visant à faciliter la tâche du concepteur de l'assistance en l'aidant à identifier les besoins d'assistance des utilisateurs, notamment grâce à l'exploitation de leurs traces d'utilisation. Cette méthode permettra également à un concepteur d'assistance d'améliorer son système d'assistance par l'exploitation des traces d'utilisation du système d'assistance et en permettant aux utilisateurs finaux de lui fournir des feedbacks relatifs à l'assistance fournie et à celle souhaitée.

Bibliographie

Dufresne A., Paquette G. (2000). ExploraGraph: a flexible and adaptive interface to support distance learning, Ed-Media, Victoria, Canada, p. 304-309.

Carlier F., Renault F. (2010). Educational webportals augmented by mobile devices with iFrimousse architecture, ICALT, Sousse, Tunisia, p. 236-240.

Cordier A., Lefevre M., Jean-Daubias S., Guin, N. (2010). Concevoir des assistants intelligents pour des applications fortement orientées connaissances : problématiques, enjeux et étude de cas. IC, Nîmes, France, p 119-130.

Gapenne O., Lenay C., Boullier D. (2002). Defining categories of the human/technology coupling: theoretical and methodological issues, ERCIM Workshop on User Interface for All, Paris, France, p. 197-198.

Ginon B., Champin P.-A., Jean-Daubias S. (2013). Collecting fine-grained use traces in any application without modifying it, Worshop EXPPORT of ICCBR, New-York, USA.

Ginon B., B., Thai L. V., Jean-Daubias S., Lefevre M., Champin P.-A. (2014). Adding epiphytic assistance systems in learning applications using the SEPIA system, Ec-Tel, Graz, Austria, p. 138-151.

Ginon B., Jean-Daubias S., Lefevre M. (2011). Evolutive learners profile, ED-Media, Lisbon, Portugal.

Harper S., Khan G., Stevens R. (2005). Design Checks for Java Accessibility, Accessible Design in the Digital World, Dundee, Scotland.

Haverty R. (2005). New accessibility model for Microsoft Windows and cross platform development, ACM SIGACCESS Accessibility and Computing, p 11-17.

Malacria S., Bailly G., Harrison J., Cockburn A. et Gutwin C. (2013). Promoting Hotkey Use through Rehearsal with ExposeHK, ACM CHI, Paris, France, p. 573-582.

Paquette G., Pachet F., Giroux S., Girard J. (1996). EpiTalk, a generic tool for the development of advisor systems, IJAIED, p. 349-370.

Paquette G., Rosca I., Mihaila S., Masmoudi A. (2007). TELOS: A Service-Oriented Framework to Support Learning and Knowledge Management, E-Learning Networked Environments and Architectures, Pierre, S. (Ed.), p. 79-109.

Richard B., Tchounikine P. (2004). Enhancing the adaptivity of an existing Website with an epiphyte recommender system, New review of hypermedia and multimedia, vol. 10, p. 31-52.