Modèles et outils génériques pour mettre en place des systèmes d'assistance épiphytes

Blandine Ginon

Université de Lyon
Institut National des Sciences Appliquées de Lyon
Laboratoire d'InfoRmatique en Image et Systèmes d'information

Modèles et outils génériques pour mettre en place des systèmes d'assistance épiphytes

Thèse pour l'obtention du titre de docteur de l'INSA de LYON
Spécialité informatique
Présentée par

Blandine GINON

Soutenance prévue le 15 septembre 2014 devant le jury composé de :

Rapporteurs :

Gilbert Paquette	Professeur titulaire, Université TELUQ, Canada
Serge Garlatti	Professeur, Université Telecom Bretagne, France

Examinateurs :

Monique Grandbastien	Professeure émérite, Université de Lorraine, France
Sylvie Després	Professeure, Université Paris 13, France

Directeurs :

Stéphanie Jean-Daubias	Professeure, Université Lyon 1, France
Sylvie Calabretto	Professeure, INSA de Lyon, France
Pierre-Antoine Champin	Maître de conférences, Université Lyon 1, France

Remerciements

Je tiens tout d'abord à remercier les membres de mon jury : les rapporteurs, Gilbert Paquette et Serge Garlatti, les examinatrices, Monique Gandbastien et Sylvie Després, ainsi que Sylvie Calabretto, ma directrice de thèse côté INSA. Merci également à Pierre-Antoine qui a activement encadré ce travail et m'a fait bénéficier de sa grande expérience et de ses conseils. Merci d'avoir été si patient et disponible malgré tes nombreuses responsabilités. Un seul merci ne suffirait pas pour exprimer toute ma gratitude à l'égard de Stéphanie, qui a fait bien plus que diriger ma thèse. Tu m'as donné l'occasion de faire mes preuves en informatique alors que j'étais une jeune étudiante en licence de mathématiques, puis tu m'as guidée et soutenue tout au long de mon parcours : c'est toi qui as rendu possible cette thèse ! Merci pour ta confiance durant toutes ces années, j'espère avoir acquis un peu de ta rigueur scientifique afin de devenir un jour une enseignante et une chercheuse à la hauteur de son mentor.

Je souhaite également remercier chaleureusement tous les membres du projet Agate, à commencer par Marie, qui s'est énormément investie dans l'encadrement de ma thèse, et qui m'a beaucoup aidée, en particulier pour la préparation des présentations orales et pour la rédaction du manuscrit de thèse, qu'elle a accepté de relire avec la patience et la rigueur qui sont les siennes. Un grand merci à Vinh, pour sa constante bonne humeur et ses grandes qualités techniques : je te souhaite bonne chance pour la poursuite de ta thèse ! Enfin, merci aux stagiaires qui ont apporté leur pierre à l'édifice du projet Agate, en particulier Brice et Léa pour leurs contributions précieuses : ça a été un plaisir de travailler avec vous. Un très grand merci également à Amélie pour ses précieux conseils et pour son soutien. Merci également à toute l'équipe Silex, c'est une chance pour moi d'avoir fait partie d'une grande équipe accueillante, qui a permis des échanges scientifiques riches.

L'enseignement étant pour moi une expérience très enrichissante et complémentaire de la recherche, je remercie les enseignants qui m'ont fait confiance pour intervenir dans leurs UE : Alain, Élodie, Fabien, Florence, Frédéric, Nathalie, Stéphanie et Sylvain.

Plusieurs expérimentations ont été mises en place pendant cette thèse : merci aux centaines de participants qui ont rendu possible cet indispensable travail d'évaluation, en particulier Maëlys et les étudiants de L3 et de M1 du département informatique de l'Université Claude Bernard, qui ont répondu avec enthousiasme et bonne volonté à nos nombreuses sollicitations.

Parce qu'il n'est pas possible de travailler efficacement sans moments de détente, je remercie mes collègues de bureau, ainsi que les participants d'un grand tournoi de billard qui dure depuis déjà plusieurs années : Fatma, Florent, Gaël, Petre, Ricardo, Vincent, Vinh, Xavier et les autres. Je remercie également mes amis, informaticiens ou non, spécialement Sarah, qui est toujours là pour moi et qui sait comme personne me remonter le moral dans les moments difficiles.

Enfin, je remercie mes parents et ma famille pour leur soutien inconditionnel et leurs encouragements tout au long de mes études.

Plan du manuscrit

Introduction

Plan de l'introduction

Motivations

De nos jours, les applications informatiques sont omniprésentes et peuvent s'adresser à chacun d'entre nous, que nous soyons jeunes ou moins jeunes, novices ou informaticiens confirmés. Il peut s'agir d'applications web ou d'applications de type bureau ; ces applications peuvent être utilisées dans un contexte professionnel, éducatif ou personnel.

Ainsi, en 2012, 57 %[1] des Français ont réalisé au moins un achat sur Internet, dans des domaines divers tels que l'habillement, la réservation d'hôtel et l'électroménager. De plus, 30 %[2] des Français participent à des réseaux sociaux, que ce soit dans un but personnel ou professionnel. Le nombre de Français qui utilisent internet pour des démarches administratives augmente également d'année en année, en 2013 13,5 millions[3] de déclarations d'impôts ont été faites par internet. Ces différents exemples montrent que les applications informatiques font désormais partie du quotidien de nombreuses personnes, quel que soit leur âge ou leur catégorie socio-professionnelle.

Dans le contexte professionnel, l'usage d'applications informatiques s'est considérablement développé : la moitié des salariés utilisent un ordinateur dans le cadre de leur travail[4]. Cette progression de l'usage des applications informatiques dans le cadre professionnel a des conséquences à différents niveaux. D'une part, les enseignants du primaire au supérieur doivent s'adapter à ce besoin et former leurs élèves ou étudiants à l'usage des outils informatiques que ces derniers devront maîtriser dans leur vie professionnelle. D'autre part, certains professionnels expérimentés doivent apprendre à maîtriser des outils informatiques pour lesquels ils n'ont pas été formés pendant leurs études.

Dans le contexte éducatif également, les applications informatiques sont de plus en plus utilisées, qu'il s'agisse d'applications spécifiquement dédiées à l'enseignement, les EIAH (Environnements Informatiques pour l'Apprentissage Humain), ou d'autres types de logiciels utilisés dans le cadre d'une activité pédagogique. Les logiciels pédagogiques et les jeux sérieux sont de plus en plus intégrés dans les pratiques des enseignants et de nombreux établissements utilisent des logiciels de gestion des notes et des devoirs tels que Pronotes [Pronote, 2014] ou it's learning [It's_learning, 2014]. Ces logiciels sont destinés d'une part aux enseignants, et d'autre part aux élèves ainsi qu'à leur famille. En ce qui concerne les logiciels non spécifiquement dédiés à l'apprentissage, les enseignants en utilisent de plus en plus comme supports d'activités pédagogiques. Ainsi, des logiciels de traitement de texte peuvent être utilisés pour la saisie de rédactions et de devoirs, un logiciel de présentation comme PowerPoint peut être utilisé par les élèves pour créer un support visuel pour un

[1] Source : INSEE (http://www.insee.fr/fr/themes/document.asp?ref_id=ip1489#inter1)
[2] Source : INSEE (http://www.insee.fr/fr/themes/document.asp?ref_id=ip1452#inter3)
[3] Source : Gouvernement (http://www.economie.gouv.fr/teledeclarations-progressent)
[4] Source : INSEE (http://www.insee.fr/fr/ffc/docs_ffc/ip651.pdf)

exposé, et Excel peut être utilisé par exemple pour favoriser l'apprentissage de l'algèbre [Ritter et Koedinger, 1995].

Enfin, dans le contexte privé, l'usage d'applications informatiques autrefois réservées aux professionnels s'est généralisé, comme c'est le cas pour les outils de retouche d'images, de création d'albums photos ou de montage vidéo par exemple.

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 également autant d'écueils potentiels pour les utilisateurs d'applications informatiques.

Pour pallier les difficultés de prise en main et d'utilisation, et ainsi réduire l'abandon et la sous-exploitation des applications informatiques, deux principales solutions complémentaires sont envisageables : l'amélioration de l'application informatique et l'assistance à l'utilisateur.

Tout d'abord, une application informatique pour laquelle des problèmes de prise en main et d'utilisation ont été constatés peut être améliorée de différentes façons. Ainsi, la réingénierie permet la modification d'une application dans le but de l'améliorer, en l'étudiant, puis en la redéveloppant sous une nouvelle forme [Chikofsky et Cross, 1990]. Ce processus inclut différents participants autres que les développeurs de l'application, par exemple des utilisateurs ou des experts du domaine. Ainsi, la réingénierie pédagogique d'un EIAH permet de mettre en place de nouvelles situations pédagogiques, prenant mieux en compte les usages observés et l'évolution des pratiques pédagogiques des enseignants et des formateurs [Choquet, 2007].

De plus, des améliorations ergonomiques peuvent être réalisées afin notamment d'alléger l'interface d'une application et de la rendre plus compréhensible pour les utilisateurs. En particulier, la mise en place d'une interface respectant les critères d'ergonomie (tels que ceux proposés par Bastien et Scapin [Bastien et Scapin, 2004] ou par Meinadier [Meinadier, 1991]) peuvent éviter le rejet ou la mauvaise utilisation d'une application tout en augmentant le plaisir et la simplicité d'utilisation et de prise en main. Par ailleurs, le respect des normes d'accessibilité [W3C, 2014] permet de rendre une application plus accessible et utilisable par tous et plus particulièrement pour les utilisateurs en situation de handicap.

Bien que de telles améliorations puissent effectivement limiter les difficultés de prise en main et d'utilisation des applications informatiques, elles ne peuvent pas les éliminer totalement. De plus, de telles améliorations ne peuvent pas être réalisées par n'importe qui : seul le développeur de l'application ou un informaticien ayant accès à son code source peut

la modifier. Par ailleurs, il est très difficile pour le concepteur d'une application informatique de déterminer à l'avance tous les usages possibles de l'application, et certains utilisateurs font un usage détourné d'une application. Pour cette raison, l'assistance à l'utilisateur est une solution complémentaire à l'amélioration d'une application informatique pour pallier les difficultés tant de prise en main que d'utilisation courante, et ainsi limiter le risque d'abandon de l'activité.

Dans le cadre de nos travaux de recherche, nous définissons l'assistance comme l'ensemble des moyens qui permettent de faciliter la prise en main et l'utilisation d'une application, de manière adaptée à l'utilisateur et au contexte d'utilisation. L'assistance vise à permettre à l'utilisateur d'exploiter pleinement toutes les possibilités d'une application, et elle facilite l'appropriation des connaissances et compétences nécessaires à l'utilisation de cette application. L'assistance, qu'elle soit humaine ou logicielle, peut revêtir différentes formes, humaines et logicielles, dont nous synthétisons les forces et les faiblesses en Fig. I-1.

		Assistance humaine		**Assistance logicielle**	
		En présentiel	À distance	Intégrée à l'application	Extérieure à l'application
Capacités de l'assistance	Compréhension des objectifs de l'utilisateur	**++**	**++**	**+/-**	**+/-**
	Identification d'un problème	**++**	**--**	**+/-**	**+/-**
	Résolution du problème identifié	**++**	**+**	**+**	**+**
	Pas de disposition du code source requis	**++**	**++**	**--**	**++**
Travail de l'expert humain	Pas de préparation requise	**++**	**++**	**--**	**-**
	Accessible à un expert non-informaticien	**++**	**++**	**--**	**+**
	Pas de disponibilité de l'expert requise	**--**	**--**	**++**	**++**

Fig. I-1. Comparaison entre quatre approches d'assistance à l'utilisateur, humaines et logicielles

Tout d'abord, Il est possible de proposer à un utilisateur en difficulté une **assistance humaine en présentiel**. En effet, si un expert de l'application est présent aux côtés de l'utilisateur, il peut dialoguer avec ce dernier afin de comprendre ses objectifs. En fonction de ces objectifs et de l'observation qu'il fait de la situation, l'expert peut identifier un problème. L'utilisateur peut également solliciter lui-même l'aide de l'expert. La résolution du problème de l'utilisateur est facilitée par la présence d'un expert capable de fournir à l'utilisateur des explications et des conseils, si besoin accompagnés de gestes, par exemple pour montrer à l'utilisateur où se trouve le bouton concerné par l'action d'assistance. De plus, l'expert pourra si besoin prendre la place de l'utilisateur afin de réaliser tout ou une partie de sa tâche. Un expert pourra également adapter l'aide qu'il fournit à la situation dans laquelle se trouve l'utilisateur et aux réactions de ce dernier. Ainsi, l'utilisateur peut à tout moment

exprimer son incompréhension face à une consigne, préciser à l'expert qu'il a compris et ne souhaite pas d'explications supplémentaires ou encore que les explications fournies ne correspondent pas à son besoin. Cependant, l'assistance humaine présentielle est extrêmement contraignante : d'une manière générale, il est inconcevable qu'un expert soit présent aux côtés de chaque utilisateur à chaque utilisation d'une application.

Une **assistance humaine à distance** est également possible, par téléphone ou via internet, par exemple dans le cadre d'un service d'assistance de type hot-line. Si cette solution est moins irréaliste à grande échelle qu'une assistance humaine en présentiel, elle reste très contraignante pour l'expert. De plus, une telle solution impose à l'utilisateur d'identifier lui-même qu'il rencontre un problème avant de solliciter l'assistant. De même, la résolution du problème est plus difficile à distance qu'en présentiel, puisque l'expert ne voit pas ce que l'utilisateur en difficulté voit et il ne peut pas accompagner ses explications par des gestes. Dans le domaine de l'apprentissage à distance, une partie des travaux de recherche visent ainsi à pallier ces manques d'informations pour faciliter le tutorat [Labat, 2002 ; Guéraud et al., 2004].

Si l'assistance humaine est particulièrement efficace pour comprendre les objectifs d'un utilisateur et identifier son éventuel problème, il s'agit d'une question importante dans le cas d'une **assistance logicielle**. Une mauvaise compréhension des objectifs de l'utilisateur ou une mauvaise identification d'une situation problématique peut entraîner un agacement de l'utilisateur et une défiance vis-à-vis de l'assistance, voire un rejet total de cette assistance [Swartz, 2003]. De plus, il peut être difficile d'anticiper les compétences des utilisateurs finaux et d'adapter l'assistance en fonction de ces compétences [Leplat, 1998]. En revanche, une fois correctement identifiés l'objectif de l'utilisateur et le problème qu'il rencontre, une assistance logicielle peut se révéler efficace pour résoudre ce problème. De plus, contrairement à l'assistance humaine, l'assistance logicielle n'impose pas la disponibilité d'un expert lors de l'utilisation de l'application par un utilisateur en difficulté.

Pour ajouter une assistance logicielle dans une application existante, une première approche consiste à redévelopper cette application pour y inclure une **assistance logicielle intégrée à l'application**. Néanmoins, il s'agit d'un travail conséquent qui requiert des connaissances en programmation pour l'expert qui développe le module d'assistance. De plus, cette approche est impossible lorsque le code source de l'application n'est pas disponible. En effet, l'expert qui souhaite ajouter une assistance logicielle à une application n'est pas toujours le concepteur de l'application. Or, de nombreuses applications ne sont pas open-source, en particulier les applications commerciales, ce qui rend impossible le redéveloppement de ces applications par des personnes autres que leur développeur. De plus, cet expert n'est pas toujours programmeur.

Une alternative au développement d'un module d'assistance intégrée à l'application consiste à permettre à un expert de définir l'assistance qu'il souhaite pour l'application, puis à exécuter cette assistance de manière extérieure à l'application, sans nécessiter de redéveloppement ni perturber son fonctionnement. Cette dernière approche d'**assistance logicielle extérieure à l'application** est celle que nous adoptons dans cette thèse. Cette approche présente l'avantage d'être applicable y compris lorsque le code source de

l'application-cible n'est pas disponible, puisqu'aucune modification n'est apportée à l'application, et elle ne requiert pas de connaissances en programmation de la part de l'expert de l'assistance.

Contexte

Cette thèse en informatique se situe dans le cadre du projet AGATE[5] (Approche Générique d'Assistance aux Tâches complexEs), qui vise à proposer des modèles et des outils pour permettre la mise en place *a posteriori* de systèmes d'assistance dans des applications existantes, que nous appelons applications-cibles. Dans le cadre du projet AGATE, nous adoptons une démarche entièrement générique : les modèles et outils du projet ne sont pas spécifiques à un environnement ou à un domaine, mais peuvent concerner n'importe quelle application-cible, sans contrainte sur celle-ci, ce qui signifie que les applications-cibles n'ont pas besoin d'avoir été spécifiquement conçues pour permettre l'ajout d'assistance par les outils du projet AGATE.

Problématique et questions de recherche

La problématique générale de cette thèse peut s'exprimer de la manière suivante :

Comment permettre la mise en place de systèmes d'assistance dans des applications existantes en adoptant une démarche générique ?

Cette problématique soulève plusieurs questions de recherche que nous détaillons ci-après.

Tout d'abord, nous étudierons **comment intégrer un système d'assistance à une application informatique, sans avoir à la redévelopper ou à la modifier et sans contrainte sur cette application ?** L'ajout d'assistance doit être rendu possible dans des applications très variées, quels que soient leur domaine et leur public cible. Les applications concernées peuvent être utilisées sous différents systèmes d'exploitation et avec des supports variés (ordinateur, tablette, smartphone…). Elles ne doivent pas nécessiter d'être spécifiquement conçues pour permettre l'ajout d'assistance. De plus, ces applications-cibles peuvent avoir été développées dans différents langages de programmation, et leur code source n'est pas toujours accessible pour le concepteur potentiel de l'assistance.

Par ailleurs, nous nous demanderons **comment permettre à une personne de mettre en place un système d'assistance dans une application sans compétence en programmation ?** Le premier enjeu de cette question de recherche se situe au niveau de la spécification de l'assistance souhaitée, qui doit être accessible à un concepteur d'assistance potentiellement non-informaticien. Le second enjeu est de permettre l'exécution automatique de l'assistance spécifiée, ce qui suppose qu'elle soit suffisamment formalisée.

[5] http://liris.cnrs.fr/stephanie.jean-daubias/projets/p-agate.html

Enfin, nous chercherons à déterminer **comment proposer une assistance adaptée aux besoins de l'utilisateur ?** Des études ont en effet montré qu'un système d'assistance peut être rejeté par ses utilisateurs s'il les interrompt trop souvent dans leur tâche, ou s'il leur propose une assistance non pertinente, soit parce que l'utilisateur n'en a pas besoin, soit parce qu'elle ne correspond pas à son objectif [Randall et Pedersen, 1998 ; Galluccio, 2006]. Le premier enjeu de cette question de recherche est donc de permettre une contextualisation fine de l'assistance proposée à un utilisateur, en prenant en compte les spécificités de l'application et de son état au moment de l'exécution de l'assistance. Le deuxième enjeu de cette question de recherche est de permettre la personnalisation de l'assistance. En effet, il s'agit d'une approche pertinente pour favoriser l'acceptation de l'assistance par l'utilisateur auquel elle est destinée. L'assistance doit pouvoir prendre en compte les spécificités de chaque utilisateur, comme ses expériences, ses objectifs, ses connaissances, ses capacités et éventuels handicaps, ainsi que ses préférences. Elle doit également pouvoir s'adapter en fonction de l'assistance déjà proposée à l'utilisateur. Enfin, le troisième enjeu de cette question de recherche est de permettre la mise en place d'assistances riches, capables de prendre des formes très variées. Cette diversité permettra en effet une plus grande personnalisation de l'assistance.

Approche épiphyte et générique

Pour permettre l'ajout *a posteriori* d'un système d'assistance à une application-cible quelconque, sans avoir à la redévelopper ou à la modifier, sans posséder son code source et sans qu'elle ait été spécifiquement conçue pour permettre l'ajout d'assistance, une solution consiste à adopter une démarche épiphyte.

Le terme épiphyte vient du grec et signifie littéralement « sur (épi) une plante (phyte) ». Il est utilisé en biologie pour désigner les plantes qui grandissent avec comme support une autre plante, l'hôte. Il ne s'agit pas de plantes parasites qui puisent leur nourriture de leur hôte : les plantes épiphytes ne perturbent pas la croissance de leur hôte. Les orchidées tropicales qui poussent sur certains arbres sont des exemples de plantes épiphytes.

Suivant la métaphore biologique de la plante épiphyte, en informatique, une application épiphyte [Paquette et al., 1994] est une application pouvant être greffée sur une autre application, qualifiée d'hôte ou d'application-cible. L'application-cible est indépendante du système épiphyte potentiellement greffé sur elle : son fonctionnement n'est pas perturbé par le système épiphyte. Un système d'assistance épiphyte est donc un système d'assistance qui peut être greffé sur une application-cible pour les utilisateurs de cette application sans perturber son fonctionnement.

Dans le cadre de cette thèse, nous adoptons une démarche entièrement épiphyte. Les modèles et outils que nous proposons permettent de greffer de manière épiphyte des systèmes d'assistance sur leur application-cible. Nous appelons *épi-assistants* les assistants capables de réaliser des actions d'assistance *sur* une application-cible. Nous appelons également *épi-détecteurs* et *épi-inspecteurs* les outils permettant de surveiller de manière épiphyte une application-cible afin de détecter les interactions entre un utilisateur et l'interface de cette application-cible, ou de connaître l'état de ses composants.

Nous avons également fait le choix d'adopter une démarche générique. Les modèles que nous proposons dans le cadre de cette thèse ne sont pas spécifiques à une application ou un domaine donné, mais peuvent être utilisés pour ajouter de l'assistance dans de nombreuses applications, appartenant à différents domaines et utilisées dans des contextes variés.

Ainsi, en plus d'être épiphytes, les outils qui mettent en œuvre nos propositions théoriques sont génériques. En revanche, les systèmes d'assistance définis à l'aide de nos modèles et outils génériques sont quant à eux spécifiques à l'application-cible dans laquelle l'assistance sera greffée. En effet l'assistance que nos propositions visent à mettre en place peut être finement contextualisée en fonction des spécificités de leur application-cible et de son contexte d'utilisation.

Scénarios d'usage

Dans cette section, nous présentons quatre scénarios d'usage dans lesquels des utilisateurs sont confrontés à des difficultés de prise en main ou d'utilisation d'applications informatiques. Ces scénarios mettent en jeu des situations variées dans lesquelles les applications informatiques impliquées ne permettent pas de répondre aux besoins particuliers des acteurs du scénario. Ainsi, les premier et troisième scénarios concernent un usage personnel de logiciels grand public, et impliquent respectivement des adultes et des enfants. Le deuxième scénario concerne l'usage d'une application informatique dans le cadre d'une activité pédagogique. Enfin, le quatrième scénario concerne la mise en place d'une assistance pour l'usage personnel de son concepteur.

Scénario 1 : une alternative au dépannage par téléphone pour PhotoScape

Stéphane envoie régulièrement par mail des photos de ses enfants à sa mère Martine qui n'habite pas dans la même région que lui. Lors d'une de ses visites, Stéphane a installé sur l'ordinateur de Martine le logiciel de retouche d'images PhotoScape pour permettre à sa mère de corriger les yeux rouges sur une photo, de la recadrer, de créer des images de type pêle-mêle à partir d'un ensemble de photos et d'enregistrer ses photos dans un format compressé afin de pouvoir ensuite facilement les envoyer par mail. Martine est très intéressée par les fonctionnalités proposées par PhotoScape. Néanmoins, elle oublie d'une utilisation sur l'autre comment utiliser le logiciel pour réaliser les tâches que lui a présentées son fils. Lorsqu'elle se trouve bloquée, Martine téléphone à Stéphane pour qu'il lui explique ce qu'elle doit faire.

Stéphane souhaiterait que Martine puisse être plus autonome et parvienne seule à utiliser PhotoScape pour corriger les yeux rouges d'une photo, et effectuer les autres manipulations de photos qui l'intéressent.

Scénario 2 : une prise en main facilitée de NetBeans

François enseigne l'ergonomie des logiciels à des étudiants de 3ème année de licence d'informatique. Dans ce cadre, les étudiants de François créent des applications graphiques en Java à l'aide de l'environnement de développement intégré NetBeans. Chaque semestre, François est confronté à un public très hétérogène : certains étudiants ne connaissent ni la

programmation Java ni NetBeans, alors que d'autres qui ont une grande expérience de la programmation Java n'ont cependant jamais utilisé NetBeans pour créer des interfaces graphiques.

François désire mettre en place un tutoriel qui permette à ses étudiants de découvrir NetBeans et d'acquérir des notions de base de la programmation Java pour la création d'interfaces graphiques. François souhaiterait que ce tutoriel soit flexible pour que ses étudiants puissent étudier à leur rythme chaque notion présentée par le tutoriel, en ayant la possibilité d'ignorer les parties du tutoriel qui ne les intéressent pas. Ce tutoriel serait proposé aux étudiants lors du premier TP du semestre, afin de leur donner les bases que François juge utiles dans le cadre de son cours. François désire de plus que ce tutoriel soit entièrement intégré à NetBeans, afin que les étudiants puissent réaliser les exercices du tutoriel directement dans NetBeans, sans avoir à suspendre le tutoriel à chaque action à réaliser dans NetBeans, comme cela serait le cas pour un tutoriel en ligne par exemple.

Scénario 3 : une utilisation ludique de Géogébra

Mathilde, 12 ans, est inscrite au club Géogébra de son collège, dans lequel les élèves apprennent à utiliser de manière ludique ce logiciel éducatif de géométrie. Les élèves de ce club ont notamment appris à utiliser les outils de Géogébra pour tracer et colorier des rosaces. Mathilde a montré des exemples de rosaces à Thomas, son petit frère de 7 ans. Thomas aimerait pouvoir créer lui aussi de telles rosaces, néanmoins cet exercice requiert la manipulation de plusieurs notions mathématiques abordées au collège qui ne sont pas encore à sa portée. Mathilde aimerait toutefois, qu'en bénéficiant d'une assistance appropriée, Thomas puisse jouer avec Géogébra pour créer simplement des rosaces colorées sans qu'il ait besoin de maîtriser les notions mathématiques mises en jeu.

Scénario 4 : des tâches automatisées dans Gimp

Anthony utilise occasionnellement l'éditeur d'images GIMP afin de rendre transparent le fond des images qu'il télécharge. D'une fois sur l'autre, il oublie comment procéder. Pour éviter d'avoir à rechercher comment faire, Anthony souhaiterait créer un système d'assistance pour son propre usage, afin de gagner du temps sur cette tâche. Après qu'il ait sélectionné dans GIMP le fond de l'image qu'il souhaite modifier, le système d'assistance qu'il imagine automatiserait la tâche d'ajout d'un canal alpha et de transformation de la couleur sélectionnée vers ce canal, puis l'export de l'image au format png.

Dans les quatre scénarios que nous avons présentés, les protagonistes sont confrontés à des difficultés qui pourraient être résolues par l'ajout *a posteriori* d'un système d'assistance dans une application informatique existante : PhotoScape, NetBeans, Géogébra et Gimp. Les protagonistes n'ont pas la possibilité de développer eux-mêmes ces systèmes d'assistance, que ce soit par manque de compétences en informatique, par manque de temps ou parce qu'ils n'ont pas accès au code source de l'application.

L'objectif de cette thèse est de permettre la mise en place *a posteriori* de systèmes d'assistance, dans des applications existantes, sans les modifier et sans recourir à la

programmation. Nous reviendrons sur ces scénarios d'usage en conclusion de ce manuscrit afin de montrer de quelles manières nos contributions permettent de résoudre les difficultés de leurs protagonistes.

Contributions

Afin de répondre à la première question de recherche, qui concerne l'ajout d'assistance dans une application quelconque sans la modifier, nous avons choisi d'adopter une démarche entièrement *épiphyte*. En effet, une telle démarche permet de greffer une assistance à une application existante, sans avoir à la modifier et sans contrainte sur cette application. Nous avons proposé un **processus d'adjonction d'un système d'assistance** à une application-cible de manière épiphyte. Il est constitué de deux phases : la spécification de l'assistance, qui concerne le concepteur de l'assistance, et la phase d'exécution de l'assistance, qui permet de fournir l'assistance spécifiée aux utilisateurs finaux de l'application-cible.

Pour répondre à notre deuxième question de recherche, qui concerne la mise en place d'assistance dans une application sans recourir à la programmation, nous avons proposé **aLDEAS**, un **langage pivot** entre la phase de spécification de l'assistance par un concepteur potentiellement non-informaticien et la phase d'exécution de cette assistance par un outil générique. Il s'agit d'un **langage graphique**, ce qui facilite son utilisation par un non-informaticien. De plus, le langage aLDEAS est complété par plusieurs **patrons** qui facilitent la définition de blocs aLDEAS par le concepteur de l'assistance. Ainsi, nous avons proposé un patron de règles d'assistance qui permet de définir des règles équivalentes à celles proposées dans les approches d'assistance épiphytes existantes. Nous avons également proposé des patrons d'actions d'assistance fréquemment rencontrées dans les assistances existantes. Nous avons également proposé **aMEAS**, le modèle d'exécution d'un système d'assistance aLDEAS qui détermine le fonctionnement de la phase d'exécution d'un système d'assistance défini par un ensemble de règles aLDEAS.

Pour répondre à la troisième question de recherche, qui concerne l'adaptation de l'assistance aux besoins de l'utilisateur, nous avons proposé à travers le langage aLDEAS une grande variété d'**actions d'assistance**, associées à des paramètres optionnels qui permettent la prise en compte des préférences des utilisateurs finaux. De plus, le déclenchement de ces actions d'assistance peut être finement **contextualisé** en fonction des actions réalisée par l'utilisateur final dans l'application. L'assistance peut également être **personnalisée** en fonction des spécificités de l'utilisateur final exprimées dans son profil, notamment ses préférences relatives à l'assistance, ainsi que ses connaissances, compétences et éventuels handicaps.

Plan du manuscrit

Ce manuscrit de thèse est divisé en quatre parties, en complément de cette introduction et d'une conclusion qui présente notamment les perspectives de cette recherche.

La première partie concerne l'état de l'art relatif à notre problématique de recherche. Nous y présentons les différentes dimensions d'un système d'assistance dans le Chapitre 1, avant de proposer une typologie de l'assistance à l'utilisateur d'applications informatiques. Nous présentons dans le Chapitre 2 les systèmes d'assistance épiphytes existants. Le Chapitre 3 est consacré à la surveillance des applications informatiques.

La deuxième partie de ce manuscrit présente nos propositions théoriques. Le Chapitre 4 présente tout d'abord notre processus de surveillance d'une application-cible de manière épiphyte. Ce processus rend possible la surveillance des interactions entre un utilisateur et l'interface d'une application-cible qui n'est pas conçue spécifiquement pour permettre cette surveillance. Le Chapitre 5 présente quant à lui la contribution théorique majeure de nos travaux de thèse : le langage de définition de systèmes d'assistance épiphytes aLDEAS, complété par des patrons d'actions d'assistance. Nous expliquons dans le Chapitre 6 les choix que nous avons faits afin de permettre la consultation de trois types de ressources pour l'assistance : le profil de l'utilisateur, l'historique de l'assistance et les traces de l'utilisateur. Enfin, nous exposons dans le Chapitre 7 aMEAS, le modèle d'exécution d'un système d'assistance aLDEAS.

La troisième partie de ce manuscrit de thèse présente le système SEPIA, qui met en œuvre nos propositions théoriques. Le Chapitre 8 présente tout d'abord l'éditeur d'assistance qui met en œuvre la partie spécification de l'assistance du processus d'adjonction d'un système d'assistance à une application existante. Il est à destination des concepteurs d'assistance et leur fournit une interface pour la définition des règles respectant le patron de règles d'assistance complétant le langage aLDEAS. Le Chapitre 9 concerne la phase d'exécution de l'assistance avec SEPIA. Il présente en particulier le moteur d'assistance de SEPIA, ainsi que les différents outils qui le complètent.

Enfin, la quatrième partie est consacrée aux évaluations des propositions faites dans le cadre de cette thèse. Dans cette partie, nous proposons tout d'abord dans le Chapitre 10 une grille d'évaluation dans laquelle nous identifions les différents points à évaluer et proposons une méthode d'évaluation pour chaque point. Nous présentons ensuite dans le Chapitre 11 les différentes expérimentations réalisées, avant d'exposer dans le Chapitre 12 nos conclusions quant à la validation de nos travaux.

L'ensemble de ce manuscrit est ponctué d'exemples qui appuient nos propos : certains de ces exemples sont volontairement simples, afin d'illustrer de manière concise le point précis qu'ils concernent ; nous les avons le plus souvent choisis en lien avec les scénarios d'usage que nous avons présentés et les expérimentations que nous avons menées. Un glossaire est par ailleurs à la disposition du lecteur en Annexe H.

Partie 1. État de l'art

Plan de la partie « État de l'art »

Dans le cadre de ce travail de recherche, nous nous intéressons à la mise en place *a posteriori* de systèmes d'assistance dans des applications existantes. Pour cette raison, nous avons réalisé un état de l'art de l'assistance à l'utilisateur d'applications informatiques. Nous nous appuyons pour cela sur une analyse bibliographique et sur une étude d'assistances existantes. Ce travail nous a permis d'identifier treize dimensions de l'assistance que nous présentons de manière synthétique dans la section 1.1, en les illustrant d'exemples provenant de différents domaines d'application.

Nous présentons ensuite de manière plus détaillée dans le Chapitre 2 les systèmes d'assistance non intégrés à leur application, qu'il s'agisse de systèmes d'assistance indépendants ou épiphytes vis-à-vis de l'application-cible.

Enfin, la mise en place d'une assistance épiphyte nécessitant une surveillance et une inspection de l'application-cible, nous avons étudié les différentes techniques permettant une telle surveillance dans des applications non spécifiquement conçues pour être surveillées. Le Chapitre 3 propose une synthèse de ces techniques.

Chapitre 1.

Chapitre 1. Assistance à l'utilisateur d'applications informatiques

Objectif du chapitre

Étudier et caractériser les assistances proposées dans les applications informatiques existantes, qu'elles soient ou non issues de la recherche, quel que soit leur domaine d'application.

Points clés

Nous avons réalisé un état de l'art de l'assistance à l'utilisateur d'applications informatiques, que nous avons synthétisé et illustré. Il présente les 15 dimensions d'un système d'assistance que nous avons identifiées. Trois de ces dimensions sont particulièrement centrales par rapport à notre problématique : les objectifs d'un système d'assistance, c'est-à-dire les besoins d'assistance auxquels il tente de répondre ; les moyens techniques mis en œuvre par le système d'assistance pour répondre à ces besoins ; ainsi que les approches d'assistances exploitées. Ces trois dimensions sont détaillées sous la forme d'une typologie de l'assistance à l'utilisateur d'applications informatiques.

Contributions

- 15 dimensions des systèmes d'assistance à l'utilisateur d'applications informatiques
- Typologie de l'assistance à l'utilisateur d'applications informatiques

Publication liée à ce chapitre

[Ginon et al., 2013b]

Ginon B., Jean-Daubias S. et Champin P.-A., Une typologie de l'assistance aux utilisateurs : exemple d'application aux EIAH, RR-LIRIS-2013-007, 2013b.

Dans le cadre de nos travaux de recherche, nous nous sommes intéressée à l'assistance à l'utilisateur d'applications informatiques. Grâce à une étude bibliographique et à une étude de systèmes d'assistances existants, nous avons identifié plusieurs dimensions des systèmes d'assistance, que nous présentons dans la section suivante. Nous proposons également une typologie de l'assistance qui confronte les trois dimensions que nous considérons comme centrales dans notre travail de recherche. Nous illustrons nos propos en nous appuyant sur des exemples d'assistances existantes variées. Certains exemples sont issus de travaux de recherche, notamment dans le domaine des Interactions Homme-Machine et des EIAH, d'autres proviennent d'applications grand public, commerciales ou en ligne.

1.1. Dimensions de l'assistance

Nous avons identifié différentes dimensions qui peuvent être utilisées pour caractériser un système d'assistance (cf. Fig. 1-1). Tout d'abord, un système d'assistance concerne une **application**-cible dans laquelle l'assistance sera exécutée, un **public** à qui l'assistance est destinée et un **concepteur** qui peut être le *développeur* de l'application-cible ou un *utilisateur expert* qui souhaite aider d'autres utilisateurs. Un système d'assistance peut être compatible avec une ou plusieurs **plateformes**. Les systèmes d'assistance peuvent se distinguer au niveau de leur **intégration** vis-à-vis de l'application-cible : un système d'assistance peut être *externe* à une application-cible, comme une vidéo de démonstration en ligne, ou *intégré* à l'application-cible. Dans ce dernier cas, nous distinguons le cas des systèmes d'assistance *internes* à l'application-cible de celui des systèmes d'assistance *épiphytes*, qui sont liés à l'application-cible, mais pour lesquels l'architecture logicielle de l'application-cible est totalement indépendante du système d'assistance. Concernant la **temporalité**, qu'il soit externe ou intégré à l'application-cible, un système d'assistance peut avoir été mis en place *a priori*, *a posteriori* ou *en parallèle* du développement de l'application-cible.

Un système d'assistance a pour **objectif** de répondre à différents besoins d'assistance auxquels les utilisateurs finaux de l'application-cible peuvent être confrontés. Pour répondre à ces besoins, le système peut utiliser plusieurs moyens **techniques**, comme les messages d'aide et les mises en valeur de composants de l'interface de l'application. Ces techniques correspondent à différentes **approches** d'assistance, comme les systèmes conseillers et les agents conversationnels animés qui peuvent notamment fournir de l'assistance sous forme de messages d'aide. Les dimensions **objectif, technique et approche** sont détaillées et confrontées dans la typologie de l'assistance présentée en section 1.2. Un système d'assistance peut se manifester pour l'utilisateur final à différents **emplacements**, par exemple une zone dédiée à l'assistance dans l'application ou une fenêtre adjacente à l'application.

Dimension	Définition	Exemples
Application	Application-cible pour laquelle le SA est conçu	Site web commercial, logiciel grand public, logiciel pédagogique…
Concepteur	Personne ayant conçu le SA	Développeur de l'application-cible, utilisateur expert de l'application-cible, enseignant…
Public	Utilisateurs finaux à qui l'assistance est destinée	Utilisateur d'un site web ou d'un logiciel, membre d'une communauté d'utilisateurs, apprenant…
Plateforme	Plateforme informatique compatible	Windows, Linux, Mac OS, Web, Plateforme mobile
Intégration	Lien entre le SA et l'application-cible	Interne, externe, épiphyte
Temporalité	Moment de la mise en place du SA par rapport au développement de l'application-cible	*A priori*, en parallèle, *a posteriori*
Objectif	Besoins d'assistance auxquels le SA cherche à répondre	Découverte du logiciel ou d'une fonctionnalité, guidage pour la réalisation d'une tâche…
Technique	Moyens techniques mis en œuvre par le SA pour fournir l'assistance	Messages d'aide, vidéo de démonstration, mises en valeur d'éléments de l'interface…
Emplacement	Endroit dans lequel se manifeste le système d'assistance	Zone dédiée dans l'application, fenêtre adjacente à l'application…
Approche	Approches d'assistance utilisées par le SA	Manuel d'aide, système conseiller, agent conversationnel animé…
Intervention	Degré d'intervention du SA	Substitution, suppléance, assistance, support
Initiative	Définit si l'assistance est fournie à l'initiative du SA ou à la demande de l'utilisateur final	Proactif, réactif, mixte, paramétrable
Évolutivité	Capacité du SA à faire évoluer l'assistance fournie à l'utilisateur final	Adaptivité par raisonnement à partir de cas ou de traces…
Contextualisation	Capacité du SA à prendre en compte le contexte	Prise en compte de l'état de l'application-cible ou des interactions entre l'application-cible et l'utilisateur final…
Personnalisation	Capacité du SA à prendre en compte les spécificités de l'utilisateur final	Prise en compte d'un handicap ou du niveau de maîtrise de l'application-cible…

Fig. 1-1. Les dimensions d'un système d'assistance

Concernant le degré d'**intervention** de l'assistance, Olivier Gapenne [Gapenne, 2006] propose la classification suivante : *substitution*, lorsque la technologie prend en charge de manière autonome tout ou partie d'une tâche donnée, *suppléance*, lorsque l'usage de la technologie aidante modifie le pouvoir d'action de son utilisateur et que l'on peut observer de nouveaux schèmes d'action, *assistance*, lorsque la technologie aidante a un rôle annexe dans le sens où elle n'est pas nécessaire pour mener à bien l'activité, mais qu'elle facilite l'utilisation de l'application-cible, *aide*, lorsque la technologie aidante permet l'appropriation et l'usage d'un schème nouveau pour l'utilisateur final. Par ailleurs, Ted Selker [Selker, 1994] distingue les systèmes d'assistance de type *conseiller*, qui donnent des informations et proposent des solutions que l'utilisateur final est libre de suivre ou non, des systèmes de type *assistant*, capables d'intervenir directement dans l'application-cible pour exécuter des tâches répétitives à la place de l'utilisateur final. Ces systèmes conseillers assistants sont de type substitution dans la classification de Gapenne.

Gilbert Paquette [Paquette et al., 1994] propose de classifier les systèmes d'assistance selon qui est à l'**initiative** de l'assistance : *proactif*, si le système détecte un besoin d'assistance et propose de l'aide à l'utilisateur, *réactif*, si le système agit à la demande de l'utilisateur, ou *mixte* s'il combine ces deux approches. Certains systèmes d'assistance peuvent également être *paramétrables* : l'utilisateur peut alors désactiver l'aide totalement ou partiellement et éventuellement définir le niveau d'aide souhaité. Les systèmes proactifs sont également qualifiés de *conversants* par Ted Selker [Selker, 1994] et Henry Lieberman [Lieberman, 1997], à l'opposé des systèmes autonomes.

Jörg Rech [Rech et al., 2007] propose quant à lui une autre classification des systèmes d'assistance, qui s'appuie sur quatre questions : *quand assister ?*, qui correspond à notre dimension initiative, *où assister ?*, qui correspond à notre dimension emplacement, *comment assister ?*, qui correspondent à notre dimension technique (par exemple proposer un message textuel dans une infobulle dans l'application-cible ou dans une fenêtre pop-up adjacente à l'application-cible, ou afficher une vidéo de démonstration dans une partie de l'application-cible dédiée à l'assistance), et *pourquoi assister ?*, qui correspond à notre dimension objectif.

Par ailleurs, certains systèmes d'assistance sont capables de faire **évoluer** l'assistance qu'ils fournissent aux utilisateurs finaux de l'application-cible. C'est le cas notamment des systèmes d'assistance *adaptatifs* [Brusilovsky, 2001], capables d'adapter l'assistance de manière autonome. D'autres systèmes s'appuient sur une base de connaissances qu'il est possible de faire évoluer, mais qui requiert une intervention extérieure au système, par exemple celle d'un expert humain. De plus, un système d'assistance peut être **contextualisé**, et ainsi adapter l'assistance à l'état de l'application-cible. Par exemple, un système d'assistance peut proposer une assistance différente selon la fonctionnalité de l'application-cible utilisée, comme c'est le cas des manuels d'aide contextualisée de type WinHelp [WinHelp, 2014]. Enfin, certains systèmes peuvent proposer une assistance **personnalisée** en fonction des spécificités de l'utilisateur final. Pour cela, ils s'appuient sur des profils d'utilisateurs, qui peuvent contenir des informations variées sur les utilisateurs, comme leurs préférences en matière d'assistance, ou leurs connaissances relatives à l'application-cible.

1.2. Typologie de l'assistance à l'utilisateur

Dans la suite de cette section, nous présentons une typologie de l'assistance à l'utilisateur (cf. Fig. 1-2) qui confronte les trois dimensions de l'assistance qui sont centrales dans notre travail de thèse [Ginon et al., 2013d]. La dimension **objectif** correspond aux besoins d'assistance des utilisateurs finaux auxquels un système d'assistance vise à répondre. La dimension **technique** montre les différentes formes que peut adopter l'assistance pour atteindre ces objectifs, ce qui correspond à l'un des enjeux de notre troisième question de recherche. La dimension **approche** quant à elle nous permet d'identifier les travaux à étudier plus particulièrement pour répondre à notre problématique générale et avec lesquels comparer par la suite notre approche.

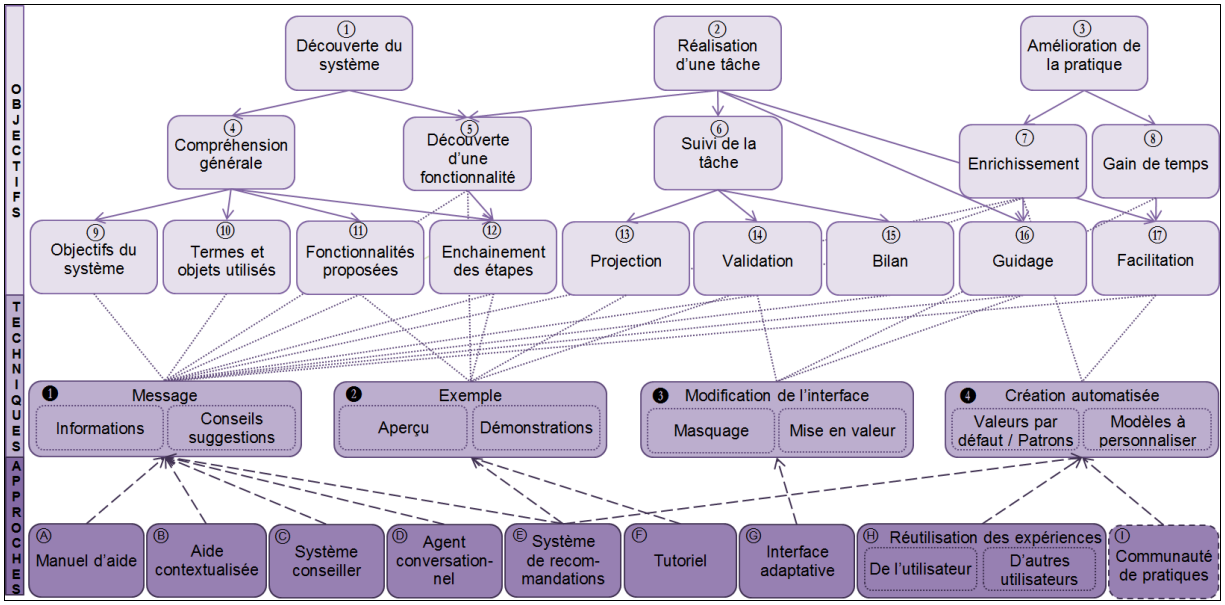


Fig. 1-2. Typologie de l'assistance à l'utilisateur d'applications informatiques

1.2.1. Objectifs de l'assistance

Nous avons identifié différents besoins d'assistance des utilisateurs, qui correspondent à des objectifs d'assistance. Nous les avons regroupés en catégories, puis hiérarchisés. Nous avons identifié trois principaux besoins d'assistance (cf. partie haute de la Fig. 1-2) : la découverte du système, la réalisation d'une tâche et l'amélioration de la pratique.

Les besoins d'assistance pour la **découverte du système** (cf. ① Fig. 1-2) regroupent les besoins de compréhension générale du système ④ et de découverte d'une fonctionnalité particulière ⑤, qui concernent principalement les utilisateurs novices : à quoi sert le système ⑨, quels sont les termes et objets manipulés dans le système et comment peut-on les utiliser ⑩, quelles sont les fonctionnalités proposées par le système ⑪ et le cas échéant quelles sont les étapes nécessaires à la réalisation d'une tâche ⑫. L'aide en ligne de la suite bureautique Office 2010 [Office_2010, 2014] propose par exemple de découvrir le système ④ et ses fonctionnalités ⑪ à l'aide de courtes vidéos et d'un glossaire ⑩.

Les besoins d'assistance pour la **réalisation d'une tâche** ② concernent tous les utilisateurs, qu'ils soient ou non novices. Parmi ces besoins, la découverte d'une nouvelle fonctionnalité ⑤ peut être nécessaire pour un utilisateur lorsqu'il effectue, pour la première fois notamment, une tâche qui requiert l'utilisation d'une ou plusieurs fonctionnalités du système. De plus, lorsqu'une tâche est complexe ou longue à réaliser, un suivi de la progression de la tâche ⑥ peut aider l'utilisateur à mieux comprendre ce qu'il est en train de faire. Une projection ⑬ peut aider l'utilisateur à mieux comprendre ce qu'il a fait, et éventuellement à repérer ce qu'il doit rectifier dans son travail pour obtenir le résultat souhaité. Cette projection peut notamment se faire sous la forme d'un aperçu du travail réalisé par l'utilisateur, ou d'une instanciation de son travail sur un exemple. Ainsi, dans l'interface enseignant de l'EIAH Ambre-Add [Jean-Daubias et Guin, 2009], l'enseignant peut contraindre des problèmes à créer pour l'EIAH, avec la possibilité de visualiser des exemples de problèmes qui correspondent aux contraintes définies. Parmi les besoins qui concernent le suivi de la tâche, une validation ⑭ peut être nécessaire pour certaines tâches, pour confirmer à l'utilisateur que ce qu'il a fait est correct, ou au contraire pour mettre en évidence les problèmes ou les incohérences de ses productions. Dans Educ@ffix, un logiciel de réalisation à distance de TP de chimie [Marzin et al., 2005], le module d'assistance permet aux apprenants de consulter à tout moment la liste et la description des éventuelles erreurs pour chaque étape. Dans Ambre-Add, les apprenants peuvent demander au système d'assistance de vérifier si ce qu'ils ont fait est juste. Parmi les besoins qui concernent le suivi de la tâche, un bilan de ce qui a été fait et de ce qu'il reste à faire peut être utile à l'utilisateur ⑮. Ainsi, dans Educ@ffix, les apprenants peuvent visualiser leur progression dans la réalisation du TP. Par ailleurs, la réalisation d'une tâche peut entraîner un besoin de guidage ⑯, afin notamment d'aider l'utilisateur à comprendre ce qu'il doit ou peut faire à un moment donné. Ainsi, dans Ambre-Add, si un apprenant clique sur « j'ai terminé » sans avoir effectué toutes les étapes, le système d'assistance lui explique ce qu'il lui reste à faire. Enfin, Il est possible de faciliter la tâche de l'utilisateur ⑰, en pré-réalisant cette tâche partiellement ou totalement, tout en lui permettant de modifier ou compléter les propositions qui lui sont faites. Dans Exploragraph par exemple, les enseignants ont la possibilité de créer des scénarios pédagogiques [Villiot-Leclercq, 2006]. Un module d'assistance leur propose pour cela de choisir entre six scénarios types qu'ils pourront ensuite modifier pour les adapter à leurs besoins.

Les besoins d'assistance relatifs à l'**amélioration de la pratique** ③ concernent tous les utilisateurs. L'amélioration peut porter d'une part sur l'enrichissement de la pratique de l'utilisateur ⑦, notamment en mettant en évidence des options ou fonctionnalités offertes par le système et non utilisées, comme dans USCSH, un système d'assistance aux utilisateurs d'Unix [Mattews et al., 2000], et d'autre part sur le temps de réalisation d'une tâche ⑧, par l'automatisation de certaines parties de la tâche par exemple.

1.2.2. Techniques d'assistance

Nous avons vu que les besoins d'assistance sont divers. Ils dépendent d'une part de l'application en elle-même, de sa complexité, de sa richesse, et d'autre part de l'utilisateur, de son niveau de maîtrise de l'application et des objectifs pour lesquels il utilise cette

application. Afin de répondre à la variété de ces besoins, qu'ils soient explicitement identifiés ou non, il existe différentes techniques, présentées dans la partie centrale de la Fig. 1-2. Nous avons regroupé les techniques d'assistance observées dans les systèmes existants en quatre catégories : les messages, les exemples, les modifications de l'interface et la création automatisée.

Les **messages d'assistance** ❶ permettent de communiquer des informations à l'utilisateur. Ils constituent un moyen à la fois simple et efficace d'assister l'utilisateur et sont très utilisés dans de nombreuses applications. Ils sont souvent proposés sous forme de fenêtres pop-up, de bulles d'aide [Grossman et Fitzmaurice, 2010] ou par un compagnon s'adressant directement à l'utilisateur. Richard [Richard, 2008], qui s'intéresse aux systèmes conseillers pour les sites web, distingue les messages d'aide selon trois types de contenus : du texte (recommandations, explications), des liens (vers une page web), et des raccourcis vers une fonctionnalité.

Les aides de type **exemple** ❷ permettent de réduire la distance entre l'explication et la tâche concrète de l'utilisateur. Il peut s'agir d'exemples fournis par l'application dans le but d'illustrer les possibilités de l'application, ou d'un aperçu du travail déjà réalisé par l'utilisateur. Ainsi, dans l'interface enseignant d'Ambre-Add, l'enseignant peut visualiser des exemples de problèmes qui correspondent aux contraintes qu'il a définies. Il peut également s'agir de démonstrations, par exemple sous forme de vidéos, expliquant à l'utilisateur comment réaliser une tâche ou prendre en main l'application, comme cela est proposé dans l'aide en ligne d'Office 2010.

L'assistance par la **modification de l'interface** ❸ consiste à effectuer des changements directement sur l'écran en jeu. Les modifications peuvent être de type masquage, pour griser une fonctionnalité indisponible par exemple, ou au contraire de type mise en valeur, par exemple pour indiquer à l'utilisateur qu'il n'a pas rempli le champ d'un formulaire ou pour mettre en valeur une fonctionnalité qui pourrait lui être utile.

La **création automatisée** ❹, parfois qualifiée de substitution [Gapenne, 2006], permet de simplifier la tâche de l'utilisateur et de lui faire gagner du temps en réalisant tout ou partie de sa tâche à l'aide de valeurs par défaut, de patrons ou de modèles à personnaliser. Les valeurs par défaut permettent de pré-remplir automatiquement les champs d'un formulaire. Elles peuvent être prédéfinies par le système, ou proposées à l'utilisateur en fonction des valeurs qu'il utilise habituellement, ou encore en fonction des valeurs utilisées par d'autres utilisateurs ayant des besoins, des objectifs ou des caractéristiques proches de ceux de l'utilisateur. Un patron est une solution, qui peut être partielle, à un problème récurrent dans un contexte donné [Goodyear et al., 2004], établie par un concepteur ou un utilisateur averti. Il est exprimé par un ensemble de spécifications qui sont exploitées par l'application lors de la réalisation de la tâche. Un modèle est quant à lui un objet du même type que l'objet à produire par la tâche en cours. Il a été défini préalablement par un concepteur ou un utilisateur averti, à des fins de réutilisation. Dans ExploraGraph [Villiot-Leclercq, 2006] par exemple, les enseignants peuvent créer des scénarios pédagogiques à partir de six scénarios types qu'ils peuvent adapter à leurs besoins.

1.2.3. Approches d'assistance

Nous venons de voir que les techniques permettant de répondre aux besoins des utilisateurs les plus variés sont nombreuses et peuvent être mises en œuvre grâce à différentes approches d'assistance qui peuvent être complémentaires. Nous présentons dans cette section une vue d'ensemble des approches que nous avons identifiées.

Un **manuel d'aide** Ⓐ rassemble un ensemble d'instructions ou d'informations concernant un produit, souvent présenté sous forme textuelle. Ces manuels sont fréquemment utilisés pour l'assistance aux utilisateurs de logiciels [Anacleto et al., 2007]. Souvent accessibles en ligne pour les applications informatiques, ils peuvent être consultés à tout moment par l'utilisateur selon ses besoins. Une aide en ligne peut ainsi être destinée à des employés en formation [Anacleto et al., 2007], l'aide fournie s'appuie souvent sur des exemples de problèmes dont la solution est connue par le système. Les manuels d'aide proposés par Microsoft pour assister les utilisateurs d'Office 2010 [Office_2010, 2014] sont par exemple constitués de démonstrations sous forme de vidéos et de messages que l'utilisateur peut consulter lorsqu'il le souhaite. Les manuels d'aide, souvent adaptés aux utilisateurs de niveau intermédiaire, ne conviennent pas toujours aux utilisateurs novices ou experts. En effet, un utilisateur novice peut ne pas se rendre compte qu'il a besoin d'aide, ou bien être incapable d'identifier ou de formuler son besoin d'aide, ce qui limite l'efficacité du manuel d'aide [Kantner et Rusinsky, 1998]. Par ailleurs, un utilisateur expérimenté peut trouver fastidieuse la recherche d'une information qui lui manque parmi de nombreuses informations qu'il connait déjà [Mattews et al., 2000].

L'**aide contextualisée** Ⓑ constitue une réponse à ces problèmes. Il s'agit en effet d'une assistance directement liée au contexte dans lequel se trouve un utilisateur [Capobianco et Carbonell, 2001] [Mattews et al., 2000]. Ainsi, dans ExploraGraph, les scénarios pédagogiques intègrent une aide contextualisée sous forme de messages d'assistance pour les apprenants si l'enseignant le souhaite : cette assistance peut tenir compte des réponses déjà fournies par l'apprenant.

Un **système conseiller** Ⓒ est « un système qui propose une aide active à l'utilisateur d'un logiciel particulier, aide fondée sur une analyse des actions et des productions de l'utilisateur » [Paquette et Tchounikine, 2002]. Le système conseiller proposé par [Richard et Tchounikine, 2004] pour les utilisateurs de sites web affiche des messages et conseils sous forme de fenêtres pop-up. Celui proposé par [Conejo et al., 2005] assiste des apprenants utilisateurs d'un site web de QCM par l'affichage de bulles d'aide. Les systèmes conseillers proposent souvent à l'utilisateur de choisir entre plusieurs modes d'assistance qui peuvent notamment être catégorisés par le degré d'intervention du système d'assistance [Paquette et al., 1996].

Les **agents conversationnels** Ⓓ sont des personnifications de la fonction d'assistance qui proposent une aide à l'utilisateur. Leur but est d'inciter au dialogue et de le faciliter par effet de sympathie [Leray et Sansonnet, 2007]. Ils peuvent avoir plusieurs apparences [Bouchet, 2006]: textuelle [Stevens et Collins, 1977] [Mattews et al., 2000] ou graphique [Dufresne et

Promtep, 2006]. Les agents graphiques peuvent être animés et exprimer des émotions (empathie, surprise, mécontentement…), ce qui facilite la communication avec l'utilisateur et augmente la crédibilité de l'agent [Johnson et al., 2000]. Parmi les agents conversationnels animés, l'un des plus tristement célèbres est Clippy [Swartz, 2003], l'assistant sous forme de trombone proposé par Microsoft. Très bien accueilli par les utilisateurs dans un premier temps, il est rapidement devenu très impopulaire en raison du manque de pertinence de l'assistance qu'il fournit et de son apparition intempestive et non contrôlable par l'utilisateur. D'autres agents conversationnels remportent cependant un plus grand succès, comme l'assistant de Micame [Réty et al., 2003], un hypermédia adaptatif. Il existe également des agents conversationnels capables de comprendre des questions en langage naturel oral, ce qui peut faciliter le dialogue entre l'utilisateur et l'assistance [Ferguson et al., 2009].

Un **système de recommandations** Ⓔ est un « système qui produit des recommandations individualisées ou qui a pour effet de guider l'utilisateur de manière personnalisée vers des objets utiles ou intéressants parmi un large choix d'options possibles » [Burke, 2002]. Les systèmes de recommandations sont souvent mis en œuvre dans des applications commerciales sur internet, pour suggérer aux utilisateurs des produits susceptibles de les intéresser. Reviewer's Assistant [Dong et al., 2012] est quant à lui un système de recommandation qui vise à aider les clients de sites web à rédiger des critiques de qualité, en leur suggérant des mots clé. Ils peuvent également aider un utilisateur à gagner du temps en le guidant vers des choix pertinents, comme dans Spyglass [Rushing et al., 2009] destinés aux programmeurs. Les recommandations peuvent être transmises dans des fenêtres pop-up qui apparaissent pendant l'activité (et de manière moins fréquente via des agents animés, comme dans le système de recommandations pour la planification de visites touristiques CT-Planner [Kurataa et Hara, 2014]), ou dans une zone de l'application dédiée aux recommandations et dont le contenu évolue durant l'activité, comme pour le système de recommandation basé sur les traces de Wanaclip [Zarka et al., 2012].

Un **tutoriel** Ⓕ est un outil pédagogique permettant à un utilisateur de se former de manière autonome à l'utilisation d'un logiciel. Un tutoriel peut se présenter sous diverses formes (application, vidéo, texte descriptif…) et contient des explications détaillées pas à pas. Ils sont fréquemment utilisés pour assister les utilisateurs, en particulier pour des logiciels grand public. Un tutoriel peut être intégré dans une application ou être indépendant : il existe ainsi de nombreux tutoriels en ligne, qu'ils soient créés par les concepteurs de l'application concernée ou par des utilisateurs désireux de faire partager leur expérience. Le site [Tutoriels_animes, 2014] en propose de nombreux exemples, pour des applications variées. Vismod, un EIAH permettant la visualisation de profils d'apprenants [Zapata-Rivera et Greer, 2004], offre aux apprenants un tutoriel intégré pour les aider à comprendre la représentation de leur profil et à utiliser le logiciel. Les MOOC (Massively Online Open Courses) [Gilliot et al., 2013] proposent des supports de cours en ligne, qui peuvent notamment intégrer des tutoriels.

Une **interface adaptative** Ⓖ est « capable d'adapter son comportement aux besoins, capacités et préférences de l'utilisateur courant pendant l'interaction, grâce à ses capacités de perception et d'interprétation de l'interaction et de son contexte » [Simonin et Carbonell,

2007]. Elles permettent d'aider l'utilisateur en personnalisant l'interface de l'application, comme le fait par exemple Pixed [Mille et Héraud, 2009], qui adapte des documents pédagogiques pour les apprenants, ou les menus adaptatifs d'Office 2003 [McGrenere et al., 2002 ; Office_2003, 2014], qui placent en tête dans les menus les items les plus fréquemment utilisés par l'utilisateur. Bien qu'une telle adaptation de l'interface puisse permettre à l'utilisateur de gagner du temps en accédant plus rapidement aux options qu'il utilise le plus, la modification de l'interface peut aussi le perturber s'il ne parvient plus à retrouver une option qui a été déplacée.

La **réutilisation de l'expérience** Ⓗ peut constituer une approche d'assistance en permettant à un utilisateur de réutiliser des actions passées [Mille et al., 2006]. Ces actions peuvent avoir été effectuées lors d'une utilisation antérieure de l'application, soit par l'utilisateur lui-même, soit par un autre utilisateur dans une situation analogue. Ainsi, dans Wanaclip, les actions passées d'un utilisateur sont utilisées par le système d'assistance pour lui recommander de nouvelles actions [Zarka et al., 2010]. Pixed exploite quant à lui les actions passées des apprenants pour leur proposer de nouvelles ressources

Les **communautés de pratiques** ⓘ désignent un ensemble de personnes qui partagent des pratiques communes, rassemblées par des relations informelles [Lave et Wenger, 1991], par une expertise partagée ou un centre d'intérêt commun [Wenger et Snyder, 2000]. Par exemple, la communauté Sésamath [Sésamath, 2012] réunit des enseignants de mathématiques, et leur permet notamment d'échanger des cours et des exercices. Les communautés de pratiques peuvent constituer une approche d'assistance en regroupant les utilisateurs d'une même application et en leur permettant de s'entraider, par exemple via des forums. Dans un tout autre contexte, le site web de Lancôme [Lancome, 2014] propose à ses utilisateurs une fenêtre de *chat* d'assistance dans laquelle ils peuvent non seulement poser une question, mais aussi répondre à une question posée par un autre utilisateur. Même lorsque l'utilisateur n'est pas personnellement actif dans un *chat* ou un forum, il peut trouver des informations pertinentes en consultant les questions posées et les réponses qui ont été données par d'autres utilisateurs.

1.3. Conclusion

Nous avons vu dans ce chapitre qu'il existe de nombreuses approches permettant d'assister les utilisateurs d'applications informatiques. Certaines semblent plus adaptées à un type d'utilisateurs (novice, occasionnel, standard, expert) ou à un besoin particulier d'assistance (découverte du système, réalisation d'une tâche…). Par exemple, un tutoriel présentant globalement une application conviendra à un utilisateur novice qui souhaite découvrir cette application, mais ne sera pas adapté à un utilisateur expert qui a besoin d'une aide plus précise.

Chapitre 2. Systèmes d'assistance non intégrés à l'application-cible

Objectif du chapitre

Étudier les forces et les faiblesses des assistances existantes compatibles avec une approche épiphyte.

Points clés

En réalisant notre état de l'art de l'assistance à l'utilisateur d'applications informatiques, nous nous sommes plus particulièrement intéressée aux systèmes d'assistance compatibles avec une approche épiphytes, c'est-à-dire non intégrés à leur application-cible. Nous en avons identifié deux catégories : les systèmes d'assistance extérieurs à l'application et les systèmes d'assistance épiphytes.

Les systèmes d'assistance extérieurs à l'application-cible, comme les tutoriels, les forums, les manuels d'aide et les foires aux questions, ne possèdent pas de lien avec leur application-cible. Ainsi, l'utilisateur doit lui-même identifier son problème avant de consulter ces systèmes d'assistance. Ils constituent néanmoins d'intéressantes ressources, qui pourraient être proposées à l'utilisateur lorsqu'un besoin d'assistance est détecté par un système d'assistance qui serait lié à l'application-cible.

Les systèmes d'assistance épiphytes paraissent intégrés à leur application-cible aux yeux de l'utilisateur final. Les systèmes d'assistance épiphytes existants ne sont toutefois pas génériques, mais spécifiques, soit à un environnement donné, soit aux applications web. De plus, ils ne mettent pas en œuvre toutes les techniques d'assistance identifiées dans notre typologie de l'assistance à l'utilisateur (cf. Section 1.2.2).

Contribution

- Analyse des systèmes d'assistance compatibles avec une approche épiphyte

Dans cette section, nous présentons notre état de l'art relatif aux systèmes d'assistance compatibles avec une approche épiphyte. Cet état de l'art a été fait indépendamment des domaines des applications-cibles. Il en ressort que les systèmes d'assistance compatibles avec une approche épiphytes sont de deux types : les systèmes d'assistance indépendants de l'application-cible et les systèmes d'assistance épiphytes. La distinction entre ces deux catégories tient au fait qu'un système d'assistance épiphyte est greffé sur son application-cible. Bien que l'application-cible soit indépendante de l'assistance épiphyte qui lui est ajoutée, cette assistance est liée à l'application-cible, ce qui conduit l'utilisateur à le considérer comme intégré à cette application. Au contraire, dans le cas d'un système d'assistance non épiphyte, l'utilisateur a conscience d'utiliser un système indépendant de l'application-cible.

2.1. Systèmes d'assistance indépendants de l'application-cible

Une assistance externe peut être mise en place *a posteriori* pour des applications existantes, sans les modifier et sans accès à leur code source. Une assistance externe peut prendre plusieurs formes complémentaires : un tutoriel, un forum, une foire aux questions ou un manuel d'aide. Son concepteur n'est pas nécessairement le développeur de l'application : il peut par exemple s'agir d'un utilisateur expert qui souhaite faire bénéficier d'autres personnes de son expérience dans le cadre d'une communauté d'utilisateurs. À l'exception de notion de programmation HTLM pour la mise en ligne éventuelle de ces assistances, aucune compétence en programmation n'est généralement requise. Toutefois, des outils auteurs existent pour faciliter la tâche du concepteur de l'assistance. À titre d'exemple, nous présentons par la suite des outils permettant de faciliter la mise en place de telles assistances.

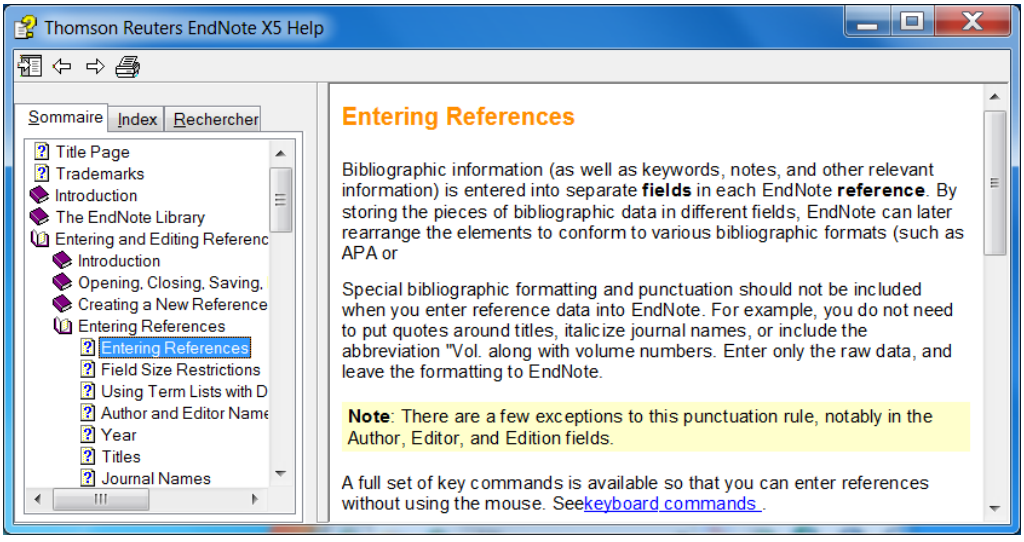


Fig. 2-1 – Manuel d'aide de l'application de gestion bibliographique EndNote [EndNote, 2014]

Les formats Winhelp [WinHelp, 2014] et Microsoft Compressed HTLM permettent d'ajouter à des applications un manuel d'aide, en ligne ou en local. Il s'agit d'une assistance à

la demande de l'utilisateur final, accessible à tout moment en appuyant sur la touche F1. Ces manuels d'aide se présentent sous la forme de pages structurées et de définitions, et proposent une fonctionnalité de recherche par mots clés (cf. Fig. 2-3). Par rapport à un manuel d'aide classique et totalement externe à l'application, l'avantage ce type de manuels est que l'aide fournie est contextualisée : en pointant avec le curseur un bouton donné avant d'appuyer sur la touche F1 de demande d'aide, l'utilisateur final recevra de l'aide relative à la fonctionnalité correspondante. De plus, l'association entre la touche F1 et la demande d'aide est très répandue dans les applications existantes sous Windows, ce qui rend les utilisateurs finaux plus susceptibles d'y avoir recours. Néanmoins, l'utilisateur peut avoir du mal à trouver l'information qu'il cherche parmi la grande quantité d'informations généralement présente dans ce type de manuels [Kantner et Rusinsky, 1998]. De plus, pour bénéficier d'un tel manuel d'aide contextualisé, les applications doivent avoir été conçues en prévision de l'ajout de ce manuel : seules les exécutables Windows qui exploitent la bibliothèque MFC (Microsoft Foundation Class) sont compatibles. Par ailleurs, la création d'un tel manuel d'aide contextualisée peut être longue, bien qu'il existe des outils auteurs pour en faciliter la création, tels que [HelpScribble, 2014] pour WinHelp ou PowerCHM [PowerCHM, 2014], FlyHelp [FlyHelp, 2014] et HTML Help COM Assistant [html-help-com-assistant, 2014] pour Microsoft Compressed HTLM.

2.2. Systèmes d'assistance épiphytes

L'assistance épiphyte a fait l'objet de plusieurs travaux de recherche : nous présentons dans cette section chacun de ces travaux. Bien que notre étude ait été menée indépendamment du domaine dont l'application-cible est issue, les travaux que nous avons identifiés sont tous issus du domaine des EIAH. Ceci s'explique selon nous par le fait que, dans ce domaine, un intérêt particulier est apporté à l'assistance à l'utilisateur.

2.2.1. Epitalk

Le système Epitalk [Paquette et al., 1996] permet la spécification d'un système conseiller sous forme de graphe conceptuel, puis son exécution dans n'importe quelle application-cible développée en Smalltalk-80.

Pour la phase de spécification de l'assistance, Epitalk propose des outils pour décrire l'application-cible sous la forme d'un graphe organisationnel et pour décrire le système conseiller sous forme d'un graphe de tâches (cf. Fig. 2-2), qui permettra par la suite de générer automatiquement ce système conseiller lors de l'exécution de l'assistance. Le concepteur de l'assistance peut spécifier les conseils à donner, et leur associer des métadonnées. Ainsi, les conseils peuvent être associés à un niveau d'importance qui permettra au système conseiller de sélectionner le conseil le plus pertinent lorsque plusieurs conseils peuvent être donnés en même temps. Les conseils peuvent également être associés à un nombre maximal d'utilisations avant leur désactivation temporaire. Le concepteur peut également déterminer si un conseil sera donné à l'initiative de l'utilisateur final ou à l'initiative du système conseiller. Dans Epitalk, un conseil se présente sous la forme d'une fenêtre pop-up, qui peut être accompagnée d'une alarme sonore pour attirer l'attention de

l'utilisateur. Le contenu d'un conseil peut être contextualisé avec des informations spécifiées par le concepteur de l'assistance.

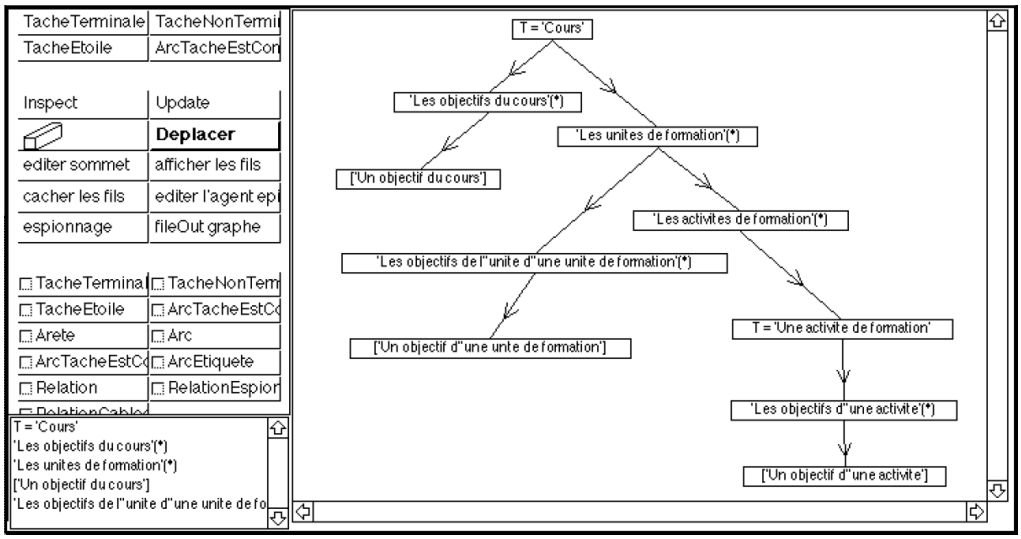


Fig. 2-2. Édition d'un graphe de tâches dans Epitalk

Pour permettre la phase d'exécution de l'assistance, Epitalk propose des outils, nommés « espions » pour surveiller de manière épiphyte l'application-cible développée en Smalltalk, par l'observation des messages qui transitent entre les objets. Les « espions » sont paramétrés par le concepteur de l'assistance, qui définit quels composants de l'application-cible doivent être surveillés, ainsi que le type de messages à surveiller. Ces messages concernent notamment les interactions entre l'utilisateur et l'interface de l'application-cible, par exemple les clics.

De plus, Epitalk génère automatiquement un graphe d'agents épiphytes, isomorphe au graphe des tâches : un agent épiphyte est associé à chaque tâche. Par défaut, les agents automatiquement créés par Epitalk ne font rien. Le concepteur peut définir le comportement d'un agent, notamment en lui associant une base de règles respectant le formalisme NéOpus [Pachet, 1995], une extension de Smalltalk-80. Lors de l'exécution de l'assistance, les agents épiphytes sont informés des événements survenant dans l'application-cible par les « espions » et déclenchent l'affichage des conseils selon les règles spécifiées par le concepteur de l'assistance.

La principale force de l'approche mise en œuvre dans Epitalk est de rendre possible la mise en place d'un système conseiller dans une application-cible existante qui n'a pas été conçue spécifiquement pour permettre l'ajout d'un tel système conseiller. De plus, les systèmes conseillers mis en place via Epitalk sont greffés sur l'application-cible, ce qui permet de proposer une assistance fortement contextualisée. Néanmoins, Epitalk ne concerne que les applications développées en Smalltalk-80. De plus, des connaissances en programmation en Smalltalk-80 sont requises pour le concepteur de l'assistance, même si Epitalk limite l'usage de la programmation.

Les systèmes conseillers mis en place via Epitalk peuvent proposer à l'utilisateur final des conseils sous forme de messages textuels. Ces conseils peuvent porter sur les connaissances

manipulées dans l'application-cible comme sur des points de méthodologie. Les conseils peuvent être déclenchés de manière proactive à l'initiative du système d'assistance ou de manière réactive à l'initiative de l'utilisateur final.

2.2.2. ExploraGraph

L'environnement collaboratif d'apprentissage à distance ExploraGraph [Dufresne, 2001] est un hypermédia adaptatif. Il a pour but de faciliter les activités pédagogiques de l'apprenant, en le guidant à travers la navigation dans les graphes conceptuels d'activités, en lui permettant de demander et de recevoir de l'aide et en facilitant l'appropriation de la matière enseignée et la gestion de l'apprentissage. L'environnement ExploraGraph est composé de trois outils : un éditeur à destination des concepteurs pédagogiques pour la description des contenus pédagogiques ; un éditeur à destination des concepteurs d'assistance pour la description des règles d'assistance ; et un navigateur à destination des apprenants pour la réalisation des activités pédagogiques. L'assistance définie à l'aide de l'éditeur de règles d'assistance peut être exécutée dans le navigateur apprenant, pour n'importe quel scénario d'ExploraGraph. En revanche, cette assistance ne peut pas être exécutée dans une application autre qu'ExploraGraph.

Côté apprenants, ExploraGraph propose un graphe conceptuel dans lequel l'apprenant peut naviguer. L'exemple de la Fig. 2-3 montre un graphe conceptuel dont les nœuds sont des activités liées à un cours de design de sites web [Dufresne et Promtep, 2006]. Sous chaque nœud du graphe, deux barres de progression sont affichées : elles représentent la progression du groupe et celle de l'apprenant dans l'activité correspondante ; ces informations étant issues du profil du groupe ou du profil de l'apprenant. Les arcs du graphe sont associés à un code couleur et représentent un lien entre les activités (précédence, production ou de composition par exemple).

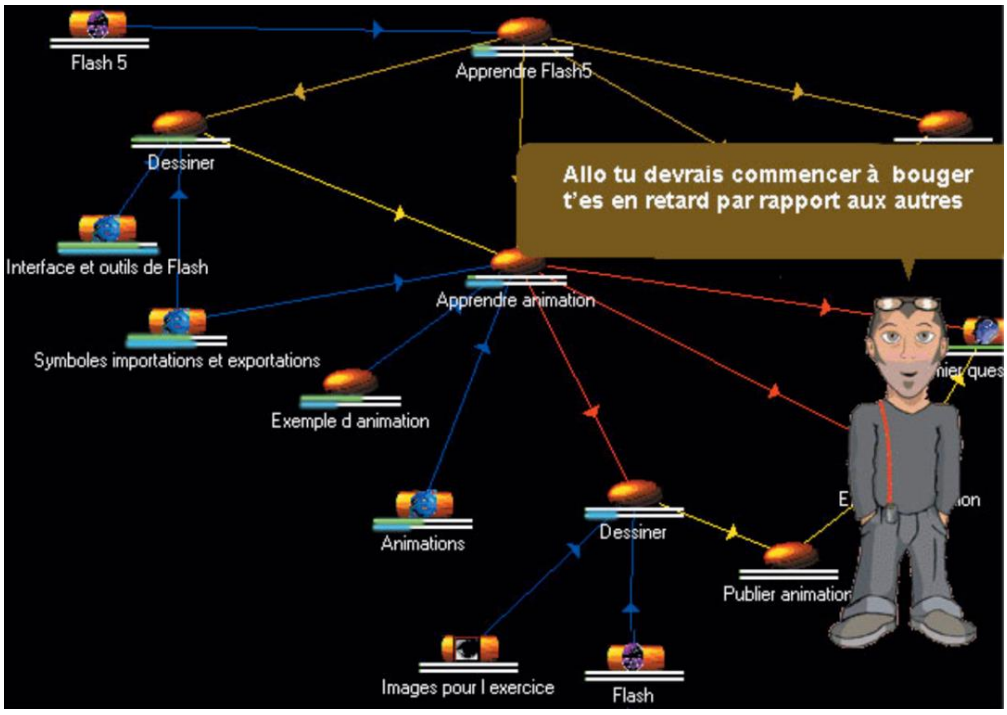


Fig. 2-3. Navigateur d'ExploraGraph : exécution d'une action d'assistance de type message

Côté concepteur pédagogique, l'éditeur de règles d'assistance d'ExploraGraph permet à un concepteur pédagogique de définir sous forme d'un ensemble de règles l'assistance qui sera proposée à l'apprenant dans le navigateur. Les règles d'assistance exploitées dans ExploraGraph sont de la forme <événement déclencheur, condition de déclenchement, actions d'assistance> [Dufresne et al., 2003]. Chaque règle est de plus associée à un type de conseil (présentation, explication, rappel, feedback, motivation) qui définit l'objectif de la règle d'assistance.

Les événements déclencheurs permettent de contextualiser l'assistance, ils peuvent concerner l'ouverture ou la fermeture d'un graphe, le clic sur un nœud et l'inactivité de l'apprenant pendant une durée donnée. Les conditions de déclenchement permettent de personnaliser l'assistance, elles peuvent concerner le profil de l'apprenant ou le profil de groupe, elles peuvent être relatives à une période (pour imposer par exemple qu'une action d'assistance ne puisse être déclenchée que dans une période donnée), ou être relative à une question posée directement à l'apprenant (pour lui demander s'il souhaite de l'assistance par exemple). Les actions d'assistance permettent de fournir à l'apprenant de l'aide, dans ExploraGraph elles sont principalement de type *message* et *animation* : un agent animé MSagents peut afficher ou lire un message, se déplacer à l'écran, mettre en valeur un point donné de l'environnement ou réaliser des animations (par exemple saluer, sourire ou applaudir). Le système d'assistance peut également réaliser des actions de type *action automatisée*, pour ouvrir un graphe de l'environnement ou cliquer sur un nœud à la place de l'apprenant, ou de type *modification du profil* de l'apprenant ou d'un groupe d'apprenants. Par exemple, sur la Fig. 2-3, une action d'assistance de type *message* affiché par un agent animé est déclenchée car, à une date donnée, le profil de l'apprenant indique qu'il a moins progressé que la moyenne du groupe dans certains activités.

La Fig. 2-4 présente une capture d'écran de l'éditeur de règles d'assistance d'ExploraGraph. Cette interface permet au concepteur pédagogique de définir des actions d'assistance. Dans cet exemple, le concepteur pédagogique définit une action de type message : le message « bonjour, bonne chance c'est dur aujourd'hui » sera lu par l'agent animé « Prof ».

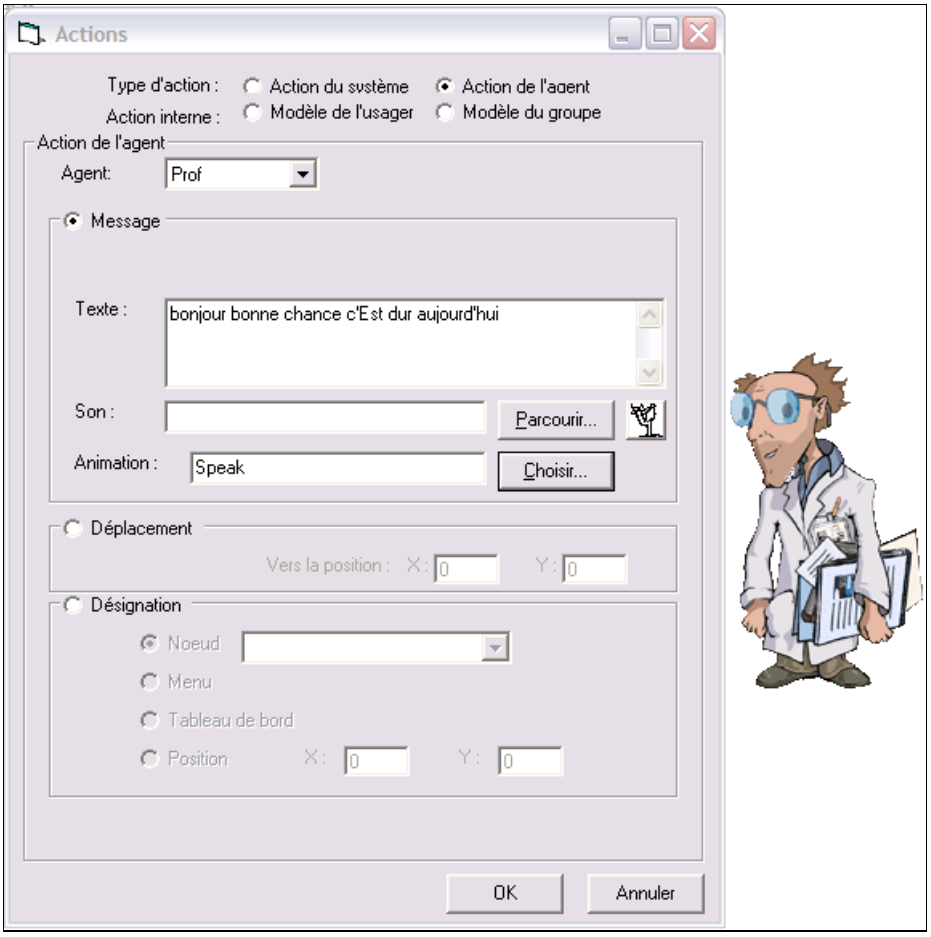


Fig. 2-4. Éditeur de règles d'assistance d'ExploraGraph : définition d'une action de type message

L'environnement ExploraGraph propose par ailleurs à l'apprenant un tableau de bord qui lui permet de spécifier ses intentions (explorer une matière ou un scénario donné, planifier une tâche ou faire le point). À partir de ce tableau de bord, l'apprenant peut demander de l'aide en fonction de ses intentions ou définir des préférences vis-à-vis de l'aide qui lui est fournie. Ainsi, l'apprenant peut ajuster le niveau d'aide qui lui sera proposée à l'initiative du système d'assistance, en définissant un seuil d'aide souhaitée qui peut être utilisé comme condition de déclenchement des règles d'assistance ; l'apprenant peut également définir un personnage préféré pour les agents animés (Prof, la religieuse, le coach ou le hacker) [Dufresne et Promtep, 2006].

L'approche d'assistance à base de règles mise en œuvre dans ExploraGraph est riche et intéressante, elle permet notamment une contextualisation et une personnalisation fine de l'assistance. Cependant, la portée de cette approche est limitée car elle est spécifique à

l'environnement Exploragraph : seuls les scénarios pédagogiques de cet environnement sont concernés par cette assistance.

2.2.3. Telos

L'environnement web Telos permet la définition de scénarios d'activités ou de scénarios pédagogiques multi-acteurs s'appuyant sur des ontologies [Paquette, 2012]. Telos est composé de cinq principaux outils : un éditeur graphique de scénarios, un éditeur de connaissances, un éditeur d'ontologie, un gestionnaire de ressources et un gestionnaire de tâches qui permet l'exécution d'un scénario en générant une interface graphique à partir de laquelle les acteurs du scénario pourront réaliser les activités du scénario.

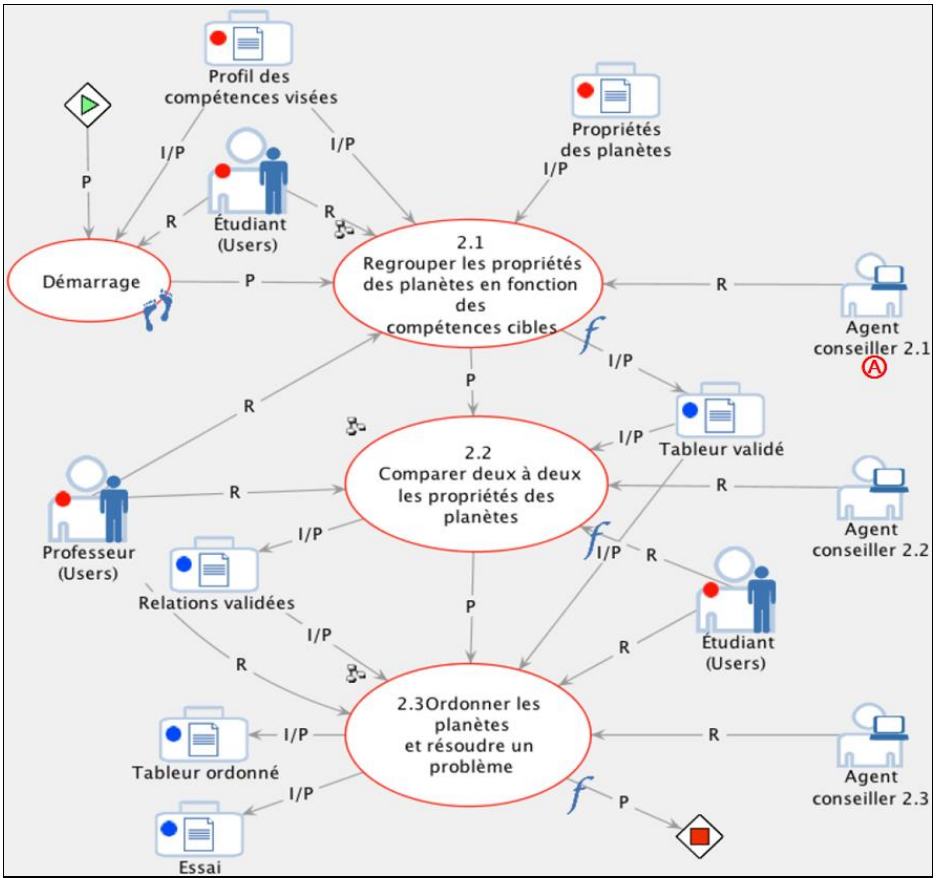


Fig. 2-5. Exemple de scénario pédagogique définit en MOT dans Telos

L'éditeur de scénarios de Telos utilise le langage graphique MOT [Paquette et al., 2008]. Un exemple de scénario pédagogique défini dans Telos est donné en Fig. 2-5. À partir de l'éditeur de scénarios de Telos, il est possible d'ajouter dans un scénario un agent assistant, ou agent conseiller, lié à une tâche du scénario. Dans l'exemple de scénario donné par la Fig. 2-5, on peut voir par exemple qu'un agent conseiller a été associé à la tâche « Regrouper les propriétés des planètes en fonction des compétences cibles » (cf. A Fig. 2-5).

Les agents conseillers utilisés dans Telos sont définis par le concepteur du scénario via un ensemble de règles de la forme <événement déclencheur, condition de déclenchement,

actions d'assistance>, comme dans ExploraGraph. Chaque règle d'assistance est associée à l'acteur du scénario auquel l'agent conseiller fournira l'assistance, par exemple un apprenant ou un enseignant.

Les événements déclencheurs permettent de contextualiser l'assistance et sont liés à une tâche ou à une ressource du scénario, par exemple « tâche complétée » ou « ressource ouverte ». Les conditions de déclenchement permettent de personnaliser l'assistance en fonction des connaissances et compétences de l'acteur ciblé par la règle d'assistance. Par exemple, il est possible de déclencher une règle d'assistance uniquement pour les apprenants dont le niveau est plus faible que le niveau attendu dans une compétence prérequise pour la tâche à laquelle l'agent conseiller est lié ; en effet ces apprenants peuvent se trouver en difficulté.

Les actions d'assistance peuvent être de type *modification du profil* de l'acteur concerné, par exemple pour mettre à jour les compétences d'un apprenant suite à une activité, elles peuvent également être de type *message*, affiché dans une fenêtre pop-up, par exemple pour donner un conseil à un apprenant ou notifier à un enseignant qu'un apprenant est en difficulté.

La Fig. 2-6 présente une capture de l'écran d'édition de règles d'assistance inclus dans l'éditeur de scénarios de Telos. Sur cet exemple, le concepteur définit une règle d'assistance qui sera déclenchée lorsque l'acteur concerné par l'assistance, un étudiant, aura complété l'activité « Regrouper les propriétés des planètes en fonction des compétences cibles », à laquelle l'agent conseiller est associé. La règle ne sera déclenchée que pour un étudiant dont une compétence est plus faible que la compétence visée par l'activité. Dans ce cas, une notification sera envoyée à un autre acteur du scénario, un facilitateur, par exemple un enseignant. Cette notification a pour contenu « Veuillez donner un conseil à l'étudiant en utilisant la ressource suivante », avec un lien vers la ressource « Le système solaire ».

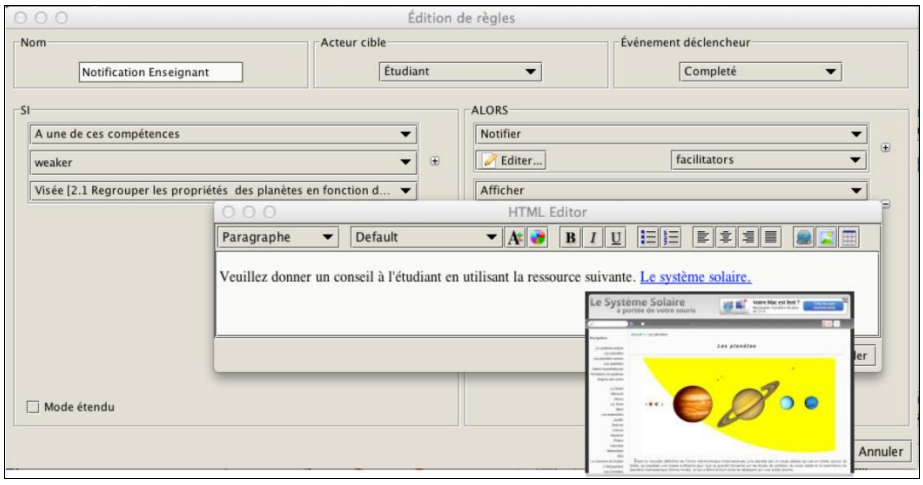


Fig. 2-6. Édition de règles d'assistance dans l'éditeur de scénarios de Telos

L'approche d'assistance à base de règles mise en œuvre dans Telos est particulièrement intéressante, notamment pour la richesse des actions d'assistance proposées et pour les possibilités de personnalisation qu'elle offre au concepteur de l'assistance. Cependant, tout

comme l'approche proposée dans Exploragraph, la portée de cette approche est limitée car elle est spécifique à un unique environnement : Telos.

2.2.4. iFrimousse

Le modèle CAMELEON [Carlier et Renault, 2010], mis en œuvre dans iFrimousse, permet quant à lui d'ajouter un agent conversationnel animé, sous la forme d'un compagnon, dans une application web, grâce à des balises insérées dans la page web. Le compagnon est capable de se déplacer au niveau de l'élément de la page web associée à une balise donnée, (par exemple pour montrer un bouton de l'interface), de réaliser des animations permettant l'expression d'émotions (par exemple sourire) et de communiquer avec l'utilisateur par des messages textuels. En effet, le compagnon peut répondre à une question saisie par l'utilisateur en langage naturel. Sur l'exemple donné par la Fig. 2-7, le compagnon (qui a l'apparence d'un petit champignon) s'est déplacé depuis sa position courante, une zone dans laquelle l'utilisateur peut saisir des questions, pour se positionner près de la zone de connexion.

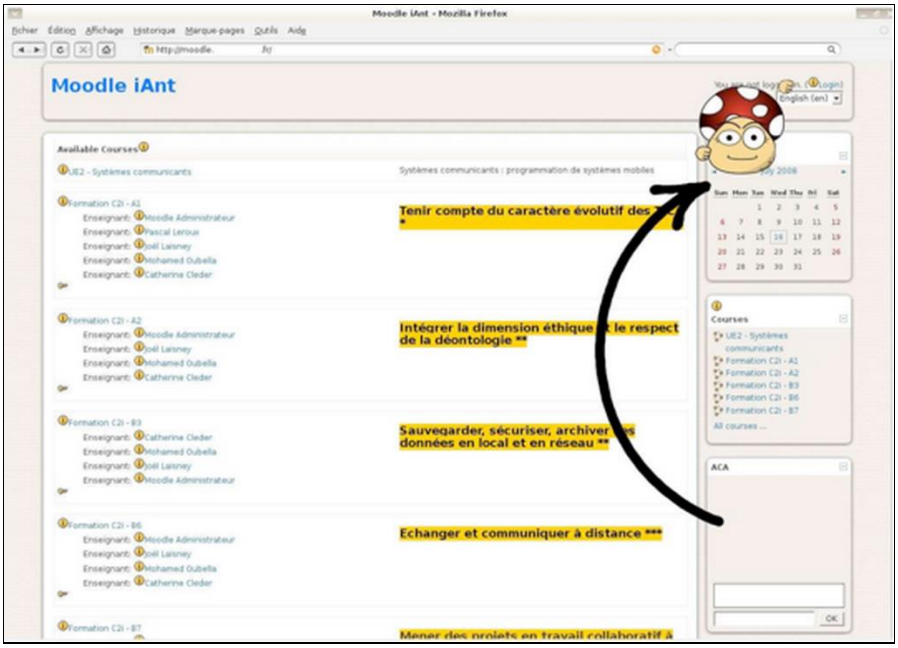


Fig. 2-7. Exemple d'animation d'un agent conversationnel animé développé selon le modèle CAMELEON

Cette approche, basée sur l'ajout de balises, n'est cependant pas compatible avec des applications bureau, ce qui limite sa portée aux seules applications web. De plus, les actions d'assistance proposées (les messages, les mises en valeur de composants et les animations) impliquent toutes un agent animé, ce qui limite les choix du concepteur d'assistance et les possibilités de personnalisation. De plus, les agents animés peuvent ne pas être appréciés par certains utilisateurs finaux, en particulier les adultes.

2.2.5. Systèmes conseillers épiphytes pour le web

L'approche proposée par Bruno Richard [Richard, 2008] permet l'élaboration d'un système conseiller pour une application web, avec une architecture épiphyte. Le système conseiller exploite un modèle de tâche sous la forme d'un graphe modélisant les différentes actions de navigation que l'utilisateur peut effectuer au sein de l'application web. Sur ce graphe, les nœuds représentent les pages et documents proposés par l'application web et les arcs représentent les liens permettant d'accéder à une autre page ou document du site. Le modèle d'utilisation du système conseiller épiphyte correspond au modèle de tâche de l'application web dans lequel des conseils sont associés à certains arcs : ces conseils seront déclenchés pour aider l'utilisateur lorsque celui-ci cliquera sur le lien correspondant à l'arc. Dans cette approche, les conseils ont pour but d'apporter directement de l'information supplémentaire à l'utilisateur, de simplifier sa navigation ou de le guider. Les conseils sont présentés à l'utilisateur sous forme d'une fenêtre pop-up au contenu textuel pouvant inclure des liens internes au site web ou vers une ressource externe. À titre d'exemple, la Fig. 2-8 présente une capture d'écran du site EducaSource[6] pendant l'affichage d'un message d'aide. Le déclenchement de conseils peut être conditionné en fonction de l'historique de navigation de l'utilisateur.

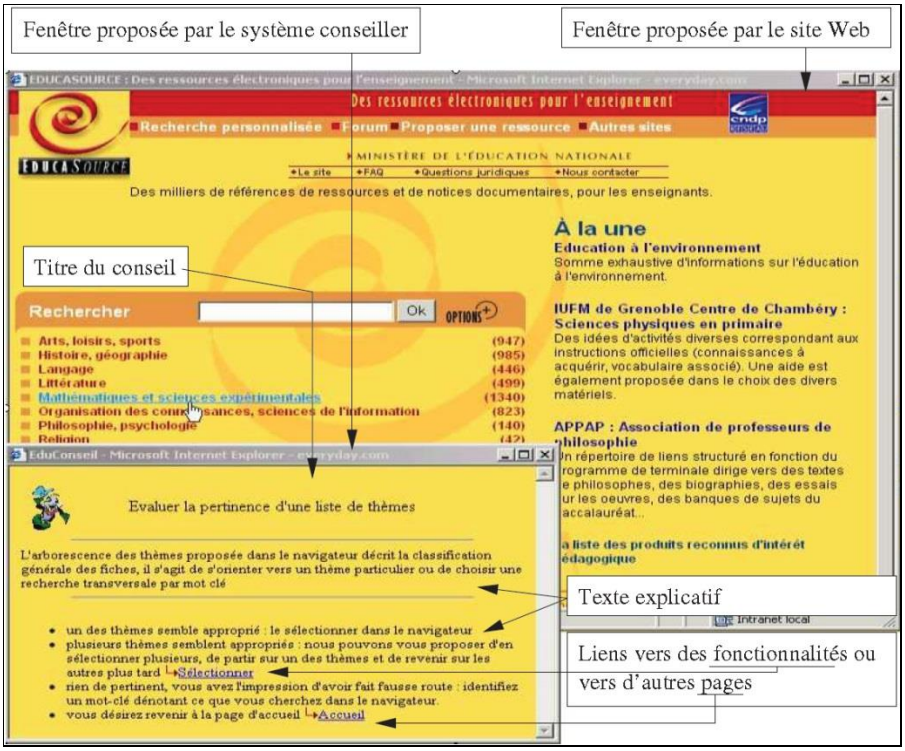


Fig. 2-8. Exemple d'un message d'aide pour le site EducaSource

[6] http://www.educasources.education.fr/

Cette approche intéressante concerne cependant uniquement les applications web. De plus, le déclenchement des messages d'aide peut uniquement avoir lieu lors d'un clic sur un lien. Il n'est pas possible avec cette approche d'associer un conseil au passage de la souris sur une image ou au clic sur un bouton donné par exemple.

2.3. Conclusion

Les différentes approches présentées dans ce chapitre permettent la mise en place d'assistance dans des applications existantes, sans avoir à les modifier.

Dans la section 2.1, nous avons identifié quatre catégories d'approches d'assistance extérieure à l'application-cible (forum, foire aux questions, tutoriel, manuel d'aide). En conséquence, l'utilisateur doit être capable d'identifier son besoin d'assistance avant de se tourner vers ce type d'assistance. Une telle démarche requiert un certain recul de la part de l'utilisateur vis-à-vis de son utilisation de l'application-cible. Cela demande également à l'utilisateur d'avoir la volonté de trouver une réponse à son besoin d'assistance. Or, une situation problématique peut entraîner une démotivation de la part de l'utilisateur final, qui peut aboutir à un abandon de l'application-cible si celle-ci ne propose spontanément aucune solution pour résoudre la situation problématique. Une alternative consiste à identifier le besoin de l'utilisateur final et à lui proposer de manière proactive une ressource d'assistance répondant à ce besoin. Une ressource peut être extérieure à l'application-cible, comme un forum ou un manuel d'aide par exemple.

Dans la section 2.2, nous avons présenté les approches d'assistance épiphytes à l'application-cible que nous avons identifiées. Ainsi, bien que l'assistance épiphyte paraisse intégrée à l'application-cible aux yeux de l'utilisateur, ce n'est pas le cas d'un point de vue logiciel. La force d'une telle assistance, qui paraît être intégrée à l'application-cible, est qu'elle n'entraîne pas nécessairement une interruption de la tâche de l'utilisateur, comme cela est le cas lorsque l'utilisateur consulte une assistance extérieure à l'application.

Ces approches ne peuvent cependant pas être utilisées dans n'importe quelle application. Elles sont en effet spécifiques à un environnement donné, comme c'est le cas pour les approches d'assistance proposées dans Telos et ExploraGraph, à un langage donné, comme c'est le cas d'Epitalk, ou au web, comme c'est le cas pour iFrimousse et pour l'approche proposée par Bruno Richard. À notre connaissance, il n'existe aucune approche d'assistance épiphyte qui soit générique. De plus, pour chacune de ces approches, les actions d'assistance proposées sont de types peu variés : principalement des messages textuels ou des actions d'agents animés.

Dans le cadre du projet AGATE, nous nous sommes intéressés à l'adjonction *a posteriori* de systèmes d'assistance dans des applications existantes les plus variées, sans que ces applications soient spécifiques à un domaine ou à un environnement. De plus, nous souhaitons proposer des actions d'assistance très variées, pour correspondre aux désirs des concepteurs d'assistance et pour permettre une adaptation fine de l'assistance en fonction des spécificités et de l'état de l'application-cible, ainsi que des spécificités des utilisateurs finaux.

Chapitre 3. Surveillance d'applications informatiques

Objectif du chapitre

Étudier les forces et les faiblesses des techniques de surveillance d'une application par rapport à nos besoins dans le cadre de l'assistance à l'utilisateur : détection des événements et identification de leur source, accès aux propriétés des composants et inspection de leur état.

Points clés

Afin de permettre la mise en place d'assistances épiphytes finement contextualisées dans des applications-cibles, il est nécessaire pour le système d'assistance de connaître les événements ayant lieu à l'interface de l'application-cible (comme les clics et les déplacements de la souris). Il est également nécessaire d'accéder à certaines propriétés des composants de l'interface de l'application-cible (par exemple la description d'une image ou l'infobulle d'un bouton), et plus particulièrement à leur état (par exemple la valeur textuelle d'une zone de saisie ou l'état coché ou décoché d'une case à cocher).

Pour cette raison, nous avons réalisé un état de l'art des systèmes de surveillance d'applications informatiques non spécifiquement conçues pour permettre une telle surveillance. Nous avons identifié quatre catégories de systèmes de surveillance que nous présentons dans ce chapitre : les systèmes d'enregistrement de logs, les normes pédagogiques, les systèmes s'appuyant sur la reconnaissance visuelle et les systèmes exploitant des bibliothèques d'accessibilité. C'est cette dernière catégorie de systèmes qui répond le mieux à nos besoins relatifs à la détection des événements ayant lieu à l'interface d'une application-cible, ainsi qu'à l'accès aux propriétés des composants de cette interface et à l'inspection de leur état.

Contribution

- Analyse des techniques de surveillance d'applications informatiques compatibles avec une approche épiphyte

Publication liée à ce chapitre

[Ginon et al., 2013a]

> Ginon B., Champin P.-A. et Jean-Daubias S., Collecting fine-grained use traces in any application without modifying it, Worshop EXPPORT of ICCBR, 2013a.

Pour qu'un système d'assistance épiphyte puisse fournir une assistance contextualisée, il lui est nécessaire de connaître les événements ayant lieu à l'interface de l'application-cible, tels que les clics de l'utilisateur, les déplacements de la souris, ainsi que les modifications de l'état de l'application-cible, tels que la sélection d'items dans des listes déroulantes, la saisie de texte dans des zones de saisie ou le fait de cocher une case à cocher ou un bouton radio. Nous présentons donc dans ce chapitre un état de l'art des systèmes de surveillance d'une application informatique. Les systèmes que nous avons identifiés peuvent être regroupés en quatre catégories : les systèmes d'enregistrement de logs, les normes pédagogiques, les systèmes s'appuyant sur la reconnaissance visuelle et les systèmes exploitant des bibliothèques d'accessibilité.

3.1. Enregistrement de logs

Il existe des systèmes tels que SelfSpy [Gurgeh, 2014], Self Quantified [QuantifedSelf, 2014] ou Wolfram's personal key logging [Wolframe, 2014], qui permettent de recueillir à des fins statistiques des logs, c'est-à-dire des données relatives à l'activité d'un utilisateur. Ces données peuvent concerner les caractères saisis, les programmes utilisés, les recherches effectuées sur le web ou le temps passé sur une activité.

Si ces informations sont intéressantes et peuvent se révéler utiles, elles sont néanmoins insuffisantes pour fournir une assistance finement contextualisée. En effet, grâce à ces informations, il serait par exemple possible de fournir une assistance lorsqu'un utilisateur lance un programme donné, et éventuellement, cette assistance pourrait être proposée uniquement lors du premier lancement de ce programme. En revanche, ces informations ne sont pas suffisamment précises pour fournir de l'assistance lorsque l'utilisateur utilise une fonctionnalité donnée d'un programme, lorsqu'il clique sur un bouton donné ou lorsqu'il coche une case à cocher donnée.

De plus, les informations fournies par ces outils d'enregistrement de logs ne permettent pas systématiquement de connaître l'état d'une application. Par exemple, ces outils permettent de connaître les caractères qui ont été saisis par un utilisateur, mais ne permettent pas de connaître le texte contenu dans une zone de saisie. Cette information peut être déduite si la zone de saisie était initialement vide et que l'utilisateur a saisi le texte avec le clavier. Néanmoins, une zone de saisie peut avoir été remplie initialement avec des valeurs par défaut, que l'utilisateur peut avoir complétées ou modifiées partiellement. De même, le contenu d'une zone de saisie peut avoir été effacé par le programme, par exemple si l'utilisateur a validé un formulaire qui contenait des valeurs incorrectes. Dans ces cas-ci, il est difficile de déduire avec certitude le contenu actuel d'une zone de saisie à partir des informations fournies par ces outils d'enregistrement de logs.

3.2. Normes pédagogiques

SCORM [Bohl et al., 2002 ; Scorm, 2014] et AICC [AICC, 2014] sont des normes qui permettent notamment l'interopérabilité entre les contenus pédagogiques et les plateformes de e-learning qui respectent cette norme, ainsi que la traçabilité de l'activité pédagogique de l'apprenant, c'est-à-dire l'utilisateur en situation d'apprentissage. Le respect

de ces normes permet ainsi de connaître le temps passé par un utilisateur sur une activité pédagogique, le score qu'il a obtenu à un exercice ou sa progression dans un parcours pédagogique. Néanmoins, ces normes concernent uniquement les plateformes de e-learning, ainsi elles ne permettent pas la traçabilité d'applications de type bureau, par exemple un serious game ou un micromonde installé sur l'ordinateur de l'utilisateur.

TINCAN, ou xAPI [Tincan, 2014], est un successeur de SCORM, qui vise à en combler certaines lacunes. Ainsi, TINCAN permet de tracer des activités pédagogiques issues de différentes sources, alors que SCORM ne trace que les activités identifiées comme faisant partie d'une même plateforme de e-learning. De plus, TINCAN concerne davantage de supports, comme les applications mobiles.

Les informations tracées lors d'une activité pédagogique qui respecte ces normes sont riches et utiles pour assister l'apprenant. Néanmoins, pour que cette traçabilité soit possible, le concepteur de l'application doit avoir respecté ces normes lors du développement de l'application, par exemple en utilisant des outils auteurs spécifiques, tels que SCORM Driver [Scorm_driver, 2014].

3.3. Reconnaissance visuelle

En utilisant les techniques de reconnaissance visuelle, il est possible de distinguer des zones rectangulaires de pixels de l'écran correspondant à des composants terminaux de l'interface d'une application, tels que les boutons et les cases à cocher. Prefab [Dixon et Fogarty, 2010] permet la surveillance d'applications existantes quelconques en associant les clics et déplacements de la souris avec des composants de l'interface d'une application. Cette technique présente l'avantage d'être compatible avec différents systèmes d'exploitation et de concerner toutes les applications existantes pourvues d'une interface graphique, qu'il s'agisse d'applications web ou d'applications bureau. Cette technique permet également d'identifier les composants d'images ou des vidéos d'applications, puisqu'elle n'utilise que les pixels pour reconnaitre les composants. Prefab permet de plus de réaliser des mises en valeur, sous la forme de coloration de la zone entre la souris et le composant à mettre en valeur (cf. Fig. 3-1).

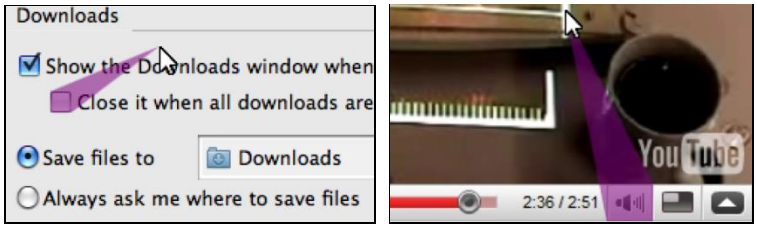


Fig. 3-1. Exemples de mises en valeur avec Prefab

Les scripts Sikuli [Sikuli, 2014] s'appuient également sur la reconnaissance visuelle des composants de l'interface, et permettent de réaliser des actions automatisées en simulant un clic sur une zone de l'écran qui correspond à un composant. Il est ainsi possible de simuler un clic de l'utilisateur sur un bouton, ou de cocher des cases à cocher pour lui.

[Chang et al., 2011] et [Hurst et al., 2010] proposent une technique mêlant la reconnaissance visuelle des composants de l'interface avec une exploitation des informations fournies par les bibliothèques d'accessibilité (cf. Section 3.4). User Interface Façades [Stuerzlinger et al., 2006] permet l'adaptation des interfaces graphiques aux utilisateurs finaux. Pour cela, ce système s'appuie sur la reconnaissance visuelle et sur l'exploitation de bibliothèques d'accessibilité pour dupliquer, puis modifier tout ou une partie de l'écran : l'utilisateur peut agir sur la façade, la zone de l'écran modifiée et éventuellement personnalisée, ses actions étant automatiquement répercutées sur l'écran original.

Les techniques reposant sur la reconnaissance visuelle sont pertinentes pour la surveillance d'une application de manière épiphyte. Elles présentent en particulier l'avantage d'être multiplateformes et adaptées quel que soit le type d'application-cible. En revanche, ces techniques ne semblent pas adaptées à l'inspection d'applications-cibles : elles ne permettent pas d'accéder aux informations non affichées à l'écran et qui peuvent être pertinentes pour fournir de l'assistance, telles que la description d'une image, l'infobulle associée à un bouton, ou l'état éditable ou non-éditable d'une zone de saisie.

3.4. Exploitation de bibliothèques d'accessibilité

Il existe un ensemble de bibliothèques d'accessibilité associées aux principaux systèmes d'exploitation ou environnements de programmation. Elles permettent notamment aux dispositifs d'accessibilité, tels que les lecteurs d'écrans et les barrettes braille, d'accéder aux applications qui sont compatibles avec ces bibliothèques. Au-delà de ces usages typiques, d'autres usages des bibliothèques d'accessibilité sont cependant possibles. Ainsi, elles permettent d'accéder à la hiérarchie des composants des applications compatibles, et d'inspecter les propriétés de ces composants, par exemple leur éventuelle description d'accessibilité ou leur état (actif, visible, coché…). Java Monkey [Java_Monkey, 2014] pour la bibliothèque JavaAccessibility [Harper et al., 2005] permet ainsi d'inspecter les applications tournant dans la machine virtuelle Java, Inspect [Inspect, 2014] (le successeur de UISpy [UISpy, 2014]) pour la bibliothèque UIAutomation [Haverty, 2005] permet d'inspecter les exécutables Windows, Accerciser [Accerciser, 2014] pour la bibliothèque ATK/T-SPI [ATK/AT-SPI, 2014] permet d'inspecter les applications GTK et Qt sous Gnome.

De plus, Java Ferret [Java_Ferret, 2014] pour la bibliothèque JavaAccessibility, permet de s'abonner à certains types d'événements ayant lieu à l'interface des applications tournant dans la machine virtuelle Java, tels que les déplacements de la souris et les changements de focus. De même, Patina [Matejka et al., 2013] permet de collecter et de visualiser des traces d'interactions entre les utilisateurs finaux et les exécutables Windows ; RUI [Kukreja et al., 2006] pour les applications sous Mac OSX.

Ces mêmes bibliothèques d'accessibilité permettent également d'automatiser certaines actions à l'interface des applications compatibles, afin par exemple d'automatiser des tests d'ergonomie, tels que Marathon [Marathon, 2014] ou Jacareto [Jacareto, 2014] pour les applications Java, et AutoIt [AutoIt, 2014], AutoHotKey [AutoHotKey, 2014] et AppMonitor [Alexander et al., 2008] pour les exécutables Windows. MenuInspector [Bailly et Malacria, 2013 ; Malacria, 2014] et ExposeHK [Malacria et al., 2013] exploitent quant à eux la

bibliothèque Apple Api Accessibility [Apple_API_Accessibility, 2014] pour analyser les menus des applications sous Mac OSX.

3.5. Conclusion

Nous avons présenté dans ce chapitre différentes techniques permettant la surveillance d'applications non préalablement instrumentées, ou d'applications respectant certaines normes.

Les systèmes d'enregistrement de logs permettent de détecter des événements bas niveau tels que les clics de l'utilisateur et l'ouverture de fenêtres. En revanche, ils ne fournissent pas d'informations suffisamment riches pour l'exploitation que nous souhaitons en faire : par exemple, pour proposer une assistance contextualisée, il est nécessaire de savoir précisément sur quel composant un clic a eu lieu. De plus, ces systèmes ne permettent ni d'accéder aux propriétés d'un composant ni d'en inspecter l'état.

Les normes qui permettent de tracer des activités pédagogiques, telles que Scorm et TinCan, constituent une approche intéressante pour la surveillance d'applications les respectant. Cependant, le respect de ces normes n'est actuellement pas systématique et de nombreuses applications proposant des activités pédagogiques ne les respectent pas. De plus, nous souhaitons pouvoir mettre en place de l'assistance dans des applications qui ne sont pas dans le domaine pédagogique.

Les systèmes s'appuyant sur la reconnaissance visuelle de composants sont particulièrement efficaces pour surveiller n'importe quelle application graphique, sans imposer aucune contrainte sur cette application. Ces systèmes fournissent des informations relatives à la détection d'événements suffisamment précises et riches pour permettre la mise en place d'une assistance finement contextualisée. En revanche, ces systèmes s'appuient uniquement sur ce qui est affiché à l'écran, et ne permettent donc pas d'accéder à toutes les propriétés d'un composant de l'interface d'une application. Ainsi, ils ne permettent pas, par exemple, de connaître le texte de l'infobulle d'un bouton, ni de connaître la valeur maximale et minimale d'une zone de saisie numérique. Ces connaissances de l'application-cible constituent pourtant un atout pour proposer une assistance pertinente et adaptée aux besoins de l'utilisateur final.

Les systèmes exploitant des bibliothèques d'accessibilité répondent pleinement à nos besoins relatifs à la détection d'événements, à l'accès aux propriétés d'un composant, ainsi qu'à l'inspection de l'état d'un composant. En revanche, les applications-cibles doivent être compatibles avec la bibliothèque d'accessibilité utilisée. De nombreuses applications sont d'ores et déjà compatibles avec une bibliothèque d'accessibilité et nous faisons l'hypothèse que de plus en plus d'applications sont appelées à être conformes aux standards d'accessibilité.

Cette analyse des forces et faiblesses des différentes approches de surveillance d'une application informatique nous orientent vers l'utilisation des bibliothèques d'accessibilité. Pour cela, nous souhaitons identifier les catégories d'applications-cibles compatibles avec chaque bibliothèque d'accessibilité, puis proposer une unique technique de surveillance

exploitant ces différentes bibliothèques d’accessibilité (cf. Chapitre 4). Cette technique offrirait ainsi une large couverture d’applications-cibles et pourrait être exploitée pour mettre en place une assistance épiphyte finement contextualisée dans ces applications.

Partie 2. Contributions théoriques

Plan de la partie « Contributions théoriques »

Dans le cadre de cette thèse, nous nous intéressons à la mise en place *a posteriori* d'un système d'assistance pour une application existante, appelée application-cible, sans avoir à modifier ou redévelopper cette application et sans que celle-ci soit conçue spécifiquement pour permettre l'adjonction d'une assistance. Pour cela, nous proposons d'adopter une démarche entièrement **épiphyte** et **générique** : elle n'est pas spécifique à une application-cible donnée, ni à un domaine ou à un environnement, mais concerne au contraire n'importe quelle application existante pour laquelle un concepteur d'assistance souhaite définir une assistance épiphyte.

Une application épiphyte, que nous appelons *épi-application*, est une application capable de réaliser une action sur une application externe sans perturber son fonctionnement [Paquette et al., 1994]. Un système d'assistance épiphyte est donc capable de réaliser des actions d'assistance sur une application-cible sans avoir à modifier cette application. Une assistance épiphyte est donc externe à l'application-cible, néanmoins, aux yeux de l'utilisateur final pour lequel le système d'assistance a été conçu, l'assistance fournie peut paraître intégrée à l'application-cible.

Par ailleurs, nous souhaitons permettre à une personne, qui n'est pas nécessairement le concepteur de l'application-cible, de définir un système d'assistance pour cette application. Cette personne, que nous appelons *concepteur de l'assistance*, est un expert de l'application-cible qui n'est pas obligatoirement un informaticien. Le terme de concepteur de l'assistance peut également faire référence à un groupe de personnes travaillant ensemble à la définition d'un système d'assistance pour une application-cible. Il peut s'agir par exemple d'une équipe pédagogique assistée par un expert de l'application-cible.

Afin de permettre la spécification *a posteriori* d'un système d'assistance pour une application-cible donnée par un concepteur d'assistance qui n'est pas nécessairement un informaticien, puis l'exécution de ce système d'assistance de manière épiphyte pour les utilisateurs finaux de l'application-cible, nous proposons un processus d'adjonction de systèmes d'assistance épiphytes dans une application existante (cf. Fig. P2-1) [Ginon, 2012 ; Ginon et al., 2012 ; Ginon et al., 2013c ; Ginon et al., 2013b]. Ce processus est constitué de deux phases. La première concerne le concepteur de l'assistance : elle lui permet de spécifier l'assistance qu'il souhaite pour une application-cible. La seconde phase concerne les utilisateurs finaux de l'application-cible : elle consiste en l'exécution de l'assistance souhaitée par le concepteur.

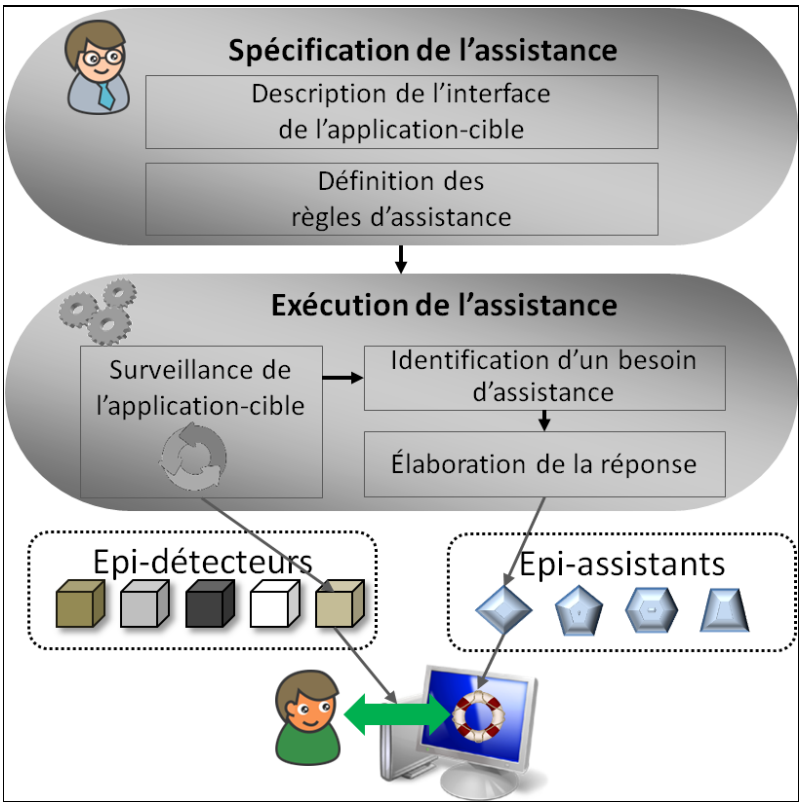


Fig. P2-1 – Processus d'adjonction d'un système d'assistance dans une application existante

La phase de **spécification de l'assistance** permet au concepteur de l'assistance de définir, par un ensemble de règles, l'assistance qu'il souhaite pour une application-cible donnée. Cette phase comporte deux étapes : la description de l'interface de l'application-cible et la définition des règles d'assistance.

La **description de l'interface de l'application-cible** est nécessaire pour obtenir des informations relatives à l'application-cible, et plus particulièrement à son interface. En effet, il est indispensable d'identifier de manière unique chaque composant de l'interface de l'application-cible, afin de pouvoir par la suite désigner un composant donné de l'interface de l'application-cible. La description de l'interface doit être effectuée lorsqu'un concepteur souhaite assister une application n'ayant jamais été décrite. Par la suite, si ce concepteur ou un autre concepteur souhaite définir un nouveau système d'assistance pour cette application-cible, il pourra réutiliser la description de l'interface existante.

La **définition des règles d'assistance** est l'étape principale de la spécification de l'assistance. Elle permet au concepteur d'exprimer l'assistance qu'il souhaite pour l'application-cible par un ensemble de règles de la forme <événement déclencheur, condition de déclenchement, actions d'assistance, événement de fin>. La définition des règles d'assistance a lieu à chaque fois qu'un concepteur souhaite spécifier un nouveau système d'assistance, ou modifier un système d'assistance existant.

La phase d'**exécution de l'assistance** a lieu à chaque utilisation de l'application-cible par un utilisateur final et se compose de trois processus : la surveillance de l'application-cible, l'identification d'un besoin d'assistance et l'élaboration d'une réponse à ce besoin. Cette phase exploite les règles d'assistance définies par le concepteur de l'assistance durant la phase de spécification de l'assistance : en effet, ce sont ces règles qui déterminent le comportement de l'assistance.

La **surveillance de l'application-cible** est nécessaire pour observer en continu les interactions entre l'utilisateur et l'interface de l'application-cible. Cette surveillance est rendue possible par des épi-détecteurs qui surveillent l'application-cible selon le processus présenté au Chapitre 4. Durant l'étape de surveillance de l'application-cible, les épi-détecteurs détectent chaque événement lié aux interactions entre l'utilisateur et l'interface de l'application-cible. Lorsqu'un événement est détecté, les épi-détecteurs font appel à la description de l'interface de l'application-cible produite durant la spécification de l'assistance afin d'identifier le composant-source de l'événement. Chaque notification d'un événement détecté par un épi-détecteur déclenche le processus qui cherche à identifier un besoin d'assistance en fonction des règles définies par le concepteur de l'assistance.

En parallèle du processus de surveillance de l'application-cible, le processus d'**identification d'un besoin d'assistance** exploite les règles d'assistance définies par le concepteur et déclenche le processus d'élaboration d'une réponse au besoin identifié. Lors de la notification d'un événement par les épi-détecteurs, le processus d'identification d'un besoin d'assistance parcourt les règles d'assistance à la recherche d'une ou plusieurs règles dont l'événement déclencheur correspond à l'événement détecté et dont les éventuelles conditions de déclenchement sont vérifiées. Pour chaque règle ainsi trouvée, le processus d'élaboration d'une réponse au besoin identifié est déclenché.

La **réponse à un besoin d'assistance** permet de fournir de l'assistance à l'utilisateur final, sous la forme d'une action d'assistance, réalisée sur l'application-cible par un épi-assistant. En fonction des règles d'assistance définies par le concepteur, le processus d'élaboration d'une réponse à un besoin d'assistance consiste à élaborer une ou plusieurs actions d'assistance, contextualisées et personnalisées pour répondre de manière la plus efficace possible au besoin d'assistance de l'utilisateur. Un ou plusieurs épi-assistants sont ensuite choisis pour réaliser ces actions parmi les épi-assistants disponibles et capables de réaliser les actions élaborées.

Chapitre 4. Processus de surveillance d'une application informatique

Objectif du chapitre

Permettre la surveillance d'applications existantes de manière épiphyte, afin de rendre possible la mise en place d'une assistance épiphyte finement contextualisée.

Points clés

Nous proposons un **processus de surveillance** d'une application informatique. Il permet la détection d'événements ayant lieu à l'interface d'une application-cible, l'identification du composant qui est la source de cet événement (par exemple le bouton sur lequel un clic a eu lieu), puis la notification de cet événement à une application exploitant ces informations, par exemple un système d'assistance contextualisé.

D'un point de vue technique, nous distinguons le cas des **applications bureau**, qui ne sont généralement pas conçues pour permettre une surveillance ni un accès à ses composants, et le cas des **applications web**, qui sont généralement ouvertes. Nous proposons donc deux techniques pour rendre possible ce processus de surveillance. La première technique est destinée à la surveillance des applications bureau ; elle s'appuie sur des **bibliothèques d'accessibilité**. La seconde technique est destinée à la surveillance des applications web ; elle s'appuie sur un **script utilisateur**.

Contributions

- Processus de surveillance d'une application informatique
- Technique de surveillance des applications bureau : exploitation de bibliothèques d'accessibilité
- Technique de surveillance des applications web : exploitation d'un script utilisateur

Validation

- ✓ Faisabilité : mise en œuvre opérationnelle et expérimentations
- ○ Couverture : plusieurs catégories d'applications-cibles d'ores et déjà prises en compte

(Le symbole ✓ indique qu'un point a été validé, alors que le symbole ○ indique qu'un point n'a pas encore été validé)

Publication liée à ce chapitre

[Ginon et al., 2013a]

> Ginon B., Champin P.-A. et Jean-Daubias S., Collecting fine-grained use traces in any application without modifying it, Worshop EXPPORT of ICCBR, 2013a.

Afin de permettre la mise en place d'assistances épiphytes finement contextualisées dans des applications-cibles, il est nécessaire d'accéder à certaines informations relatives à l'application-cible et plus particulièrement aux interactions entre l'utilisateur et l'interface de cette application-cible (comme les clics et les déplacements de la souris). En effet, il est difficile de proposer une assistance pertinente et contextualisée sans avoir accès aux traces d'interaction des utilisateurs ou sans connaître l'état de l'application-cible à tout instant.

Pour cette raison, nous nous sommes intéressée à la surveillance en temps réel d'une application-cible existante, de manière épiphyte, c'est-à-dire sans avoir à modifier l'application-cible ni perturber son fonctionnement. Le processus de surveillance d'une application-cible que nous présentons dans ce chapitre peut être réalisé sur une application quelconque, sans contrainte sur l'ouverture de cette application. Cela signifie qu'il n'est pas nécessaire qu'elle ait été conçue spécifiquement pour permettre la surveillance des interactions entre son interface et ses utilisateurs, ou pour permettre l'inspection de l'état de des composants de son interface [Ginon et al., 2013a].

Pour permettre la surveillance des applications-cibles les plus variées avec le processus de surveillance que nous proposons, nous distinguons le cas des applications de type bureau (cf. Section 4.2) et le cas des applications web (cf. Section 4.3), pour lesquels les techniques proposées pour permettre une telle surveillance diffèrent.

4.1. Processus de surveillance d'une application-cible

Notre processus de surveillance en temps réel d'une application cible de manière épiphyte est constitué de trois étapes (cf. Fig. 4-1) : la détection d'un événement lié aux interactions entre l'utilisateur et l'interface de l'application-cible, l'identification du composant source de cet événement et la notification de cet événement lorsque sa source est identifiée.

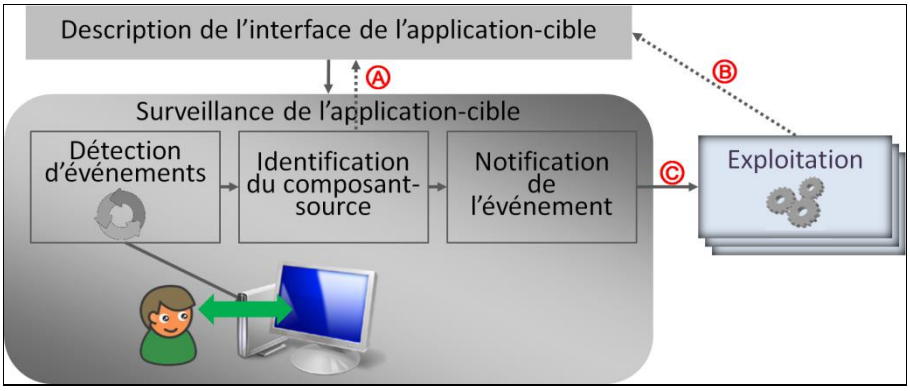


Fig. 4-1. Processus de surveillance d'une application-cible

4.1.1. Détection d'événements

La détection d'événements se fait en continu tout au long de la surveillance de l'application-cible. Cette étape permet de détecter tout événement lié à une interaction entre un utilisateur et l'interface de l'application-cible : clic, déplacement de la souris, saisie de texte, ouverture d'un menu, sélection d'un item d'une liste déroulante, etc. Chaque

détection d'événements déclenche le processus d'identification de la source de cet événement sans pour autant mettre fin à la détection d'événements. En effet, la détection d'événements se termine seulement lorsque la surveillance de l'application-cible prend fin.

4.1.2. Identification du composant source

Le processus d'identification du composant source d'un événement détecté permet d'associer cet événement à un composant précis de l'interface de l'application-cible. Nous avons vu dans le Chapitre 3 qu'il existe des techniques permettant de détecter les clics des utilisateurs. Grâce à l'étape d'identification de la source d'un événement, le processus de surveillance d'une application-cible que nous proposons permet d'associer chaque événement détecté à un composant de l'interface de l'application-cible, ce qui permet une exploitation plus riche de ces événements. En effet, s'il peut être utile de savoir que l'utilisateur a cliqué dans une fenêtre, il est plus utile encore de connaître le composant de la fenêtre le plus précis possible sur lequel il a cliqué, comme un bouton, un menu ou une case à cocher. Il est également utile de pouvoir détecter une grande variété d'événements en complément des clics, des déplacements de la souris et des saisies, qui sont les plus fréquemment collectés. Ainsi, le processus de surveillance d'une application-cible que nous proposons permet également de collecter des événements liés par exemple à une sélection (de texte, d'item…) ou liés à une ouverture ou fermeture (de fenêtre, de menu, d'infobulle, de liste déroulante…).

4.1.3. Notification d'un événement

Les événements détectés et dont la source est identifiée sont notifiés aux applications qui écoutent le processus de surveillance de l'application-cible afin d'exploiter ces informations. Les événements sont notifiés sous la forme <type d'événement, composant-source>. Ils peuvent alors être exploités en temps réel par une ou plusieurs applications et/ou être stockés dans une trace en vue d'une exploitation future. De nombreuses applications intéressantes s'appuient en effet sur l'exploitation de traces : l'assistance à l'utilisateur et la visualisation de traces pour la découverte de connaissances ou pour la réflexivité en sont des exemples typiques [Terrat et Sagot, 2011 ; Champin et al., 2012 ; Thevenet et al., 2012 ; Zarka, 2013]. De nombreuses applications exploitant les traces pourraient ainsi bénéficier de notre processus de surveillance épiphyte pour obtenir des traces riches provenant d'applications variées. Pour cela, ces applications devront toutefois être « à l'écoute » des notifications envoyées par la surveillance de l'application (cf. © Fig. 4-1).

4.1.4. Utilisation d'une description de l'interface de l'application-cible

La *description de l'interface de l'application-cible* apporte des connaissances relatives aux composants de l'interface d'une application-cible ou d'une partie de cette application (dans les cas où l'on s'intéresse uniquement à certaines fonctionnalités de l'application-cible par exemple), ou de plusieurs applications-cibles. Dans cette description, les composants sont associés à un identifiant, ainsi qu'à diverses informations qui pourront être utiles lors de

l'exploitation des traces (cf. Ⓑ Fig. 4-1). Par exemple, pour un composant donné, en plus d'un identifiant, la description de l'interface peut préciser qu'il s'agit d'un bouton dont le texte est « Suivant », qu'il est associé à une infobulle dont le texte est « Passer à l'étape suivante », et que la fonction de ce bouton est de terminer l'étape X pour passer à l'étape Y de la tâche Z. Pour une exploitation de type assistance à l'utilisateur, nous verrons en section 8.2 un exemple d'utilisation de la description de l'interface.

Cette description est exploitée pour filtrer les événements détectés lors de l'identification du composant-source (cf. Ⓐ Fig. 4-1), afin de ne notifier que les événements en lien avec les composants contenus dans cette description. Elle peut également être exploitée pour faciliter l'exploitation des traces en fournissant des connaissances relatives aux composants de l'application-cible.

La description de l'interface de l'application-cible est une étape optionnelle préliminaire à la surveillance d'une application-cible. Sans cette description, les traces d'interactions sont moins riches et exploitables. Prenons l'exemple d'un clic : si l'étape de description de l'interface a été réalisée, l'événement notifié associera ce clic à son composant-source, c'est-à-dire au composant sur lequel le clic a eu lieu et dont la description de l'interface précise un certain nombre de connaissances. Dans le cas contraire, l'événement notifié indiquera uniquement qu'un clic a eu lieu.

Dans la suite de cette section, nous distinguons le cas des applications de type bureau de celui des applications web, pour lesquelles la technique que nous proposons pour décrire l'interface de l'application-cible et pour détecter des événements est différente.

4.2. Technique de surveillance des applications bureau

Nous proposons une technique pour collecter les traces d'interactions entre un utilisateur et l'interface d'une application-cible de type bureau quelconque. Cette technique s'appuie sur l'utilisation de bibliothèques d'accessibilité telles qu'UIAutomation [Haverty, 2005] ou JavaAccessibility [Harper et al., 2005]. En exploitant ces bibliothèques (cf. Section 3.4) , il est possible de s'abonner à différents types d'événements (par exemple les clics de la souris, les changements de focus, les ouvertures de menus…) afin d'être informé lorsqu'un événement de ce type survient, mais aussi de connaître le composant de l'interface concerné par l'événement. Par exemple, il est possible de connaître quel composant a été survolé par la souris, ou quel bouton a gagné le focus.

Il existe plusieurs bibliothèques d'accessibilité complémentaires (cf. Fig. 4-2). En effet, chaque système d'exploitation dispose de sa propre bibliothèque d'accessibilité. De plus, les applications développées avec certains langages de programmation ne sont pas ouvertes aux bibliothèques d'accessibilité utilisées par les systèmes d'exploitation, mais disposent d'une bibliothèque d'accessibilité qui leur est propre, comme c'est le cas pour les applications Java qui sont ouvertes uniquement à la bibliothèque JavaAccessibility.

Bibliothèque d'accessibilité	Couverture
UIAutomation	Windows – applications natives
Apple API Accessibility	Mac OS
ATK/AT SPI	Linux – applications Gnome et GTK Linux – applications KDE et Qt
JavaAccessibility	Applications Java

Fig. 4-2. Couverture des bibliothèques d'accessibilité

4.2.1. Description de l'application-cible

Les bibliothèques d'accessibilité fournissent un accès à la totalité de la hiérarchie des composants de l'interface de toutes les applications ouvertes sur le bureau. À titre d'exemple, la Fig. 4-3 présente l'arborescence des composants visible par la bibliothèque UIAutomation à un moment donné. Nous pouvons voir sur cet exemple que six applications sont lancées sur cet ordinateur : NetBeans, Word, EndNote, Google Chrome, et la calculatrice de Windows. Nous voyons également la barre des tâches (cf. Ⓐ Fig. 4-3) et les icônes ou fichiers situés sur le bureau (cf. © Fig. 4-3). Nous constatons que pour NetBeans (cf. Ⓑ Fig. 4-3), seul le cadre de l'application (avec les boutons Restaurer, Agrandir et Fermer) est visible pour UIAutomation. En effet, NetBeans, comme toutes les applications développées en Java, n'est pas ouverte à la bibliothèque UIAutomation, mais à la bibliothèque JavaAccessibility propre au langage Java.

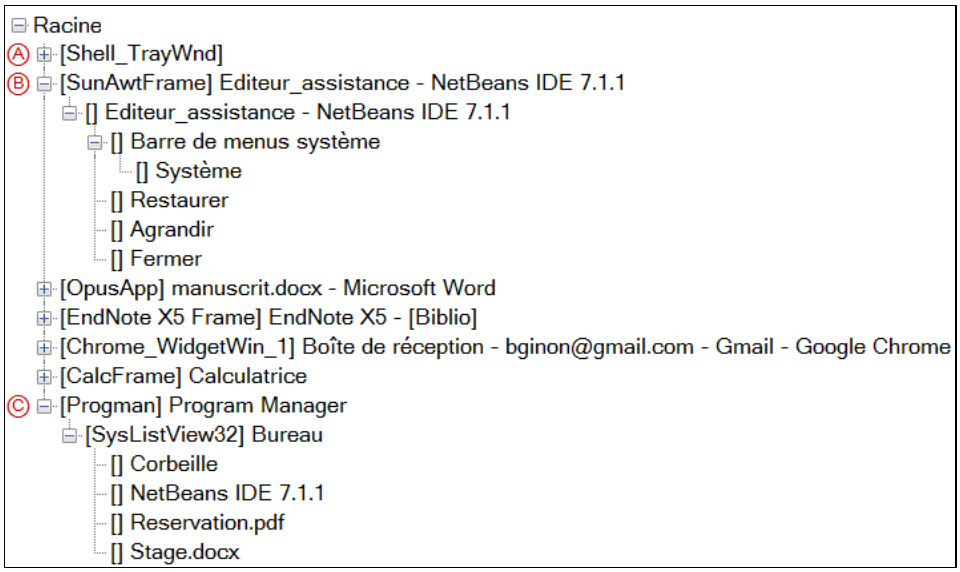


Fig. 4-3. Arborescence des composants détectés par la bibliothèque d'accessibilité UIAutomation

Grâce aux bibliothèques d'accessibilité, il est également possible de connaître à tout instant l'état d'une application lancée. En effet, ces bibliothèques fournissent de nombreuses

informations sur chaque composant de l'interface d'une application ouverte, comme son type (fenêtre, bouton, case à cocher…), son texte, sa position à l'écran, sa description d'accessibilité, son état (actif, sélectionné, maximisé, visible…).

Il est à noter que dans la plupart des cas, les composants détectés par les bibliothèques d'accessibilité ne sont pas associés à un identifiant persistant, or il est nécessaire de pouvoir identifier chaque composant en vue d'une exploitation immédiate ou future des traces d'interactions. Une solution à ce problème consiste à identifier un composant par sa position hiérarchique dans l'arborescence des composants d'une application, ainsi que par les informations complémentaires auxquelles la bibliothèque d'accessibilité donne accès, notamment son type et sa description d'accessibilité qui eux sont persistants.

Pour cela, nous considérons l'arborescence des composants d'une application comme un arbre XML. Chaque élément XML représente un composant de l'interface, avec un type, un éventuel texte ou label et une éventuelle description d'accessibilité. Chaque composant est alors caractérisé par son chemin XML contenant toutes les informations persistantes concernant le composant et ses parents.

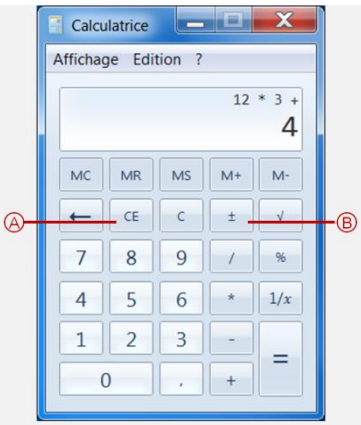


Fig. 4-4. Capture d'écran de la calculatrice Windows

La Fig. 4-5 donne l'exemple de l'arbre XML décrivant la hiérarchie des composants de la calculatrice Windows (Fig. 4-4). Notons par exemple que le bouton « CE » est associé à la description d'accessibilité « Effacer l'entrée » (cf. Ⓐ Fig. 4-5 et Fig. 4-4). Ce bouton peut être caractérisé par le chemin suivant :

//fenetre[@type="CalcFrame" and @text="Calculatrice"]/ componsant [@type="CalcFrame" and @text=""]/composant[@type="Button" and @text="CE" and @description=" Effacer l'entrée "]

```
<fenetre type="CalcFrame" texte="Calculatrice">
 <composant type="TitleBar" texte="Calculatrice">
 <composant type="MenuBar" texte="Application">
 <composant type="CalcFrame" texte="">
  <composant type="Static" texte="12 * 3 +" />
  <composant type="Static" texte="4" />
  <composant type="Button" texte="MC" description="Effacer la mémoire"/>
  <composant type="Button" texte="MR" description="Rappel mémoire" />
  <composant type="Button" texte="MS" description="Stockage mémoire" />
  <composant type="Button" texte="M+" description="Ajout mémoire" />
  <composant type="Button" texte="M-" description="Soustraction mémoire"/>
  <composant type="Button" texte="" description="Retour arrière" />
Ⓐ<composant type="Button" texte="CE" description="Effacer l'entrée" />
  <composant type="Button" texte="C" description="Effacer" />
Ⓑ<composant type="Button" texte="" description="Négation" />
  <composant type="Button" texte="" description="Racine carrée" />
  <composant type="Button" texte="7" />
  <composant type="Button" texte="4" />
  <composant type="Button" texte="1" />
  <composant type="Button" texte="0" />
  <composant type="Button" texte="8" />
  <composant type="Button" texte="5" />
  <composant type="Button" texte="2" />
  <composant type="Button" texte="9" />
  <composant type="Button" texte="6" />
  <composant type="Button" texte="3" />
  <composant type="Button" texte="," description="Séparateur décimal"/>
  <composant type="Button" texte="" description="Diviser" />
  <composant type="Button" texte="" description="Multiplier" />
  <composant type="Button" texte="" description="Soustraire" />
  <composant type="Button" texte="" description="Additionner" />
  <composant type="Button" texte="" description="Pourcentage" />
  <composant type="Button" texte="" description="Réciproque" />
  <composant type="Button" texte="" description="Est égal à" />
 </composant>
</fenetre>
```

Fig. 4-5. Description de l'interface de la calculatrice Windows

4.2.2. Détection d'événements

Les bibliothèques d'accessibilité permettent de s'abonner à certains types d'événements (par exemple les clics, les déplacements de la souris, les changements de focus, les ouvertures ou fermetures de menus ou d'infobulles) afin d'être informé de l'occurrence de tous les événements de ce type détectés par la bibliothèque sur l'un des composants visibles pour elle. Ainsi, en s'abonnant avec UIAutomation aux événements de type clic gauche, il sera possible de savoir lorsque l'utilisateur clique sur un composant de la calculatrice Windows qui est ouverte à UIAutomation, mais pas lorsqu'il clique sur un composant de NetBeans qui n'est pas ouverte à UIAutomation, à l'exception de la fenêtre et des boutons du cadre de la fenêtre.

4.3. Technique de surveillance des applications web

Pour les applications web, ouvertes par nature, il n'est pas nécessaire de passer par une bibliothèque d'accessibilité pour accéder à leurs composants et pour s'abonner aux événements liés à l'interaction entre l'utilisateur final et l'interface des applications web. Nous présentons dans cette section une technique simple pour la surveillance et l'inspection des applications web ; elle s'appuie sur l'utilisation de scripts utilisateurs. On retrouve néanmoins dans ces deux techniques des éléments similaires pour identifier un composant, tels que la position dans la hiérarchie des composants et l'utilisation de la description d'accessibilité d'une part et d'autre part le XPath et le champ ALT.

4.3.1. Description de l'application-cible

Une fois exécutées dans un navigateur, les applications web peuvent être considérées comme un ensemble de pages web en HTML ou XHTML dont l'arborescence des composants peut être inspectée. Chaque composant d'une application web peut donc être identifié par son URL et son XPATH à partir de la racine de la page ou du plus proche ancêtre du composant associé à un identifiant persistant.

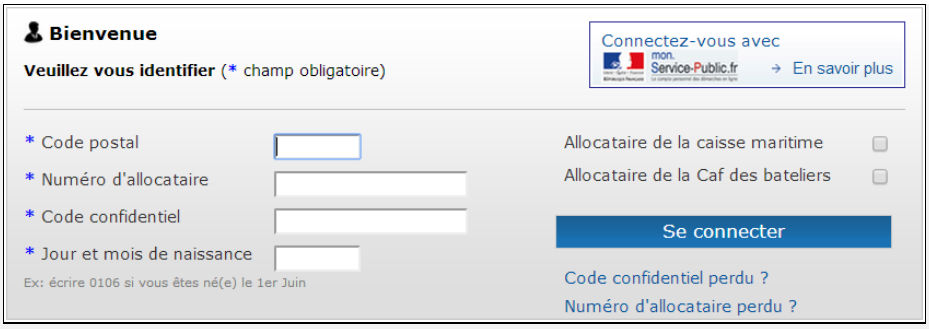


Fig. 4-6. Capture d'écran d'un formulaire d'authentification du site de la CAF[7]

La Fig. 4-7 donne l'exemple de la description de l'interface d'un formulaire d'authentification présent sur le site de la CAF (Caisse d'Allocations Familiales, cf. Fig. 4-6).

[7] http://wwwd.caf.fr/wps/portal/caffr/login

```
<fenetre id="0" url="https://wwwd.caf.fr/wps/portal/caffr/login/" >
  <composant type="lien" texte="Connectez-vous avec"
             xpath="CHEMIN/div/div[2]/div/p/strong/a" />
  <composant type="lien" texte="En savoir plus"
             xpath="CHEMIN/div/div[2]/div/p[2]/a" />
  <composant type="lien" texte="Code confidentiel perdu ?"
             xpath="CHEMIN/div[2]/fieldset/div[2]/div[4]/a" />
  <composant type="lien" texte="Numéro d'allocataire perdu ?"
             xpath="CHEMIN/div[2]/fieldset/div[2]/div[4]/a[2]" />
  <composant type="image" alt="mon.service-public.fr"
             xpath="CHEMIN/div/div[2]/div/p/strong/a/img" />
  <composant type="bouton" titre="Se connecter"
             xpath="CHEMIN/div[2]/fieldset/div[2]/div[3]/input" />
  <composant type="zone de saisie"
             xpath="CHEMIN/div[2]/fieldset/div/div/input" />
  <composant type="zone de saisie"
             xpath="CHEMIN/div[2]/fieldset/div/div[2]/input" />
  <composant type="zone de saisie"
             xpath="CHEMIN/div[2]/fieldset/div/div[3]/input" />
  <composant type="zone de saisie"
             xpath="CHEMIN/div[2]/fieldset/div/div[4]/input" />
  <composant type="case à cocher"
             xpath="CHEMIN/div[2]/fieldset/div[2]/div/input" />
  <composant type="case à cocher"
             xpath="CHEMIN/div[2]/fieldset/div[2]/div[2]/input" />
  <composant type="label" texte="Bienvenue"
             xpath="CHEMIN/div/div/span" />
  <composant type="label" texte="(* champ obligatoire)"
             xpath="CHEMIN/div/div/p" />
  <composant type="label" texte="Veuillez vous identifier"
             xpath="CHEMIN/div/div/p/strong" />
  <composant type="label" texte="* Code postal"
             xpath="CHEMIN/div[2]/fieldset/div/label" />
  <composant type="label" texte="* Numéro d'allocataire"
             xpath="CHEMIN/div[2]/fieldset/div/label[2]" />
  <composant type="label" texte="* Code confidentiel"
             xpath="CHEMIN/div[2]/fieldset/div/label[3]" />
  <composant type="label" texte="* Jour et mois de naissance"
             xpath="'CHEMIN/div[2]/fieldset/div/label[4]'" />
  <composant type="label" texte="Ex:écrire 0106 si vous êtes né(e) le 1er Juin"
             xpath="CHEMIN/div[2]/fieldset/div/div[5]"/>
  <composant type="label" texte="Allocataire de la caisse maritime"
             xpath="CHEMIN/div[2]/fieldset/div[2]/label" />
  <composant type="label" texte="Allocataire de la Caf des bateliers"
             xpath="CHEMIN/div[2]/fieldset/div[2]/label[2]" />
</fenetre>
```

Fig. 4-7. Description de l'interface d'un formulaire d'authentification du site de la CAF

4.3.2. Détection d'événements

Nous proposons une technique de collecte de traces pour les applications Web qui s'appuie sur l'utilisation de scripts utilisateurs qui permettent d'inclure des gestionnaires d'événements dans les pages web du navigateur. Pour cela, une extension doit être installée sur le navigateur de l'utilisateur, et un script de collecte doit être activé.

4.4. Conclusion

Dans ce chapitre, nous avons présenté un processus de surveillance d'une application-cible qui permet de notifier toutes les interactions d'un utilisateur avec l'interface de l'application-cible à une application qui exploite ou stocke ces informations. Le processus de surveillance que nous proposons est épiphyte : la surveillance ne perturbe pas le fonctionnement de l'application-cible et l'application-cible n'a pas besoin d'être spécifiquement « ouverte » à la surveillance.

Ce processus de surveillance ne concerne pas uniquement les clics et les déplacements de la souris, mais concerne au contraire toutes les interactions d'un utilisateur avec l'interface d'une application-cible. De plus, ce processus permet d'associer chaque événement détecté avec son composant source, c'est-à-dire avec le composant de l'interface de l'application-cible sur lequel a eu lieu l'événement. Une description de l'interface de l'application-cible peut être utilisée afin de ne notifier que les événements ayant lieu à l'interface de l'application-cible et d'ignorer les événements ayant lieu hors de cette application.

La surveillance des interactions de l'utilisateur avec l'application-cible et l'association de chaque événement détecté à son composant source permet une exploitation riche de ces traces d'interactions. Dans le chapitre suivant, nous présentons comment le processus de surveillance d'une application-cible peut être exploité par un processus d'adjonction d'un système d'assistance épiphyte à une application-cible, pour mettre en place de l'assistance dans des applications existantes.

Chapitre 5. aLDEAS : un langage de définition de systèmes d'assistance épiphytes

Objectif du chapitre

Permettre la spécification de l'assistance de manière suffisamment formalisée pour qu'elle soit exécutable automatiquement, tout en restant accessible à des concepteurs d'assistance potentiellement non-informaticiens.

Points clés

Nous proposons **aLDEAS**, un **langage pivot** entre la phase de spécification de l'assistance par un concepteur potentiellement non-informaticien et la phase d'exécution de cette assistance par un outil générique. Il s'agit d'un **langage graphique simple**, dédié à la spécification de systèmes d'assistance très variés, sous la forme d'un ensemble de règles. Le langage aLDEAS est accompagné par plusieurs **patrons** qui facilitent la définition de blocs aLDEAS par le concepteur de l'assistance, en particulier un patron de règles d'assistance.

Contributions

- Le langage graphique aLDEAS : langage pivot entre la spécification de l'assistance et l'exécution de l'assistance
- 1 patron de règles d'assistance
- 3 patrons d'actions d'assistance : pas à pas, présentation guidée et action d'agent animé

Validation

- ✓ Large couverture d'aLDEAS pour définir des systèmes d'assistance variés
- ✓ Simplicité d'utilisation d'aLDEAS par des concepteurs d'assistance informaticiens et non-informaticiens
- ✓ Intérêt d'aLDEAS pour faciliter la définition de systèmes d'assistance
- ✓ Utilité des patrons associés à aLDEAS

Publications liées à ce chapitre

[Ginon et al., 2014a]

Ginon B., Jean-Daubias S., Champin P.-A. et Lefevre M., aLDEAS : un langage de définition de systèmes d'assistance épiphytes, IC, Clermont-Ferrand, France, pp. 137-148, 2014a.

[Ginon et al., 2014b]

Ginon B., Thai V. L., jean-Daubias S., Lefevre M. et Champin P.-A., Adding epiphytic assistance systems in learning applications using the SEPIA system, Ec-Tel, Graz, Austria, 2014b.

Notre objectif est de rendre possible d'une part la spécification d'un système d'assistance pour une application-cible existante, sans contrainte de domaine ou de caractéristiques techniques, par un concepteur d'assistance qui peut être autre que le développeur de l'application-cible, et d'autre part l'exécution de l'assistance spécifiée, sans modification de l'application-cible et sans programmation informatique.

En introduction de cette partie, nous avons proposé un processus d'adjonction de systèmes d'assistance dans des applications existantes de manière épiphyte. Nous avons vu que ce processus est constitué d'une phase de spécification de l'assistance, suivie d'une phase d'exécution de l'assistance spécifiée. Afin de permettre au concepteur de spécifier l'assistance souhaitée pour les utilisateurs finaux d'une application sans mettre en œuvre de connaissances en programmation, une solution consiste à utiliser un outil générique capable d'exécuter de manière épiphyte un système d'assistance. L'utilisation d'un tel outil implique de formaliser l'assistance selon un langage pivot entre la phase de spécification qui concerne le concepteur de l'assistance et la phase d'exécution qui est réalisée par un outil générique. Ce langage doit être suffisamment générique pour permettre la spécification de systèmes d'assistance riches pour des applications-cibles variées, tout en restant suffisamment simple pour être utilisable par un concepteur d'assistance potentiellement non-informaticien. À notre connaissance, un tel langage n'existe pas. Pour ces raisons, nous proposons aLDEAS (a Language to Define Epi-Assistance Systems), un langage graphique à destination des concepteurs d'assistance, dont le but est de permettre la définition de systèmes d'assistance pour des applications existantes, sous la forme d'un ensemble de règles [Ginon et al., 2014a]. aLDEAS se situe au cœur de nos travaux de recherche sur l'assistance à l'utilisateur, et constitue le lien entre la phase de spécification et la phase d'exécution de l'assistance du processus d'adjonction de systèmes d'assistance dans une application existante. Il permet en effet au concepteur de l'assistance de représenter ses connaissances relatives à l'application-cible et à l'assistance qu'il souhaite y ajouter. Ces connaissances peuvent ensuite être exploitées par un outil générique pour fournir automatiquement l'assistance aux utilisateurs finaux de l'application-cible.

5.1. Composants du langage aLDEAS

Afin de permettre la spécification de systèmes d'assistance variés et fortement personnalisables, pleinement adaptés à leur application-cible, sans contrainte sur le domaine ou les caractéristiques techniques de celle-ci, nous avons besoin d'un langage suffisamment riche et expressif. Pour cette raison, le langage aLDEAS est constitué de différents types de composants (cf. Fig. 5-1). Nous montrerons en section 5.1.4 de quelles manières ces composants peuvent être combinés dans des blocs, afin de permettre la spécification de systèmes d'assistance les plus variés.

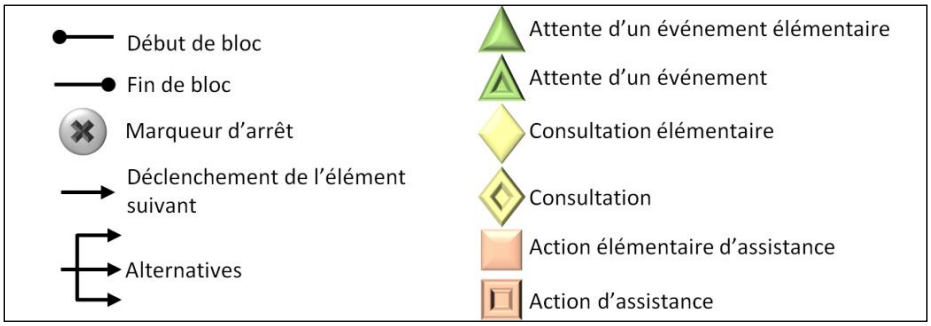


Fig. 5-1 – Composants du langage aLDEAS

Parmi ces composants, les attentes d'événement permettent de suspendre l'exécution d'un bloc aLDEAS, par exemple pour contextualiser le déclenchement ou la fin d'une action d'assistance. Les consultations permettent au système d'assistance d'obtenir des informations qui peuvent conditionner la poursuite de l'exécution d'un bloc, afin d'adapter l'assistance, par exemple, pour proposer une action d'assistance uniquement pour certains types d'utilisateurs. Les actions d'assistance permettent de fournir à l'utilisateur final de l'assistance sous des formes très variées. Les actions d'assistance proposées par aLDEAS correspondent aux différents moyens techniques identifiés dans notre état de l'art de l'assistance à l'utilisateur d'applications informatiques (cf. Chapitre 1). Le marqueur d'arrêt permet quant à lui mettre fin à toutes les actions d'assistance déclenchées depuis le début du bloc.

À titre d'exemple, la Fig. 5-2 présente un bloc aLDEAS, sur lequel nous reviendrons en Section 5.2.1. Ce bloc aLDEAS permet de déclencher de l'assistance lorsque l'utilisateur clique sur un bouton donné de l'application-cible, ce qui est représenté par le permier  du bloc. L'assistance fournie prend la forme d'une mise en valeur d'un bouton et d'un message d'aide, représentés par deux  . Cette assistance prendra fin lorsque que l'utilisateur aura cliqué sur le bouton mis en valeur, ce qui est représenté par le second  suivi de  . Il s'agit donc d'une assistance contextualisée à l'initiative du système d'assistance, mais qui n'est ni personnalisée ni évolutive, bien que cela soit possible avec aLDEAS, comme nous le verrons par la suite. Cette assistance a pour objectif de répondre à un besoin relatif à la réalisation d'une tâche.

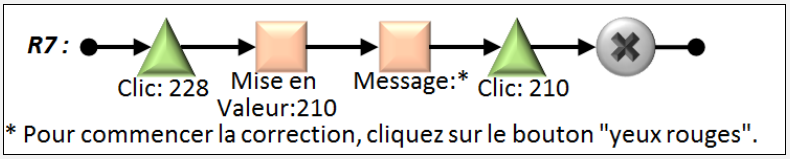


Fig. 5-2 – Exemple de bloc aLDEAS

Par la suite, nous présentons de manière détaillée les composants d'aLDEAS.

5.1.1. Attentes d'événements

Afin de permettre la spécification de systèmes d'assistance réactifs et à l'écoute des événements qui surviennent d'une part dans l'application-cible et d'autre part au niveau de l'assistance, aLDEAS propose les **attentes d'événements**, des éléments qui permettent

d'attendre qu'un événement donné ait eu lieu avant de déclencher un autre élément. Ces composants pourront être utilisés notamment afin de déclencher une action d'assistance à la suite d'un événement donné.

La Fig. 5-3 montre comment aLDEAS caractérise les attentes d'événement élémentaire. Ces éléments sont représentés dans le langage par ▲, peuvent concerner un événement lié à l'utilisateur, un événement lié à l'assistance ou un timer. Les composants élémentaires d'aLDEAS sont hiérarchisés sous forme arborescente avec des traits pleins. Les paramètres sont indiqués en italique et les paramètres optionnels sont précédés d'un ?.

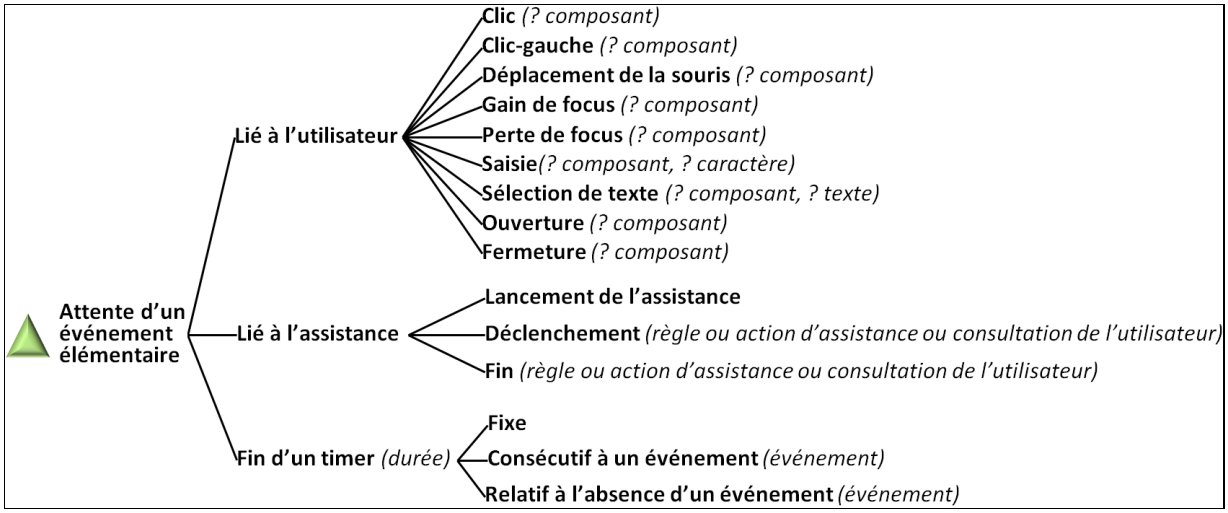


Fig. 5-3 – aLDEAS : attentes d'événement élémentaire

Une **attente d'événement lié à l'utilisateur** implique d'attendre que l'utilisateur réalise une action donnée à l'interface de l'application-cible. De telles attentes d'événements permettent notamment de contextualiser le déclenchement ou la fin d'actions d'assistance, et de proposer une assistance à l'initiative du système d'assistance. Une action de l'utilisateur est caractérisée par un type : clic (ou clic-droit), clic-gauche, déplacement de la souris, gain ou perte de focus, saisie, sélection de texte, ouverture ou fermeture (d'infobulles, de menus ou de fenêtres). Une action de l'utilisateur peut concerner un composant de l'interface de l'application-cible. Le paramètre *composant* est optionnel : si aucun composant n'est spécifié l'événement concerne n'importe quel composant de l'écran, qu'il appartienne ou non à l'application-cible. De plus, un événement de type *saisie* a pour paramètre optionnel un caractère ; un événement peut donc concerner la saisie d'un caractère donné ou une saisie quelconque. De même, un événement de type *sélection de texte* a pour paramètre optionnel un texte.

La Fig. 5-4 présente quatre exemples d'attentes d'événements élémentaires liés à l'utilisateur : E0 permet d'attendre que l'utilisateur clique sur le composant d'identifiant 364 (par exemple un bouton) pour déclencher le composant qui suit le symbole ⟶, E1 permet d'attendre l'ouverture du composant 12 (par exemple une fenêtre ou une liste déroulante) ; E2 permet d'attendre qu'une saisie du caractère @ ait lieu dans le composant 109 (une zone de saisie dans cet exemple) ; et E3 permet d'attendre que l'utilisateur déplace la souris. Notons que dans l'exemple de E3, le composant concerné n'est pas précisé, ce qui signifie que E3

concerne un déplacement quelconque de la souris et non pas le passage de la souris sur un composant donné.

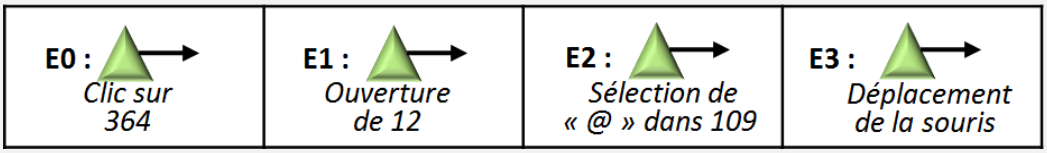


Fig. 5-4 – Exemples d'attentes d'événements élémentaires liés à l'utilisateur

Une **attente d'événement lié à l'assistance** implique d'attendre que le système d'assistance démarre, ou qu'il déclenche ou mette fin à une règle ou action d'assistance. De telles attentes d'événements permettent notamment de fournir une assistance étape par étape, en déclenchant une nouvelle étape de l'assistance uniquement lorsque la précédente a pris fin par exemple. Un événement lié à l'assistance est caractérisé par un type (lancement de l'assistance, déclenchement ou fin) et par une règle ou action d'assistance concernée, si l'événement ne concerne pas le lancement de l'assistance.

La Fig. 5-5 présente trois exemples d'attentes d'événements élémentaires liés à l'assistance : E4 correspond au lancement de l'assistance, E5 implique d'attendre que l'action d'assistance A7 soit déclenchée, et E6 implique d'attendre que la règle d'assistance R11 prenne fin.

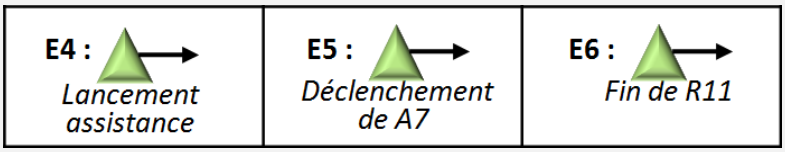


Fig. 5-5 – Exemples d'attentes d'événements élémentaires liés à l'assistance

Une **attente d'événement élémentaire lié à la fin d'un timer** implique d'attendre qu'un timer atteigne sa durée, passée en paramètre. De telles attentes d'événements, suivies du marqueur d'arrêt, permettent en particulier de mettre fin à des actions d'assistance au bout d'une certaine durée, comme une mise en valeur ou un message d'aide qui disparait après 30 secondes. Dans le cas d'un timer fixe, le timer est déclenché lorsque le symbole est atteint, et l'élément suivant ne sera déclenché qu'après la fin de ce timer. Un timer peut également être consécutif à un événement donné. Dans ce cas, le symbole correspond à l'attente de cet événement puis à l'attente de la fin du timer. Enfin, un timer peut être relatif à l'absence d'un événement donné. Dans ce cas, le symbole représente l'attente qu'un timer atteigne sa durée donnée en paramètre, sachant que ce timer est initialisé lors du lancement de l'assistance puis remis à zéro à chaque fois que l'événement donné a lieu.

La Fig. 5-6 présente trois exemples d'attentes d'événements élémentaires liés à l'assistance : E7 correspond à une attente de 30 secondes, E8 correspond à attendre 2 minutes après que E5 ait eu lieu, c'est-à-dire 2 minutes après le déclenchement de l'action d'assistance A7 (cf. E5 Fig. 5-4), et E9 correspond à une attente de 1 minute sans que E3 ait eu lieu, c'est-à-dire sans déplacement de la souris (cf. E3 Fig. 5-4).

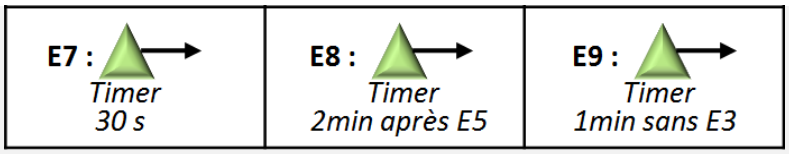


Fig. 5-6 – Exemples d'attentes d'événements élémentaires de type timer

Les attentes d'événements élémentaires peuvent être combinées pour former une succession d'attentes d'événements élémentaires définissant une **attente d'événement composé**, représentée dans le langage par ⚠. Ainsi, les attentes d'événements permettent d'attendre des événements de plus haut niveau que les événements élémentaires. Par exemple, une action de correction des yeux rouges sur une photo est un événement non élémentaire, constitué d'une suite donnée d'événements élémentaires, comme des clics et des déplacements de la souris. Il est à noter que, dans le cas où une application permet d'effectuer une action de plusieurs manières, un événement non élémentaire peut être défini par plusieurs suites d'événements élémentaires distinctes mais équivalentes.

Par exemple, une action classique de « copier-coller » peut être réalisée à l'aide de la souris (cf. l'exemple d'E10 sur la Fig. 5-7), du clavier (cf. l'exemple d'E11 sur la Fig. 5-7), ou d'un mélange des deux.

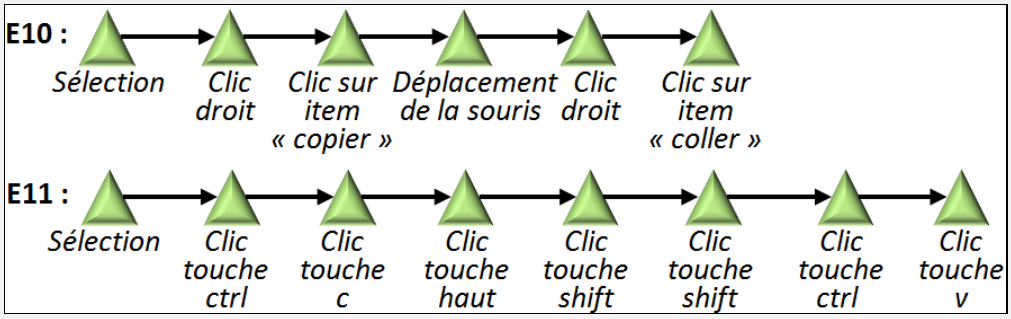


Fig. 5-7 – Exemples d'attentes d'événements non élémentaires correspondant à une action de « copier-coller »

Notons que dans aLDEAS le symbole △ est utilisé uniquement pour représenter une attente d'événement élémentaire, alors que le symbole ⚠ peut être utilisé pour représenter une attente d'événement composé ou une attente d'événement dont on ne précise pas s'il s'agit d'un événement élémentaire ou composé. Le symbole ⚠ est donc plus général que le symbole △ et englobe ce dernier.

5.1.2. Consultations

Afin de permettre une personnalisation et une contextualisation de l'assistance, le système d'assistance doit disposer d'informations relatives à l'utilisateur, à l'application-cible et à l'assistance déjà fournie. Pour cette raison, aLDEAS propose plusieurs types de consultations élémentaires (cf. Fig. 5-8), représentées dans le langage par ◇ : la consultation de l'utilisateur, de l'état de l'application-cible ou d'informations liées à l'assistance. Une consultation renvoie une valeur dont le type varie selon la consultation, il peut s'agir d'un

nombre, d'un booléen ou d'un texte. Le type de valeur renvoyé est indiqué en gras sur la Fig. 5-8 (« -> Texte », « -> Booléen » ou -> « Nombre »).

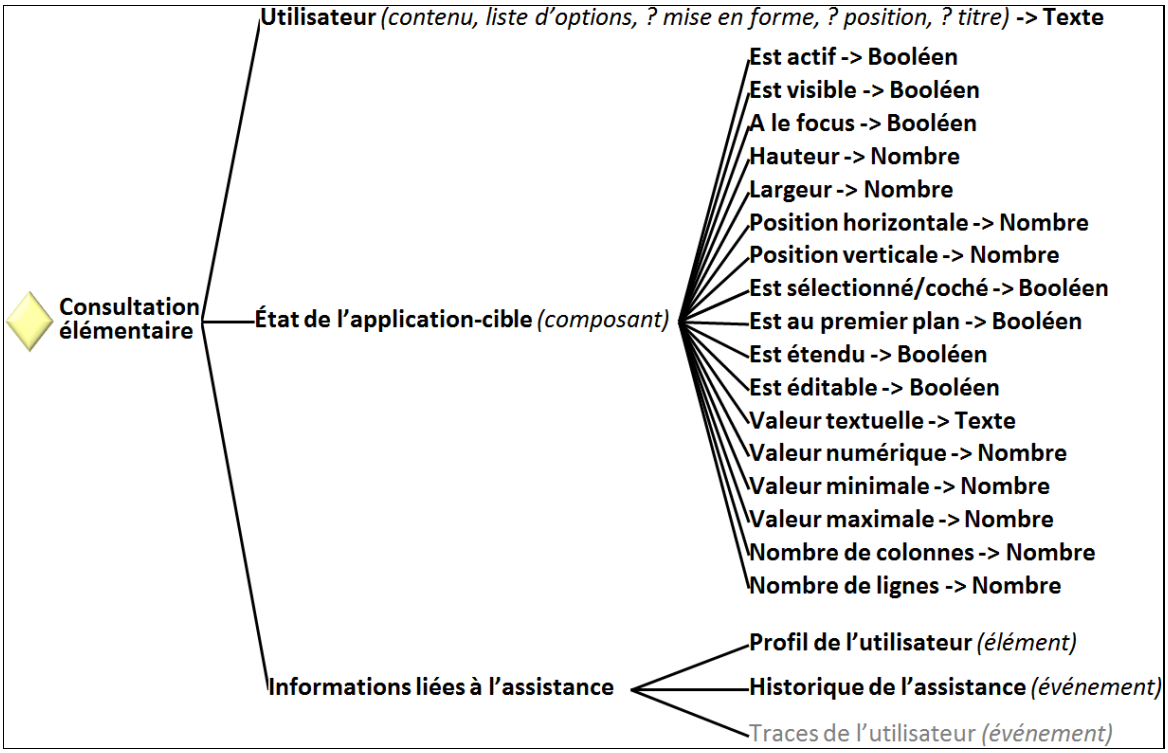


Fig. 5-8 – aLDEAS : consultations élémentaires

aLDEAS permet la **consultation de l'utilisateur**, pour lui proposer de l'aide ou lui demander ce qu'il souhaite faire par exemple. De telles consultations permettent d'adapter l'assistance aux choix de l'utilisateur, afin notamment d'éviter de lui proposer une assistance qui ne corresponde pas à ses besoins, et qu'il pourrait juger non pertinente ou trop envahissante. Une consultation de l'utilisateur est caractérisée par un contenu, par exemple une phrase, et une liste d'options entre lesquelles l'utilisateur peut choisir. Une consultation de l'utilisateur peut être associée à des paramètres optionnels, notamment de mise en forme. Elle renvoie un texte qui correspond à l'option choisie par l'utilisateur lors de la consultation. Ce texte est vide dans le cas où l'utilisateur n'a choisi aucune option. Par exemple, une consultation affichée dans une fenêtre pop-up renvoie un texte vide si l'utilisateur supprime la fenêtre sans avoir fait un choix parmi les options proposées.

La Fig. 5-9 présente un exemple de consultation de l'utilisateur, qui lui propose de découvrir Windows 8 et lui permet de choisir entre trois options : « oui », « me redemander plus tard » et « non ».

Fig. 5-9 – Exemple d'une consultation de l'utilisateur

Le langage aLDEAS permet également la **consultation de l'état de l'application-cible**, afin de connaître le texte d'une zone de saisie, ou de savoir quel item est sélectionné dans une liste déroulante par exemple. De telles consultations permettent de contextualiser l'assistance en fonction de l'état de l'application-cible, par exemple dans le but d'attirer l'attention de l'utilisateur sur des options sélectionnées par défaut par l'application-cible, ou sur une erreur dans un formulaire pré-rempli. Une consultation de l'état de l'application-cible est caractérisée par le composant de l'interface à laquelle elle se réfère. Selon le type de composant, différentes propriétés peuvent être consultées (cf. Fig. 5-10). Une consultation de l'état de l'application-cible renvoie une valeur dont le type dépend de la propriété consultée. Ainsi, une consultation de la propriété « Hauteur » d'un composant renvoie un nombre, qui correspond à sa hauteur en pixels, alors que la consultation de la propriété « Est visible » d'un composant renvoie un booléen qui indique si le composant est visible à l'écran au moment de la consultation.

Type de composant	**Propriété ->Type de valeur de retour**	
Tout composant	Est actif Est visible A le focus Hauteur Largeur Position horizontale Position verticale	-> booléen -> booléen -> booléen -> nombre -> nombre -> nombre -> nombre
Item, bouton radio, case à cocher	Est sélectionné Est coché	-> booléen -> booléen
Fenêtre	Est au premier plan	-> booléen
Arbre	Est étendu	-> booléen
Zone de saisie, liste déroulante, liste	Est éditable Valeur textuelle	-> booléen -> texte
Spinner, slider, barre de défilement	Valeur numérique Valeur minimale Valeur maximale	-> nombre -> nombre -> nombre
Table	Nombre de colonnes Nombre de lignes	-> nombre -> nombre

Fig. 5-10 –Propriétés consultables selon le type de composant dans aLDEAS

La Fig. 5-11 présente trois exemples de consultations de l'état de l'application-cible : C1 permet de connaître la valeur textuelle du composant d'identifiant 47 (il peut s'agir par exemple du texte contenu dans une zone de saisie) ; C2 permet de savoir si le composant 60 (par exemple un item d'une liste déroulante ou un bouton radio) est sélectionné ; et C3 permet de connaître la hauteur du composant 124 (par exemple une image ou une fenêtre).

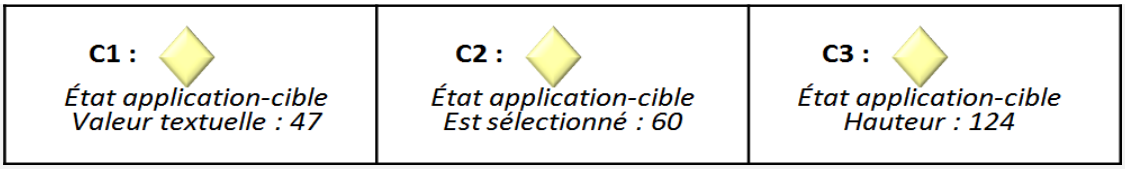


Fig. 5-11 – Exemples de consultations de l'état de l'application-cible

aLDEAS permet également la consultation d'**informations liées à l'assistance**, à condition que celles-ci soient exprimées selon un formalisme connu (cf. Chapitre 6). Nous regroupons dans cette catégorie toutes les informations qui peuvent être utilisées pour adapter l'assistance à un utilisateur final, en particulier son profil, l'historique de l'assistance qui lui a déjà été fournie et ses traces d'interactions avec l'application-cible. Une consultation du profil de l'utilisateur, qui peut notamment contenir des informations sur ses préférences en matière d'assistance, permet de connaître la valeur d'un élément donné du profil de l'utilisateur. De telles consultations permettent de personnaliser l'assistance en fonction des spécificités de l'utilisateur final. Une consultation de l'historique de l'assistance permet d'accéder à des informations relatives à un événement lié à l'assistance, comme le nombre de déclenchements d'une règle d'assistance dans une période donnée, ou la durée écoulée entre deux déclenchements d'une action d'assistance pour l'utilisateur. De telles consultations permettent d'adapter l'assistance en fonction de l'assistance déjà proposée, et ainsi de la faire évoluer. Une consultation des traces de l'utilisateur permet d'obtenir des informations relatives aux événements liés aux actions de l'utilisateur, afin de savoir par exemple s'il a déjà réalisé une action donnée et depuis combien de temps. De telles consultations permettent notamment d'adapter l'assistance en fonction de l'expérience de l'utilisateur final, en tenant compte notamment des tâches qu'il a déjà réalisées.

La Fig. 5-12 présente trois exemples de consultations élémentaires : C4 permet de connaître la valeur associée à l'élément « Niveau d'assistance souhaité » du profil de l'utilisateur ; C5 est une consultation de l'historique de l'assistance qui permet d'obtenir le nombre de déclenchements de l'action d'assistance A4 dans la journée courante ; et C6 est une consultation des traces de l'utilisateur qui permet de connaître la durée écoulée depuis la dernière occurrence de l'événement E0, c'est-à-dire la durée écoulée depuis que l'utilisateur a cliqué sur le composant 364 de l'application-cible pour la dernière fois (cf. E0 sur la Fig. 5-4).

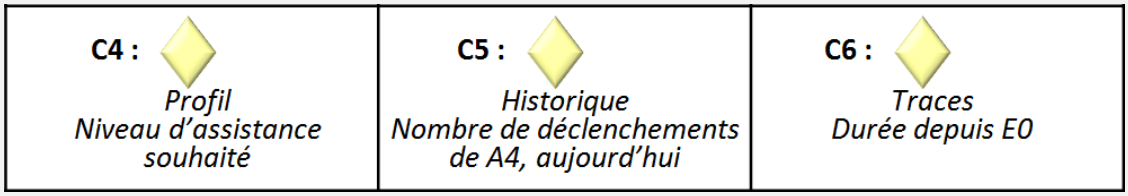


Fig. 5-12 – Exemples de consultations de ressources liées à l'assistance

Les consultations élémentaires renvoient une valeur dont le type dépend de la nature de la consultation : booléen, texte, entier ou durée ; une durée étant un entier associé à une unité de temps. Pour les exemples donnés sur la Fig. 5-12, notons ainsi que C4 renvoie une valeur textuelle, le niveau d'assistance souhaité par l'utilisateur (par exemple « beaucoup d'assistance », « peu d'assistance » ou « aucune assistance »), C5 renvoie un entier

correspondant au nombre de déclenchements de A4 ; et C6 renvoie une durée correspondant au temps écoulé depuis E0.

Dans aLDEAS, les éléments renvoyant une valeur peuvent être suivis d'une alternative à n branches chacune associée à un test dont le type dépend du type de valeur renvoyé. Par exemple pour un texte, les tests peuvent imposer qu'il soit égal, qu'il contienne, commence ou se termine par un texte donné.

La Fig. 5-13 présente trois exemples de consultations élémentaires suivies d'alternatives. Si un test est vérifié, l'élément suivant de la branche est déclenché.

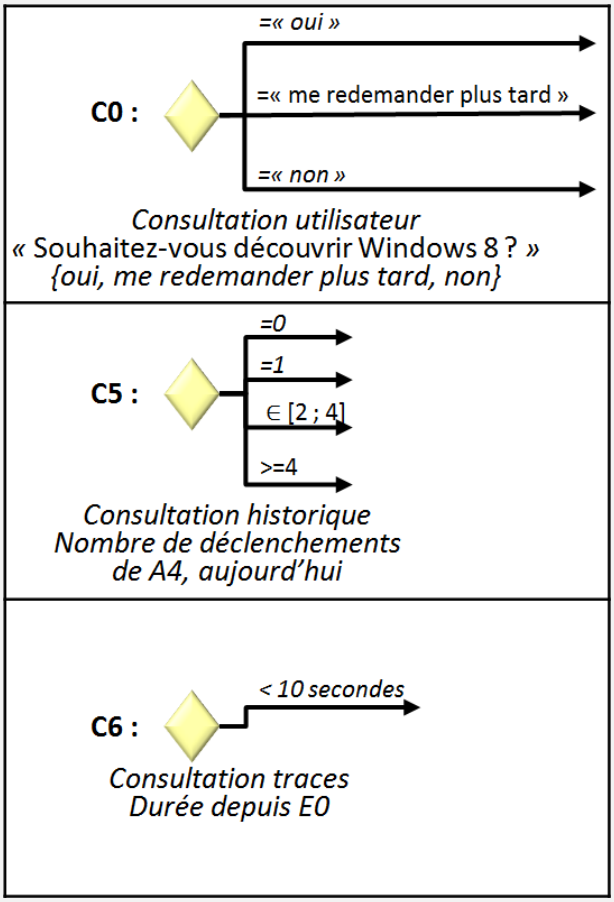


Fig. 5-13 – Exemples de consultations élémentaires suivies d'alternatives

Notons que plusieurs éléments peuvent être déclenchés en parallèle si la valeur de retour vérifie plusieurs tests. Par exemple, en supposant que la valeur de retour de C5 soit 4 (cf. C5 sur la Fig. 5-13), alors 4 vérifie $4 \in [2 ; 4]$ et $4 \geq 4$. De même, il est possible qu'aucun test ne soit vérifié, par exemple en supposant que la valeur de retour de C6 soit 3 minutes (cf. C6 sur la Fig. 5-13), alors le test 3 minutes < 10 secondes de l'unique alternative qui suit C6 n'est pas vérifié.

Les consultations élémentaires peuvent être combinées par une formule logique afin de créer une consultation composée, représentée avec aLDEAS par . Une formule logique permet de combiner uniquement des valeurs booléennes, c'est pourquoi, pour permettre la combinaison de consultations élémentaires renvoyant une valeur autre qu'un booléen, il est nécessaire de les associer à un test. Par exemple, la consultation élémentaire de l'utilisateur

C0 (cf. C0 sur la Fig. 5-9) renvoie un texte, néanmoins, C0 associé au test « = oui » renvoie un booléen et peut donc être utilisé dans une formule logique afin de créer une consultation non élémentaire. On notera C(t) l'association d'une consultation C avec un test t.

La Fig. 5-14 présente l'exemple de C7, une consultation composée de deux consultations élémentaires, C4 et C5, chacune associée à un test. Toutes les consultations composées, à l'exemple de C7, renvoient un booléen, puisqu'il s'agit de formules logiques.

C7 : C4(= « beaucoup d'assistance ») && C5(<3)

Fig. 5-14 – Exemple de C7, une consultation composée de deux consultations élémentaires

Notons que dans aLDEAS le symbole est utilisé uniquement pour représenter une consultation élémentaire, alors que le symbole peut être utilisé pour représenter une consultation composée ou une consultation dont on ne précise pas si elle est élémentaire ou composée.

5.1.3. Actions élémentaires d'assistance

Afin de répondre efficacement et de manière personnalisée aux besoins d'assistance les plus variés des utilisateurs, un système d'assistance doit être capable de fournir des actions d'assistance riches et diversifiées. aLDEAS propose deux catégories d'actions élémentaires, représentées dans le langage par : des actions intégrées et des actions extérieures à l'interface de l'application-cible (cf. Fig. 5-15). Certaines actions d'assistance peuvent être spécialisées : les possibilités de spécialisation d'une action d'assistance sont indiquées par des traits en pointillés. Sur la Fig. 5-15, les quatre éléments grisés correspondent à des actions d'assistance qui n'ont actuellement pas été mises en œuvre dans SEPIA : les mises en valeur par animation (comme les fenêtres de connexion sous Mac OS, agitées de secousses visuelles en cas d'erreur), la transmission de messages en Braille ou en langue des signes et le retour haptique.

Les actions intégrées à l'interface de l'application-cible (cf. partie supérieure de la Fig. 5-15) agissent directement sur l'interface de l'application-cible et concernent un composant donné, comme un bouton ou une zone de saisie. Le langage propose quatre types d'actions intégrées à l'interface de l'application-cible : **action automatisée**, pour agir à la place de l'utilisateur ; **ajout de composant**, pour enrichir l'interface de l'application-cible, par exemple par un bouton permettant de demander de l'aide ; **mise en valeur**, pour guider l'utilisateur et attirer son attention sur un composant ; et **masquage**, pour simplifier aux yeux de l'utilisateur l'interface de l'application-cible.

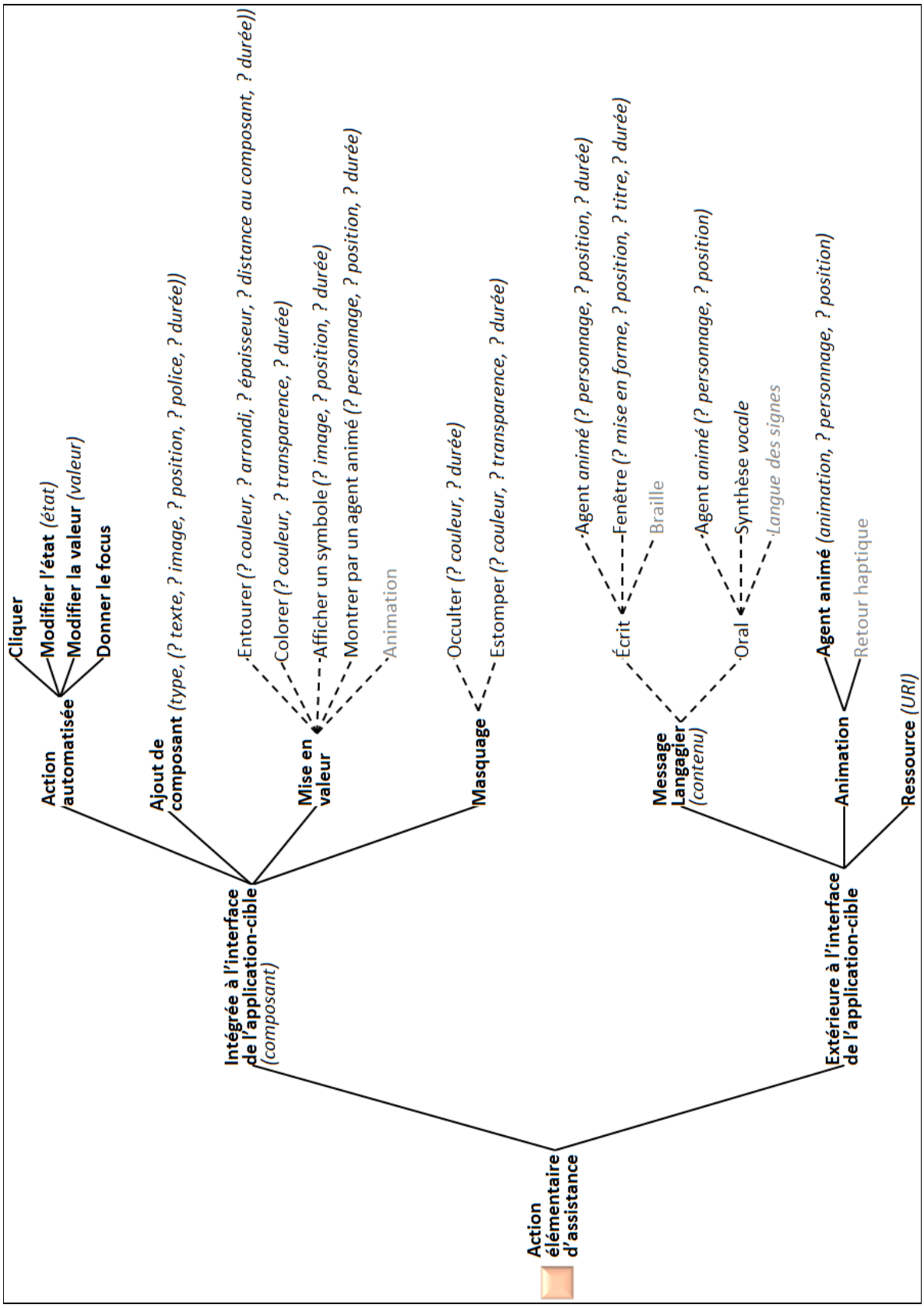


Fig. 5-15 – aLDEAS : actions élémentaires d'assistance proposées

Les actions extérieures à l'interface de l'application-cible permettent quant à elles de proposer à l'utilisateur de l'assistance non associée à un composant de l'interface de l'application-cible. Notre langage en propose trois types (cf. partie inférieure de la Fig. 5-15) : les **messages langagiers**, associés à un contenu textuel pouvant être affiché et/ou lu, par exemple un conseil ou une suggestion ; les **animations**, par exemple un agent animé qui applaudit ; et les **ressources** qui peuvent être proposées à l'utilisateur, par exemple une vidéo de démonstration, un forum ou une application comme la calculatrice.

La Fig. 5-16 présente l'exemple de trois actions élémentaires d'assistance : A0 est une action de modification de l'interface de l'application-cible qui permet de sélectionner automatiquement le composant d'identifiant 46 (par exemple un bouton radio) ; A1 est une action de modification de l'interface de l'application-cible qui permet de mettre en valeur le composant 78 en l'entourant en bleu ; et A2 est une action extérieure à l'interface de l'application-cible qui permet de transmettre à l'utilisateur le message « N'oublie pas de saisir ton adresse mail ».

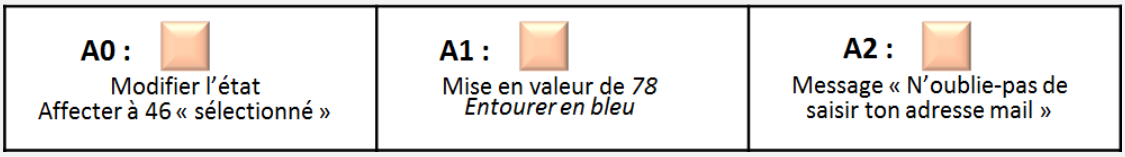


Fig. 5-16 – Exemples d'actions élémentaires d'assistance

Il est à noter qu'une action extérieure à l'interface de l'application-cible peut malgré tout être liée à un composant de cette interface. En effet, certaines actions extérieures à l'interface de l'application-cible (les messages langagiers et les animations) ont un paramètre optionnel de position. Or, une position peut être définie par des coordonnées absolues (par exemple x=160, y=90), relatives à l'écran (par exemple centré sur l'écran en x et en haut de l'écran en y), ou relatives à la position d'un composant (par exemple à droite ou au-dessus du composant). Ainsi, l'action A2 (cf. Fig. 5-16) pourrait avoir un paramètre de position relatif au champ de saisie de l'adresse mail, afin que le message soit affiché à proximité du composant sur lequel l'utilisateur est invité à agir.

5.1.4. Définition d'actions composées

Tous les éléments proposés par le langage aLDEAS peuvent être combinés pour créer un bloc de composants correspondant à des actions d'assistance composées, représentées dans le langage par . Une action composée est un **bloc aLDEAS** délimité par les symboles ●— et —● ; le symbole —▶ déclenchant l'élément suivant.

Lors de l'exécution d'un bloc aLDEAS, lorsqu'un composant de type attente d'événement (élémentaire ou composé) ou de type consultation de l'utilisateur est atteint, l'exécution est bloquée jusqu'à ce que l'événement attendu ait eu lieu ou jusqu'à ce que l'utilisateur ait fourni une réponse. Si un composant de type alternative est atteint, l'exécution se poursuit en parallèle dans toutes les branches dont le test est vérifié ; si aucun test n'est vérifié, l'exécution du bloc prend automatiquement fin, ce qui équivaut à dire que chaque composant de type alternative contient implicitement une branche associée au test « else » qui conduit à la fin du bloc, ce qui permet d'alléger la notation.

Notons que dans aLDEAS le symbole ▭ est utilisé uniquement pour représenter une action élémentaire, alors que le symbole ▯ peut être utilisé pour représenter une action composée ou une action dont on ne précise pas si elle est élémentaire ou composée.

La Fig. 5-17 donne l'exemple de deux actions composées : A3 est composée d'une consultation élémentaire qui renvoie un texte correspondant à la valeur d'un composant, si ce texte est vide, une action de type message est déclenchée. A4 est composée de deux actions élémentaires d'assistance : un message et une mise en valeur, suivies d'une attente d'un événement élémentaire (un timer fixe de 30 secondes) et du marqueur de fin qui provoquera la suppression du message et de la mise en valeur 30 secondes après leur apparition.

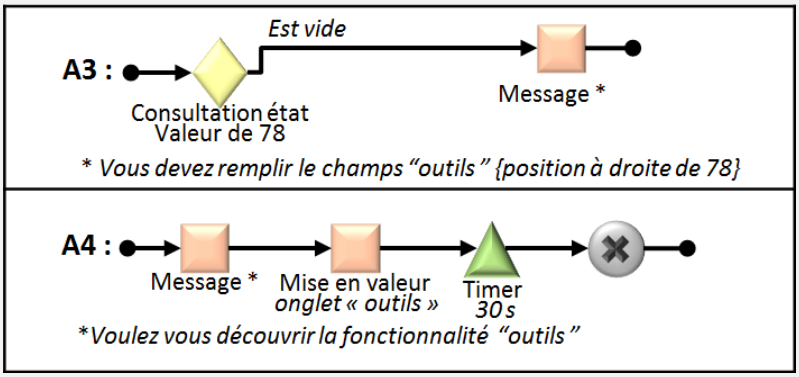


Fig. 5-17 – Exemple de deux actions d'assistance composées

Une action peut renvoyer une valeur de type texte, indiquée dans le langage sous le symbole de fin de bloc. Les actions composées renvoyant une valeur peuvent être suivies d'une alternative à n branches dont le fonctionnement est identique à celui des alternatives qui suivent une consultation, présenté en section 5.1.2.

La Fig. 5-18 présente l'exemple de trois actions composées : A5, A6 et A7. Nous observons en particulier que l'action A5 renvoie une valeur qui peut être égale à « indice » ou à « réponse » selon le choix de l'utilisateur lors de la consultation. Les actions A5 et A6 sont exploitées par l'action A7 : A5 est tout d'abord déclenchée puis en fonction de sa valeur de retour, soit A6 est déclenchée et un message est affiché, soit il ne se passe rien dans le cas où A5 ne renvoie rien, c'est-à-dire lorsque l'utilisateur a répondu « non merci » à la consultation contenue dans A5.

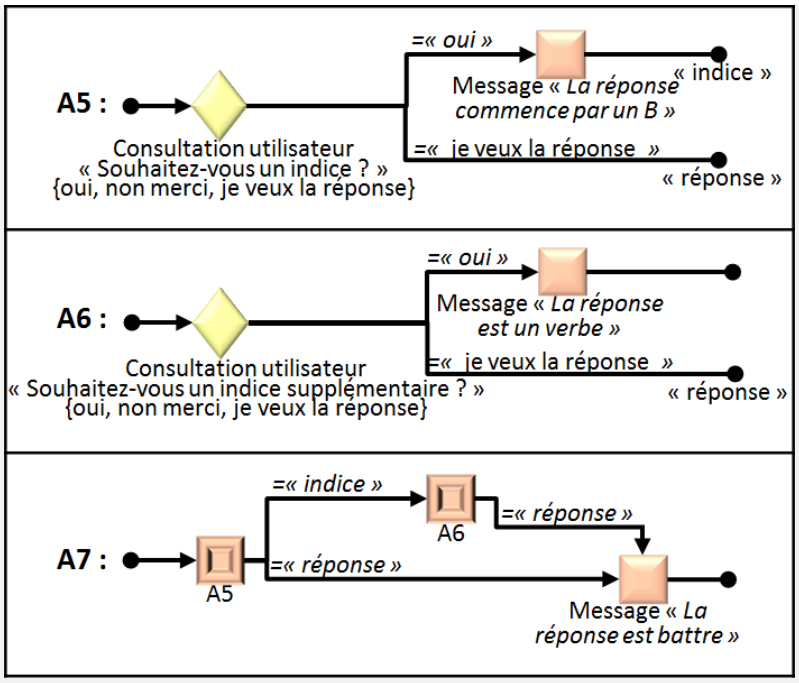


Fig. 5-18 – Exemple des trois actions composées

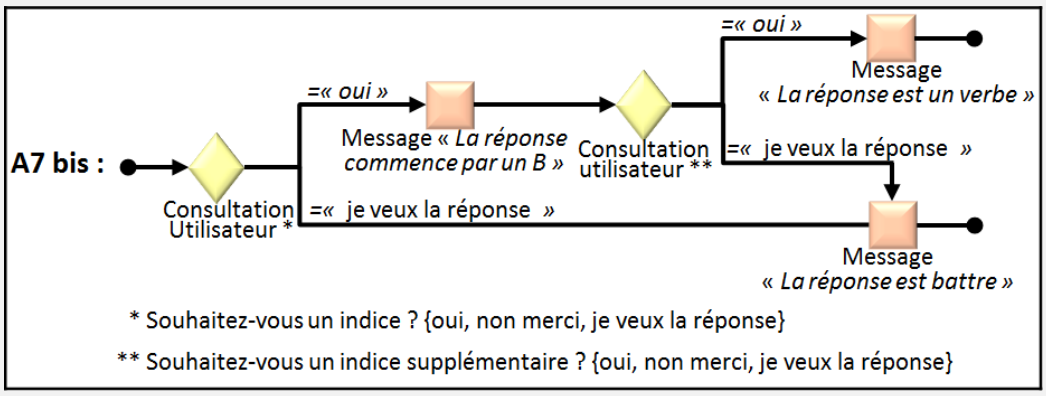


Fig. 5-19 – Exemple de A7 bis, une action composée équivalente à A7 (cf. Fig. 5-18)

Notons que aLDEAS permet de définir de différentes manières des actions pourtant équivalentes. Ainsi, dans l'exemple présenté par la Fig. 5-18, il serait possible avec aLDEAS de définir une unique action équivalente à A7 sans exploiter des actions « intermédiaires » comme A5 et A6 (cf. Fig. 5-19). Néanmoins, dans le cas d'une action composée de très nombreux composants, l'exploitation d'actions « intermédiaires » permet de gagner en clarté et en lisibilité.

5.2. Patrons d'actions composées

Afin de faciliter la définition avec aLDEAS d'actions composées, nous proposons un ensemble de patrons. Pour cela nous enrichissons notre langage avec deux structures supplémentaires : l'embranchement « ou » et les éléments optionnels précédés d'un « ? ». Contrairement aux composants d'aLDEAS (cf. Fig. 5-1) présentés dans la section précédente,

le « ou » et le « ? » ne décrivent pas l'*exécution* de l'assistance : ils représentent des choix à faire par le concepteur de l'assistance au moment de l'*instanciation* du patron.

Par la suite, on appelle **événement déclencheur** (cf. Fig. 5-20) une attente d'événement, élémentaire ou composé, qui débute un bloc. Un événement déclencheur permet de définir dans quel cas une assistance va être proposée à l'utilisateur final. On appelle **événement de fin** (cf. Fig. 5-20) une attente d'événement, élémentaire ou composé, immédiatement suivie du marqueur de fin représenté dans le langage par ⊗. Le marqueur de fin provoque en effet la fin de toutes les actions lancées depuis le début du bloc et non encore terminées. Un événement de fin permet donc de définir quand une assistance proposée va prendre fin.

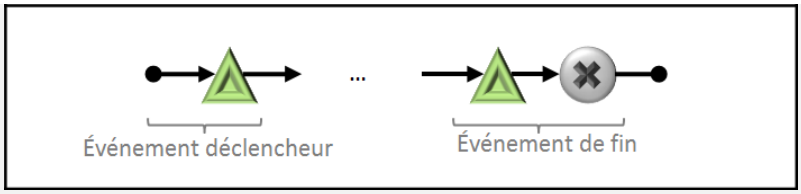


Fig. 5-20 – Exemple d'une action composée contenant un événement déclencheur et un événement de fin

5.2.1. Patrons de règles d'assistance

Le but de notre langage est de permettre à des concepteurs de spécifier par un ensemble de règles l'assistance qu'ils souhaitent pour une application-cible. Dans aLDEAS, nous définissons une règle d'assistance comme une action composée qui respecte le patron donné en Fig. 5-21.

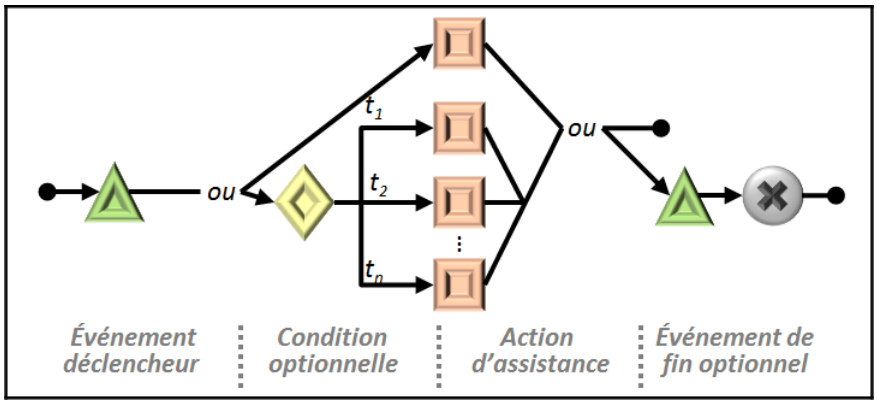


Fig. 5-21 – Patron de règles d'assistance

Une règle contient tout d'abord un événement déclencheur, qui correspond à l'attente d'un événement élémentaire ou composé. Une règle contient ensuite une action d'assistance ou une consultation avec alternatives, chacune associée à une action d'assistance. Une règle peut se terminer par un événement de fin. Si une règle n'est pas associée à un événement de fin, elle ne se terminera que lorsque toutes les actions qu'elle a déclenchées seront achevées. En effet, une action d'assistance peut être associée à son propre événement de fin, et dans certains cas, l'utilisateur peut lui-même mettre fin à une action. Par exemple,

dans le cas d'un message affiché par une fenêtre pop-up, l'utilisateur peut mettre fin à l'action en fermant la fenêtre.

Comme on peut le voir sur la Fig. 5-21, une règle d'assistance dans aLDEAS est de la forme <événement déclencheur, condition, action, événement de fin>. Le patron de règles d'assistance ne prévoit donc qu'une seule action assistance par branche représentée par . Rappelons néanmoins que si le symbole peut représenter une action élémentaire, il peut aussi représenter une action composée qui peut contenir une combinaison complexe de nombreuses actions élémentaires et d'autres composants d'aLDEAS.

La Fig. 5-22 donne l'exemple de deux règles d'assistance, R0 et R7, créées pour l'application-cible PhotoScape[8] (cf. Annexe B), un logiciel gratuit de retouche d'images. La règle R0 contient un événement déclencheur (le lancement de l'assistance) et une consultation de l'utilisateur lui demandant s'il souhaite de l'aide. R1 sera déclenchée si l'utilisateur choisit l'option « oui, je veux de l'aide ». La règle R7 est déclenchée par le clic de l'utilisateur sur le composant d'identifiant 228 (qui correspond à l'onglet « outils » de PhotoScape). R7 déclenche une action d'assistance constituée de deux actions élémentaires d'assistance : la mise en valeur du composant d'identifiant 210 (le bouton « yeux rouges » de PhotoScape) et un message enjoignant à l'utilisateur de cliquer sur ce bouton. Le clic de l'utilisateur sur le bouton « yeux rouges » mettra fin à la règle R7 et provoquera donc l'effacement de la mise en valeur et du message.

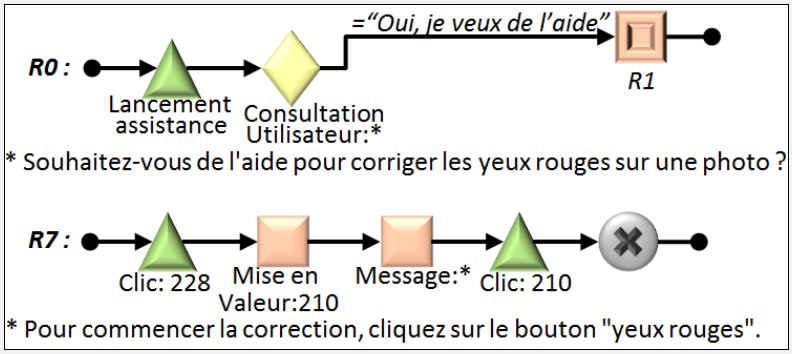


Fig. 5-22 – Exemple des règles d'assistance R0 et R7 pour PhotoScape

5.2.2. Patrons d'actions d'assistance composées

Le langage aLDEAS permet la définition d'actions d'assistance complexes, combinant de nombreux éléments d'assistance. La définition de telles actions peut être difficile, or aLDEAS s'adresse principalement à des concepteurs d'assistance qui ne sont pas nécessairement informaticiens. Pour cette raison, nous avons défini des patrons d'actions composées, associés à notre langage, afin de faciliter la définition de certaines actions composées fréquemment présentes dans les systèmes d'assistance existants : les actions d'agent animé, les présentations guidées et les pas à pas.

[8] http://www.photoscape.org

Une **action d'agent animé** permet de combiner plusieurs actions élémentaires d'un même personnage : messages, mises en valeur de composants, animations (montrer un composant, applaudir, saluer…), et déplacements à l'écran. Nous proposons en Fig. 5-23 un patron pour définir de telles actions. Ces actions contiennent une succession de n étapes représentées par [], où n est un entier défini lors de l'instanciation du patron. Les étapes d'action d'agent animé doivent respecter le patron d'étape d'action d'agent animé donné en Fig. 5-24 : une étape contient une action élémentaire d'agent animé (message, animation, mise en valeur ou déplacement), et éventuellement un événement de fin.

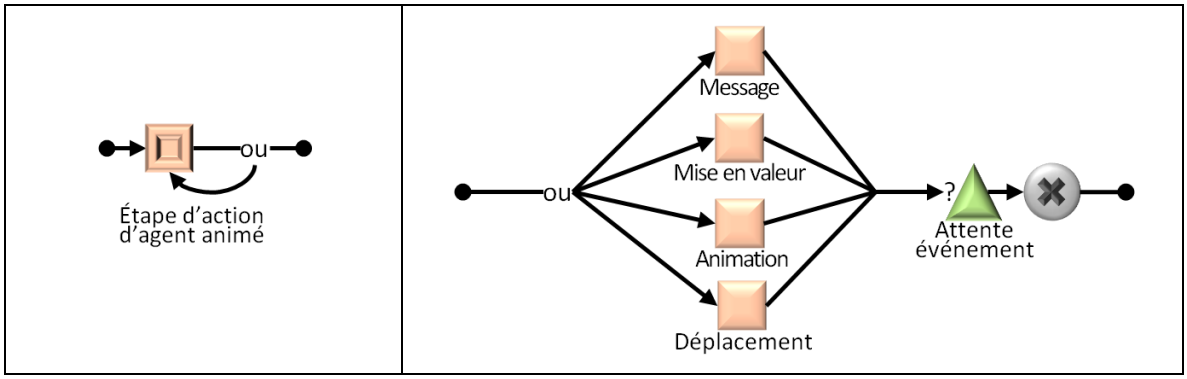


Fig. 5-23 – Patron d'action d'agent animé

Fig. 5-24 – Patron d'étape d'action d'agent animé

La Fig. 5-25 donne un exemple d'instanciation du patron d'action d'agent animé avec trois étapes : l'agent animé se place à droite du champ e-mail (désigné par « composant 78 »), il affiche le message « n'oublie-pas de saisir ton adresse mail », puis montre le champ e-mail jusqu'à ce que l'utilisateur ait modifié sa valeur.

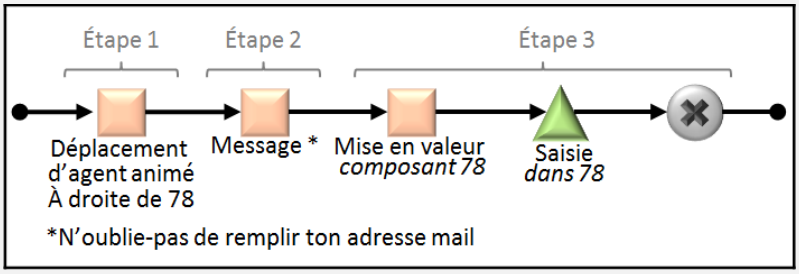


Fig. 5-25 – Exemple d'instanciation du patron d'action d'agent animé

Une **présentation guidée** comporte plusieurs étapes, dans lesquelles un composant de l'interface de l'application-cible est mis en valeur et éventuellement présenté par un message. Une présentation guidée permet de représenter une connaissance relative aux fonctionnalités proposées par l'application-cible. Une action d'assistance de type *présentation guidée* a pour objectif de répondre à un besoin de découverte de l'application-cible (cf. Section 1.2.1). On retrouve fréquemment ces actions d'assistance dans les applications existantes, notamment lorsqu'une application est lancée pour la première fois afin de présenter à l'utilisateur les fonctionnalités de l'application, ou à la suite d'une mise à jour afin de présenter à l'utilisateur les modifications apportées à l'application. L'0 donne un exemple d'instanciation de ce patron, pour la présentation guidée du jeu en ligne « Dora l'exploratrice : le spectacle de fin d'année » (cf. Fig. 7, Annexe A-1). Nous avons proposé un

patron pour les présentations guidées (cf. Fig. 5-26) : il permet de définir des actions comportant une succession de n étapes représentées par , où n est un entier défini lors de l'instanciation du patron, ces étapes pouvant être précédées et suivies d'un message. Les étapes de présentation guidée doivent respecter un patron donné en Fig. 5-27 : une étape de présentation guidée peut contenir un message dont le but est de présenter à l'utilisateur le composant concerné par l'étape, ainsi qu'une mise en valeur de ce composant et un timer indiquant la durée de l'étape.

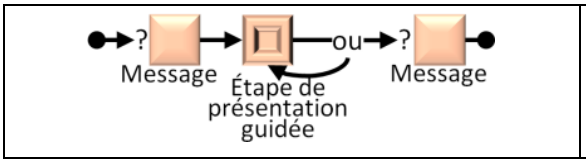


Fig. 5-26 – Patrons de présentation guidée

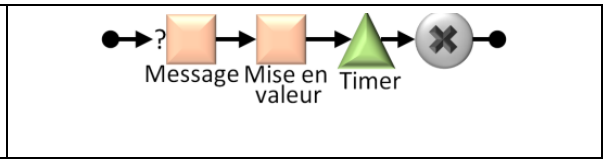


Fig. 5-27 – Patrons d'étape de présentation guidée

La Fig. 5-28 présente un exemple d'action d'assistance respectant le patron de présentation guidée (cf. Fig. 5-26). Il s'agit d'une présentation guidée des 10 principaux menus de Word 2010. On note que cette présentation guidée contient un message avant les étapes mais pas de message après : ce choix relève du concepteur de l'assistance lors de l'instanciation du patron de présentation guidée proposé en complément d'aLDEAS. La Fig. 5-29 présente quant à elle un exemple d'étape de présentation guidée respectant le patron donné en Fig. 5-27. Cette étape permet de présenter le menu « Fichier » de Word 2010 par un message et une mise en valeur durant 10 secondes.

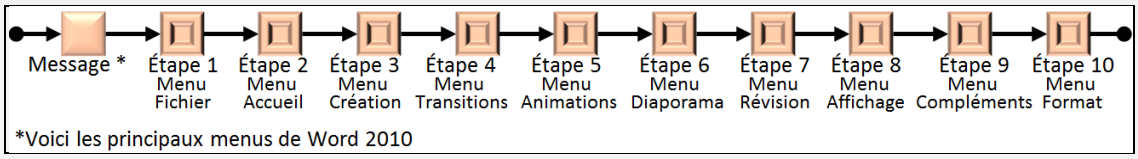


Fig. 5-28 – Exemple de présentation guidée pour Word 2010

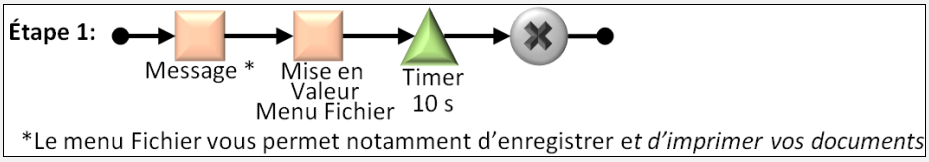


Fig. 5-29 – Exemple d'étape de présentation guidée pour Word 2010

Un **pas à pas** vise à faciliter la réalisation d'une tâche en la détaillant sous forme de plusieurs étapes. Chaque étape correspond à une action à réaliser sur un composant de l'interface de l'application-cible. Un pas à pas permet de représenter une connaissance relative à la réalisation d'une tâche donnée dans l'application-cible. Une action d'assistance de type *pas à pas* a pour objectif de répondre à un besoin relatif à la réalisation d'une tâche (cf. Section 1.2.1). Nous appelons *pas à pas automatisé* un pas à pas dans lequel le système d'assistance va réaliser les actions à la place de l'utilisateur. Nous appelons *pas à pas guidé* un pas à pas dans lequel le système va demander à l'utilisateur de réaliser lui-même les actions. Les patrons d'étapes de pas à pas automatisé et guidé sont donnés respectivement

en Fig. 5-31 et Fig. 5-32 : ces deux patrons d'étapes sont exploités par le patron de pas à pas (cf. Fig. 5-30), sur lequel les étapes sont représentées par ▯.

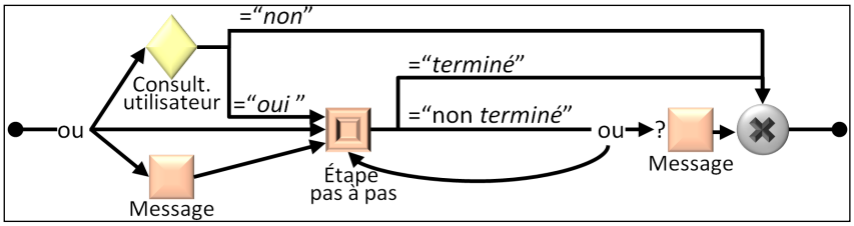


Fig. 5-30 – Patron de pas à pas

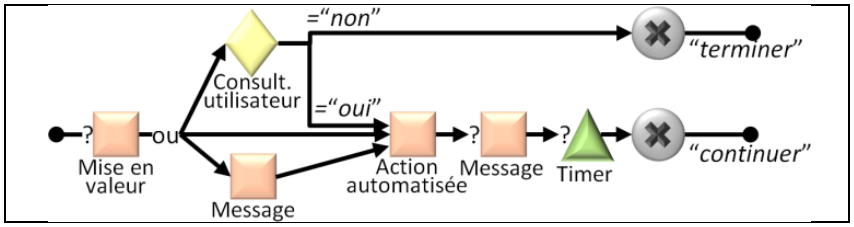


Fig. 5-31 – Patron d'étape de pas à pas, en mode automatisé

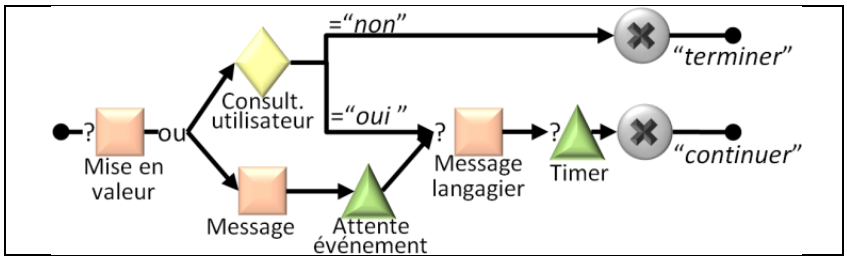


Fig. 5-32 – Patron d'étape de pas à pas, en mode guidé

La Fig. 5-33 présente un exemple d'action composée respectant le patron de pas à pas donné en Fig. 5-30. Ce pas à pas concerne la calculatrice Windows et comporte 5 étapes respectant le patron d'étape de pas à pas et permet la saisie automatique de la formule « 42/6 = ». Une telle action d'assistance peut par exemple être proposée dans le cadre d'un système d'assistance pour un logiciel pédagogique de mathématiques. Ainsi, si les objectifs pédagogiques ne concernent pas directement le calcul, l'enseignant peut juger pertinent de proposer à un apprenant en difficulté une assistance qui lance la calculatrice, puis automatise la saisie d'une formule, afin d'éviter qu'il ne soit bloqué dans l'activité pédagogique. La Fig. 5-34 présente à titre d'exemple la première étape de ce pas à pas : cette étape concerne le bouton « 4 » de la calculatrice. Pendant cette étape, le bouton 4 est actionné automatiquement et mis en valeur durant 3 secondes avant que cette étape ne prenne fin et que l'étape suivante ne soit déclenchée.

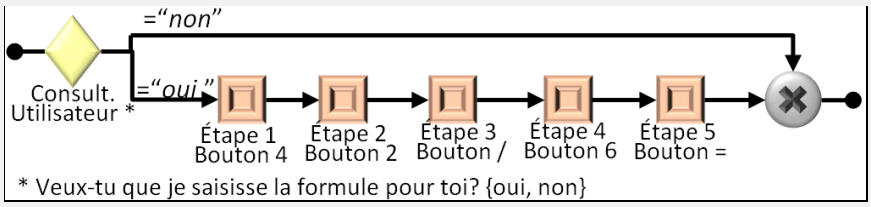


Fig. 5-33 – Exemple de pas à pas pour la calculatrice Windows

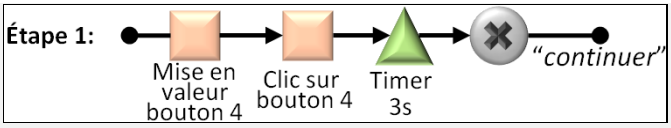


Fig. 5-34 – Exemple d'étape de pas à pas automatisé pour la calculatrice Windows

5.3. Conclusion

Dans ce chapitre, nous avons présenté le langage graphique aLDEAS, qui est un langage pivot entre la phase de spécification de l'assistance par un concepteur d'assistance et la phase d'exécution automatique de cette assistance par un outil générique. aLDEAS permet la définition formelle de systèmes d'assistance pouvant être intégrés aux applications les plus variées. Nous avons complété aLDEAS par un ensemble de patrons permettant de faciliter la définition d'actions composées fréquemment utilisées dans les systèmes d'assistance existants et qui seraient difficiles à définir sans l'usage de patrons en raison de leur complexité ou de leur nombre important de composants. Dans la suite de ce manuscrit, nous définissons un système d'assistance aLDEAS comme un système d'assistance défini par un ensemble de règles respectant le patron de règles d'assistance aLDEAS.

L'assistance définie à l'aide d'aLDEAS peut être adaptée et contextualisée finement à son application-cible par l'emploi d'attentes d'événements liés aux interactions entre l'utilisateur et l'interface de l'application-cible, ainsi que par l'emploi de consultations de l'état de l'application-cible. En effet, pour être efficace, une assistance doit être fournie uniquement lorsqu'elle est nécessaire et un besoin d'assistance peut être détecté en fonction des interactions entre l'utilisateur et l'interface de l'application-cible et en fonction de l'état de l'application-cible.

De plus, l'assistance proposée à l'utilisateur final de l'application-cible peut être personnalisée par l'emploi de consultations de son profil et de ses traces d'interactions avec l'interface de l'application-cible. La définition d'une assistance pleinement personnalisée pour l'utilisateur peut augmenter son efficacité et sa bonne acceptation par ce dernier. L'assistance doit donc être adaptée autant que possible aux spécificités de chaque utilisateur, notamment son expérience, ses préférences en matière d'assistance, ses connaissances et compétences relatives à l'application-cible, son expérience générale et son expérience d'utilisation de l'application-cible en particulier. Un besoin d'assistance peut en effet être détecté en fonction des traces de l'utilisateur et de son profil lorsque celui-ci contient des informations sur des connaissances impliquées dans la réalisation de sa tâche courante. Par exemple, un besoin d'assistance pourra être supposé lorsque les traces d'un utilisateur indiquent qu'il n'a jamais utilisé une certaine fonctionnalité de l'application-cible et que son profil indique qu'il est novice dans l'utilisation de cette application ou s'il indique que l'utilisateur souhaite être assisté pour les fonctionnalités qu'il découvre.

L'emploi de consultations de l'utilisateur peut également rendre l'assistance plus adaptée à l'utilisateur, à ses préférences et à ses objectifs. En effet, fournir à un utilisateur une assistance correspondant à un besoin mal identifié ou qui ne correspond pas à ses objectifs peut perturber l'utilisateur et lui faire perdre confiance dans le système d'assistance, ce qui pourra le conduire à ignorer les prochaines actions d'assistance fournies par le système, voire

à désactiver le système d'assistance lorsque cela est possible [Randall et Pedersen, 1998 ; Galluccio, 2006]. En consultant l'utilisateur, on peut donc s'assurer d'avoir bien détecté ses besoins et objectifs. Ce type de consultation doit néanmoins être utilisé avec parcimonie. En effet une intervention trop fréquente du système d'assistance pour solliciter l'avis de l'utilisateur peut le rendre trop intrusif et perturber l'utilisateur autant qu'en lui proposant une assistance inadaptée.

Par ailleurs, l'assistance définie à l'aide d'aLDEAS peut évoluer automatiquement en fonction de l'expérience de l'utilisateur, grâce à l'emploi de consultations des traces de l'utilisateur, du profil de l'utilisateur et de l'historique de l'assistance. Ainsi, aLDEAS permet de fournir à un utilisateur une assistance différente selon les actions d'assistance qui lui ont déjà été proposées et l'effet qu'elles ont eu. En effet, si l'on constate que l'utilisateur se retrouve pour la seconde fois dans une situation problématique identique et que les actions d'assistance qui lui avaient été proposées la première fois n'avaient pas permis de répondre à ce besoin d'assistance, alors il peut être pertinent de lui proposer une autre action d'assistance qui tient compte des difficultés rencontrées par l'utilisateur la première fois.

Enfin, aLDEAS permet la personnalisation de l'assistance fournie à l'utilisateur en fonction de ses préférences, de ses capacités et éventuels handicaps. Ainsi, le mode de transmission d'une action d'assistance pourra être personnalisé en fonction de son profil. Par exemple, dans le cas d'un utilisateur avec une vision imparfaite, le système d'assistance peut utiliser les polices et couleurs par défaut de l'ordinateur afin d'adapter une action de type message textuel, et accompagner celle-ci d'un message oral équivalent. Dans le cas d'un ordinateur partagé, par exemple lors de l'utilisation à l'école d'un logiciel pédagogique, le système d'assistance pourra utiliser les polices et couleurs préférées de l'utilisateur si celui-ci les a indiquées dans son profil. La prise en compte des préférences de l'utilisateur peut également permettre de rendre l'assistance plus attractive et mieux acceptée par l'utilisateur, notamment si ses préférences indiquent le niveau d'assistance qu'il souhaite ou les paramètres optionnels qu'il préfère en fonction du type d'action d'assistance fournie, tel que le mode de transmission d'un message (comme une fenêtre pop-up, un agent animé ou une synthèse vocale), ou le mode d'affichage d'une mise en valeur.

Chapitre 6. Adaptation de l'assistance

Objectif du chapitre

Rendre possible l'adaptation de l'assistance en fonction des spécificités de l'utilisateur final, ainsi que de l'assistance qui lui a déjà été proposée.

Points clés

Le langage aLDEAS permet de consulter des informations contenues dans le profil de l'utilisateur et dans l'historique de l'assistance pour définir des systèmes d'assistance. Cette consultation n'est possible que si le **formalisme** de ces différentes informations est connu.

Nous présentons ici le formalisme de **profils d'utilisateurs** que nous avons choisi d'utiliser. Nous proposons également un formalisme simple pour l'**historique de l'assistance**.

Contribution

- Formalisme d'historique de l'assistance

Validation

- ✓ Identification d'un désir de mettre en place de l'assistance personnalisée pour les concepteurs d'assistance
- ○ Identification d'un désir de recevoir de l'assistance personnalisée pour les utilisateurs finaux
- ○ Efficacité d'une assistance personnalisée pour la réalisation d'une tâche

Publications liées à ce chapitre

[Ginon et al., 2011]

Ginon B., Jean-Daubias S. et Lefevre M., Evolutive learners profiles, ED-MEDIA 2011 - World Conference on Educational Multimedia, Hypermedia & Telecommunications, 2011.

[Ginon et al., 2012a]

Ginon B., Champin P.-A. et Jean-Daubias S., Taking into account users' knowledge, abilities and preferences to personalize animated assistant agents, Workshop PALE associated with UMAP, Montreal, Canada, pp. 29-34, 2012a.

[Jean-Daubias et Ginon, 2010]

Jean-Daubias S. et Ginon B., Des profils d'apprenant évolutifs, TICE 2010 - Colloque Technologies de l'Information et de la Communication pour l'Enseignement, 2010.

Pour fournir une assistance à la fois pertinente, efficace pour répondre aux besoins d'assistance des utilisateurs et bien acceptée par ces derniers, une solution consiste à lui proposer une assistance personnalisée en fonction de ses spécificités, comme ses connaissances et compétences, ses capacités et éventuels handicaps, ainsi que ses préférences. Une solution complémentaire consiste à proposer à l'utilisateur une assistance qui s'adapte en fonction de l'assistance déjà proposée.

Le langage aLDEAS, et notamment le patron de règles, permet le déclenchement d'assistance conditionné par des consultations de différentes sources d'informations, telles que le profil de l'utilisateur ou l'historique de l'assistance.

Le **profil de l'utilisateur** peut contenir toute information relative à l'utilisateur jugée pertinente par son créateur. Ainsi, un profil d'utilisateur peut contenir des informations relatives à la maîtrise de l'application-cible. En prenant en compte de telles informations, l'assistance fournie pourra être différente pour un utilisateur qui maîtrise une fonctionnalité de l'application-cible et pour un utilisateur ne la maîtrisant pas.

L'**historique de l'assistance** contient quant à lui toutes les informations relatives à l'exécution d'un système d'assistance. Ainsi, si un message d'aide a déjà été proposé à un utilisateur lorsqu'il était confronté à une situation problématique, et qu'il se retrouve à nouveau dans cette même situation problématique, il est possible de lui proposer un autre message d'aide, ou un autre type d'action d'assistance, par exemple une vidéo de démonstration.

Pour que ces informations puissent être consultées par aLDEAS, il est nécessaire qu'elles soient exprimées selon un formalisme connu. Nous présentons dans ce chapitre les formalismes de profils et d'historique de l'assistance que nous avons choisi d'utiliser.

6.1. Exploitation du profil de l'utilisateur

Pour permettre la personnalisation de l'assistance en fonction des spécificités de l'utilisateur, une solution consiste à exploiter un profil de l'utilisateur que le système d'assistance peut consulter afin de contraindre le déclenchement d'actions d'assistance. Le langage aLDEAS, présenté dans le chapitre précédent, permet d'effectuer des tests sur une valeur, issue d'une consultation du profil de l'utilisateur, afin de déclencher ou non des actions d'assistance. Or, pour consulter un profil et pour récupérer une valeur de ce profil, il est nécessaire d'en connaître le formalisme.

Par ailleurs, nous souhaitons permettre l'exploitation de profils d'utilisateurs au contenu très varié. En effet, il n'est pas possible de proposer un unique modèle de profil d'utilisateur qui contiendrait toutes les informations que les concepteurs d'assistance pourraient souhaiter utiliser pour personnaliser l'assistance destinée à n'importe quelle application-cible. Nous souhaitons donc exploiter un formalisme générique, suffisamment expressif pour permettre la définition de profils d'utilisateurs au contenu très varié.

Le langage PMDLe, Profiles MoDeling Language [Eyssautier-Bavay, 2008 ; Jean-Daubias et al., 2011] répond à notre besoin de définition de profils d'utilisateurs très variés selon un formalisme commun. PMDLe est associé au modèle de contraintes sur profils cPMDLe

[Lefevre, 2009 ; Ginon et Jean-Daubias, 2011], qui permet de consulter un profil respectant le formalisme PMDLe et de réaliser certaines opérations sur les données contenues dans de tels profils, comme des moyennes ou des sommes.

PMDLe et son complément cPMDLe ont été proposés comme réponses à des manques identifiés concernant l'exploitation de profils avec une démarche générique [Eyssautier-Bavay, 2008 ; Lefevre, 2009]. Ces travaux ont été proposés dans le cadre du projet PERLEA[9]. Plusieurs membres de l'équipe qui le constituait, dont nous faisions partie, sont également impliqués dans le projet AGATE, contexte de cette thèse. Pour ces deux raisons, continuité des travaux et généricité validée des propositions, nous avons choisi d'exploiter ces formalismes dans le cadre de cette thèse

Notons que dans le cadre de nos travaux de recherche sur l'assistance à l'utilisateur, nous nous intéressons uniquement à l'exploitation de profils d'utilisateurs pour personnaliser l'assistance. Nous considérons que la constitution de ces profils selon le formalisme PMDLe est réalisée en amont.

6.1.1. PMDLe : un formalisme de représentation de profils d'utilisateurs

Le langage de modélisation de profils PMDLe permet la définition de profils très variés selon un même formalisme. Pour cela, PMDLe distingue la partie *structure* du profil, qui peut être commune à plusieurs utilisateurs, par exemple les utilisateurs d'une même application-cible, et la partie *données*, qui contient les valeurs propres à chaque utilisateur. La Fig. 6-1 présente l'exemple de deux profils d'utilisateurs respectant le formalisme PMDLe : ces profils partagent la même structure, mais les valeurs qu'ils contiennent sont propres à Lucas et à Manon, deux étudiants dans le cadre d'un cours de programmation orientée objet.

La structure des profils PMDLe est constituée d'un ensemble d'éléments hiérarchisés nommés *briques*. Chaque brique peut être constituée de composantes et sous-composantes. Dans l'exemple donné par la Fig. 6-1, les profils de Lucas et Manon contiennent chacun trois briques : « Modélisation », « Design pattern » et « Outils ». La brique « Outils » est constituée des composantes « IDE » et « Gestion de projet ». La composante « IDE » contient deux sous-composantes : « NetBeans » et « Éclipse ».

Chaque brique d'un profil PMDLe est associée à une échelle sPMDLe qui définit l'ensemble des valeurs possibles pour les composantes et sous-composantes de cette brique. sPMDLe permet la création de deux types d'échelles, en addition de l'échelle *texte libre* : les échelles numériques et les échelles textuelles. Une échelle numérique sPMDLe est définie par un intervalle numérique et un pas. Ainsi, un pourcentage peut être défini par l'échelle numérique d'intervalle [0, 100] avec un pas de 1. Une échelle textuelle sPMDLe est définie par un ensemble de niveaux. Dans l'exemple des profils de Lucas et Manon, les briques « Modélisation » et « Designs patterns » sont associées à l'échelle textuelle « Non maîtrisé /

[9] http://liris.cnrs.fr/stephanie.jean-daubias/projets/p-perlea.html

partiellement maîtrisé / maîtrisé », alors que la brique « Outils » est associée à l'échelle textuelle « Novice / Standard / Expert ».

	Lucas	Manon
Modélisation *(Non maîtrisé / partiellement maîtrisé / maîtrisé)*		
Diagrammes UML	Partiellement maîtrisé	Partiellement maîtrisé
Ⓐ Diagrammes de classes	Maîtrisé	Partiellement maîtrisé
Diagrammes d'objets	Non maîtrisé	Maîtrisé
Diagrammes de séquences	Partiellement maîtrisé	Maîtrisé
Design patterns *(Non maîtrisé / Partiellement maîtrisé / Maîtrisé)*		
MVC	Partiellement maîtrisé	Partiellement maîtrisé
Ⓑ Observer	Non maîtrisé	
Factory	Maîtrisé	Maîtrisé
Outils *(Novice / Standard / Expert)*		
IDE		
NetBeans	Expert	Novice
Éclipse	Standard	Expert
Gestion de projet		
SVN	Standard	Standard
Forge	Standard	Expert

Fig. 6-1 – Exemples de profils d'utilisateurs respectant le formalisme PMDLe

Les composantes ou sous-composantes terminales, c'est-à-dire qui ne contiennent pas de sous-composantes, peuvent contenir une ou plusieurs valeurs respectant l'échelle de la brique. Par exemple, pour la composante « Modélisation/Diagrammes de classes » (cf. Ⓐ Fig. 6-1), le profil de Lucas contient la valeur *Maîtrisé*, et le profil de Manon contient la valeur *Partiellement maîtrisé* ; pour la composante « Design pattern/Observer » (cf. Ⓑ Fig. 6-1), le profil de Lucas contient la valeur *Maîtrisé*, alors que le profil de Manon ne contient aucune valeur. Les profils en PMDLe permettent par ailleurs de conserver l'historique de l'évolution des valeurs. Ainsi, une composante ou sous-composante terminale peut être associée à plusieurs valeurs correspondant à une date et éventuellement à une source. Par exemple, dans le cas d'un profil d'apprenant utilisé par un enseignant, chaque composante ou sous-composante terminale peut contenir une valeur pour chaque trimestre, ou pour chaque contrôle. Sur l'exemple donné par la Fig. 6-1, seules les valeurs les plus récentes sont représentées : le profil complet de Lucas est donné à titre d'exemple en annexe (cf. Fig. 44, Annexe F).

6.1.2. cPMDLe : un modèle de contraintes sur profils

Le modèle de contraintes sur profils cPMDLe est un complément de PMDLe. Il permet dans un premier temps de consulter un profil dont la structure est définie en PMDLe, afin de récupérer la valeur associée à l'élément du profil à consulter. Dans un second temps, cPMDLe permet de vérifier si la valeur vérifie certaines contraintes, qu'elle soit directement récupérée dans le profil de l'utilisateur ou calculée à partir de ce profil. En effet, dans le cas d'un élément non terminal, cPMDLe permet d'effectuer certaines opérations, telles que la somme et la moyenne, afin de déterminer la valeur de cet élément. Par exemple, la brique « Modélisation » du profil de Lucas et Manon (cf. Fig. 6-1) n'est pas associée à une valeur puisqu'il ne s'agit pas d'un élément terminal. Le modèle de contraintes sur profils cPMDLe permet néanmoins d'obtenir une valeur pour cette brique, en calculant la moyenne des composantes qui la constituent : Lucas a une moyenne de *Partiellement maîtrisé* pour la brique « Modélisation », alors que Manon a une moyenne de *Maîtrisé*.

Dans le cadre de cette thèse, nous n'exploitons de cPMDLe que sa capacité à récupérer ou à calculer des valeurs à partir d'un profil d'utilisateur dont la structure est définie en PMDLe. En effet, la capacité de cPMDLe à vérifier des contraintes sur les valeurs obtenues est équivalente aux alternatives proposées dans aLDEAS : suite à une consultation du profil de l'utilisateur réalisée selon cPMDLe, une alternative aLDEAS à n branches, chacune associée à un test, peut être exécutée afin de vérifier si la valeur de retour de la consultation respecte certaines contraintes.

À titre d'exemple, la Fig. 6-2 montre dans sa partie gauche une condition sur le profil de l'utilisateur définie en cPMDLe, et son équivalent en aLDEAS dans sa partie droite : une consultation de l'utilisateur suivie d'une alternative. Cette condition n'est vérifiée que si l'élément « Modélisation » du profil de l'utilisateur a la valeur *Maîtrisé*. Cet élément n'étant pas terminal, la valeur est calculée par une moyenne[10] sur les valeurs des composantes de cette brique. Ainsi, si cette consultation du profil de l'utilisateur est utilisée pour conditionner le déclenchement d'une action d'assistance, par exemple un conseil, cette action d'assistance pourra être déclenchée pour Manon, mais pas pour Lucas, dont la valeur pour l'élément « Modélisation » est différent de *Maîtrisé*.

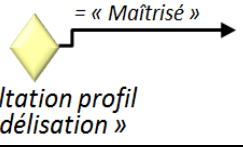
{« Modélisation », (« Maîtrisé »)}	

Fig. 6-2 – Définitions équivalentes d'une condition sur le profil de l'utilisateur, en cPMDLe (à gauche) et en aLDEAS (à droite)

[10] L'opération d'agrégation (moyenne, somme, nombre d'occurrences) est choisie en fonction du type de brique utilisé [Lefevre, 2009]

6.2. Exploitation de l'historique de l'assistance

Un historique de l'assistance permet de conserver une trace de tous les événements liés à l'assistance qui a été fournie à un utilisateur, c'est-à-dire les règles et actions d'assistance déclenchées, ainsi que les consultations de l'utilisateur réalisées et les choix effectués par l'utilisateur. Nous avons identifié deux principaux usages de l'historique de l'assistance : l'adaptation de l'assistance et *a posteriori* l'analyse de l'assistance fournie.

Tout d'abord, les informations relatives à l'historique de l'assistance peuvent être utiles pour adapter l'assistance, par exemple pour ne pas consulter plusieurs fois l'utilisateur pour une même question. Ensuite, ces informations peuvent également être utiles pour le concepteur de l'assistance, qui peut les utiliser afin de savoir comment son système d'assistance est utilisé, et éventuellement pour l'améliorer. Par exemple, si beaucoup d'utilisateurs demandent « plus d'aide » lors d'une même consultation, le concepteur peut considérer que l'assistance actuelle est insuffisante et choisir de fournir dorénavant systématiquement plus d'aide, ou une aide différente.

Afin d'être consultable lors de la phase d'exécution de l'assistance, l'historique de l'assistance doit être exprimé selon un formalisme connu du système mettant en œuvre le langage aLDEAS et réalisant les consultations de l'historique. Nous avons choisi d'utiliser un formalisme très simple, que nous présentons ici. Il permet de conserver la trace de tous les éléments d'aLDEAS perceptibles par l'utilisateur final : les règles et actions d'assistance, ainsi que les consultations de l'utilisateur.

Dans le cadre de cette thèse, un historique de l'assistance concerne un système d'assistance aLDEAS. Dans un historique de l'assistance, les informations sont regroupées par utilisateur final. En effet, lors de l'exécution de l'assistance, l'utilisateur final doit s'identifier ou se déclarer « anonyme » (cf. Chapitre 7). Pour plus de lisibilité de l'historique de l'assistance, les informations sont également regroupées par séance d'utilisation pour un même utilisateur final. Nous définissons une séance d'utilisation comme une période entre le lancement de l'assistance et son arrêt. Une séance d'utilisation est associée à une date, une heure de début et une heure de fin.

Pour chaque séance d'utilisation, l'historique de l'assistance conserve la trace de toutes les règles et actions d'assistance déclenchées. Chaque règle ou action d'assistance peut être déclenchée plusieurs fois pendant une même séance et l'historique mémorise chacun de ces déclenchements. Un déclenchement est lui-même associé à une heure de début et éventuellement une heure de fin. En effet, il n'est pas toujours possible de connaître l'heure de fin d'une action ou d'une règle d'assistance, dans le cas où elles ne sont pas associées à un événement de fin. Par exemple, l'action d'assistance A7 (cf. Fig. 6-3), le lancement d'une vidéo de démonstration, n'est associée à aucun événement de fin : la vidéo est lancée dans le lecteur vidéo par défaut de l'utilisateur final, qui peut la mettre en pause et la relancer comme il le souhaite, il est donc difficile de déterminer précisément la fin de cette action d'assistance. L'historique de l'assistance conserve également la trace de toutes les consultations de l'utilisateur réalisées par le système d'assistance : chaque consultation de l'utilisateur peut être déclenchée plusieurs fois pendant une même séance et l'historique

mémorise chacun de ces déclenchements. Un déclenchement est lui-même associé à une heure de début et une heure de fin, ainsi qu'à un choix de l'utilisateur, c'est-à-dire l'option choisie par l'utilisateur lors de la consultation. Un exemple de fichier xml d'historique de l'assistance généré par SEPIA suite à l'exécution d'un système d'assistance pour un utilisateur final est donné en annexe (cf. Fig. 25 en Annexe B-3).

À titre d'exemple, la Fig. 6-3 présente un extrait de l'historique du système d'assistance pour PhotoScape, pour une séance d'utilisation et un utilisateur donné. Les heures de début et de fin sont exprimées en millisecondes depuis 1970. La règle R0 et la consultation de l'utilisateur C0 ont tout d'abord été déclenchées : l'utilisateur final a choisi l'option « Oui je veux de l'aide », ce qui a entraîné la fin de la consultation C0 et de la règle d'assistance R0, ainsi que le déclenchement de la règle R1. En soustrayant l'heure de début de celle de fin, l'historique de l'assistance nous permet de savoir que l'utilisateur a mis 2453 ms pour répondre à cette consultation. Nous constatons que la consultation de l'utilisateur C6 a été déclenchée trois fois durant cette séance. Lors des deux premiers déclenchements de C6, l'utilisateur a choisi l'option « Plus d'aide », alors que la troisième fois, il a choisi l'option « J'ai terminé ».

Règles et actions d'assistance	**Début**	**Fin**	
R0	1402734846406	1402734848859	
A7	1402734868759		
	1402734869401		
Consultations de l'utilisateur	**Début**	**Fin**	**Choix**
C0	1402734846406	1402734848859	« Oui je veux de l'aide »
C6	1402734864087	1402734868759	« Plus d'aide »
	1402734868756	1402734869401	« Plus d'aide »
	1402734868758	1402734869923	« J'ai terminé »

Fig. 6-3 – Extrait de l'historique du système d'assistance pour PhotoScape

L'historique de l'assistance supporte les éventuelles modifications effectuées dans le système d'assistance associé. En effet, si un élément est supprimé du système d'assistance, l'historique conserve les traces de cet élément. Au contraire, si un élément est ajouté au système d'assistance, il sera ajouté à l'historique de l'assistance lorsque cet élément se déclenchera lors des prochaines séances d'utilisation.

6.3. Conclusion

Dans ce chapitre, nous avons montré de quelle manière le langage aLDEAS permet la consultation du profil de l'utilisateur et de l'historique de l'assistance pour personnaliser l'assistance. Pour pouvoir consulter et exploiter ces informations, il est nécessaire qu'elles soient exprimées selon un formalisme connu du système exécutant l'assistance. Nous avons présenté le formalisme de profils PMDLe que nous avons choisi de réutiliser, ainsi qu'un formalisme simple pour les historiques de l'assistance.

En complément du profil de l'utilisateur et de l'historique de l'assistance, d'autres types d'informations peuvent être pertinents à consulter afin d'adapter l'assistance. Ainsi, une assistance peut être personnalisée en fonction de l'expérience d'un utilisateur, et plus particulièrement en fonction de ses actions passées. Pour disposer de telles informations, une solution consiste à surveiller les interactions entre l'utilisateur et l'application-cible, et à les stocker sous forme de traces qui pourront par la suite être consultées par un système d'assistance afin d'adapter l'assistance en fonction des actions passées de l'utilisateur. Par ailleurs, l'ajout de connaissances du domaine de l'application-cible serait un atout pour adapter l'assistance. Bien que ces deux points ne soient pas traités dans le cadre de cette thèse, ils sont développés en perspectives (cf. Section 12.7).

Chapitre 7. aMEAS : modèle d'exécution d'un système d'assistance aLDEAS

Objectif du chapitre

Permettre l'exécution d'un système d'assistance défini avec aLDEAS sous forme d'un ensemble de règles.

Points clés

Nous proposons **aMEAS**, un modèle qui explicite le fonctionnement de la phase d'exécution d'un système d'assistance aLDEAS. Il détaille en particulier la gestion des **événements déclencheurs** et des **événements de fin** des règles qui définissent le système d'assistance, ainsi que le cas particulier de la gestion des événements de type **timer**.

Contribution

- aMEAS, modèle d'exécution d'un système d'assistance aLDEAS

Validation

- ✓ Exécution conforme à leur définition de systèmes d'assistance aLDEAS variés dans différentes applications-cibles

Dans le Chapitre 5, nous avons présenté le langage aLDEAS, un langage pivot entre la phase de spécification de l'assistance et celle d'exécution de l'assistance. aLDEAS permet de définir des systèmes d'assistance sous la forme d'un ensemble de règles d'assistance respectant le patron de règles aLDEAS. Ce chapitre a pour objectif la présentation d'aMEAS (a Model to Execute aLDEAS Assistance Systems), le modèle qui explicite le fonctionnement de la phase d'exécution d'un système d'assistance aLDEAS, depuis son lancement jusqu'à son arrêt. En particulier, nous montrons de quelle manière le processus de surveillance d'une application-cible présenté en section 4.1 s'intègre dans le fonctionnement d'un système d'assistance aLDEAS. Nous détaillons également la gestion des événements de fin et des événements déclencheurs des règles aLDEAS, ainsi que la gestion des événements de type timers.

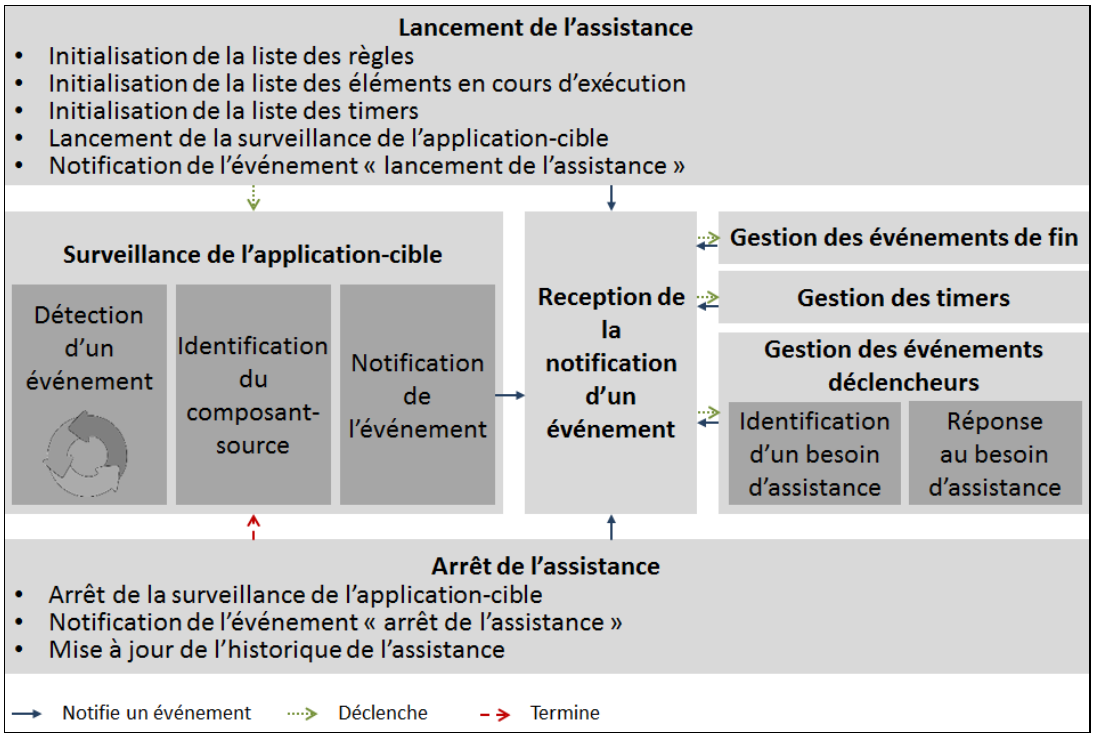


Fig. 7-1 – aMEAS : modèle d'exécution d'un système d'assistance aLDEAS

L'exécution d'un système d'assistance aLDEAS, présentée en Fig. 7-1, se déroule entre le lancement de l'assistance et l'arrêt de cette assistance. L'exécution de toutes les règles d'assistance est lancée en même temps, au lancement de l'assistance. Comme ces règles débutent par un événement déclencheur, qui correspond à une attente d'événement suspendant l'exécution de la règle, l'exécution de chaque règle est suspendue jusqu'à ce que son événement déclencheur ait lieu. aMEAS explicite donc de quelle manière les événements déclencheurs des règles du système d'assistance peuvent être gérés. Lorsque l'événement déclencheur d'une règle d'assistance a eu lieu, des consultations de l'utilisateur et des actions d'assistance peuvent être déclenchées. La règle peut être associée à un événement de fin, qui indique quand ces éléments devront être supprimés. aMEAS explicite donc de quelle manière les événements de fin des règles du système d'assistance peuvent être gérés. Par ailleurs, certains événements, qu'ils soient déclencheurs ou de fin, peuvent correspondre

à une attente d'événement d'aLDEAS de type timer, qui nécessite une gestion particulière, détaillée par aMEAS. En effet, les attentes d'événements (cf. Section 0) entraînent une suspension de l'exécution d'un bloc aLDEAS jusqu'à ce que l'événement attendu ait lieu. Dans le cas d'une attente d'événement de type timer, lorsque l'événement attendu a lieu, l'exécution du bloc est encore suspendue pour la durée du timer.

La surveillance de l'application-cible a lieu en continu, tout le long de l'exécution de l'assistance et notifie les événements détectés dans l'application-cible. Qu'il s'agisse d'un événement en provenance de l'application-cible ou d'un événement interne à l'exécution de l'assistance, la notification d'un événement déclenche la gestion des événements de fin, des timers et des événements déclencheurs.

7.1. Lancement de l'assistance

Lors du lancement d'un système d'assistance aLDEAS, l'ensemble des règles d'assistance le constituant est parcouru afin d'initialiser la liste des règles, en fonction de leur événement déclencheur. La liste des timers est également initialisée afin de permettre la gestion des attentes d'événements de type timers qui sont éventuellement contenus dans les règles d'assistance. La liste des règles, actions et conditions en cours est initialement vide lors du lancement de l'assistance.

La surveillance de l'application-cible est également déclenchée lors du lancement de l'assistance : elle notifiera le système d'assistance de tout événement en lien avec un composant de l'interface de l'application-cible et ne prendra fin que lors de l'arrêt de l'assistance.

Enfin, l'événement « lancement de l'assistance » est notifié, il s'agit d'un événement d'aLDEAS lié à l'assistance (cf. Fig. 5-8). Le lancement de l'assistance peut correspondre à l'événement déclencheur d'une règle d'assistance du système. Par exemple, un système d'assistance peut permettre à l'utilisateur de choisir via une consultation pour quelle fonctionnalité de l'application-cible il souhaite être assisté. Cet événement peut n'avoir aucun effet, comme c'est le cas pour n'importe quel événement notifié. En effet, si un événement notifié n'est ni l'événement déclencheur ou de fin d'une règle du système, ni associé à un timer du système, alors sa notification n'aura aucun effet.

7.2. Notification d'un événement

Un événement notifié au système d'assistance peut provenir de trois sources : la surveillance de l'application-cible notifie le système d'assistance de tout événement lié à l'utilisateur (comme ses clics), le système d'assistance se notifie de tout événement lié à l'assistance (comme les déclenchements d'actions d'assistance) et chaque timer géré par le système d'assistance notifie le système d'assistance lorsqu'il arrive au terme fixé par sa durée.

La notification d'événements est au cœur du fonctionnement d'un système d'assistance aLDEAS. En effet, les règles qui définissent l'assistance sont des blocs aLDEAS commençant par une attente d'événement, qui correspond à l'événement déclencheur de la règle. Seule la

notification de l'événement qui est attendu permet la progression dans le bloc aLDEAS et donc le déclenchement d'actions d'assistance pour l'utilisateur de l'application-cible. Par ailleurs, les règles d'assistance peuvent également être associées à un événement de fin, qui correspond à une attente d'événement suivie par le marqueur d'arrêt qui provoque la fin de la règle et de toutes les actions déclenchées dans cette règle. Ainsi, toutes les règles déclenchées contenant un événement de fin demeurent actives jusqu'à la notification de leur événement de fin qui indique au système d'assistance qu'il doit mettre fin à cette règle. De plus, pour les attentes d'événement de type timer associées à un événement, qu'il soit lié à l'utilisateur ou à l'assistance, la notification de cet événement implique le déclenchement ou la réinitialisation du timer, selon que le timer concerne une durée depuis l'événement notifié ou une durée relative à l'absence de cet événement.

En conséquence, chaque notification d'un événement déclenche les processus de gestion des événements déclencheurs, des timers et des événements de fin, que nous détaillons dans la suite de ce chapitre.

7.3. Gestion des événements de fin

Il est nécessaire pour le système d'assistance de gérer les événements qui doivent mettre fin aux règles et actions d'assistance et aux consultations de l'utilisateur en cours d'exécution. Lors de la notification d'un événement, quel que soit l'émetteur de cette notification, le système d'assistance consulte la liste des règles éléments en cours d'exécution (cf. Ⓐ Fig. 7-2) afin de mettre fin aux éléments dont l'événement de fin est celui notifié (cf. Ⓑ Fig. 7-2).

Pour chaque élément E d'aLDEAS qui doit prendre fin, le système d'assistance notifie l'événement « Fin de E » (cf. © Fig. 7-2). Le système d'assistance retire également cet élément de la liste des règles, actions et conditions en cours.

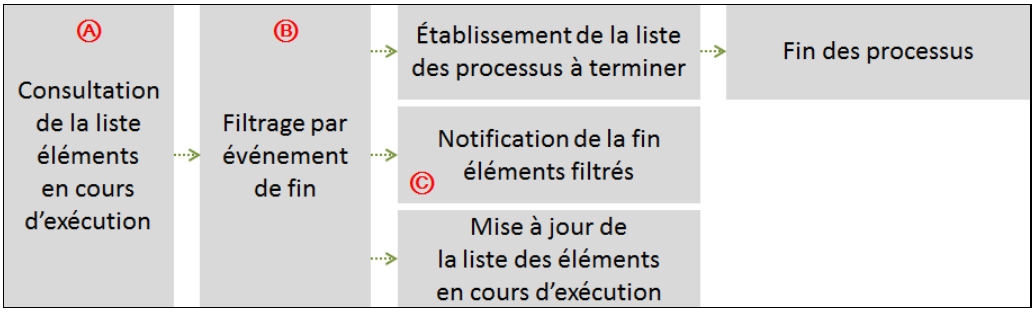


Fig. 7-2 – Gestion des événements de fin dans aMEAS

7.4. Gestion des timers

Les systèmes d'assistance aLDEAS peuvent contenir des attentes d'événements liées à des timers, qu'il s'agisse d'événements déclencheurs (par exemple pour déclencher une règle d'assistance 1 minute après le clic de l'utilisateur sur un bouton donné), ou d'événements de fin (par exemple pour effacer une mise en valeur après 30 secondes). La gestion de ces

timers dans aMEAS est représentée en Fig. 7-3. Lorsqu'il arrive au terme fixé par son paramètre *durée*, un timer notifie au moteur d'assistance l'événement qu'il a en paramètre.

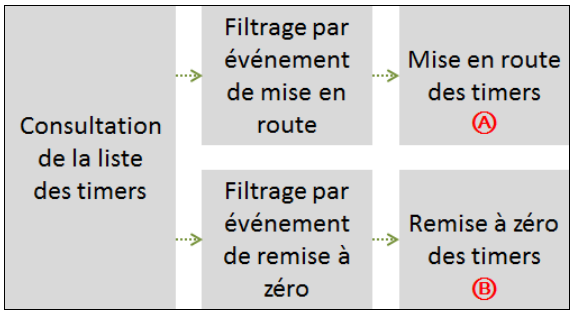


Fig. 7-3 – Gestion des timers dans aMEAS

Prenons l'exemple d'un timer qui permet d'attendre qu'il se soit écoulé 3 minutes sans que l'utilisateur final n'ait effectué une action dans l'application-cible. Lors de la notification du lancement de l'assistance, ce timer est mis en route avec comme durée 3 minutes (cf. Ⓐ Fig. 7-3). Lorsque cette durée arrive à son terme, le timer notifie l'événement « 3 minutes sans action de l'utilisateur », puis se remet à 0. Si une action de l'utilisateur final est notifiée (par exemple un clic) avant que ce timer n'arrive à son terme, celui-ci se remet à 0 sans notifier d'événement (cf. Ⓑ Fig. 7-3).

7.5. Gestion des événements déclencheurs

La gestion des événements déclencheurs a pour but l'identification d'un besoin d'assistance et l'élaboration d'une réponse à ce besoin. Elle a lieu lors de chaque notification d'un événement. Elle est représentée en Fig. 7-4.

Tout d'abord, afin d'identifier un éventuel besoin d'assistance de l'utilisateur de l'application-cible, la liste des règles d'assistance du système est consultée puis filtrée afin de ne conserver que les règles dont un événement déclencheur correspond à l'événement qui a été notifié (cf. Ⓐ Fig. 7-4). Pour toutes les règles dont l'événement déclencheur correspond à l'événement notifié, le système d'assistance vérifie l'éventuelle condition de déclenchement (cf. Ⓑ Fig. 7-4). Chaque règle d'assistance dont un événement d'assistance a été notifié et dont la condition de déclenchement est vraie correspond à un besoin d'assistance pour lequel le système va élaborer une réponse sous la forme d'une ou plusieurs actions d'assistance.

Pour cela, le système établit tout d'abord la liste des actions d'assistance à déclencher (cf. © Fig. 7-4). Parmi ces actions, définies en aLDEAS, certaines peuvent comporter des paramètres optionnels. Si certains paramètres optionnels n'ont pas été spécifiés par le concepteur de l'assistance, les préférences de l'utilisateur final peuvent être exploitées (cf. Ⓓ Fig. 7-4) : par exemple pour déterminer le mode de transmission d'un message (fenêtre pop-up, agent animé ou synthèse vocale), ou la couleur et l'épaisseur du trait d'une mise en valeur par encadré. Si l'utilisateur n'a pas exprimé de préférences relatives à ces paramètres optionnels, des valeurs par défaut sont utilisées (cf. Ⓔ Fig. 7-4). Notons que le choix des paramètres optionnels se fait toujours en fonction des épi-assistants disponibles, le système

peut être amené à modifier certains paramètres pour tenir compte de cette disponibilité. Ainsi, si le paramètre d'une action d'assistance indique qu'un message d'assistance doit être transmis à l'utilisateur par un agent animé et qu'aucun épi-assistant capable de faire appel à un agent animé n'est installé sur l'ordinateur de l'utilisateur final, ce paramètre sera modifié en fonction de valeurs par défaut pour que le message puisse être transmis, par exemple par une fenêtre pop-up. Finalement, le système fait appel aux épi-assistants adaptés pour réaliser les actions d'assistance ainsi paramétrées, notifie leur déclenchement et met à jour la liste des éléments actifs.

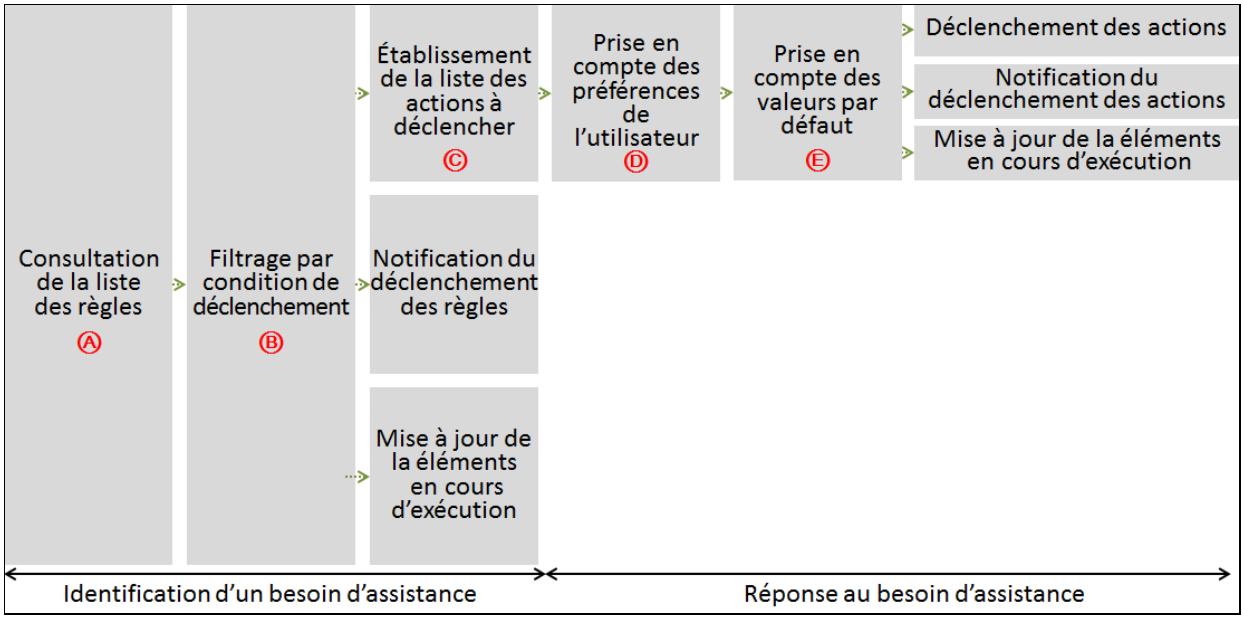


Fig. 7-4 – Gestion des événements déclencheurs dans aMEAS

7.6. Exemple de l'exécution d'un système d'assistance avec aMEAS

Dans toute cette section, nous prenons l'exemple du système d'assistance pour la correction des yeux rouges sur une photo avec PhotoScape. La définition en aLDEAS de ce système d'assistance est donnée en annexe (cf. Fig. 20, Annexe B). Au lancement de l'assistance par un utilisateur final, le moteur générique parcourt l'ensemble des règles d'assistance qui composent le système d'assistance, afin d'initialiser la liste des règles. Si certaines règles contiennent des événements de type *timer*, le moteur générique ajoute les événements concernés à la liste des timers. La liste des règles, consultations et actions en cours est également initialisée à vide. Suite à ces étapes préliminaires, le moteur d'assistance lance l'épi-détecteur adapté au type de l'application-cible, en lui fournissant le fichier de description de l'interface de l'application-cible. Ce fichier permettra à l'épi-détecteur de notifier au moteur d'assistance pour chaque événement détecté son type et l'identifiant du composant source de cet événement, et d'ignorer les événements détectés qui ne concernent pas l'application-cible. Le moteur d'assistance se notifie ensuite l'événement « lancement de l'assistance » qui marque véritablement le début de l'exécution de l'assistance.

La partie gauche de la Fig. 7-5 schématise l'état initial du moteur d'assistance. Nous constatons que le système d'assistance est composé de 9 règles d'assistance, R0 à R8. Initialement, aucun élément n'est en cours d'exécution. Aucune des règles d'assistance de ce système ne contenant d'attente d'événement de type *timer*, la liste des timers est vide. Le premier événement notifié est celui du lancement de l'assistance. Pour chaque notification qu'il reçoit, le moteur d'assistance lance les processus de gestion des événements de fin, des timers et des événements déclencheurs.

Moteur d'assistance	**Moteur d'assistance**
• Règles {R0, R1, R2, R3, R4, R5, R6, R7, R8} • Éléments en cours d'exécution { } • Timers { }	• Règles {R0, R1, R2, R3, R4, R5, R6, R7, R8} • Éléments en cours d'exécution { R1 } • Timers { }

Fig. 7-5 – État du moteur d'assistance avant (à gauche) et après (à droite) la notification de l'événement « lancement de l'assistance »

Le processus de gestion des événements de fin n'a aucun effet puisque la liste des éléments en cours d'exécution est vide au moment de la notification. Le processus de gestion des timers ne fait rien dans le cas du système d'assistance pour PhotoScape puisque la liste des timers est vide.

Le processus de gestion des événements déclencheurs va lui avoir des conséquences (cf. Fig. 7-6). En effet, l'une des règles du système, R0, a pour événement déclencheur « lancement de l'assistance » : R0 est donc déclenchable (cf. Ⓐ Fig. 7-6). Lors du filtrage par condition de déclenchement, une consultation de l'utilisateur est réalisée, afin de proposer de l'aide à l'utilisateur, conformément à la définition de R0 (cf. Ⓑ Fig. 7-6). Pendant cette consultation, l'exécution de R0 est suspendue jusqu'à ce que l'utilisateur ait fait un choix. Cependant, le moteur d'assistance continue de gérer les événements qui lui sont notifiés, et d'autres règles peuvent être déclenchées pendant cette période. Une fois que l'utilisateur a répondu à la consultation et s'il a répondu « Oui je veux de l'aide », R0 est une règle à déclencher. Le moteur d'assistance notifie donc l'événement « Lancement de R0 » (cf. © Fig. 7-6) et ajoute R0 à la liste des éléments en cours d'exécution (cf. Ⓓ Fig. 7-6). Le moteur consulte ensuite les actions de R0 qui doivent être déclenchées : pour le cas de R0, seule la règle R1 doit être déclenchée (cf. Ⓔ Fig. 7-6). Les étapes de prise en compte des préférences de l'utilisateur, de choix d'un assistant épiphyte, de prise en compte de valeurs par défaut et de déclenchement d'actions concernent uniquement les actions d'assistance à l'exclusion des règles d'assistance. En revanche, l'événement « Lancement R1 » est notifié (cf. Ⓕ Fig. 7-6), et la liste des éléments en cours d'exécution est mise à jour : à la fin de la gestion des événements déclencheurs, seule la règles R1 est en cours d'exécution (cf. Ⓖ Fig. 7-6).

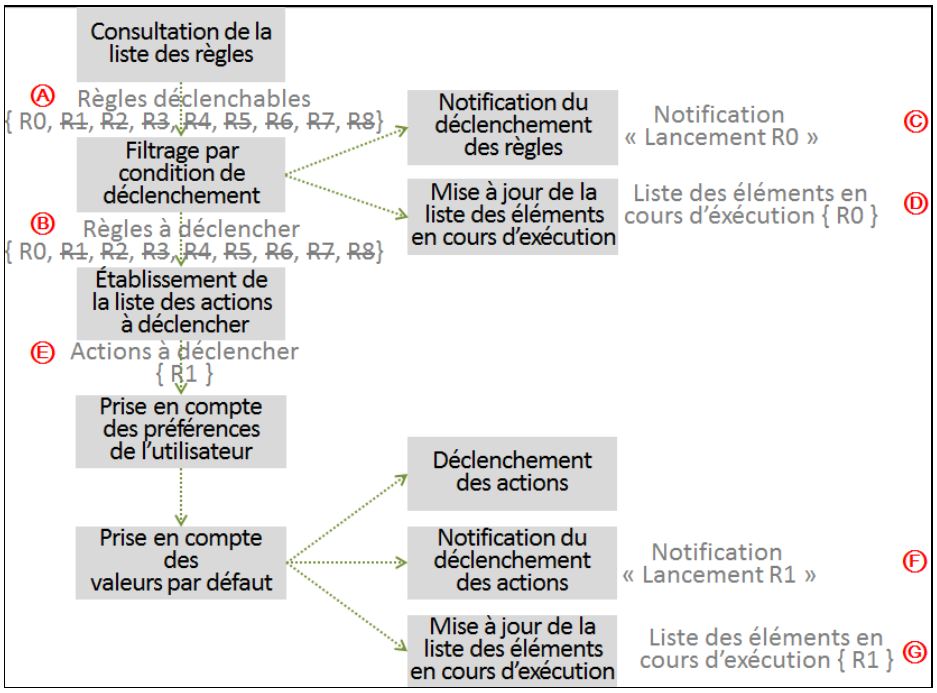


Fig. 7-6 – Gestion des événements déclencheurs lors de la notification de l'événement « lancement de l'assistance »

Reprenons maintenant notre exemple un peu plus loin dans l'exécution du système d'assistance pour la correction des yeux rouges dans PhotoScape. La partie gauche de la Fig. 7-7 schématise l'état du moteur d'assistance avant la notification de l'événement « clic 228 » (illustré par la Fig. 7-8) : à cet instant, R4 ainsi que les actions d'assistance A3 et A4 sont en cours d'exécution (cf. Fig. 7-8). L'action A3 est une mise en valeur du composant d'identifiant 228, l'onglet « Outils ». L'action A4 est un message, affiché dans une fenêtre pop-up. La partie droite de la Fig. 7-7 schématise l'état du moteur d'assistance après la notification de l'événement « clic 228 » : à cet instant, R6 ainsi que les actions d'assistance A6 et A7 sont en cours d'exécution (cf. Fig. 7-9).

Moteur d'assistance	**Moteur d'assistance**
• Règles {R0, R1, R2, R3, R4, R5, R6, R7, R8} • Éléments en cours d'exécution { R4, A3, A4 } • Timers { }	• Règles {R0, R1, R2, R3, R4, R5, R6, R7, R8} • Éléments en cours d'exécution { R6, A6, A7 } • Timers { }

Fig. 7-7 – État du moteur d'assistance avant (à gauche) et après (à droite) la notification de l'événement « clic 228 »

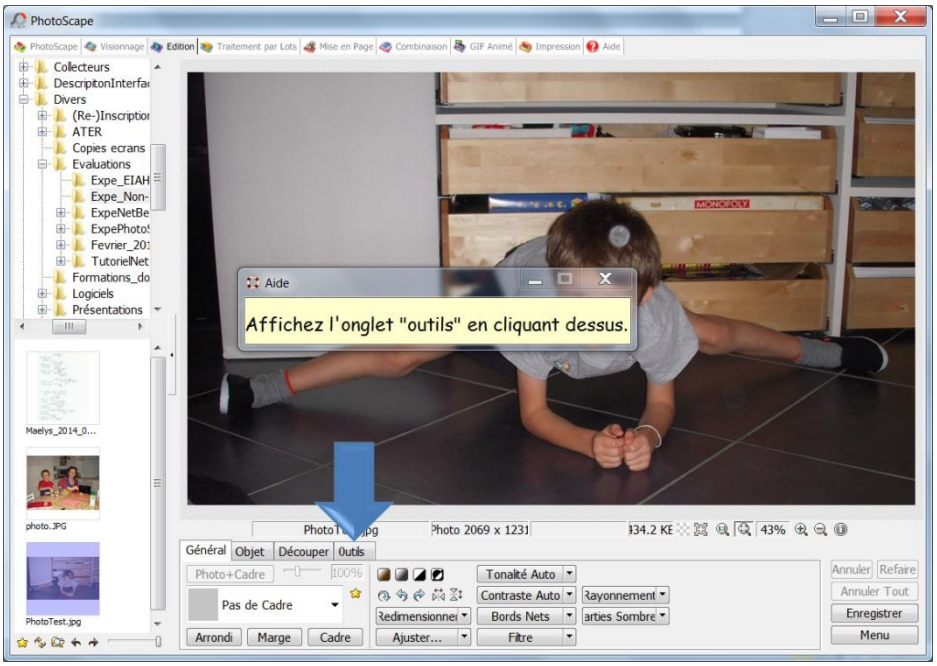


Fig. 7-8 – Exécution de l'assistance avant la notification de l'événement « clic 228 »

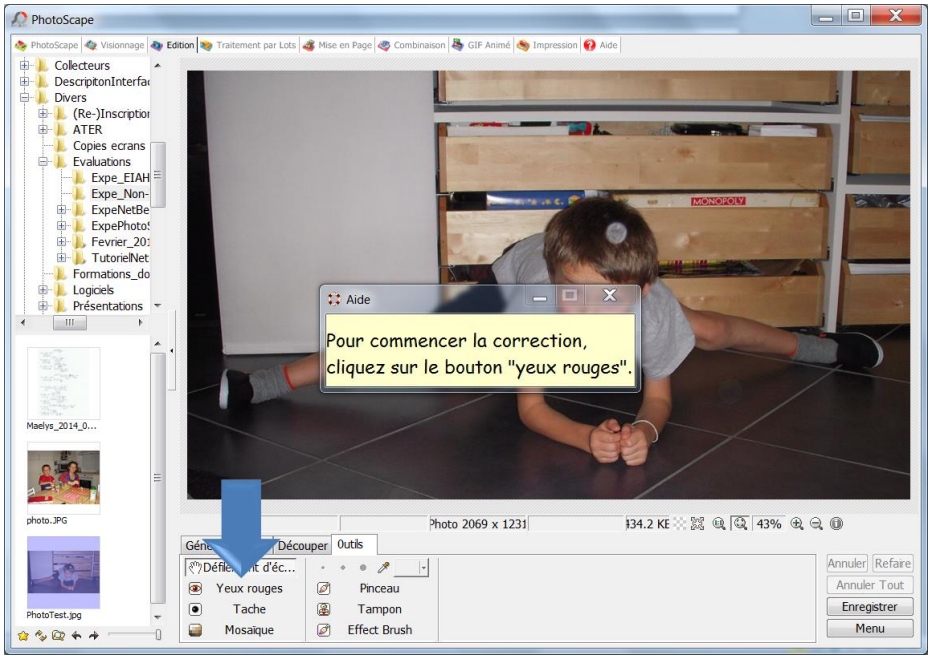


Fig. 7-9 – Exécution de l'assistance après la notification de l'événement « clic 228 »

Lorsque le moteur d'assistance reçoit la notification « clic 228 », qui correspond à un clic de l'utilisateur final sur l'onglet « Outils » de PhotoScape, la gestion des événements de fin est tout d'abord déclenchée (cf. Fig. 7-10). Le moteur d'assistance commence par regarder si les éléments en cours d'exécution, R4, A3 et A4, sont associés à un événement de fin « clic 228 » (cf. Ⓐ Fig. 7-10). Comme c'est le cas pour les trois éléments, le moteur établit la liste des processus à terminer (cf. Ⓑ Fig. 7-10). R4 ne correspond pas à un processus à terminer puisqu'il ne s'agit pas d'une action élémentaire d'assistance. Par contre A3 et A4

correspondent à des processus auxquels le moteur d'assistance met fin (cf. © Fig. 7-10), ce qui fait disparaître à l'écran la flèche et la fenêtre pop-up. Le moteur se notifie alors les événements de fin de R4, A3 et A4 (cf. ⓓ Fig. 7-10). Suite à cette gestion des événements de fin, plus aucun élément n'est en cours d'exécution (cf. ⓔ Fig. 7-10).

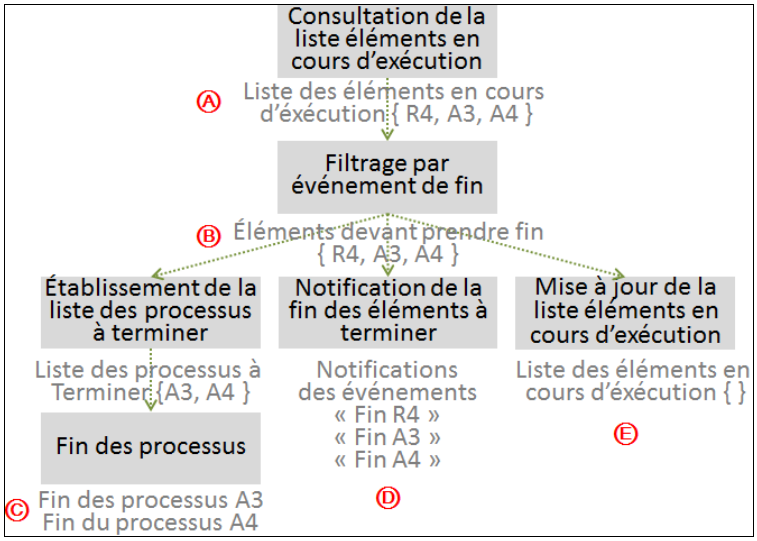


Fig. 7-10 – Gestion des événements de fin lors de la notification de l'événement « clic 228 »

La gestion des timers n'a pas lieu dans ce système d'assistance puisqu'aucun timer n'est impliqué.

La gestion des événements déclencheurs a alors lieu (cf. Fig. 7-11). En consultant la liste des règles, le moteur détermine que la règle R6 est déclenchable, puisque son événement déclencheur est « clic 228 » (cf. Ⓐ Fig. 7-11). R6 n'ayant pas de condition de déclenchement, le moteur générique établit ensuite la liste des actions qui doivent être déclenchées selon la définition en aLDEAS de R0 : la mise en valeur A6 et le message A7 (cf. Ⓑ Fig. 7-11). Ces deux types d'actions élémentaires ont des paramètres optionnels qui n'ont pas été spécifiés par le concepteur de l'assistance, ce qui rend possible une prise en compte des préférences de l'utilisateur. Supposons que l'utilisateur actuel soit Lucas Dupuis, dont une partie du profil de préférences est représenté en Fig. 9-21 page 157. Les paramètres optionnels de la mise en valeur A6 sont complétés selon les préférences de Lucas : la mise en valeur sera réalisée par l'affichage d'un symbole, une flèche qui pointera sur le haut du composant. Le message A7 sera affiché dans une fenêtre pop-up, avec une mise en forme correspondant également aux préférences de Lucas (cf. © Fig. 7-11).

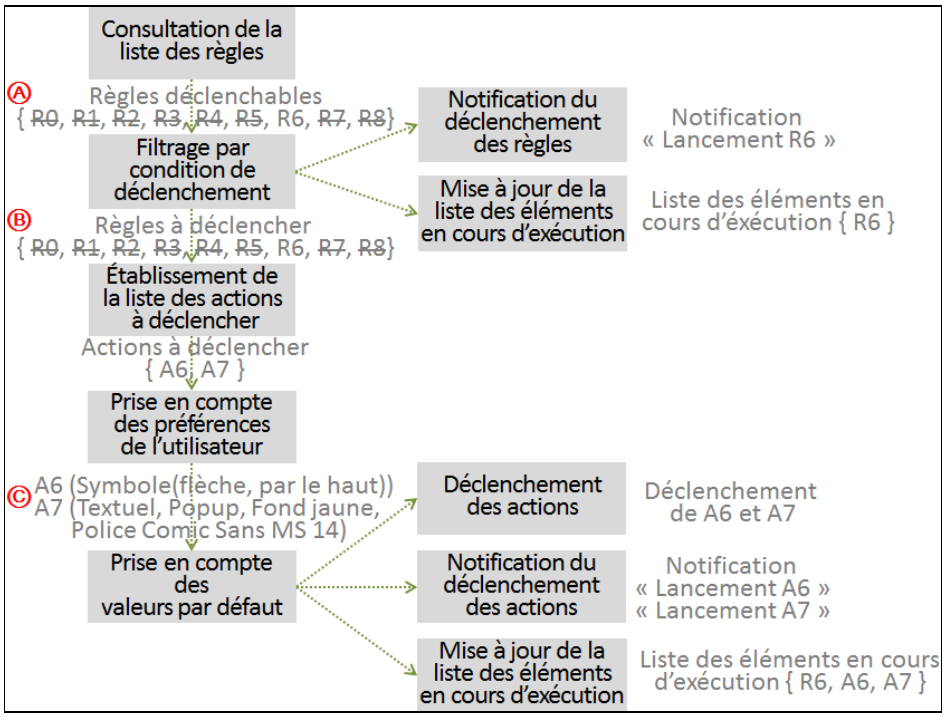


Fig. 7-11 – Gestion des événements déclencheurs lors de la notification de l'événement « clic 228 »

7.7. Conclusion

Dans ce chapitre, nous avons présenté aMEAS, le modèle d'exécution d'un système d'assistance aLDEAS. Ce modèle décrit une manière d'opérationnaliser le langage aLDEAS, en s'appuyant sur le patron de règles d'assistance. Une alternative consiste à proposer un moteur aLDEAS, capable d'exécuter n'importe quel bloc défini en aLDEAS. Nous avons cependant fait le choix de ne pas opérationnaliser la totalité du langage aLDEAS : selon nous, l'exploitation des patrons, et en particulier le patron de règles d'assistance, facilite fortement le travail de spécification de l'assistance, et le rend plus accessible aux concepteurs d'assistance, en particulier les non-informaticiens.

Dans la partie suivante, nous présentons la mise en œuvre des propositions théoriques faites dans le cadre de notre thèse, à travers le système SEPIA.

Partie 3. Le système SEPIA

Plan de la partie « Le système SEPIA »

Afin de mettre en œuvre les différentes contributions théoriques présentées dans la partie précédente, nous avons développé le système SEPIA (Specification and Execution of Personnalised Intelligent Assistance). En particulier, SEPIA met en œuvre le processus d'adjonction d'un système d'assistance à une application-cible, présenté en introduction de la Partie 2. Nous présentons ici l'architecture du système SEPIA (cf. Fig. P3-1), en séparant les outils de SEPIA qui concernent la phase de spécification de l'assistance, de ceux qui concernent la phase d'exécution de l'assistance. Cette architecture présente une vue d'ensemble du système SEPIA, qui sera présenté en détail dans les chapitres suivants.

SEPIA est composé de différents outils, interagissant les uns avec les autres et destinés à deux catégories d'utilisateurs : les concepteurs d'assistance souhaitant spécifier un système d'assistance pour une application-cible, et les utilisateurs finaux de ces applications-cibles qui souhaitent recevoir de l'aide. Les deux principaux outils de SEPIA sont l'éditeur d'assistance, pour la phase de spécification, et le moteur générique d'assistance, pour la phase d'exécution. Le langage aLDEAS est mis en œuvre à la fois à travers l'éditeur et le moteur d'assistance de SEPIA. D'une part, l'éditeur d'assistance de SEPIA permet au concepteur de l'assistance de manipuler les éléments d'aLDEAS via son interface, pour définir l'assistance souhaitée sous la forme de règles instanciant le patron présenté en Section 5.2.1. D'autre part, le moteur d'assistance de SEPIA exécute les règles aLDEAS pour fournir de l'assistance aux utilisateurs finaux de l'application-cible.

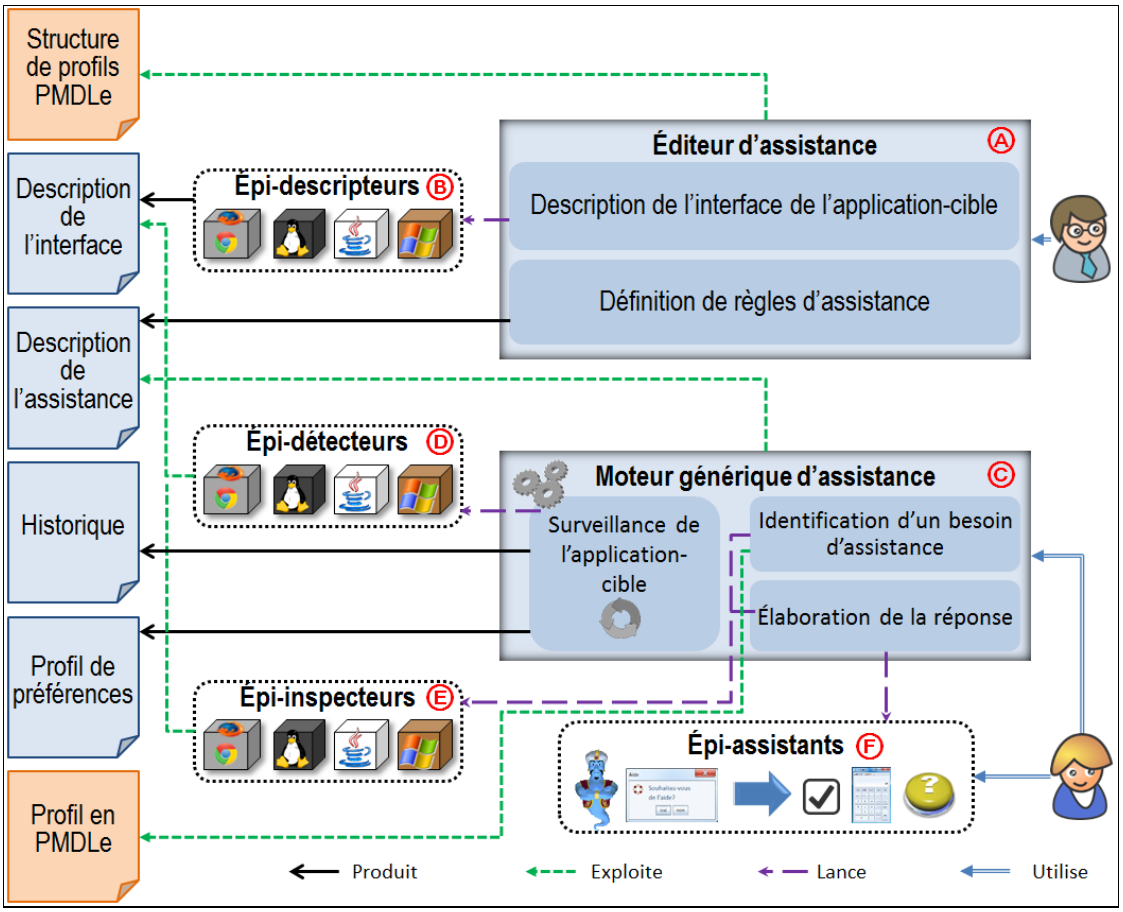


Fig. P3-1 – Architecture du système SEPIA

Dans le système SEPIA, la **spécification de l'assistance** se fait principalement via l'éditeur d'assistance. La Fig. P3-2 présente les outils, les acteurs et les fichiers impliqués dans l'étape de spécification de l'assistance.

Outils	Acteurs concernés	Fichiers produits	Fichiers édités	Fichiers exploités
Éditeur d'assistance	Concepteurs d'assistance	Description de l'assistance	Description de l'interface de l'application-cible Description de l'assistance	Structure de profils PMDLe
Épi-descripteurs	Concepteurs d'assistance	Description de l'interface de l'application-cible	-	-

Fig. P3-2 – Outils, acteurs et fichiers pour la spécification de l'assistance avec SEPIA

L'éditeur d'assistance de SEPIA (cf. Ⓐ Fig. P3-1) est un outil à destination du concepteur de l'assistance, qui spécifie l'assistance en deux étapes : la description de l'interface de l'application-cible, puis la définition des règles d'assistance.

Pour l'étape de description de l'interface de l'application-cible, l'éditeur d'assistance fait appel à un épi-descripteur (cf. Ⓑ Fig. P3-1) pour produire un fichier dans lequel chaque composant de l'application-cible est associé à un identifiant. Le concepteur de l'assistance utilise les épi-descripteurs, bien qu'il n'ait pas conscience qu'il s'agit de programmes externes lancés par l'éditeur d'assistance. Le fichier de description de l'interface, produit par l'épi-détecteur adapté au type de l'application-cible, peut ensuite être enrichi par le concepteur de l'assistance via l'éditeur d'assistance. Il peut notamment associer des commentaires à un composant, par exemple pour expliquer quelle est sa fonction.

L'étape de définition des règles d'assistance produit un fichier contenant l'ensemble des règles décrites. Ce fichier est lié à celui de description de l'interface de l'application-cible, puisque les règles d'assistance peuvent contenir des éléments faisant référence à l'identifiant d'un composant, comme les actions élémentaires d'assistance de type *mise en valeur*, ou les consultations de l'état de l'application-cible. Le concepteur de l'assistance peut choisir de personnaliser les règles qu'il crée en fonction du profil de l'utilisateur. Dans SEPIA, les profils des utilisateurs manipulés doivent respecter le formalisme PMDLe (cf. Section 6.1.1). Le concepteur doit donc indiquer la structure de profils PMDLe qu'il souhaite utiliser pour personnaliser son système d'assistance.

L'**exécution de l'assistance** avec SEPIA est réalisée par le moteur générique d'assistance (cf. © Fig. P3-1), qui pilote d'autres outils : les épi-détecteurs, les épi-inspecteurs et les épi-assistants. La Fig. P3-3 présente les outils, acteurs et fichiers impliqués dans la spécification de l'assistance avec SEPIA.

Outils	Acteurs concernés	Fichiers produits	Fichiers édités	Fichiers exploités
Moteur générique d'assistance	Utilisateurs finaux de l'application-cible	Historique de l'assistance	Historique de l'assistance	Description de l'assistance Description de l'interface de l'application-cible Historique de l'assistance Profils de préférences des utilisateurs finaux Profils PMDLe des utilisateurs finaux
Épi-détecteurs	-	-	-	Description de l'interface de l'application-cible
Épi-inspecteurs	-	-	-	Description de l'interface de l'application-cible
Épi-assistants	Utilisateurs finaux de l'application-cible	-	-	-

Fig. P3-3 – Outils, acteurs et fichiers pour l'exécution de l'assistance avec SEPIA

Le moteur générique d'assistance concerne les utilisateurs finaux de l'application-cible qui vont recevoir de l'assistance. Pour fournir à chaque utilisateur l'assistance qui a été spécifiée par le concepteur de l'assistance, le moteur générique manipule les fichiers de description de l'assistance et de l'interface de l'application-cible, le profil de préférences de l'utilisateur final, ainsi que son profil respectant la structure de profils PMDLe si le concepteur a souhaité personnaliser l'assistance en fonction des spécificités de l'utilisateur exprimées dans un profil PMDLe.

Pour surveiller l'application-cible, le moteur d'assistance fait appel à un ensemble d'épi-détecteurs (cf. Ⓓ Fig. P3-1) qui l'informent des événements survenant dans l'application-cible. De plus, lorsqu'il a besoin d'une information sur l'état de l'application-cible, le moteur d'assistance fait appel à un épi-inspecteur (cf. Ⓔ Fig. P3-1), capable d'accéder aux propriétés des composants de l'application-cible. Les épi-détecteurs comme les épi-inspecteurs sont invisibles aux yeux de l'utilisateur final.

Enfin, pour fournir sous des formes variées l'assistance aux utilisateurs finaux, le moteur d'assistance pilote un ensemble d'épi-assistants (cf. Ⓕ Fig. P3-1), capables de réaliser des actions d'assistance dans l'application-cible de manière épiphyte. Contrairement aux épi-détecteurs et aux épi-inspecteurs, l'action des épi-assistants est perceptible par les utilisateurs finaux. Certains épi-assistants permettent une interaction directe avec l'utilisateur, comme l'assistant qui réalise des consultations de l'utilisateur sous forme de fenêtres pop-up contenant des boutons de choix destinés à l'utilisateur final.

Les outils que nous proposons sont épiphytes : ils peuvent agir sur des applications-cibles existantes de manière externe, sans nécessiter de modification de ces applications et sans

qu'elles aient été spécifiquement conçues pour permettre l'utilisation des outils de SEPIA. Par ailleurs, les outils de SEPIA ne sont spécifiques ni à une application ni à un domaine. Au contraire, ils peuvent être utilisés avec des applications très variées, depuis certains petits gratuiciels, intéressants mais rudimentaires, développés par des amateurs, jusqu'aux applications professionnelles proposant des fonctionnalités très riches et élaborées.

Dans les prochains chapitres, nous présentons de manière détaillée le fonctionnement des outils de SEPIA que nous venons d'introduire.

Chapitre 8. Spécification de l'assistance avec SEPIA

Objectif du chapitre

Proposer un outil informatique rendant possible la spécification de l'assistance avec aLDEAS via une interface permettant à un concepteur d'assistance potentiellement non-informaticien de manipuler les éléments du langage.

Points clés

Nous présentons l'**éditeur d'assistance** de SEPIA, qui permet à un concepteur d'assistance, de spécifier avec **aLDEAS** l'assistance qu'il souhaite pour une application-cible. Cet éditeur permet de faire appel à un **épi-descripteur** adapté pour générer un fichier de description de l'interface de l'application-cible. Ce fichier permet d'identifier ses composants et fournit des connaissances relatives à l'application-cible. L'éditeur fournit également au concepteur de l'assistance une interface pour **manipuler les éléments du langage aLDEAS** afin de définir les règles d'assistance qui constituent le système d'assistance. Aucune connaissance d'aLDEAS ni de programmation n'est requise pour utiliser l'éditeur.

Contributions

- Éditeur d'assistance de SEPIA exploitant le langage aLDEAS
- Épi-descripteurs de SEPIA

Validation

- ✓ Utilisabilité de l'éditeur de SEPIA par des concepteurs d'assistance informaticiens et non-informaticiens
- ✓ Utilité de l'éditeur de SEPIA pour définir des systèmes d'assistance variés et conformes aux attentes de leur concepteur

Publications liées à ce chapitre

[Ginon et al., 2013c]

Ginon B., Jean-Daubias S. et Champin P.-A., Mise en place d'un système d'assistance personnalisée dans une application existante, IC, Lille, France, 2013c.

[Ginon et al., 2013d]

Ginon B., Jean-Daubias S. et Champin P.-A., Adjonction de systèmes d'assistance personnalisée à des EIAH existants, poster pour EIAH, Toulouse, 2013d.

[Ginon et al., 2014a]

Ginon B., Jean-Daubias S., Champin P.-A. et Lefevre M., aLDEAS : un langage de définition de systèmes d'assistance épiphytes, IC, Clermont-Ferrand, France, pp. 137-148, 2014a.

[Ginon et al., 2014b]

Ginon B., Thai V. L., jean-Daubias S., Lefevre M. et Champin P.-A., Adding epiphytic assistance systems in learning applications using the SEPIA system, Ec-Tel, Graz, Austria, 2014b.

Nous avons développé un éditeur d'assistance à destination des concepteurs d'assistance : il leur permet de spécifier l'assistance qu'ils souhaitent pour une application-cible. Cette assistance pourra ensuite être exécutée pour les utilisateurs finaux de cette application-cible. Les concepteurs d'assistance n'ont besoin d'aucune connaissance en programmation pour définir un système d'assistance avec l'éditeur de SEPIA, mais une bonne maîtrise de l'application-cible est nécessaire. L'éditeur d'assistance permet de spécifier l'assistance en deux étapes : la description de l'interface de l'application-cible, puis la définition de règles d'assistance. L'éditeur d'assistance est le premier des deux principaux outils qui composent le système SEPIA. Il met en œuvre la partie spécification de l'assistance du processus d'adjonction d'un système d'assistance à une application-cible. Dans ce chapitre, nous décrivons le fonctionnement de l'éditeur d'assistance du point de vue des utilisateurs.

Les éléments manipulés dans l'éditeur par le concepteur d'assistance correspondent aux éléments du langage aLDEAS (cf. Chapitre 5). Néanmoins, dans l'éditeur, le vocabulaire utilisé pour désigner ces éléments n'est pas exactement le même que celui employé dans le langage aLDEAS, dans le but de rendre l'utilisation de l'éditeur accessible à un plus large public. Ainsi, les attentes d'événements proposées par aLDEAS sont désignées dans l'éditeur par le terme « événements », car ils sont utilisés comme événements déclencheurs ou événements de fin dans les règles d'assistance. Les consultations proposées par aLDEAS sont quant à elles désignées dans l'éditeur par le terme « conditions », car elles sont exploitées comme conditions de déclenchement dans les règles d'assistance.

8.1. Spécification d'un système d'assistance

L'enchaînement des écrans de l'éditeur d'assistance de SEPIA est illustré en Fig. 8-1, chacun de ces écrans étant présenté en détail par la suite. Nous constatons que les écrans de l'éditeur peuvent être regroupés pour correspondre aux deux principales étapes de la création d'un système d'assistance : la description de l'interface de l'application-cible (cf. Ⓐ Fig. 8-1) et la définition des règles d'assistance (cf. Ⓑ Fig. 8-1).

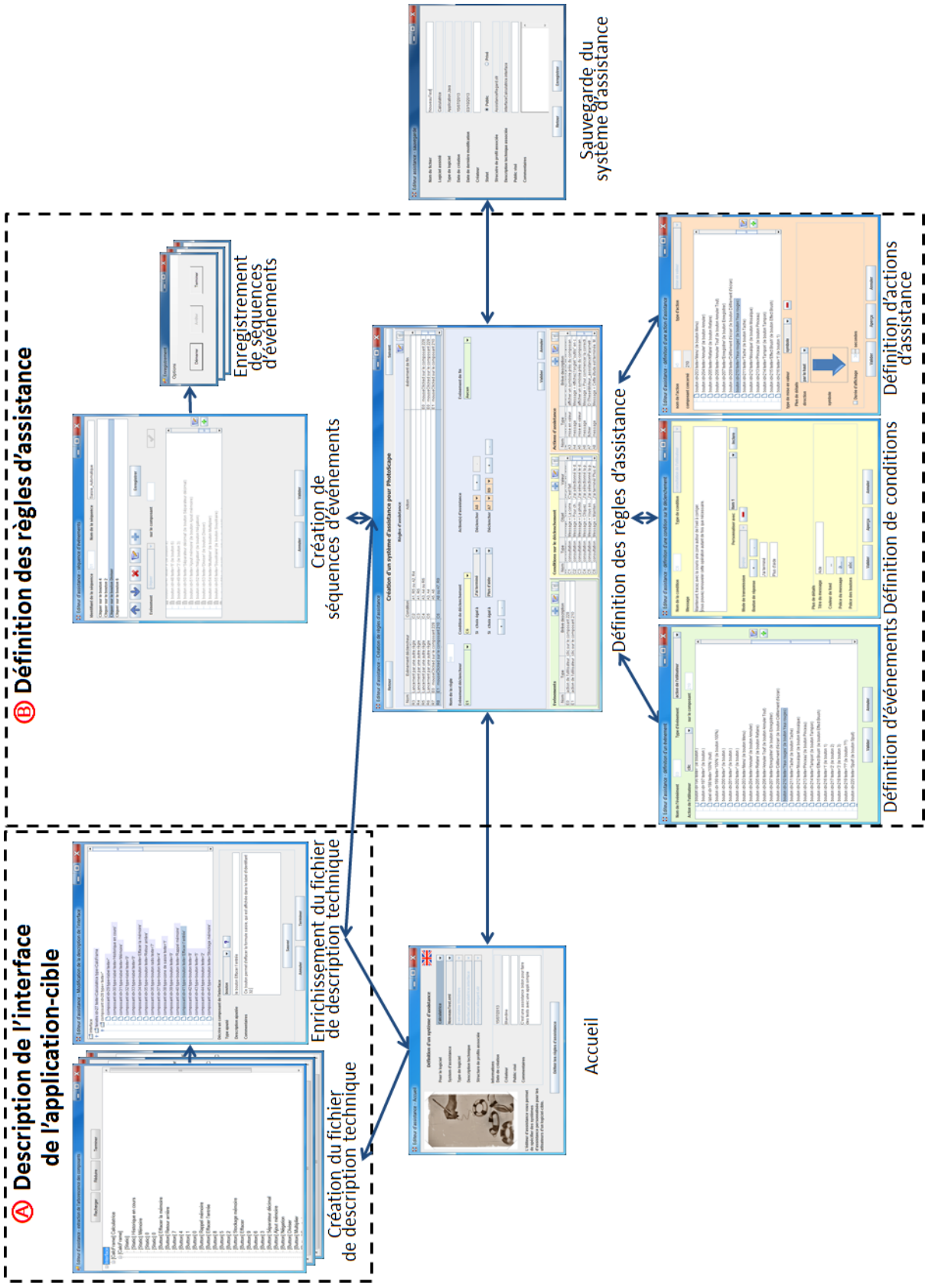


Fig. 8-1 – Enchaînement des écrans de l'éditeur d'assistance

Un système d'assistance créé avec l'éditeur de SEPIA est spécifique à une application-cible (cf. Ⓐ Fig. 8-2), associée à un type (application web, exécutable Windows, application Java ou application GTK / Qt[11], cf. © Fig. 8-2). De plus, un système d'assistance s'appuie sur une description de l'interface de son application-cible (cf. Ⓓ Fig. 8-2) qui permet d'identifier les composants de cette application. Un système d'assistance peut également être associé à une structure de profils (cf. Ⓔ Fig. 8-2) respectant le modèle PMDLe (cf. Section 6.1.1), afin de permettre les consultations du profil de l'utilisateur pour personnaliser l'assistance en fonction des spécificités de chaque utilisateur.

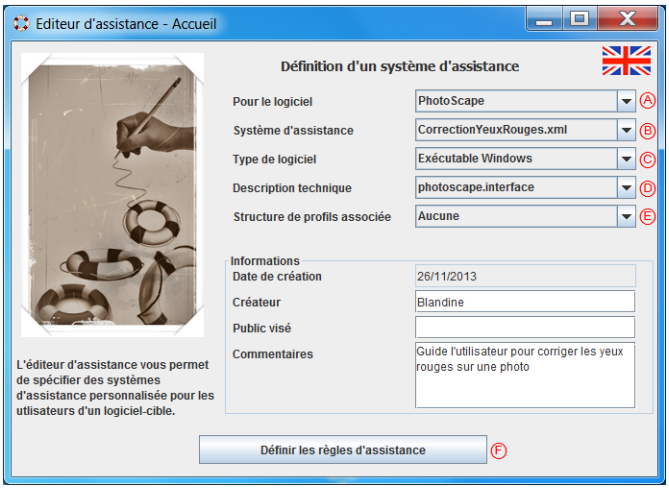


Fig. 8-2 – Écran d'accueil de l'éditeur de SEPIA

Pour créer un système d'assistance avec l'éditeur, le concepteur d'assistance doit sélectionner l'application-cible, ou l'ajouter si besoin à la liste des applications-cibles prises en charge par le système (cf. Ⓐ Fig. 8-2). Il doit ensuite nommer son système d'assistance (cf. Ⓑ Fig. 8-2). Le concepteur peut créer un nouveau fichier de description de l'interface de l'application-cible en sélectionnant l'item « nouvelle description technique… », ou en sélectionner un parmi ceux déjà créés (cf. Ⓓ Fig. 8-2). La création d'un nouveau fichier de description correspond à l'étape de description de l'interface de l'application-cible (cf. Ⓐ Fig. 8-1). Nous détaillons cette étape dans la section 8.2.

Après avoir renseigné ces informations indispensables à la création d'un système d'assistance, le concepteur peut accéder à l'étape principale de la spécification de l'assistance. Cette étape consiste à définir les règles qui décrivent l'assistance souhaitée pour l'utilisateur final (cf. Ⓕ Fig. 8-2, Ⓑ Fig. 8-1), ainsi que les constituants des règles : événements déclencheurs, conditions de déclenchement, actions d'assistance et événements de fin. Nous détaillons cette étape dans la section 8.3.

[11] Ces catégories d'applications correspondent à celles actuellement compatibles avec le système SEPIA

8.2. Description de l'interface d'une application-cible

L'étape de description de l'interface de l'application-cible permet la prise en charge d'une nouvelle application-cible pour laquelle des concepteurs d'assistance pourront par la suite spécifier des systèmes d'assistance. Elle peut être réalisée par un concepteur d'assistance ou par un expert de l'application-cible ou de SEPIA. Cette étape peut être réalisée une seule fois : plusieurs systèmes d'assistance pour une même application-cible peuvent exploiter une même description d'interface. Néanmoins, s'il le souhaite, le concepteur d'assistance peut ne pas réutiliser les fichiers de description déjà créés, mais en créer un nouveau pour le système d'assistance qu'il définit. Ceci peut être pertinent dans le cas où le concepteur de l'assistance souhaite enrichir personnellement ce fichier de description d'interface, afin de s'en approprier plus facilement le contenu (cf. Section 8.2.2).

L'étape de description de l'interface de l'application-cible aboutit à la création d'un fichier XML dans lequel chaque composant de l'interface de l'application-cible est associé à un identifiant utilisé par la suite par les différents outils de SEPIA pour le désigner.

8.2.1. Création du fichier de description de l'interface

Lors de la création d'un nouveau fichier de description de l'interface d'une application-cible, l'utilisateur doit préciser quel est le type de l'application-cible, afin que l'éditeur détermine quel outil de description, nommé **épi-descripteur**, doit être utilisé. Cette multiplicité des épi-descripteurs est représentée sur la Fig. 8-1 par un empilement des écrans (cf. © Fig. 8-1). Du point de vue de l'utilisateur, ces épi-descripteurs ont exactement la même interface, ce qui fait que le concepteur d'assistance n'a pas conscience de l'existence de plusieurs outils, ni du fait que ces outils sont externes à l'éditeur d'assistance.

Chaque épi-descripteur permet donc la création d'un fichier de description d'interface adapté à un type d'applications-cibles. La Fig. 8-3 présente par exemple une capture d'écran de l'épi-descripteur destiné aux exécutables Windows, qui exploite la bibliothèque d'accessibilité UIAutomation, pour décrire l'interface de la Calculatrice Windows. Les épi-descripteurs affichent automatiquement l'arborescence des composants de toutes les applications qu'ils détectent. Chacun de ces épi-descripteurs permet à l'utilisateur de valider ou supprimer un nœud sélectionné dans l'arbre des composants, afin de conserver dans la description uniquement les composants qui appartiennent à l'application-cible ou aux applications-cibles.

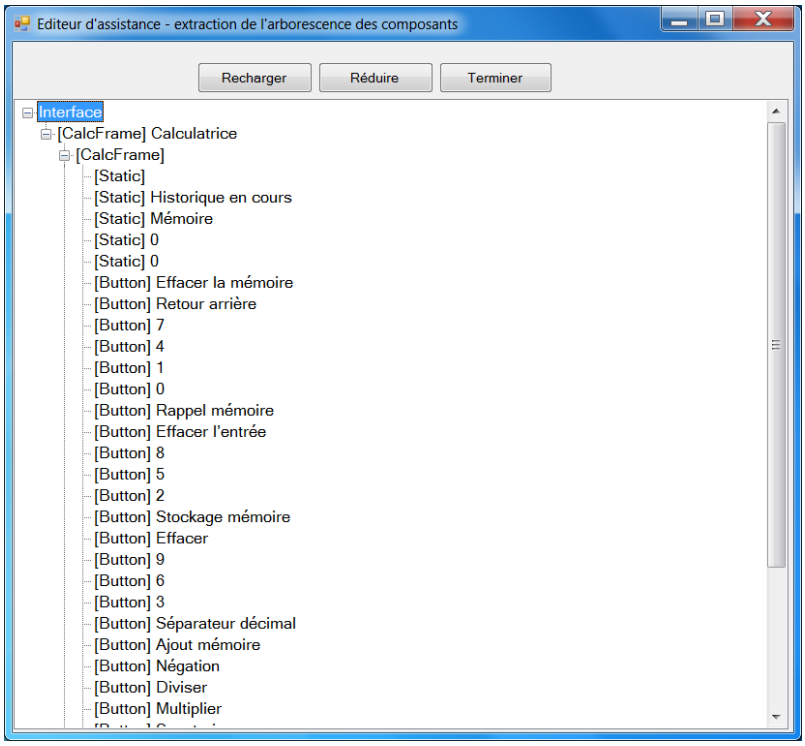


Fig. 8-3 – Création du fichier de description de l'interface de l'application-cible dans l'éditeur de SEPIA

8.2.2. Enrichissement du fichier de description de l'interface

Un fois créé, le fichier de description de l'interface peut être enrichi par le concepteur de l'assistance à tout moment de la spécification de l'assistance. Le concepteur de l'assistance ne peut toutefois pas modifier les attributs qui sont utilisés pour identifier un composant et remplis par l'épi-descripteur. En effet, ces informations seront utilisées au moment de l'exécution de l'assistance pour faire le lien entre un composant et son identifiant dans le fichier de description de l'interface. En revanche, celui-ci peut s'il le souhaite associer à chaque composant des attributs *typeAjoute*, *descriptionAjoutee* et *commentaires*. La Fig. 8-4 présente l'écran d'enrichissement de la description de l'interface de l'application-cible. Cet écran affiche dans sa partie supérieure l'arborescence des composants de l'application-cible et dans sa partie inférieure les attributs pouvant être modifiés pour le composant sélectionné dans l'arborescence.

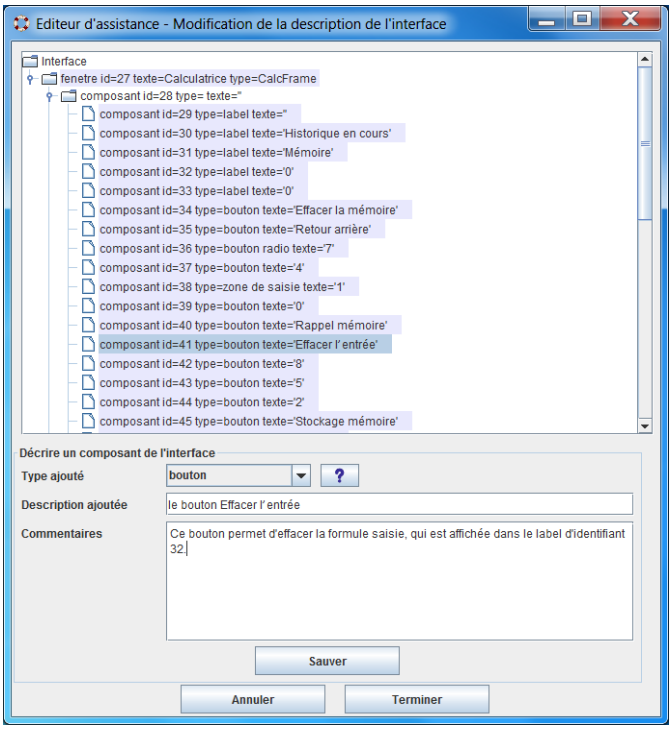


Fig. 8-4 – Enrichissement du fichier de description de l'interface de l'application-cible dans l'éditeur de SEPIA

L'attribut *typeAjoute* a pour valeur l'un des principaux types de composants d'interface graphiques que nous avons identifiés : bouton, bouton radio, zone de saisie, case à cocher, liste déroulante, item, panel, menu, label, arbre, fenêtre, table, liste, slider, lien, image, barre de progression ; ces types de composants sont illustrés en Annexe G. Le concepteur de l'assistance peut choisir la valeur de cet attribut, néanmoins elle est initialement remplie par défaut par l'éditeur d'assistance en fonction de l'attribut *type* (l'attribut *type* étant renseigné automatiquement par l'épi-descripteur et étant non modifiable). Pour cela, nous avons associé un *typeAjoute* à chaque *type* que nous avons rencontré dans les applications-cibles que nous avons décrites avec nos outils. Par exemple, le *type* JButton, que l'on retrouve dans les applications Java, est associé au *typeAjoute* bouton. L'attribut *typeAjoute* est donc à la fois plus générique et plus compréhensible par un concepteur d'assistance non-informaticien que l'attribut *type* qui est plus technique. Il est notamment utilisé lors de la définition d'une action automatisée : en effet, les types d'actions automatisées possibles dépendent du type de composant. Il est par exemple possible de modifier le texte d'une zone de saisie éditable mais pas celui d'un bouton.

Certains composants sont surlignés en bleu dans l'arborescence des composants (cf. Fig. 8-4). Cela indique que ces composants sont associés à un *typeAjoute*, soit parce que l'éditeur d'assistance a automatiquement reconnu leur *type*, soit parce que le concepteur a lui-même rempli un ou plusieurs attributs pour ces composants : ils sont donc susceptibles d'être importants aux yeux du concepteur. Partout où l'arborescence des composants est affichée, le concepteur a la possibilité de préciser qu'il ne souhaite voir que les composants ayant un *typeAjoute*. Cela permet dans certains cas de réduire fortement la taille de l'arbre des

composants visibles, en affichant uniquement les composants les plus susceptibles d'être exploités dans l'assistance, et ainsi de faciliter la recherche d'un composant dans l'arbre. Cette arborescence est par exemple affichée sur l'écran de définition d'une action de type mise en valeur d'un composant, afin de permettre au concepteur de l'assistance de sélectionner le composant à mettre en valeur (cf. Fig. 8-17 en section 8.3.4).

L'attribut *descriptionAjoutee* est un texte pouvant être rempli par le concepteur, afin par exemple d'expliquer à quoi sert ce composant. Initialement, l'éditeur d'assistance propose une *descriptionAjoutee* par défaut, construite à partir du *typeAjoute* du composant et de son texte ou de sa description d'accessibilité s'il y en a une. Par exemple, pour le bouton de la calculatrice Windows « MC », dont le *typeAjoute* est « bouton » et la description d'accessibilité « Effacer l'entrée », la *descriptionAjoutee* proposée par défaut par l'éditeur est « le bouton Effacer l'entrée » (cf. Ⓐ Fig. 8-4). L'attribut *descriptionAjoutee* est utilisé notamment pour générer automatiquement des messages par défaut, modifiables par le concepteur pour les actions d'assistance de type pas à pas ou présentation guidée ; nous en verrons un exemple en Section 8.3.4. L'attribut *descriptionAjoutee* est également affiché entre parenthèses à la suite de chaque composant dans tous les écrans de l'éditeur qui affichent l'arborescence des composants (cf. Fig. 8-5).

L'attribut *commentaires* est un texte libre dans lequel le concepteur de l'assistance peut préciser toutes les informations qu'il souhaite. Cet attribut est vide par défaut. Si le concepteur remplit cet attribut, il est alors affiché dans une bulle d'aide lors du survol par la souris du composant concerné dans tous les écrans de l'éditeur qui affichent l'arborescence des composants.

La Fig. 8-5 présente un premier exemple d'utilisation des attributs d'enrichissement de la description de l'interface d'une application-cible pour l'affichage de l'arborescence des composants. Le composant d'identifiant 210 a pour attribut *typeAjoute* la valeur « bouton » et sa *descriptionAjoutee* est « le bouton Yeux rouges ». Son attribut *commentaires* est « Permet de sélectionner une zone de correction sur la photo », il s'affiche lors du survol de ce composant.

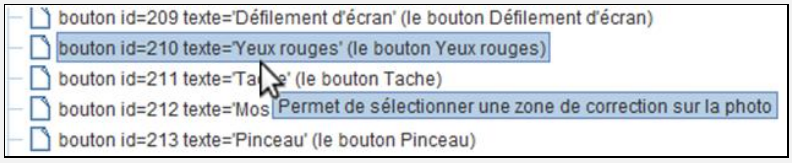


Fig. 8-5 – Exemple d'utilisation des attributs d'enrichissement de la description de l'interface d'une application-cible pour l'affichage de la hiérarchie des composants

À titre d'exemple, la Fig. 8-7 et la Fig. 8-8 présentent l'affichage de l'arborescence des composants dans l'éditeur, correspondant à un menu appartenant à l'application-cible NetBeans (dont une capture d'écran est présentée en Fig. 8-6), en affichage simplifié (cf. Fig. 8-7) et en affichage complet (cf. Fig. 8-8). Nous constatons que l'affichage simplifié permet au concepteur de l'assistance de repérer plus facilement les composants de l'interface avec lesquels l'utilisateur final de NetBeans interagit principalement (notamment les boutons et listes déroulantes du menu), en ignorant les nombreux composants de type conteneur qui hiérarchisent l'interface de NetBeans, mais qui ne sont pas matérialisés pour l'utilisateur final de NetBeans. Ces conteneurs peuvent toutefois être utiles dans l'assistance, par exemple la mise en valeur par encadrement d'un composant de type conteneur permet de mettre en valeur tous les composants qu'il contient.

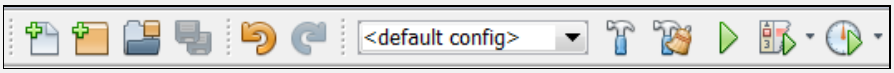


Fig. 8-6 – Menu de la fenêtre principale de NetBeans

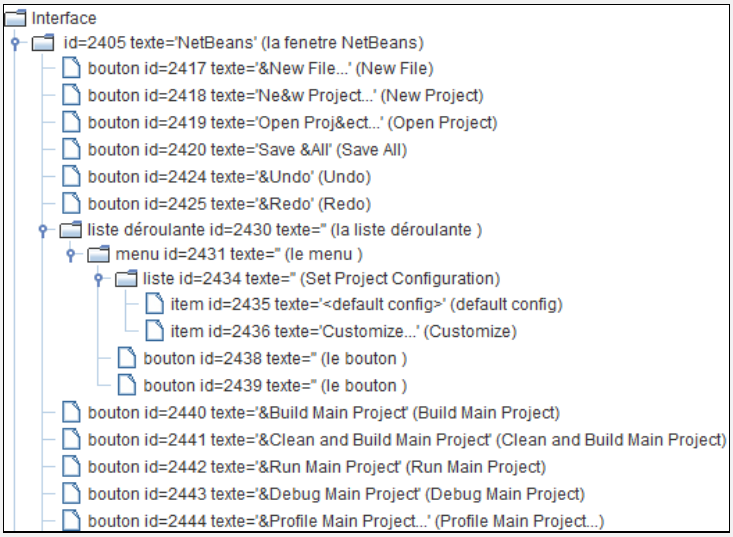


Fig. 8-7 – Affichage « simplifié » de la description de l'interface de NetBeans dans l'éditeur d'assistance

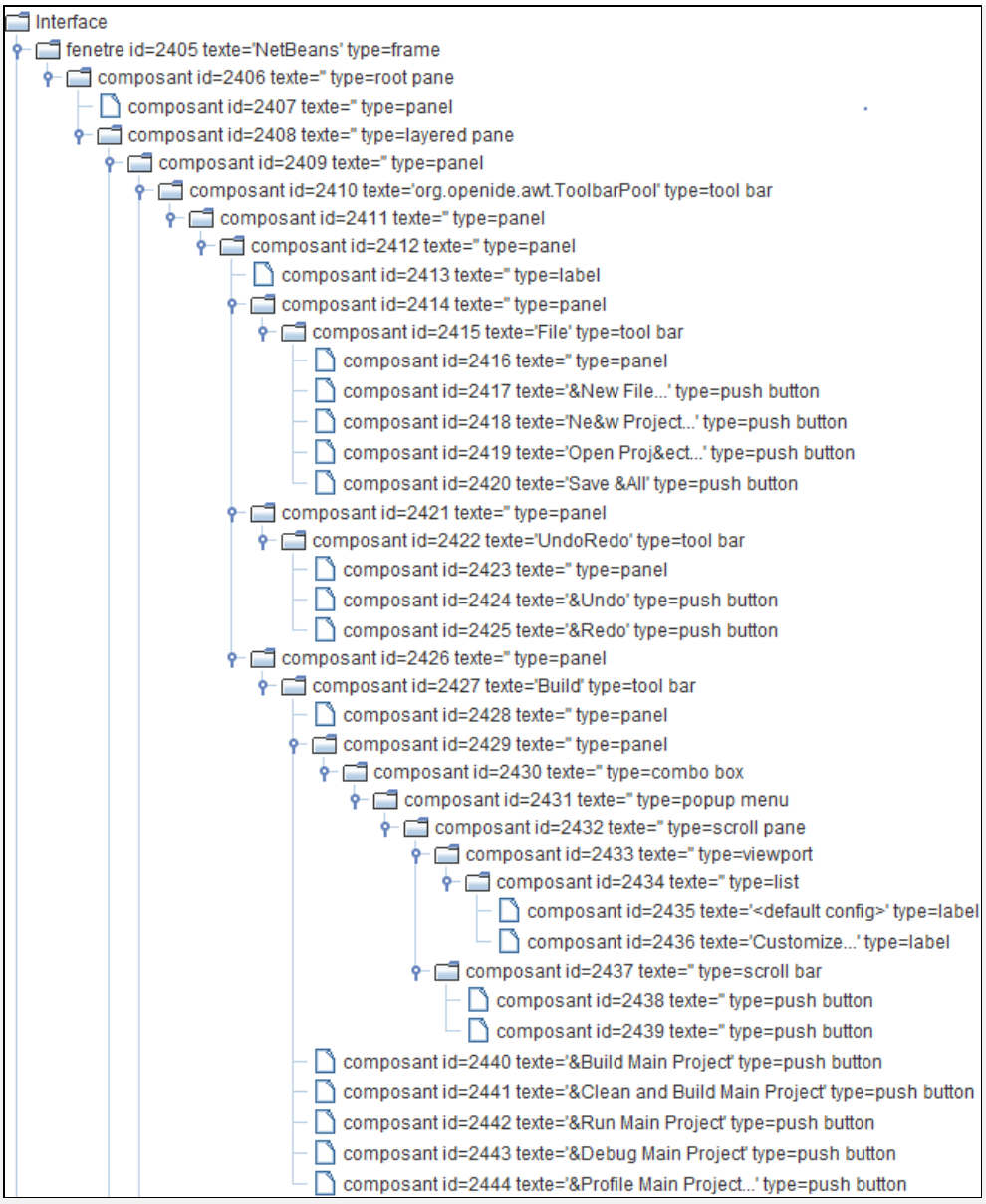


Fig. 8-8 – Affichage complet de la description de l'interface de NetBeans dans l'éditeur d'assistance

8.3. Définition des règles du système d'assistance

Après l'étape de description de l'interface de l'application-cible, la seconde et principale étape de la spécification de l'assistance consiste à définir les règles d'assistance. Cette étape met en œuvre le patron de règles d'assistance d'aLDEAS présenté en section 5.2.1, avec la restriction suivante : les actions composées d'assistance doivent soit respecter l'un des patrons d'actions d'assistance présentés en section 5.2 (c'est-à-dire être de type pas à pas, animation d'agent ou présentation guidée), soit être une succession d'actions élémentaires ou non élémentaires d'aLDEAS. En effet, nous avons fait le choix de ne pas mettre en œuvre la totalité du langage aLDEAS afin de simplifier la tâche du concepteur d'assistance en lui

proposant des interfaces spécifiquement adaptées à la création de règles et actions d'assistance dont la forme est la plus pertinente ou la plus fréquemment utilisée dans les systèmes d'assistance existants.

8.3.1. Définition d'une règle d'assistance

Définir des règles d'assistance permet de spécifier le comportement du système d'assistance lors de la phase d'exécution de l'assistance par le moteur générique. Les règles d'assistance combinent des attentes d'événements qui permettent la contextualisation de l'assistance avec des consultations suivies d'alternatives qui permettent de personnaliser et contextualiser l'assistance, ainsi qu'avec des actions d'assistance afin de répondre aux besoins d'assistance des utilisateurs finaux de l'application-cible.

L'écran principal de l'éditeur d'assistance permet la création et la modification de règles d'assistance (cf. Ⓑ Fig. 8-9). Il permet aussi de visualiser les règles créées (cf. Ⓐ Fig. 8-9), ainsi que les événements déclencheurs ou finaux (cf. Ⓒ Fig. 8-9), les conditions de déclenchement (cf. Ⓓ Fig. 8-9) et les actions d'assistance (cf. Ⓔ Fig. 8-9) créés.

Le concepteur de l'assistance peut définir l'assistance dans l'ordre qu'il souhaite : il peut commencer par créer tous les éléments nécessaires avant de définir les règles d'assistance, créer les événements, la condition et les actions nécessaires avant de définir chaque règle ou définir ces éléments lorsqu'ils sont nécessaires pendant la création d'une règle. En effet, dans la zone de création d'une règle d'assistance (cf. Ⓑ Fig. 8-9), les listes déroulantes qui permettent de sélectionner un événement, une condition ou une action contiennent également un raccourci vers la définition d'un nouvel événement, condition ou action. Ceci permet à l'utilisateur de ne pas interrompre la définition d'une règle d'assistance lorsqu'un autre élément doit être créé et rend l'utilisation de l'éditeur plus flexible.

La définition en XML d'un élément est générée automatiquement par l'éditeur d'assistance : l'éditeur de SEPIA fournit en effet une interface au concepteur d'assistance pour lui éviter d'avoir à créer à la main un fichier XML compatible avec les outils de SEPIA et notamment avec le moteur d'assistance qui exécute l'assistance.

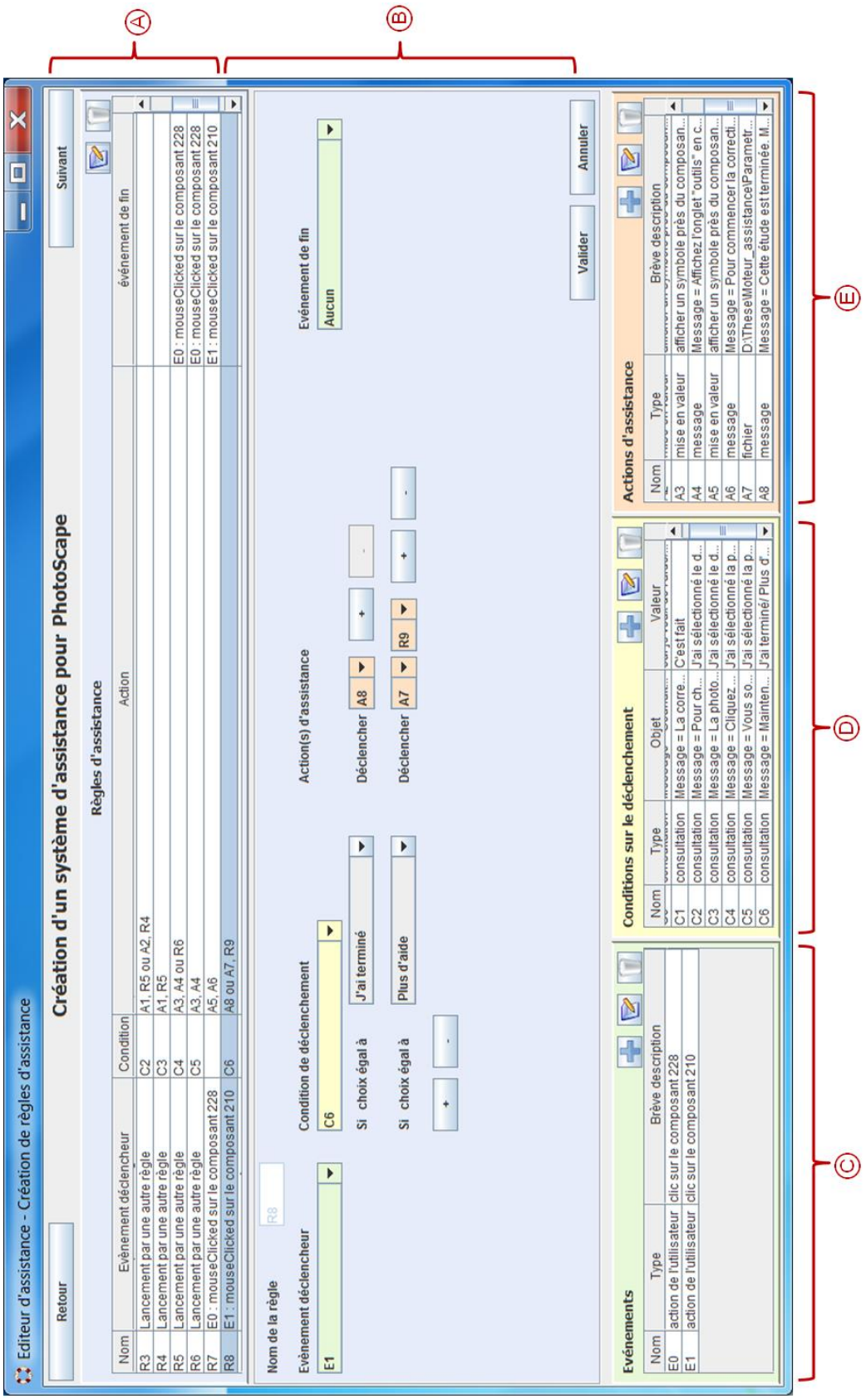


Fig. 8-9 – Écran principal de l'éditeur de SEPIA : définition de règles d'assistance

À titre d'exemple, la Fig. 8-9 présente la définition de la règle d'assistance R8, pour l'application-cible PhotoScape. Cette règle a pour objectif d'expliquer à l'utilisateur comment effectuer la dernière étape de la correction des yeux rouges sur une photo. Son événement déclencheur est une action de l'utilisateur : le clic sur le composant 210 (le bouton « Yeux rouges » de PhotoScape). La condition de déclenchement de cette règle est de type consultation de l'utilisateur : un message lui expliquant comment tracer sur la photo une sélection autour des yeux rouges à corriger est affiché (cf. * sur la Fig. 8-10) et l'utilisateur peut choisir entre deux options : « J'ai terminé » et « Plus d'aide ». Dans le premier cas, l'action d'assistance A8 (un message indiquant la fin de la tâche de correction des yeux rouges) est déclenchée ; et dans le second cas, l'action d'assistance A7 (une vidéo de démonstration), ainsi que la règle d'assistance R9 sont déclenchées. Conformément à la syntaxe d'aLDEAS, la vidéo de démonstration est déclenchée, suivie immédiatement de la règle R9 sans attendre la fin de la vidéo de démonstration. Ces deux éléments sont donc visibles par l'utilisateur final simultanément.

Les Fig. 8-10 et Fig. 8-11 présentent respectivement la définition de la règle d'assistance R8 en aLDEAS et sa définition en XML contenue dans le fichier de description de l'assistance pour PhotoScape produit à l'aide de l'éditeur de SEPIA.

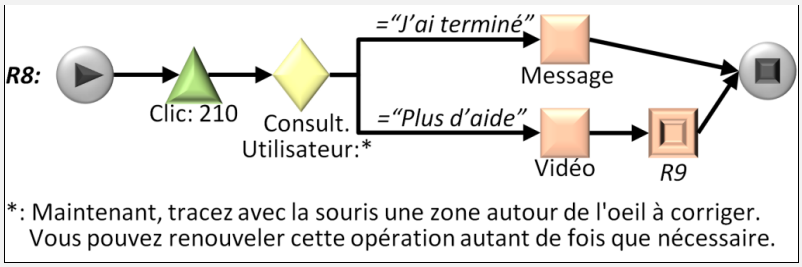


Fig. 8-10 – Définition en aLDEAS de la règle d'assistance R8 de l'assistance pour PhotoScape

```
<regle id="R8">
  <evenement_declencheur idEve="E1"/>
  <alternatives condition="C6">
    <alternative valeur="J'ai terminé">
      <action id="A8" />
    </alternative>
    <alternative valeur="Plus d'aide">
      <action id="A7" />
      <action id="R9" />
    </alternative>
  </alternatives>
</regle>
```

Fig. 8-11 – Définition en XML de la règle d'assistance R8 issue du fichier de description de l'assistance pour PhotoScape

8.3.2. Définition d'événements déclencheurs ou finaux

L'éditeur d'assistance fournit une interface (cf. Fig. 8-12) pour la définition d'attentes d'événements proposées par le langage aLDEAS (cf. Section 0). Ces attentes d'événements, simplement nommées « événements » dans l'éditeur, peuvent être utilisées comme événement déclencheur ou comme événement de fin pour une règle d'assistance.

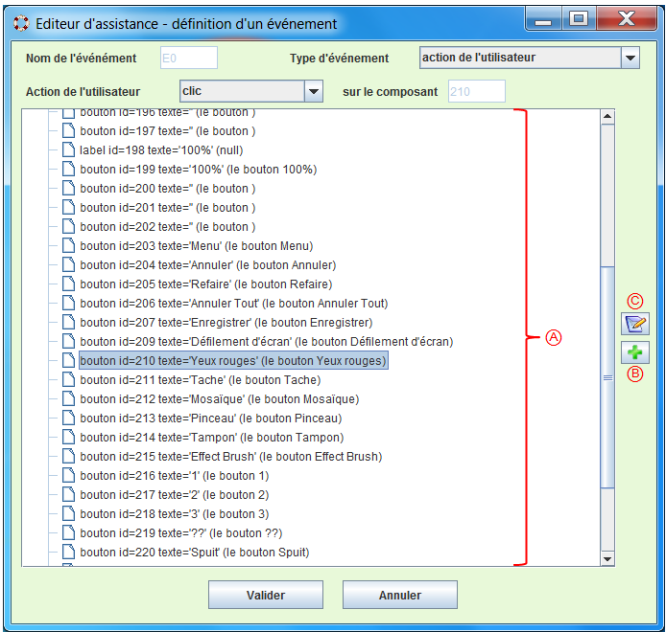


Fig. 8-12 – Définition d'événements dans l'éditeur de SEPIA : définition d'un événement lié à l'action de l'utilisateur final

Les événements supportés par SEPIA peuvent concerner : une action de l'utilisateur ou du système d'assistance ; l'attente d'une durée donnée suite à un événement donné ; ou l'attente d'une durée donnée suite à une absence d'événement. Chaque type d'événement est associé à des paramètres définis avec aLDEAS comme présenté en section 0.

Un événement concernant une **action de l'utilisateur** est associé à un type d'action de l'utilisateur (clic, passage de la souris, gain/perte de focus, ouverture/fermeture de menu ou d'infobulle, saisie) et éventuellement à un composant donné de l'interface de l'application cible. Si aucun composant n'est précisé, l'événement concerne l'action précisée sur n'importe quel composant. Par exemple, un événement peut concerner un clic de l'utilisateur sur un bouton donné, ou un déplacement quelconque de la souris.

Un événement concernant une **action du système d'assistance** est associé d'une part à un type (déclenchement ou fin) et d'autre part à une règle, condition ou action donnée ou à un ou plusieurs types d'éléments. Par exemple, un événement peut concerner la fin d'une règle d'assistance donnée, le déclenchement de n'importe quelle règle, ou le déclenchement de n'importe quelle règle ou action.

Un événement concernant l'**attente d'une durée suite à un événement** est associé à une durée et à un événement. Par exemple, un événement peut concerner une attente suite au déclenchement d'une action d'assistance ou suite à une action de l'utilisateur.

Un événement concernant une **absence d'événement** est associé à une durée et à un événement. Par exemple, un événement peut concerner une absence de déplacement de la souris, de clic sur un bouton donné, ou l'absence de déclenchement de n'importe quelle action d'assistance.

L'écran de définition d'événements concernant une action de l'utilisateur présente l'arborescence des composants de l'application-cible stockée dans le fichier de description de l'application-cible (cf. Ⓐ Fig. 8-12), afin de permettre au concepteur de l'assistance de sélectionner un composant.

Sur l'exemple de la Fig. 8-12, le concepteur d'assistance définit un événement, E0, qui concerne une action de l'utilisateur : un clic sur le composant d'identifiant 210 (le bouton Yeux rouges de l'application-cible PhotoScape).

La Fig. 8-13 présente la définition en XML de l'événement E0, en cours de définition sur la Fig. 8-12 et issue du fichier de description de l'assistance pour l'application-cible PhotoScape créé avec l'éditeur d'assistance de SEPIA.

```
<evenement typeEve="action utilisateur" id="E0"
           type="mouseClicked" idComp="210" />
```

Fig. 8-13 – Définition en XML de l'événement E0

8.3.3. Définition de conditions de déclenchement

L'éditeur d'assistance fournit une interface (cf. Fig. 8-14) pour la définition des consultations proposées par le langage aLDEAS et présentées en section 5.1.2 : consultations de l'utilisateur, de l'interface de l'application-cible, du profil de l'utilisateur ou de l'historique de l'assistance. À des fins de simplifications, les consultations proposées par aLDEAS sont nommées dans l'éditeur « conditions » puisqu'elles sont uniquement utilisées comme conditions de déclenchement de règles d'assistance. Ces conditions sont associées à des paramètres obligatoires et à des paramètres optionnels, conformément à leur définition en aLDEAS.

Pour tous les éléments pouvant être visualisés (comme les consultations de l'utilisateur mais aussi certaines actions d'assistance telles que les messages, les mises en valeur et les animations) l'éditeur propose un aperçu de l'élément en cours de définition (cf. Ⓕ Fig. 8-14), afin d'aider le concepteur à visualiser l'assistance qu'il définit. Néanmoins, si le concepteur n'a pas spécifié tous les paramètres optionnels de l'élément, l'aperçu ne correspondra pas forcément exactement à ce que l'utilisateur final visualisera lors de l'exécution de l'assistance. En effet, les paramètres optionnels non spécifiés par le concepteur de l'assistance permettent la prise en compte des préférences de l'utilisateur final de l'application-cible, s'il en a définies, et sont donc personnalisées pour chaque utilisateur.

Néanmoins, afin de permettre un aperçu, l'éditeur d'assistance exploite des valeurs par défaut permettant de compléter les paramètres optionnels non spécifiés par le concepteur.

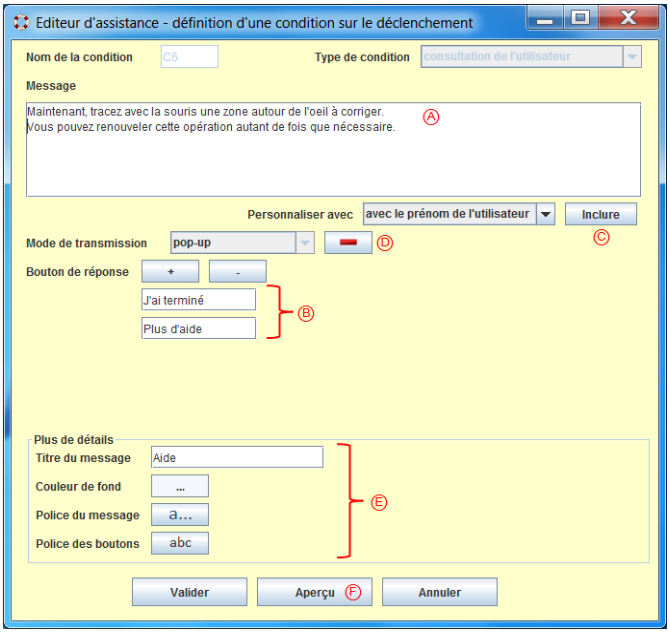


Fig. 8-14 – Définition de conditions dans l'éditeur de SEPIA : définition d'une consultation de l'utilisateur

Sur l'exemple donné par la Fig. 8-14, la condition en cours de définition, C6, est de type consultation de l'utilisateur. Les consultations de l'utilisateur sont associées à deux paramètres obligatoires : un contenu (cf. Ⓐ Fig. 8-14) et une liste d'options entre lesquelles l'utilisateur pourra choisir lors de l'exécution de cette consultation (cf. Ⓑ Fig. 8-14). Le contenu d'une consultation peut être personnalisé avec des éléments du profil de l'utilisateur (cf. Ⓒ Fig. 8-14), comme son nom, son prénom ou la valeur associée à un élément de son profil, par exemple son niveau de maîtrise d'une tâche. Les consultations de l'utilisateur sont également associées à des paramètres optionnels : mise en forme, titre et position. Le concepteur de l'assistance peut afficher ou masquer (cf. Ⓓ Fig. 8-14) la partie « plus de détails » (cf. Ⓔ Fig. 8-14) qui permet la spécification de ces paramètres.

La Fig. 8-15 présente la définition en XML de la condition de déclenchement C6, en cours de définition sur la Fig. 8-14, issue du fichier de description de l'assistance pour l'application-cible PhotoScape qui est généré par l'éditeur de SEPIA en fonction des choix effectués à l'interface par le concepteur de l'assistance. La Fig. 8-16 présente quant à elle l'aperçu correspondant à la condition C6. En effet, un aperçu a du sens pour les consultations de l'utilisateur, car contrairement aux autres types de consultations proposées par aLDEAS, comme les consultations du profil de l'utilisateur ou de l'historique de l'assistance, elles sont visibles par l'utilisateur final.

```
<condition id="C6" type="consultation">
  <texte>Maintenant, tracez avec la souris une zone
  autour de l'oeil à corriger. \nVous pouvez renouveler
  cette opération autant de fois que nécessaire.</texte>
  <titre>Aide</titre>
  <policeMessage nom="Calibri" style="0" taille="20" />
  <policeOptions nom="Calibri" style="0" taille="16" />
  <couleur police="sun.swing.PrintColorUIResource[r=46,g=79,b=118]"
           fond="java.awt.Color[r=243,g=247,b=251]" />
  <options>
    <option label="J'ai terminé" />
    <option label="Plus d'aide" />
  </options>
</condition>
```

Fig. 8-15 – Définition en XML de la condition de déclenchement C6

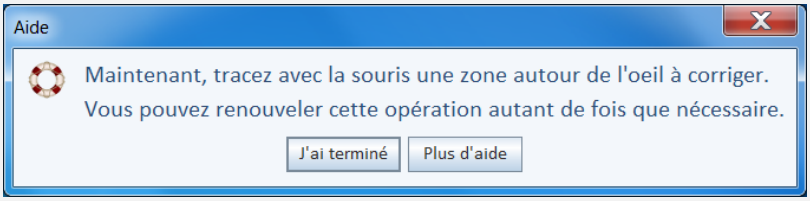


Fig. 8-16 – Exécution de la condition de déclenchement C6 pour PhotoScape

8.3.4. Définition d'actions d'assistance

L'éditeur d'assistance fournit une interface (cf. Fig. 8-17) pour la définition d'actions élémentaires d'assistance proposées par le langage aLDEAS et présentées en section 5.1.3 : messages, mises en valeur, masquages, actions sur l'interface de l'application-cible, ressources externes, animations et ajouts de composants. Cette interface permet au concepteur de l'assistance de spécifier les paramètres obligatoires d'une action d'assistance, ainsi que ses paramètres optionnels s'il le souhaite. Les différents épi-assistants développés pour permettre l'exécution de ses actions d'assistance dans l'application-cible sont présentés en Section 9.3. Dans la suite de cette section, nous détaillons l'exemple d'une action d'assistance créée à partir de l'éditeur (cf. Fig. 8-17) : nous donnons sa définition en XML, puis illustrons son exécution par le moteur d'assistance. La Section 9.3 propose d'autres exemples d'actions d'assistance définies à l'aide de l'éditeur.

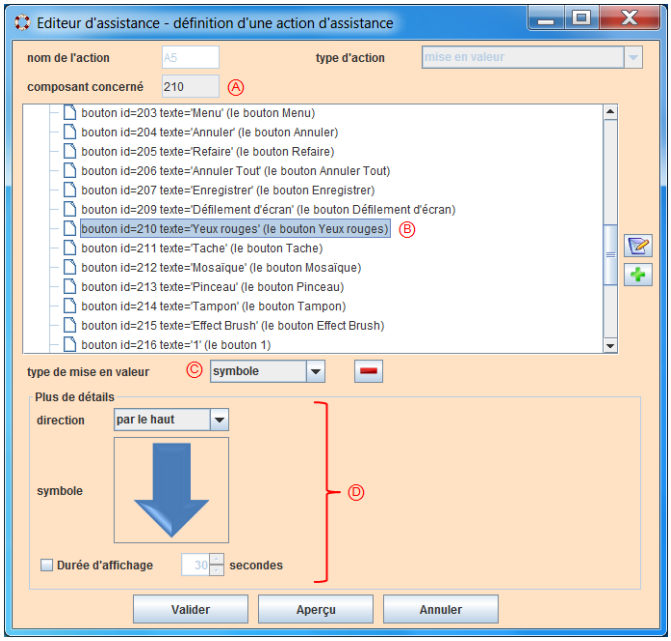


Fig. 8-17 – Définition d'actions d'assistance dans l'éditeur de SEPIA : exemple d'une mise en valeur

Sur l'exemple de la Fig. 8-17, le concepteur d'assistance définit A5, une action élémentaire intégrée à l'interface de l'application-cible de type mise en valeur. Cette action concerne le composant d'identifiant 210 (le bouton « Yeux rouges » de l'application-cible PhotoScape, cf. Ⓐ et Ⓑ Fig. 8-17). Le concepteur de l'assistance a spécifié le paramètre optionnel qui définit le type de mise en valeur : le composant sera mis en valeur par un symbole (cf. Ⓒ Fig. 8-17). Il a également spécifié certains paramètres optionnels d'une mise en valeur par symbole disponibles dans la zone « Plus de détails » (cf. Ⓓ Fig. 8-17) : le symbole (une flèche bleue) et sa direction (le symbole sera affiché au-dessus du composant concerné).

La Fig. 8-18 présente la définition en XML de l'action d'assistance A5 issue du fichier de description de l'assistance pour l'application-cible PhotoScape. La Fig. 8-19 montre l'exécution de cette action d'assistance dans PhotoScape.

```
<action id="A5" type="mise en valeur">
  <composant id="210" type="symbole"
     direction="par le haut" symbole="flecheBleue.png"/>
</action>
```

Fig. 8-18 – Définition en XML de l'action d'assistance A5

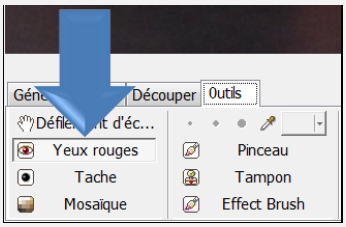


Fig. 8-19 – Exécution de l'action d'assistance A5 pour PhotoScape

L'éditeur d'assistance fournit également une interface pour la définition d'actions d'assistance qui instancient l'un des patrons d'actions d'assistance qui complètent le langage aLDEAS (cf. Section 5.2.2) : pas à pas, présentations guidées et actions d'agents animés.

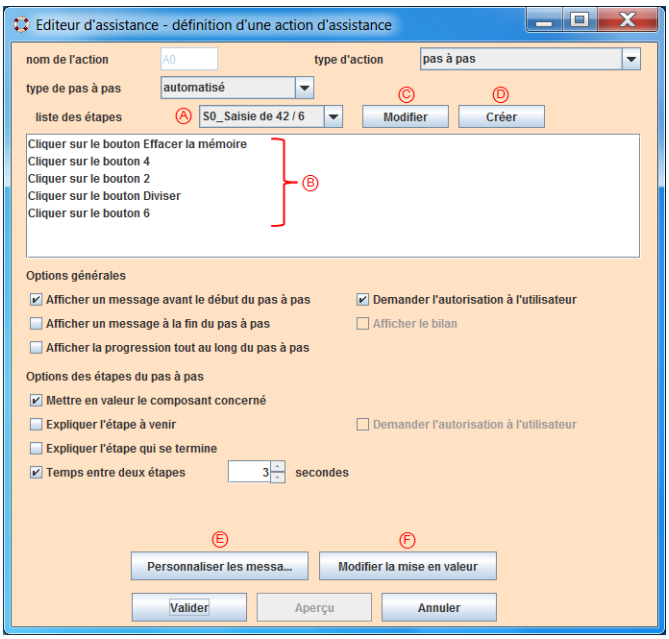


Fig. 8-20 – Définition d'actions d'assistance dans l'éditeur de SEPIA : définition d'un pas à pas

Sur l'exemple de la Fig. 8-20, le concepteur définit une action d'assistance A0, de type pas à pas en mode automatisé pour la Calculatrice Windows. Il s'agit d'une action d'assistance composée, qui instancie les patrons de pas à pas et d'étape de pas à pas automatisé (cf. Fig. 5-30 et Fig. 5-31 en Section 5.2.2). Cette action permet de proposer à l'utilisateur une assistance qui saisira automatiquement la formule 42/6 dans la calculatrice, en mettant en valeur chaque bouton pendant 3 secondes. La définition en aLDEAS de cette action d'assistance est donnée en section 5.2.2 (cf. Fig. 5-33 et Fig. 5-34). Les actions de type pas à pas, ainsi que les actions de type présentation guidée, prennent en paramètre une séquence d'événements qui détermine les étapes du pas à pas. Le pas à pas A0 est associé à la séquence d'événements « S0_Saisie de 42/6 » (cf. Ⓐ Fig. 8-20), dont les étapes sont listées en Ⓑ sur la Fig. 8-20. Lorsqu'il crée une action impliquant une séquence d'événements, le concepteur peut créer une nouvelle séquence (cf. © Fig. 8-20) ou modifier une séquence existante (cf. Ⓓ Fig. 8-20). La création et la modification de séquences d'événements sont détaillées dans la section suivante.

Lorsqu'il crée une action de type pas à pas ou présentation guidée, le concepteur a la possibilité de modifier le contenu des messages et des consultations (cf. Ⓔ Fig. 8-20), ainsi que leurs paramètres optionnels. L'éditeur d'assistance exploite la description de l'interface de l'application-cible pour générer des messages par défaut. Par exemple, pour une étape de pas à pas automatisé qui concerne le clic sur le bouton « Effacer la mémoire », si la description associée à ce bouton est « le bouton qui permet d'effacer la formule saisie », le message proposé par défaut sera « nous allons cliquer sur le bouton qui permet d'effacer la formule saisie ». La proposition de messages par défaut permet au concepteur de définir simplement et rapidement une action composée de type pas à pas, à l'aide de l'interface de l'éditeur. Le concepteur peut également définir les paramètres optionnels des mises en valeur impliquées

dans le pas à pas (cf. Ⓕ Fig. 8-20). Par défaut, les paramètres optionnels des messages et mises en valeur impliqués dans un pas à pas ne sont pas spécifiés afin de permettre la prise en compte des préférences de l'utilisateur final de l'application-cible lors de l'exécution de l'assistance.

La Fig. 8-21 présente la définition en XML de l'action d'assistance A0 de type pas à pas, issue du fichier de description de l'assistance pour l'application-cible Calculatrice Windows. On constate que le concepteur de l'assistance a modifié le message par défaut proposé par l'éditeur pour la consultation de l'utilisateur qui précède le déroulement des étapes du pas à pas (cf. Ⓐ Fig. 8-21). On note également, à leur absence, que le concepteur a choisi de ne pas spécifier les paramètres optionnels de la consultation de l'utilisateur ni des mises en valeur impliquées dans le pas à pas pour permettre une prise en compte des préférences des utilisateurs finaux. Cela lui permet également de définir plus rapidement cette action d'assistance.

```
<action id="A0" type="pas à pas" typeAuto="automatisé" >
  <message type="avant" demande="oui">
  Ⓐ <texte>Veux-tu que je saisisse la formule pour toi?</texte>
    <titre>Aide</titre>
    <options>
      <option label="oui" />
      <option label="non" />
    </options>
  </message>
  <etapes timer="3000" miseEnValeur="oui" listeEtapes="S0" >
    <evenement idComp="34" propriete="clic" />
    <evenement idComp="37" propriete="clic" />
    <evenement idComp="44" propriete="clic" />
    <evenement idComp="53" propriete="clic" />
    <evenement idComp="48" propriete="clic" />
  </etapes>
</action>
```

Fig. 8-21 – Définition en XML de l'action d'assistance A0

8.3.5. Définition de séquences d'événements

Les actions d'assistance de type pas à pas ou présentation guidée prennent en paramètre une séquence d'événements élémentaires : chaque étape du pas à pas ou de la présentation guidée concerne un événement de cette séquence. L'éditeur d'assistance propose une interface pour la définition de séquences d'événements (cf. Fig. 8-22), qui permet d'ajouter, de supprimer ou de modifier un événement de la séquence, ainsi que de réorganiser la séquence en déplaçant les événements. Chaque événement concerne un composant de l'interface de l'application-cible. Pour une séquence d'événements destinée à un pas à pas, chaque événement est associé à un type d'événement, qui correspond à une action possible de l'utilisateur (pour un pas à pas guidé) ou à une action automatisée possible (pour un pas à pas automatisé) et qui dépend du type du composant. Par exemple, pour un bouton, les événements disponibles dans le cadre d'un pas à pas automatisé sont *cliquer* et *donner le focus*. Pour une séquence d'événements destinée à une présentation guidée, le seul type d'événement disponible est *montrer*. En effet, lors d'une présentation guidée, le système d'assistance ne réalise aucune action automatisée dans l'application-cible.

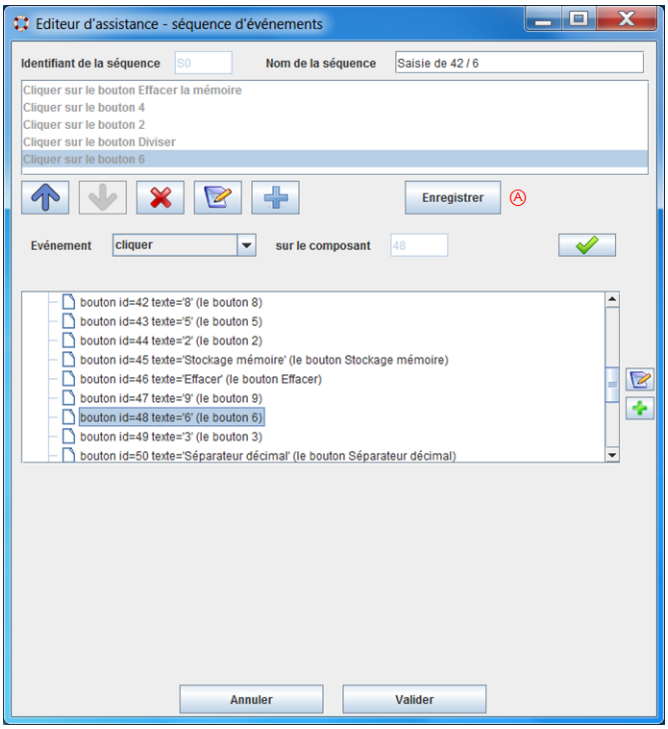


Fig. 8-22 – Définition de séquences d'événements dans l'éditeur de SEPIA

La Fig. 8-22 présente l'exemple de la définition de la séquence S0 nommée « Saisie de 42/6 » pour l'application-cible Calculatrice Windows. Cette séquence est composée de 5 événements élémentaires de type clic sur un bouton. La Fig. 8-23 présente sa définition en XML.

```
<sequence id="S0" nom="Saisie de 42/6" type="PasAPas">
  <evenement idComp="34" propriete="clic"/>
  <evenement idComp="37" propriete="clic"/>
  <evenement idComp="44" propriete="clic"/>
  <evenement idComp="53" propriete="clic"/>
  <evenement idComp="48" propriete="clic"/>
</sequence>
```

Fig. 8-23 – Définition en XML de la séquence d'événements S0 contenue dans le fichier de description de l'assistance pour la Calculatrice Windows

Afin de faciliter la tâche du concepteur de l'assistance, l'éditeur lui permet également d'enregistrer une séquence d'événements directement en manipulant l'interface de l'application-cible (cf. Ⓐ Fig. 8-22). Pour cela, un outil d'enregistrement de séquences d'événements est lancé en fonction du type de l'application-cible. En effet, la technique pour enregistrer une séquence d'événements s'appuie sur la technique de surveillance d'une application-cible. L'enregistrement peut être paramétré afin de choisir les types d'événements à enregistrer. Par exemple, il est possible d'ignorer les déplacements de la souris. Lorsqu'il est prêt, l'utilisateur peut démarrer l'enregistrement : tous les événements (choisis lors du paramétrage) sur les composants de l'interface de l'application-cible seront enregistrés puis présentés sur l'écran de définition de séquences d'événements de l'éditeur. L'utilisateur peut alors modifier et/ou compléter la séquence issue de l'enregistrement.

8.4. Conclusion

L'éditeur d'assistance de SEPIA permet la spécification de systèmes d'assistance variés. Il est destiné au concepteur de l'assistance qui n'est pas nécessairement un informaticien, et lui fournit une interface pour manipuler l'ensemble des éléments d'aLDEAS, et plus particulièrement pour définir un ensemble de règles instanciant le patron de règles d'assistance présenté en section 5.2.1 (cf. Fig. 5-21).

L'éditeur d'assistance fournit également une interface pour la définition d'attentes d'éléments élémentaires, de consultations élémentaires et d'actions élémentaires d'assistance, ainsi que d'actions d'assistance qui instancient l'un des patrons d'actions d'assistance présentés en section 5.2 : les pas à pas automatisés ou guidés, les présentations guidées et les actions d'agents animés.

Chapitre 9. Exécution de l'assistance avec SEPIA

Objectif du chapitre

Proposer un outil informatique rendant possible l'exécution automatique de l'assistance spécifiée avec aLDEAS.

Points clés

Nous présentons le **moteur générique d'assistance** de SEPIA, qui met en œuvre **aMEAS**, le modèle d'exécution d'un système d'assistance aLDEAS. Pour surveiller l'application-cible, le moteur fait appel à un ensemble d'**épi-détecteurs** qui mettent en œuvre le processus de surveillance d'une application existante. Pour effectuer les consultations de l'état de l'application-cible d'aLDEAS, le moteur fait appel à un ensemble d'**épi-inspecteurs**, capables d'accéder aux propriétés des composants de l'interface de l'application-cible. Enfin, pour fournir de l'assistance à l'utilisateur final, le moteur d'assistance fait appel à un ensemble d'**épi-assistants**, capables de réaliser de manière épiphyte les actions d'assistance d'aLDEAS sur l'application-cible.

Contributions

- Moteur générique de SEPIA exploitant le modèle aMEAS
- Épi-détecteurs de SEPIA
- Épi-inspecteurs de SEPIA
- Épi-assistants de SEPIA

Validation

- ✓ Exécution conforme à leur définition de systèmes d'assistance aLDEAS variés dans différentes applications-cibles
- ✓ Capacité à fournir une assistance efficace pour réaliser une tâche donnée et bien acceptée par les utilisateurs finaux

Publications liées à ce chapitre

[Ginon et al., 2013c]

> Ginon B., Jean-Daubias S. et Champin P.-A., Mise en place d'un système d'assistance personnalisée dans une application existante, IC, Lille, France, 2013c.

[Ginon et al., 2013d]

> Ginon B., Jean-Daubias S. et Champin P.-A., Adjonction de systèmes d'assistance personnalisée à des EIAH existants, poster pour EIAH, Toulouse, 2013d.

[Ginon et al., 2014a]

> Ginon B., Jean-Daubias S., Champin P.-A. et Lefevre M., aLDEAS : un langage de définition de systèmes d'assistance épiphytes, IC, Clermont-Ferrand, France, pp. 137-148, 2014a.

[Ginon et al., 2014b]

> Ginon B., Thai V. L., jean-Daubias S., Lefevre M. et Champin P.-A., Adding epiphytic assistance systems in learning applications using the SEPIA system, Ec-Tel, Graz, Austria, 2014b.

Les systèmes d'assistance spécifiés à l'aide de l'éditeur de SEPIA peuvent être exécutés de manière automatique par le moteur générique d'assistance de SEPIA, présenté dans la section 9.4. Pour rendre une telle exécution de l'assistance possible sur une très large variété d'applications-cibles, le moteur d'assistance de SEPIA fait appel à un ensemble d'outils épiphytes : les épi-détecteurs (cf. Section 9.1), capables de surveiller des applications-cibles, les épi-inspecteurs (cf. Section 9.2), capables d'inspecter l'état des applications-cibles et les épi-assistants (cf. Section 9.3), capables de fournir de l'assistance aux utilisateurs finaux sous des formes très variées.

9.1. Épi-détecteurs de SEPIA

Pour permettre de greffer de manière épiphyte un système d'assistance sur une application-cible et permettre une exécution contextualisée de l'assistance, il est nécessaire d'observer les événements survenant à l'interface de cette application. Pour cette raison, nous avons développé un ensemble d'épi-détecteurs, capables de surveiller des applications-cibles. Ces épi-détecteurs mettent en œuvre le processus de surveillance d'une application informatique, présenté dans l'introduction de la Partie 2. Nos épi-détecteurs ne sont pas spécifiques à une application, mais à une catégorie d'applications. Dans la suite de cette section, nous distinguons le cas des applications bureau de celui des applications web.

9.1.1. Surveillance des applications bureau

Pour les applications bureau, la technique de surveillance que nous proposons s'appuie sur l'exploitation de bibliothèques d'accessibilité (cf. Section 4.2). Ces bibliothèques permettent d'accéder à la hiérarchie des composants des applications ouvertes à ces bibliothèques et de s'abonner à des événements pouvant survenir sur ces composants, comme les clics et les déplacements de la souris sur ces composants.

Comme nous l'avons vu en Section 4.2, il existe plusieurs bibliothèques d'accessibilité : les applications informatiques ne sont pas toutes ouvertes à l$a même bibliothèque d'accessibilité (cf. Fig. 4-2). Pour permettre la prise en charge par SEPIA d'une large variété d'applications, nous avons développé plusieurs épi-détecteurs, qui exploitent chacun une bibliothèque d'accessibilité différente.

Le **détecteur Windows** permet la surveillance des exécutables Windows ; il s'appuie sur l'exploitation d'UIAutomation et est développé en C#. Ce détecteur permet par exemple la surveillance de la suite Microsoft Office (avec des logiciels tels que Word, Excel et PowerPoint) et des accessoires Windows (tels que la calculatrice et les jeux), de l'IDE (Integrated Development Environnement) CodeBlocks, ainsi que du logiciel de retouche d'images PhotoScape. Le **détecteur Java** permet la surveillance des applications Java sous Windows; il s'appuie sur l'exploitation de JavaAccessibility et est également développé en C#. Ainsi, ce détecteur permet par exemple la surveillance de l'IDE NetBeans et du logiciel éducatif de mathématiques Géogébra. Notons que l'utilisation de la bibliothèque JavaAccessibility depuis l'extérieur de la machine virtuelle Java n'est possible que via une bibliothèque native, qui n'est à notre connaissance disponible que pour Windows. Pour

permettre la surveillance d'applications Java sous Mac OS et Linux, il serait donc nécessaire de développer pour ces systèmes un équivalent de la bibliothèque native (une *DLL*) utilisée sous Windows. Le **détecteur GTK/Qt**[12] permet la surveillance des applications QTK et Qt sous Linux ; il s'appuie sur l'exploitation de ATK-SPI et est développé en Python. Ainsi, ce détecteur permet par exemple, la surveillance du logiciel de dessin vectoriel Inkscape et du logiciel de retouche d'images Gimp. Cet épi-détecteur utilise un service de communication inter-applications (un *D-Bus*) conçu spécifiquement pour Linux, qui peut cependant être installé sous Windows ou Mac OS afin de le rendre portable.

La Fig. 9-1 présente les types d'événements détectés par les trois bibliothèques d'accessibilité que nous utilisons : UIAutomation, JavaAccessibility et QtAccessibility. Concernant les événements liés à la souris, notons à titre d'exemple que ces trois bibliothèques sont capables de détecter un clic gauche sur un composant appartenant à une application accessible à cette bibliothèque. Concernant les événements liés au focus, chacune de ces bibliothèques est capable de détecter qu'un composant a gagné le focus. En revanche, seule UIAutomation est capable de détecter l'ouverture ou la fermeture d'une infobulle. Bien que certains types d'événements ne soient pas détectables par certaines bibliothèques d'accessibilité, il est souvent possible de déduire un événement grâce à la détection d'autres événements. Par exemple, avec JavaAccessibility, il n'est pas possible de détecter directement les événements de type ouverture de menu, cependant, il est possible de détecter le clic qui entraîne l'ouverture de ce menu, ou la saisie du raccourci éventuellement associé à l'ouverture du menu.

Ces épi-détecteurs sont lancés par le moteur d'assistance au lancement de l'assistance par un utilisateur final qui souhaite recevoir de l'aide. Ils prennent en paramètre le fichier de description de l'interface de l'application-cible. Ce fichier associe un identifiant à chaque composant de l'application-cible, ce qui permet de filtrer les événements notifiés par l'épi-détecteur. En effet, lorsqu'ils détectent un événement, les épi-détecteurs cherchent à identifier le composant qui est la source de cet événement en utilisant le fichier de description, par exemple le bouton sur lequel un clic a eu lieu. S'ils parviennent à trouver un identifiant correspondant au composant qui est la source de l'événement, les épi-détecteurs transmettent au moteur d'assistance la notification de cet événement, sous la forme « type d'événement, identifiant du composant source ».

[12] Ce détecteur a été développé par Brice Buffa, étudiant de Master 1 au département informatique de l'Université Lyon 1, dans le cadre d'un stage, puis complété par Léa Bonneville, dans le cadre d'un stage de 1ere année d'école d'ingénieure à l'INSA de Lyon.

Événements détectés		UIAutomation	JavaAccessibility	ATK-SPI
Action	Action réalisée	✓	✓	✓
Souris	Clic gauche	✓	✓	✓
	Clic droit	✓	✓	✓
	Passage sur un composant	✓	✓	
	Déplacement	✓	✓	✓
	Bouton appuyé	✓	✓	✓
	Bouton relâché	✓	✓	✓
	Glisser	✓	✓	
	Sortie d'un composant	✓	✓	
Saisie	Touche tapée	✓	✓	✓
	Touche relâchée	✓	✓	✓
	Touche pressée	✓	✓	✓
Focus	Gain de focus	✓	✓	✓
	Perte de focus	✓	✓	✓
	Changement de focus	✓	✓	✓
Menu	Sélection dans un menu	✓	✓	✓
	Désélection dans un menu	✓	✓	✓
	Ouverture de menu		✓	✓
	Fermeture de menu		✓	✓
Infobulle	Ouverture d'infobulle	✓		
	Fermeture d'infobulle	✓		
Fenêtre	Ouverture de fenêtre	✓	✓	✓
	Fermeture de fenêtre	✓	✓	✓
Texte	Modification de texte	✓		✓
	Sélection de texte	✓		✓
Élément	Sélection d'élément	✓	✓	
	Ajout à la sélection	✓		
	Suppression de la sélection	✓		
Propriété	Modification de propriété	✓		✓

Fig. 9-1. Types d'événements détectés par bibliothèque d'accessibilité

9.1.2. Surveillance des applications web

Pour permettre la surveillance des applications web, nous avons développé un script utilisateur en JavaScript. L'utilisation de ce script peut se faire de différentes façons. Tout d'abord, il peut être directement intégré dans l'application web par le développeur de celle-ci. Néanmoins, nous souhaitons conserver une démarche épiphyte, ce qui implique la contrainte de ne pas modifier le code source de l'application-cible. Une autre solution consiste alors à utiliser une extension de navigateur gérant ce script, telle que Grease Monkey [Grease_monkey, 2014] pour Mozilla Firefox [Mozilla, 2014] ou Tamper Monkey [Tamper_monkey, 2014] pour Google Chrome [Google, 2014]. Pour simplifier notamment l'installation de l'extension de navigateur et l'activation du script de collecte, nous avons choisi d'utiliser l'extension de navigateur TraceMe[13], dans laquelle nous avons inséré notre script de collecte.

Lorsque le script de collecte est actif, il permet la détection dans une page web du navigateur de tout événement de type clic, passage de la souris sur un composant, saisie de texte et changement de focus. Lors de la détection d'un tel événement, le script transmet au moteur d'assistance la notification de cet événement, sous la forme « type d'événement, XPath du composant source ». Contrairement aux épi-détecteurs implémentant la surveillance des applications bureau, ce script de collecte pour les applications web n'est pas lancé par le moteur générique d'assistance au lancement de l'assistance, mais il doit être activé par l'utilisateur final dans le navigateur qu'il utilise.

9.1.3. Conclusion

L'ensemble des épi-détecteurs que nous proposons permettent au système SEPIA de prendre en charge de très nombreuses applications-cibles, dans lesquelles il sera possible de mettre en place de l'assistance épiphyte via les outils de SEPIA.

Pour permettre la prise en charge des applications actuellement non compatibles avec nos épi-détecteurs, et ainsi augmenter encore la portée de SEPIA, il est nécessaire de développer de nouveaux épi-détecteurs.

C'est le cas en particulier pour les applications bureau sous Mac OS, ouvertes à la bibliothèque Apple API Accessibility. Cet épi-détecteur aura un fonctionnement très similaire à celui des épi-détecteurs Windows, Java et GTK/Qt.

En revanche, pour certaines applications, il est nécessaire de proposer une méthode différente de celle utilisée actuellement dans nos épi-détecteurs. Par exemple, les applications Flash ne sont pas ouvertes à notre script de collecte. De même, l'extension de navigateur TraceMe n'est pas compatible avec le navigateur Internet Explorer.

[13] L'extension de navigateur TraceMe a été développée par Fatma Derbel, ingénieure d'étude dans le cadre du projet COAT

9.2. Épi-inspecteurs de SEPIA

Pour permettre les consultations de l'état de l'application-cible, et ainsi permettre une contextualisation de l'assistance, nous avons développé un ensemble d'épi-inspecteurs, capables d'inspecter à la demande du moteur d'assistance les propriétés d'un composant de l'interface de l'application-cible. Les épi-inspecteurs sont également appelés lors de l'exécution d'une action élémentaire d'assistance impliquant la position d'un composant de l'interface de l'application-cible, comme celles de type *mise en valeur* ou *masquage*.

Les épi-inspecteurs exploitent les mêmes technologies que les épi-détecteurs (cf. Section 9.1), c'est pourquoi nous distinguons de nouveau le cas des applications bureau de celui des applications web.

9.2.1. Inspection des applications bureau

En plus de permettre la surveillance des événements survenant dans des applications, les bibliothèques d'accessibilité permettent d'accéder à un composant d'une application ouverte à cette bibliothèque, et d'inspecter ses propriétés.

La Fig. 9-2 présente les propriétés auxquelles les différentes bibliothèques d'accessibilité accèdent, selon le type de composant inspecté. Notons par exemple que chaque bibliothèque d'accessibilité permet d'accéder au nom et au rôle d'un composant, ainsi qu'à son champ de description, à condition toutefois que celui-ci ait été rempli par le développeur de l'application. En revanche, seule UIAutomation permet de connaître le pourcentage de défilement d'une barre de défilement et la valeur maximale autorisée d'une zone de saisie par incrément (le *spinner* dont la valeur est modifiable à l'aide de boutons + et -).

Composant	Propriété	Type	UIAuto-mation	Java Accessibility	ATK SPI
Tout composant	Actif	Booléen	✓	✓	
	Focusable	Booléen	✓	✓	
	Focus	Booléen	✓		
	Bounding rectangle	Texte	✓	✓	✓
	Nom	Texte	✓	✓	✓
	Rôle	Texte	✓	✓	✓
	Description	Texte	✓	✓	✓
	Visible	Booléen	✓	✓	✓
	Action accessible	Booléen	✓	✓	✓
	Nombre d'enfants	Entier	✓	✓	✓
	Indice depuis le parent	Entier	✓	✓	✓
Item **Bouton radio**	Sélectionné	Booléen	✓	✓	✓
	Sélectionnable	Booléen		✓	✓
Liste Calendriers **Zone de saisie**	Éditable	Booléen	✓	✓	✓
	Valeur	Texte	✓	✓	✓
Item de table	Colonne	Entier	✓		✓
	Portée en colonnes	Entier	✓		✓
	Ligne	Entier	✓		✓
	Portée en lignes	Entier	✓		✓
Table	Nombre de colonnes	Entier	✓		✓
	Nombre de lignes	Entier	✓		✓
Slider **Spinner**	Éditable	Booléen	✓	✓	✓
	Valeur maximale	Entier	✓		
	Valeur minimale	Entier	✓		
	Value	Entier	✓	✓	✓
Composant avec barre de défilement	Scrollable horizontalement	Booléen	✓		✓
	Scrollable verticalement	Booléen	✓		✓
	Défilement horizontal (%)	Entier	✓		
	Défilement vertical (%)	Entier	✓		
	Longueur visible	Entier	✓		
	Hauteur visible	Entier	✓		
	Non scrollable	Booléen	✓		✓
Liste **Groupe de boutons**	Sélection multiple permise	Booléen	✓	✓	✓
	Sélection requise	Booléen	✓		✓
Fenêtre **Item de table** **Canevas**	Déplaçable	Booléen	✓		✓
	Redimensionnable	Booléen	✓	✓	✓
	Rotatif	Booléen	✓		
Fenêtre	Maximisable	Booléen	✓		✓
	Minimisable	Booléen	✓		✓
	Modale	Booléen	✓		✓
	Premier plan	Booléen	✓		✓

Fig. 9-2. Propriétés des composants visibles par des bibliothèques d'accessibilité

Prenons l'exemple de la calculatrice Windows (cf. Fig. 9-3), et inspectons certaines propriétés pour quelques-uns de ses composants. La *Valeur* de la zone de saisie contenant le résultat de la formule saisie est « 0 » (cf. Ⓐ Fig. 9-3). La propriété *Sélectionné* du bouton radio Degrés est « true » (cf. Ⓑ Fig. 9-3). La propriété *Actif* du bouton vide est false (cf. © Fig. 9-3). La propriété *Bounding Rectangle* du bouton + est « (31 ; 31 ; 896 ; 451) » (cf. Ⓓ Fig. 9-3). Cette dernière propriété, qui correspond aux coordonnées à l'écran du rectangle qui englobe le composant, est indispensable pour réaliser des mises en valeur de ce composant.

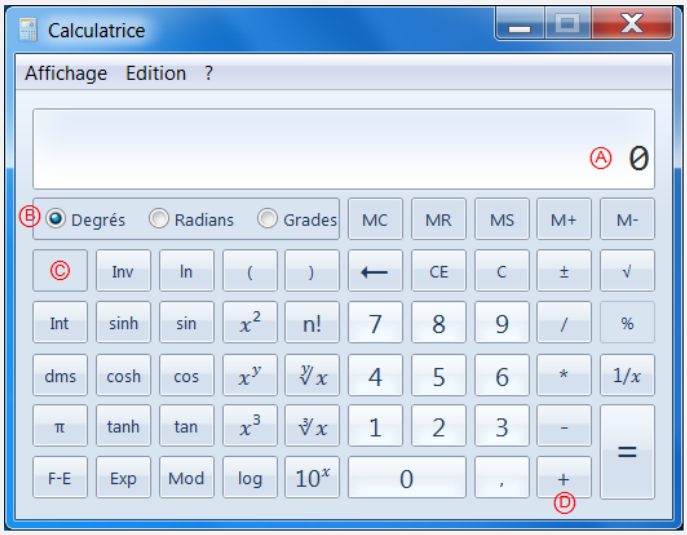


Fig. 9-3. Calculatrice Windows en affichage scientifique

Les épi-inspecteurs sont appelés par le moteur d'assistance lorsqu'il a besoin d'une information sur l'état d'un composant de l'application-cible. Les épi-inspecteurs prennent en paramètres le fichier de description de l'interface de l'application-cible, l'identifiant du composant et la propriété à inspecter.

9.2.2. Inspection des applications web

Pour inspecter les applications web, nous avons développé une fonction d'inspection que nous avons ajoutée dans le script de collecte. Le moteur générique communique avec cette fonction via un web socket lorsqu'il a besoin d'une information sur l'état d'un composant d'une application web. Cette fonction prend en paramètre l'URL de la page web, le XPath du composant et la propriété à inspecter. Toutes les propriétés d'un composant HTML peuvent être inspectées, par exemple *valeur* textuelle ou numérique, *actif*, *éditable*, *bounding rectangle*, *sélectionné*.

Prenons l'exemple du site de la CAF[14] (cf. Fig. 9-4), et inspectons certaines propriétés pour quelques-uns de ses composants. La *valeur* de la zone de saisie Code postal est « 69100 », et

[14] http://wwwd.caf.fr/wps/portal/caffr/login

sa propriété *éditable* vaut true (cf. Ⓐ Fig. 9-4). La propriété *sélectionné* de la case à cocher Allocataire de la caisse maritime vaut false (cf. Ⓑ Fig. 9-4).

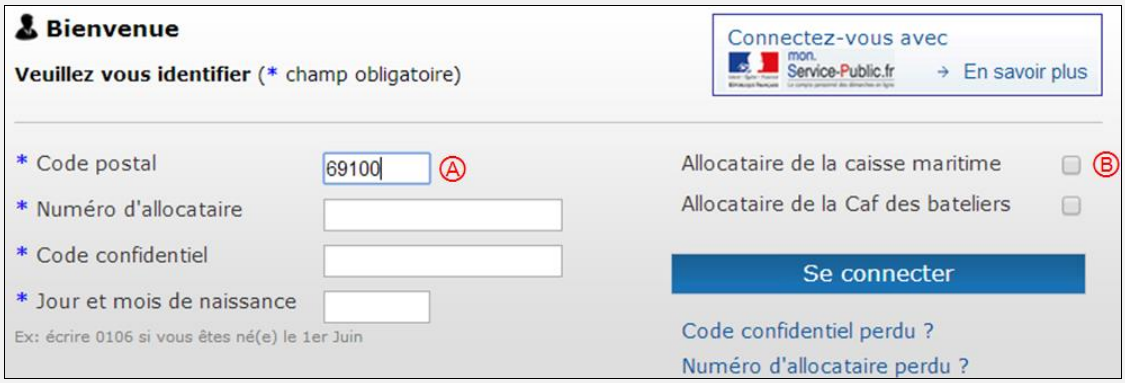


Fig. 9-4. Site de la Caisse des Allocations Familiales

9.2.3. Conclusion

Les épi-inspecteurs que nous avons développés permettent de contextualiser l'assistance en permettant de réaliser les consultations de l'état de l'application-cible proposées par le langage aLDEAS (cf. Section 5.1.2). Ils contribuent ainsi à rendre l'assistance fournie par le système SEPIA intégrée à l'application-cible aux yeux de l'utilisateur final. De plus, ils rendent possibles les actions d'assistance d'aLDEAS intégrées à l'application-cible (cf. Section 5.1.3), comme les mises en valeur de composants, qui nécessitent une inspection de leurs coordonnées.

9.3. Épi-assistants de SEPIA

Nous avons également développé un ensemble d'épi-assistants capables de réaliser dans une application-cible les actions élémentaires d'assistance proposées par aLDEAS (cf. section 5.1.3) et définies à l'aide de notre éditeur. La Fig. 9-5 présente les actions élémentaires d'assistance pouvant être réalisées par nos épi-assistants en fonction du type d'application-cible. Pour l'instant, nos épi-assistants sont capables d'agir principalement sur les applications Windows natives ou Java, ainsi que sur les applications web hors Flash quel que soit le système d'exploitation, et sur les applications GKT ou Qt sous linux.

Nos épi-assistants ne sont pas spécifiques à une application-cible, mais à une catégorie d'applications-cibles : ils peuvent être appelés pour réaliser une action dans n'importe quelle application-cible dont le type est compatible avec l'épi-assistant. Ils sont développés en langage Java, à l'exception des épi-assistants d'action automatisée, qui s'appuient sur la même technologie que les épi-détecteurs qui surveillent les applications-cibles selon leur type. Ainsi, les épi-assistants d'action automatisée pour les applications de type bureau sont développés en C# et celui pour les applications web en JavaScript.

		Actions élémentaires d'assistance					
		Action automatisée	**Mise en valeur**	**Masquage**	**Message**	**Agent animé**	**Accès ressource**
Windows	Exécutables	✓	✓	✓	✓	✓	✓
	Java	✓	✓	✓	✓	✓	✓
	GTK/Qt				✓	✓	✓
	web (hors Flash)	✓	✓	✓	✓	✓	✓
Linux	Java				✓		✓
	GTK/Qt		✓	✓	✓		✓
	web (hors Flash)	✓	✓	✓	✓		✓
Mac OS	Java				✓		✓
	GTK/Qt				✓		✓
	web (hors Flash)	✓	✓	✓	✓		✓

Fig. 9-5 – Actions élémentaires d'assistance réalisables selon le type d'application-cible

Dans la suite de cette section, nous présentons, pour chaque type d'action élémentaire d'assistance proposé par aLDEAS, comment nos épi-assistants permettent de les réaliser dans l'application-cible.

9.3.1. Action automatisée

Nous appelons *action automatisée* une action d'assistance qui permet d'agir dans l'application à la place de l'utilisateur final. Une action automatisée peut par exemple pré-remplir pour l'utilisateur une zone de saisie, ou simuler un clic sur un bouton. Pour réaliser une action d'assistance de type action automatisée, nous proposons un ensemble d'épi-assistants qui s'appuient sur la même technologie que les épi-détecteurs (cf. Section 9.1). L'épi-assistant adapté pour réaliser une action automatisée est donc choisi par le moteur générique en fonction du type de l'application-cible. Ainsi pour réaliser une action automatisée dans une application Java, c'est l'épi-assistant de création automatisée exploitant JavaAccessibility qui est appelé, alors que dans le cas d'une application web c'est le script utilisateur d'action automatisée qui est appelé.

La Fig. 9-6 présente les actions automatisées possibles selon le type de l'application-cible. Les actions automatisées ont pour but de simuler une action de l'utilisateur à l'interface de l'application-cible afin de réaliser tout ou une partie de sa tâche à sa place. Pour cette raison, seules les actions que l'utilisateur a la possibilité de faire lui-même peuvent être réalisées par le système d'assistance. Ainsi, une action automatisée de clic sur un bouton inactif n'aura aucune conséquence, comme cela serait le cas si l'utilisateur cliquait lui-même sur ce bouton. De même, une action automatisée ne peut pas donner le focus à un composant qui ne peut pas le recevoir, ni modifier la valeur d'une zone de saisie non éditable.

Une action automatisée prend en paramètre un composant de l'interface de l'application-cible, ainsi qu'un type d'action automatisée qui doit être compatible avec le composant concerné : cliquer, donner le focus, modifier l'état ou modifier la valeur.

En complément, une action automatisée de **modification de l'état** d'un composant peut prendre en paramètre l'état souhaité pour le composant. Par exemple, une action automatisée peut permettre de décocher une case à cocher, ou de sélectionner un item d'une liste déroulante. Sans paramètre *valeur*, l'action automatisée de modification de l'état inverse l'état courant du composant, par exemple pour sélectionner un bouton radio s'il ne l'est pas ou pour le désélectionner dans le cas contraire.

Enfin, une action automatisée de **modification de la valeur** d'un composant prend en paramètre la valeur à attribuer au composant. Cette valeur doit respecter le modèle des valeurs possibles du composant, dans le cas contraire, l'action n'aura aucun effet. Par exemple, il n'est pas possible d'affecter une valeur textuelle à une barre de défilement, et la valeur numérique affectée doit appartenir à l'intervalle [0, 100] ou à un sous-ensemble de cet intervalle, par exemple {10, 25, 50, 75, 100} selon les barres de défilements.

Action	Type de composant	Exécutables Windows	Applications Java	Applications GTK / Qt	Applications web
Cliquer	Tout composant	✔	✔	✔	✔
Donner le focus	Tout composant	✔	✔	✔	
Modifier l'état	Bouton radio, case à cocher, item, arbre	✔	✔	✔	✔
Modifier la valeur	Zone de saisie, spinner, slider, barre de progression, barre de défilement	✔	✔	✔	✔

Fig. 9-6 – Actions automatisées possibles selon le type de l'application-cible

La Fig. 9-7 présente la définition en XML de trois actions automatisées pour le site web de la CAF (cf. Fig. 4-6, page 64). L'action A5 concerne le composant 85 (la case à cocher « allocataire de la caisse maritime ») et permet de lui affecter l'état coché. L'action A6 concerne le composant 82 (la zone de saisie « code postal ») et permet de remplir ce champ avec le code postal « 13001 ». Enfin, l'action A7 permet de simuler un clic sur le composant 87 (le bouton « se connecter »), afin de valider le formulaire.

```
<action id="A5" type="action sur l'interface" >
  <evenement idComp="85" propriete="etat" valeur="On"/>
</action>
<action id="A6" type="action sur l'interface" >
  <evenement idComp="82" propriete="valeur" valeur="69100"/>
</action>
<action id="A7" type="action sur l'interface" >
  <evenement idComp="87" propriete="clic" />
</action>
```

Fig. 9-7 – Exemple de trois actions automatisées pour le site web de la CAF

9.3.2. Mise en valeur

Une mise en valeur est une modification de l'interface qui permet d'attirer l'attention de l'utilisateur sur un composant donné. Pour réaliser dans une application-cible une action d'assistance de type mise en valeur, nous proposons deux épi-assistants. Le premier permet de mettre en valeur un composant en l'entourant, en le colorant ou en affichant un symbole à côté de lui. Il exploite des fenêtres transparentes placées sur l'application-cible et dans lesquelles sont dessinées des formes géométriques ou dans lesquelles une image est insérée. Le second épi-assistant permet de mettre en valeur un composant en affichant un agent animé qui le désigne. Cet épi-assistant exploite les agents animés MsAgents [MsAgents, 2014] ou Double Agent [DoubleAgent, 2014], qui prennent la forme de 39 personnages variés. Cet épi-assistant est disponible uniquement sous Windows, pour des application-cibles de type exécutable, Java ou web. En effet, les MsAgents et les double Agents sont des composants activeX, propres à Windows.

Les actions de mise en valeur d'un composant prennent en paramètre l'identifiant du composant à mettre en valeur. Au moment de l'exécution d'une telle action d'assistance, le moteur fait appel à un épi-inspecteur (cf. Section 9.2) afin de compléter la définition de cette action en précisant les informations nécessaires à une mise en valeur : les coordonnées (x, y) du coin supérieur gauche du rectangle englobant le composant (appelé bounding rectangle, cf. Fig. 9-8), ainsi que la largeur et la hauteur de ce rectangle.

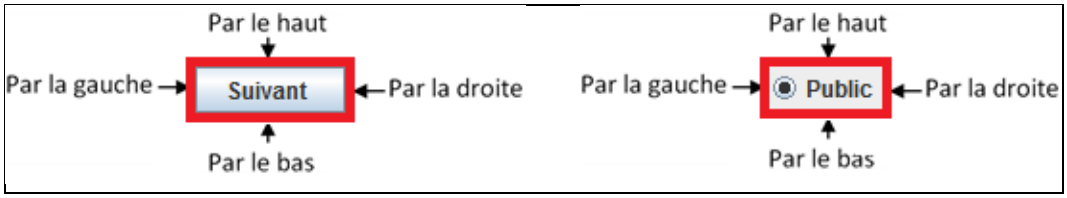


Fig. 9-8 – Deux illustrations du bounding rectangle et des quatre directions de mise en valeur possibles pour un bouton (à gauche) et un bouton radio (à droite)

Une action de mise en valeur est également associée à un paramètre optionnel, le *type* de mise en valeur (entourer, colorer, afficher un symbole, afficher un agent animé), et à un ensemble de paramètres optionnels selon le type de mise en valeur choisi. Le paramètre optionnel *direction* (par la gauche, par la droite, par le haut, par le bas, cf. Fig. 9-8) peut être spécifié pour les mises en valeur impliquant l'affichage d'un symbole ou d'un agent animé. En fonction des coordonnées du rectangle englobant et de la direction le cas échéant, les épi-assistants de mise en valeur calculent la position de l'agent animé ou de la fenêtre transparente qui contiendra la forme géométrique ou l'image. Dans le cas d'une mise en valeur par un agent animé, l'épi-assistant détermine également quel geste l'agent doit effectuer en fonction de la direction (cf. Fig. 9-11), et ajuste la position de l'agent en fonction du personnage choisi (les personnages n'étant pas tous de même taille).

À titre d'exemple, la Fig. 9-9 présente l'exécution simultanée de trois mises en valeur dans la Calculatrice Windows et la Fig. 9-10 présente la définition en XML de ces trois actions d'assistance. Le bouton + est coloré en vert (cf. Fig. 9-9 et Ⓐ Fig. 9-10), le bouton 6 est entouré en rose (cf. Fig. 9-9 et Ⓑ Fig. 9-10) et le bouton 8 est mis en valeur par un symbole (une flèche jaune affichée par le haut du composant, cf. Fig. 9-9 et © Fig. 9-10). La Fig. 9-11 présente quant à elle l'exécution d'une action d'assistance de type mise en valeur par un agent animé (cf. Fig. 9-11 et Ⓓ Fig. 9-10) sur le bouton 9 de la calculatrice Windows, en faisant varier le paramètre direction.

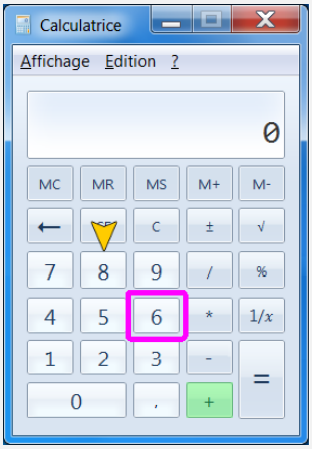


Fig. 9-9 – Exemples de trois mises en valeur simultanées dans la Calculatrice Windows

```
<action id="A0" type="mise en valeur"> Ⓐ
  <composant id="42" type="colorer" transparence="70"
             couleur="java.awt.Color[r=255,g=0,b=255]"/>
  <coordonnees x="1550" y="571" largeur="43" hauteur="33"/>
</action>
<action id="A1" type="mise en valeur"> Ⓑ
  <composant id="39" type="entourer"
             couleur="java.awt.Color[r=0,g=255,b=0]"
             arrondi="5" eloignement="5" epaisseur="5" />
  <coordonnees x="1387" y="551" largeur="43" hauteur="33" />
</action>
<action id="A2" type="mise en valeur"> Ⓒ
  <composant id="47" type="symbole" direction="par le haut"
             symbole="flecheJaune.png" />
  <coordonnees x="1452" y="451" largeur="43" hauteur="33" />
</action>
<action id="A3" type="mise en valeur"> Ⓓ
  <composant id="50" type="agent animé" personnage="Robby"/>
  <coordonnees x="1501" y="451" largeur="43" hauteur="33"/>
</action>
```

Fig. 9-10 – Définition de quatre mises en valeur pour la Calculatrice Windows

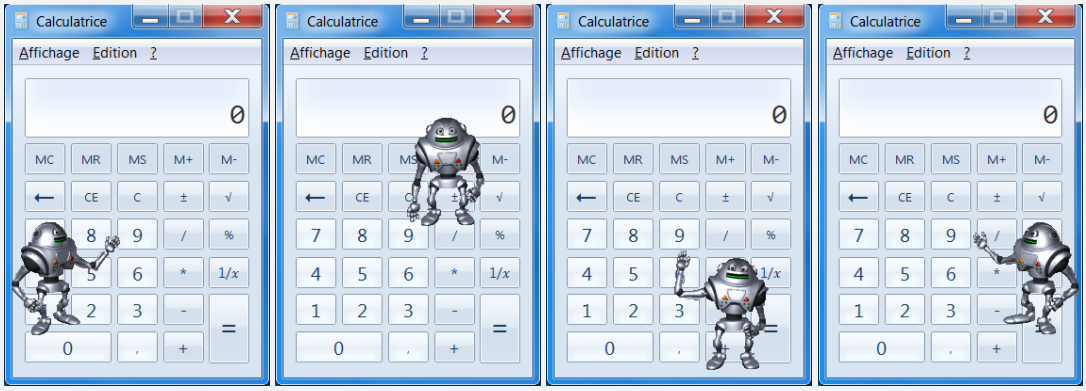


Fig. 9-11 – Exemple de quatre mises en valeur par l'agent animé «Robby » dans la Calculatrice Windows

Le paramètre optionnel *rafraichissement*, commun à tous les types de mises en valeur, indique au moteur d'assistance à quelle fréquence il doit vérifier si les coordonnées du composant mis en valeur ont été modifiées, par exemple si l'utilisateur a déplacé la fenêtre qui contient le composant ou si celui-ci n'est plus visible suite à une action de l'utilisateur. Lorsque ces coordonnées changent, le moteur d'assistance envoie l'information à l'épi-assistant concerné pour qu'il mette à jour son affichage.

9.3.3. Masquage

Une action d'assistance de type *masquage* peut être réalisée par l'épi-assistant qui effectue les mises en valeur de type *colorer*. En effet, un masquage consiste à afficher un rectangle coloré sur le composant à masquer : ce rectangle est translucide pour un masquage de type *estomper* et opaque pour un masquage de type *occulter*. Cependant, contrairement

à une mise en valeur de type *colorer*, le masquage empêche l'utilisateur d'agir sur le composant masqué. Pour qu'un composant soit totalement invisible aux yeux de l'utilisateur final de l'application-cible, le paramètre couleur de l'action de masquage doit être la couleur du composant père de celui à masquer. Pour cela, les épi-inspecteurs peuvent être utilisés par le moteur d'assistance pour récupérer la couleur d'un composant de l'interface d'une application-cible lorsque cela est nécessaire.

À titre d'exemple, la Fig. 9-12 présente l'exécution simultanée de cinq masquages dans la Calculatrice Windows et la Fig. 9-13 présente la définition en XML de ces cinq actions d'assistance. Ces masquages sont de type *occulter*, ce qui signifie que les composants sont totalement masqués, au contraire des masquages de type *estomper*, avec lesquels les composants sont encore visibles. De plus, l'attribut *couleur* de ces masquages a pour valeur *père*, ce qui signifie que le rectangle qui est affiché au-dessus du composant à masquer doit être de la couleur du père de ce composant, afin qu'il devienne totalement invisible aux yeux de l'utilisateur final. Nous constatons en effet que les cinq boutons masqués semblent avoir disparu de la calculatrice (cf. Ⓐ Fig. 9-12).

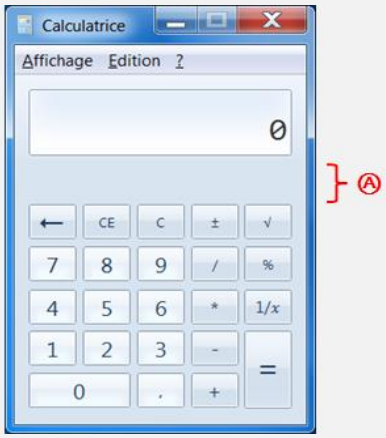


Fig. 9-12 – Exemples de cinq masquages exécutés dans la Calculatrice Windows

```
<action id="A8" type="masquage">
  <composant id="30" type="occulter" couleur="pere" />
</action>
<action id="A9" type="masquage">
  <composant id="31" type="occulter" couleur="pere" />
</action>
<action id="A10" type="masquage">
  <composant id="34" type="occulter" couleur="pere" />
</action>
<action id="A11" type="masquage">
  <composant id="40" type="occulter" couleur="pere" />
</action>
<action id="A12" type="masquage">
  <composant id="45" type="occulter" couleur="pere" />
</action>
```

Fig. 9-13 – Définition de cinq masquages pour la Calculatrice Windows

9.3.4. Message

Un message d'assistance permet de transmettre à l'utilisateur une information, un conseil, une instruction etc. Pour réaliser une action d'assistance de type message dans une application-cible, nous proposons deux épi-assistants, afin de permettre au concepteur de l'assistance de définir des messages sous des formes très variées et finement personnalisables. Le premier permet d'afficher des messages dans une fenêtre pop-up et/ou de lire ce message par synthèse vocale. Le second, disponible uniquement sous Windows, fait appel à un agent animé pour transmettre un message de manière textuelle et/ou orale.

Une action de type message prend en paramètre un contenu (le message à transmettre), et éventuellement un mode de transmission : textuel, vocal ou les deux. Dans le cas d'une action d'assistance de type message textuel et vocal, le contenu transmis oralement à l'utilisateur peut différer de celui transmis textuellement, à la demande du concepteur de l'assistance. Par exemple, une synthèse vocale peut résumer le message affiché dans une fenêtre pop-up (cf. Ⓐ et Ⓑ Fig. 9-15), ou au contraire apporter un complément.

Un message écrit peut être affiché dans une fenêtre pop-up ou dans une bulle de dialogue associée à un agent animé. Il peut être associé à des paramètres optionnels de durée d'affichage et de position à l'écran, ainsi qu'à différents paramètres optionnels selon le mode de transmission choisi : le personnage choisi pour un message transmis par un agent animé, la mise en forme et le titre pour un message affiché par une fenêtre pop-up. Un message oral peut être lu par synthèse vocale ou par un agent animé. Un message oral lu par un agent animé a comme paramètres optionnels le choix du personnage et sa position.

La position d'une fenêtre pop-up ou d'un agent animé peut être relative à l'écran, au curseur, à un composant ou à des coordonnées absolues. Une position relative à l'écran prend en paramètres une position horizontale et une position verticale, par exemple centré horizontalement et en haut de l'écran verticalement. Un message dont la position est relative au curseur signifie que l'agent animé ou la fenêtre pop-up sera affiché à l'endroit où se situe la souris au moment de l'exécution de l'action d'assistance. Une position relative à des coordonnées absolues prend en paramètre un couple d'entiers (x, y) qui définit cette position. Une position relative à un composant prend en paramètre une direction (à gauche, à droite, en haut, en bas) : une position relative à un composant permet d'afficher un message à côté d'un composant donné, qui peut par exemple être concerné par le message ; le calcul de la position du message relativement à un composant de l'interface de l'application-cible est similaire au calcul de la position d'une mise en valeur d'un composant (cf. section 9.3.2).

À titre d'exemple, la Fig. 9-14 présente l'exécution d'une action d'assistance de type message, dont la définition en XML est donnée en Fig. 9-15, pour deux utilisateurs finaux de PhotoScape : Lucas à gauche et Julie à droite. Il s'agit d'un message écrit et oral dont le mode de transmission (fenêtre pop-up et synthèse vocale ou agent animé) n'est pas spécifié par le concepteur. Le choix du mode de transmission, et donc de l'épi-assistant qui réalisera cette action, est fait par le moteur générique au moment de l'exécution de l'assistance dans l'application-cible, en fonction des épi-assistants disponibles sur le poste de travail de l'utilisateur final et en fonction de ses préférences. Notons que le contenu de ce message est

également personnalisé en fonction du prénom de l'utilisateur final : la balise %prenom% est remplacée par le moteur d'assistance au moment de l'exécution.

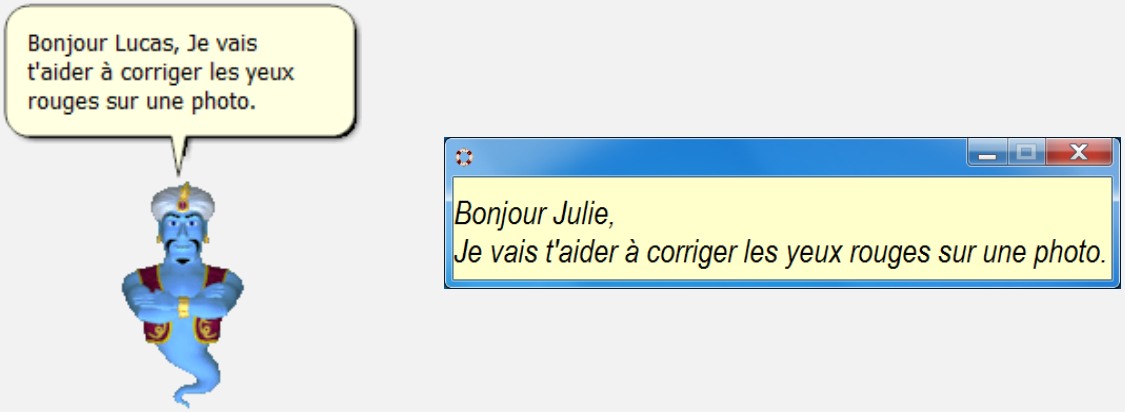


Fig. 9-14 – Deux exemples d'exécution d'une action d'assistance de type message

```
<action id="A9" type="message" sous-type="textuel_vocal" >
  <textuel>
    <texte>Bonjour %prenom%,
          Je vais t'aider à corriger les yeux rouges sur une photo.
    </texte>
  </textuel>
  <vocal different="vrai">
    <texte>Bonjour %prenom%</texte>
  </vocal>
</action>
```

Fig. 9-15 – Définition en XML d'une action d'assistance de type message textuel et vocal pour l'application-cible PhotoScape

9.3.5. Animation

Pour réaliser dans une application-cible une action d'assistance de type animation, nous proposons un épi-assistant, uniquement disponible sous Windows, qui utilise des agents animés. Chaque agent animé proposé par cet épi-assistant est capable de s'exprimer oralement et textuellement, et de se déplacer à l'écran. Les personnages peuvent également réaliser des animations qui leur sont propres. Les animations les plus courantes selon le personnage sont saluer, lire, écrire, applaudir, montrer une zone de l'écran, sourire, être triste et alerter.

À titre d'exemple, la Fig. 9-16 présente neuf illustrations d'animations d'agents animés avec des personnages différents, de gauche à droite : Cathie parle, Merlin écrit, Isabella donne un coup de pied, Léo félicite l'utilisateur, Max salue, Milton effectue une recherche, Oscar montre quelque chose à l'écran, Peddy donne une explication et Pikka lit.

Fig. 9-16 – Exemples d'animations d'agents animés

Fig. 9-17 – Exemple des étapes d'une action d'agent animé

La Fig. 9-17 présente l'exemple d'A11, une action d'agent animé impliquant plusieurs animations et messages. Cette action instancie le patron d'action d'agent animé présenté en section 5.2.2 : la Fig. 9-18 présente sa définition en XML créée à l'aide de l'éditeur d'assistance.

```
<action id="A11" type="animation" personnage="Genie">
  <sousaction type="deplacement" position_x="700" position_y="600"/>
  <sousaction type="animation" identifiant="wave" />
  <sousaction type="parole" >
    <texte>Salut %prenom%</texte>
  </sousaction>
  <sousaction type="animation" identifiant="confused" />
  <sousaction type="pense">
    <texte>hum...</texte>
  </sousaction>
  <sousaction type="animation" identifiant="gestureLeft" />
  <sousaction type="parle" >
    <texte>Tu es sûr de cette réponse ?</texte>
  </sousaction>
  <sousaction type="animation" identifiant="suggest" />
  <sousaction type="parle" >
    <texte>J'ai une idée !</texte>
  </sousaction>
  <sousaction type="animation" identifiant="read" />
  <sousaction type="parole" >
    <texte>Tu devrais relire le cours numéro 3</texte>
  </sousaction>
</action>
```

Fig. 9-18 – Définition en XML de l'action d'agent animé représentée en Fig. 9-17

9.3.6. Ressource externe

Pour réaliser dans une application-cible une action d'assistance de type ressource externe, nous proposons un épi-assistant capable de lancer une page web dans le navigateur par défaut de l'utilisateur final de l'application-cible, de lancer un exécutable ou d'ouvrir un fichier avec l'application associée par défaut au type du fichier sur l'ordinateur de l'utilisateur final.

À titre d'exemple, la Fig. 9-19 présente la définition en XML de deux actions de type ressource : A16 permet d'ouvrir un fichier nommé Cours3.pdf et A17 permet d'ouvrir la page d'accueil du site web WordReference.

```
<action id="A16" type="fichier" chemin="Cours3.pdf"/>
<action id="A17" type="URL" URL="http://www.wordreference.com/fr/"/>
```

Fig. 9-19 – Définition en XML de deux actions de type ressource

9.3.7. Conclusion

Les épi-assistants que nous avons développés permettent de réaliser dans toutes les applications-cibles prises en charge par SEPIA les actions élémentaires d'assistance proposées par le langage aLDEAS. Nos épi-assistants permettent de réaliser une même action d'assistance de différentes manières, tout particulièrement pour les actions de type mise en valeur et message pour lesquels les nombreux paramètres optionnels permettent d'aboutir à des actions d'assistance visuellement très variées. Cette diversité permet la mise en place d'assistances riches et variées, de manière adaptée aux souhaits des concepteurs d'assistance et aux spécificités des utilisateurs finaux de l'application-cible.

9.4. Moteur générique d'assistance de SEPIA

Afin de permettre l'exécution des systèmes d'assistance définis avec l'éditeur d'assistance de SEPIA, nous avons développé un moteur générique d'assistance. Il constitue avec l'éditeur d'assistance l'un des deux principaux outils de SEPIA. Le moteur générique met en œuvre la phase d'exécution de l'assistance du processus d'adjonction d'un système d'assistance à une application-cible, ainsi que le modèle d'exécution d'un système d'assistance aLDEAS (cf. Chapitre 7).

Dans cette section, nous présentons le fonctionnement du moteur générique d'assistance de SEPIA du point de vue de l'utilisateur final, avant de décrire son fonctionnement d'un point de vue plus technique. Nous donnons également plusieurs exemples d'exécution d'assistance.

9.4.1. Lancement de l'assistance

Pour exécuter un système d'assistance avec le moteur générique, l'utilisateur final doit tout d'abord préciser pour quelle application-cible il souhaite recevoir de l'assistance. La Fig. 9-20 présente l'écran principal du moteur générique d'assistance de SEPIA. Sur cet écran, l'utilisateur final peut sélectionner l'application-cible pour laquelle il souhaite de l'assistance (cf. Ⓐ Fig. 9-20), puis l'un des fichiers d'assistance associés (cf. Ⓑ Fig. 9-20), avant de lancer l'exécution de cette assistance (cf. Ⓕ Fig. 9-20).

S'il le souhaite, l'utilisateur final peut également s'identifier avant de lancer l'exécution d'une assistance, dans le cas contraire il restera anonyme en laissant le profil *anonyme* sélectionné par défaut dans la liste déroulante qui permet l'identification (cf. © Fig. 9-20). L'identification d'un utilisateur lors de l'exécution d'un système d'assistance par le moteur générique d'assistance permet la prise en compte des préférences de cet utilisateur, ainsi que la personnalisation de l'assistance en fonction du profil de cet utilisateur.

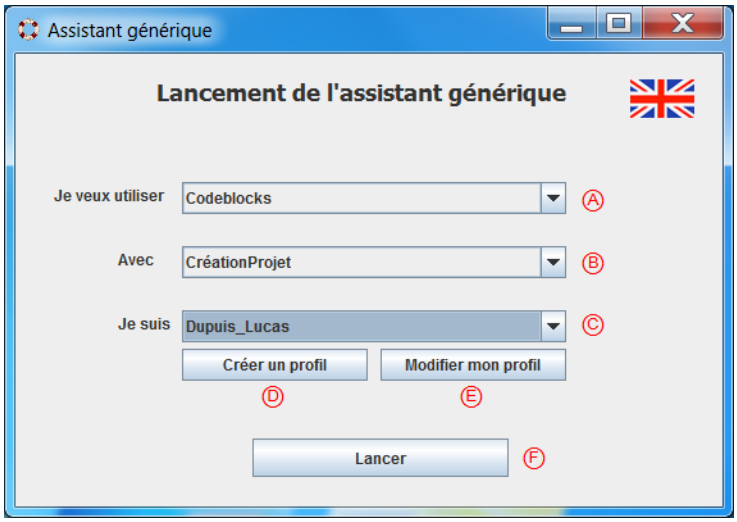


Fig. 9-20 – Écran principal du moteur générique d'assistance

9.4.2. Exploitation du profil de préférences

L'interface utilisateur du moteur générique d'assistance de SEPIA permet à l'utilisateur final de créer ou de modifier son profil de préférences (cf. Ⓓ et Ⓔ Fig. 9-20). La Fig. 9-21 présente l'écran d'édition d'un profil de préférences. Il permet à un utilisateur de définir ses préférences pour toutes les actions élémentaires d'assistance d'aLDEAS pour lesquelles une prise en compte des préférences est possible, c'est-à-dire celles associées à des paramètres optionnels : les messages, les mises en valeur, les masquages et les animations d'agents animés. En effet, lors de l'exécution d'une action élémentaire d'assistance pour laquelle le concepteur de l'assistance n'a pas spécifié un paramètre optionnel donné, le moteur générique va compléter la spécification de l'action en prenant en compte les préférences de l'utilisateur, définies dans son profil de préférences.

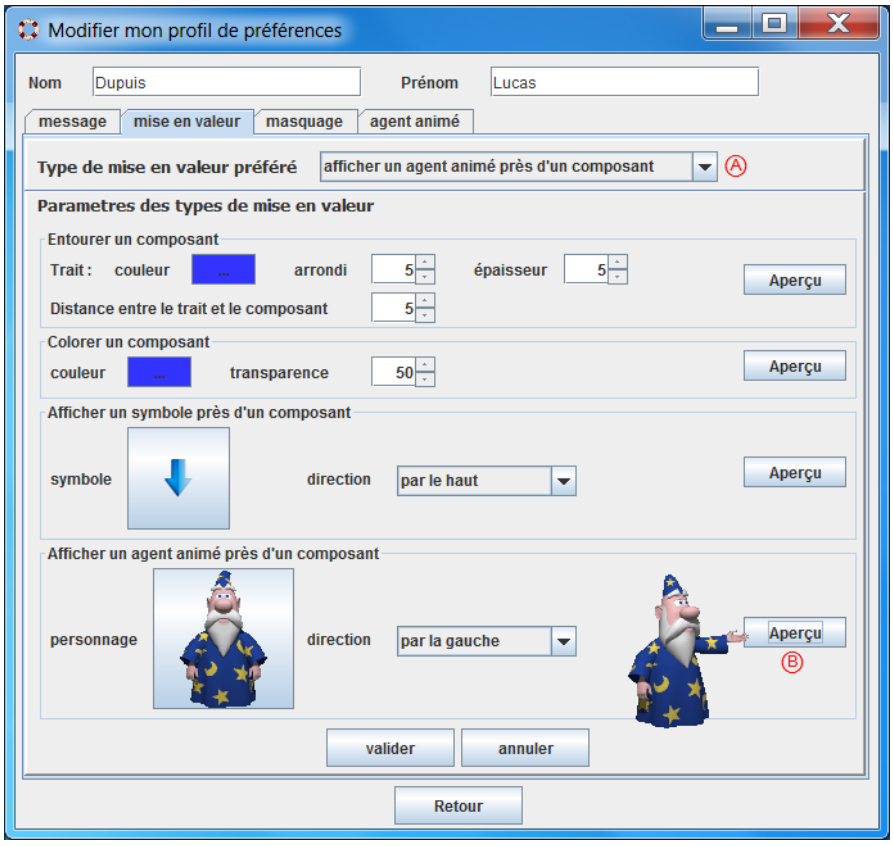


Fig. 9-21 – Définition des préférences d'un utilisateur

La Fig. 9-21 présente les préférences de Lucas Dupuis concernant les actions d'assistance de type *mise en valeur*. Lucas a défini ses préférences pour tous les types de mises en valeur proposées par aLDEAS et mises en œuvre dans le système SEPIA : mises en valeur par un entourage, par une coloration, par un symbole et par un agent animé. Lucas préfère que les mises en valeur soient réalisées par un agent animé (cf. Ⓐ Fig. 9-21). Plus précisément, il préfère que l'agent animé qui réalise les mises en valeur soit Merlin, et qu'il se place à gauche du composant à mettre en valeur. Pour chaque type de mise en valeur, le bouton *aperçu* déclenche sa propre mise en valeur suivant les paramètres définis. À titre d'exemple, l'aperçu d'une action de mise en valeur par un agent animé est donné en Ⓑ sur la Fig. 9-21.

9.4.3. Exploitation d'un profil d'utilisateur

Nous avons vu en section 8.1 que le concepteur de l'assistance peut associer à son système d'assistance une structure de profils respectant le formalisme PMDLe. L'association d'une structure de profils à un système d'assistance rend possible la personnalisation de l'assistance en fonction du profil de l'utilisateur. En effet, lors de la spécification de l'assistance avec l'éditeur de SEPIA, le concepteur de l'assistance a la possibilité de définir des consultations du profil de l'utilisateur, qui conditionneront le déclenchement d'actions d'assistance lors de l'exécution de l'assistance pour un utilisateur final.

Lors de l'exécution de l'assistance, si un utilisateur final a choisi de s'identifier, le moteur d'assistance fait un lien entre le nom de l'utilisateur final et son profil PMDLe correspondant

à la structure de profils associée au système d'assistance en exécution. Si l'utilisateur final ne s'est pas identifié, c'est-à-dire s'il s'est identifié comme utilisateur « anonyme », ou s'il n'existe pas de profil pour cet utilisateur et cette structure de profils, alors le moteur d'assistance ignorera toute règle contenant une consultation du profil de l'utilisateur.

Prenons l'exemple d'un système d'assistance dont le but est de guider l'utilisateur du logiciel pédagogique SolarySyst[15]. La règle R2, dont la définition en aLDEAS est donnée en Fig. 9-22, se déclenche lorsque l'utilisateur ouvre l'écran principal de SolarySyst (qui correspond au composant d'identifiant 2). Un message de bienvenue est alors affiché. Il contient un élément du profil de l'utilisateur : son prénom. La Fig. 9-23 présente l'exécution de la règle R2 dans SolarySyst pour Lucas. La Fig. 9-24 montre l'aperçu du même message pour Manon et pour un utilisateur qui ne s'est pas identifié au lancement de l'assistance. On constate que la balise « %prenom% » a été remplacée par « Lucas » ou par « Manon » pour ces deux utilisateurs, et par un espace pour un utilisateur anonyme. Notons également que le moteur générique a pris en compte les préférences des utilisateurs pour l'exécution du message. En effet, le message est affiché dans une fenêtre pop-up pour Lucas, avec une mise en forme correspondant à ses préférences, alors qu'il est transmis par l'agent animé « Peedy » pour Manon. Pour l'utilisateur anonyme, le moteur utilise les paramètres par défaut associés aux actions d'assistance de type *message*. Ces paramètres par défaut sont éditables depuis le moteur d'assistance.

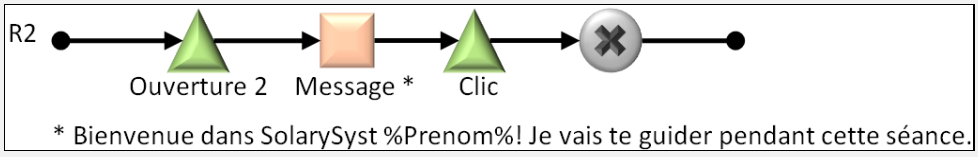


Fig. 9-22 – Définition en aLDEAS de la règle R2 du système d'assistance pour SolarySyst

Fig. 9-23 – Écran principal de SolarySyst avec un message d'assistance personnalisé

[15] http://liris.cnrs.fr/stephanie.jean-daubias/enseignement/M1/M1if22/Projets-2009-2010/

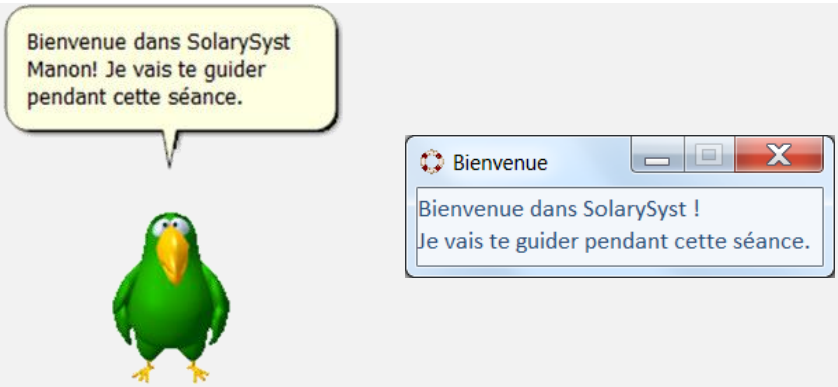


Fig. 9-24 – Exécution d'une action d'assistance de la règle R2 issue du système d'assistance pour SolarySyst : pour Manon (à gauche) et pour un utilisateur anonyme (à droite)

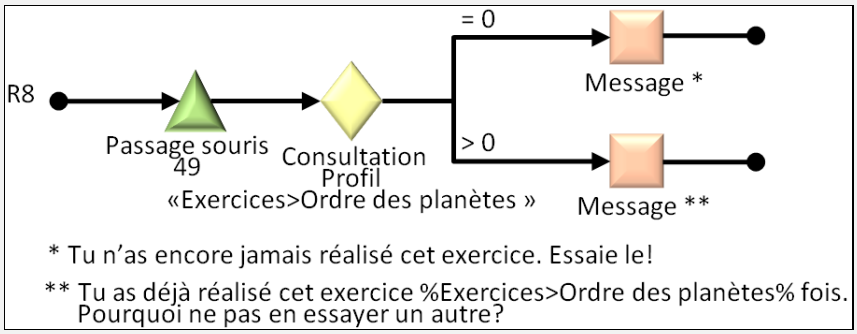


Fig. 9-25 – Définition en aLDEAS de la règle R8 du système d'assistance pour SolarySyst

La règle R8, dont la définition est donnée en Fig. 9-25, se déclenche lorsque l'utilisateur passe le curseur sur le bouton « Ordre des planètes » (qui correspond au composant d'identifiant 49). R8 contient une condition de déclenchement portant sur une consultation du profil de l'utilisateur. Pour vérifier cette condition, le moteur d'assistance consulte l'élément du profil de l'utilisateur « Exercices>Ordre des planètes » qui correspond au nombre de fois que l'utilisateur a effectué cet exercice. Pour les utilisateurs ayant déjà réalisé cet exercice, le moteur déclenche un message personnalisé en fonction de ce nombre. C'est le cas pour Lucas, qui a réalisé l'exercice 3 fois. L'exécution de ce message est présentée en Fig. 9-26. Pour les utilisateurs dont le profil indique qu'ils n'ont jamais réalisé cet exercice, comme c'est le cas de Manon, le moteur déclenche une autre action d'assistance, présentée en Fig. 9-27. Pour les utilisateurs non identifiés, le moteur générique ignore la règle R8. C'est également le cas pour les utilisateurs pour lesquels le moteur ne peut pas accéder à l'élément du profil impliqué par la consultation, soit parce que cet utilisateur n'a pas de profil dans la structure de profils associée au système d'assistance, soit parce que l'élément à consulter n'est pas renseigné.

Fig. 9-26 – Choix d'une activité pédagogique dans SolarySyst

Fig. 9-27 – Exécution d'une action d'assistance de la règle R8 issue du système d'assistance pour SolarySyst pour Manon

9.5. Conclusion

Le moteur générique d'assistance de SEPIA permet l'exécution de manière épiphyte de systèmes d'assistance définis par un ensemble de règles aLDEAS pour les utilisateurs finaux d'applications-cibles très variées. Il met en œuvre le langage aLDEAS et la personnalisation de l'assistance définie par le concepteur. Le moteur d'assistance permet également la prise en compte des préférences de l'utilisateur final, grâce à un profil de préférences éditable par l'utilisateur final.

Le fonctionnement du moteur d'assistance est défini par aMEAS, le modèle d'exécution d'un système d'assistance aLDEAS, présenté dans le Chapitre 7. Grâce aux épi-détecteurs qui surveillent de manière épiphyte les applications-cibles et qui notifient le moteur d'assistance de tout événement survenant à l'interface de ces applications, le moteur d'assistance est informé de toutes les interactions entre l'utilisateur final et l'application-cible, ce qui permet de contextualiser l'assistance.

De plus, les épi-assistants pilotés par le moteur d'assistance permettent de fournir de l'assistance à l'utilisateur final sous de multiples formes, correspondant aux actions d'assistance proposées dans le langage aLDEAS. Cette richesse des actions d'assistance possibles permet une adaptation fine de l'assistance aux souhaits de son concepteur et aux spécificités de l'utilisateur final.

Partie 4. Évaluations

Plan de la partie « Évaluations »

Cette thèse a abouti à plusieurs propositions théoriques visant à permettre la mise en place de systèmes d'assistance dans des applications-existantes avec une démarche épiphyte. Ces propositions ont été mises en œuvre de manière opérationnelle à travers le système SEPIA.

Cette partie concerne l'évaluation de nos propositions théoriques et de leur mise en œuvre. Nous présentons tout d'abord dans le Chapitre 10 une grille d'évaluation présentant sous forme synthétique les différents aspects à évaluer pour valider nos propositions. Dans le Chapitre 11, nous présentons de manière détaillée les différentes expérimentations que nous avons conduites, chacune permettant d'évaluer plusieurs points. Enfin, dans le Chapitre 12, nous reprenons notre grille d'évaluation en exposant les conclusions que nous avons tirées des différentes expérimentations menées.

Chapitre 10. Méthodologie

Objectif du chapitre

Présenter la méthodologie pour évaluer nos contributions théoriques et leur mise en œuvre à travers le système SEPIA.

Points clés

Nous présentons les différents aspects à évaluer pour valider nos propositions théoriques et leur mise en œuvre. Ces aspects concernent la **pertinence de notre approche** de mise en place *a posteriori* de systèmes d'assistance épiphyte en adoptant une démarche générique ; le langage **aLDEAS**, la **surveillance de l'application-cible**, le système **SEPIA**, l'assistance fournie aux utilisateurs finaux et la personnalisation de l'assistance. Nous détaillons chacun de ces aspects à évaluer et nous présentons la méthodologie d'évaluation envisagée, ainsi que les critères de validation associés.

Nous présentons dans ce chapitre les différents aspects à évaluer pour valider nos contributions théoriques, ainsi que leurs mises en œuvre. Pour chacun de ces aspects, nous synthétisons dans un tableau les méthodes d'évaluations envisagées, ainsi que les critères de validation associés.

10.1. Pertinence de l'approche

L'objectif de cette thèse est de permettre la mise en place *a posteriori* de systèmes d'assistance dans des applications existantes, en adoptant une démarche épiphyte. Pour déterminer la pertinence de notre approche, plusieurs questions se sont posées (cf. Fig. 10-1). Tout d'abord, existe-t-il un besoin d'assistance dans les applications-existantes ? De même, existe-t-il un désir de mettre en place de l'assistance dans des applications existantes sans les redévelopper ? Ensuite, une telle approche est-elle faisable ? Enfin, les propositions faites sont-elles réutilisables ?

Facettes à évaluer	Méthode d'évaluation	Critère de validation
Existence d'un besoin d'assistance	Questionnaire auprès d'utilisateurs finaux	Des utilisateurs finaux expriment un besoin d'assistance
	Expérimentation avec des utilisateurs finaux : réalisation sans assistance d'une tâche non connue	Des utilisateurs finaux ne parviennent pas à réaliser sans assistance une tâche donnée
Existence d'un désir de mettre en place de l'assistance	Questionnaire auprès de concepteurs d'assistance potentiels	Identification de situations dans lesquelles des personnes souhaitent mettre en place de l'assistance
Faisabilité de l'approche	Implémentation de l'approche proposée	Mises en place de systèmes d'assistance dans des applications sans les modifier
Réutilisabilité des propositions	Utilisation de nos propositions dans d'autres contextes	Exemples de réutilisations variées

Fig. 10-1 – Évaluation de la pertinence de notre approche

10.2. Le langage aLDEAS

La principale contribution théorique de cette thèse est le langage aLDEAS, qui permet la définition de systèmes d'assistance très variés pour des applications-cibles quelconques (cf. Chapitre 5). Nous avons cherché à évaluer d'une part sa couverture et d'autre part son utilisabilité et son intérêt pour faciliter la définition de systèmes d'assistance pour des concepteurs d'assistance. Nous avons également cherché à évaluer l'utilité des patrons que nous avons proposés en complément d'aLDEAS (cf. Fig. 10-2).

Facettes à évaluer	Méthode d'évaluation	Critère de validation
Couverture d'aLDEAS	Utilisation d'aLDEAS pour définir des assistances existantes	aLDEAS permet de définir des assistances existantes variées
	Expérimentation avec des concepteurs d'assistance : définition de systèmes d'assistance à l'aide d'aLDEAS	aLDEAS permet à des concepteurs d'assistance de définir l'assistance qu'ils souhaitent
Utilisabilité d'aLDEAS	Expérimentation avec des concepteurs d'assistance : définition de systèmes d'assistance à l'aide d'aLDEAS	Définition de systèmes d'assistance en aLDEAS par des concepteurs d'assistance informaticiens et non-informaticiens
Intérêt d'aLDEAS pour faciliter la spécification de systèmes d'assistance	Expérimentation avec des concepteurs d'assistance : définition de systèmes d'assistance avec ou sans aLDEAS	Les concepteurs d'assistance jugent que l'utilisation d'aLDEAS facilite la définition de l'assistance
Utilité des patrons d'aLDEAS	Utilisation des patrons pour définir des assistances existantes	L'utilisation des patrons facilitent la définition d'assistances

Fig. 10-2 – Évaluation du langage aLDEAS

10.3. Surveillance de l'application-cible

La surveillance d'une application-cible est indispensable à notre approche. En effet, cette surveillance permet de contextualiser l'assistance fournie, et ainsi de rendre cette assistance intégrée à l'application-cible aux yeux des utilisateurs finaux. Nous avons développé un ensemble d'épi-détecteurs (cf. Section 9.1) capables de surveiller une application-cible selon le processus que nous avons proposé (cf. Chapitre 4). Pour évaluer ces épi-détecteurs, plusieurs points sont à analyser (cf. Fig. 10-4). Tout d'abord, nos épi-détecteurs couvrent-ils toutes les applications existantes ? Accèdent-ils à tous les composants d'une application-cible ? Peuvent-ils détecter tous les événements liés à l'interaction entre un utilisateur final et l'interface d'une application-cible ? La source d'un événement est-elle identifiée de manière exacte ?

Facettes à évaluer	Méthode d'évaluation	Critère de validation
Couverture de nos épi-détecteurs	Classification des applications-cibles	Nous proposons un épi-détecteur pour chaque catégorie d'applications-cibles
Détection des composants d'une application-cible	Observation de l'arborescence des composants détectée par nos épi-détecteurs pour un ensemble d'applications-cibles de chaque catégorie	Obtention complète de l'arborescence des composants
Détection d'événements dans une application-cible et identification de leur source	Expérimentation avec nos épi-détecteurs	Nos épi-détecteurs détectent les événements d'interaction et identifient leur source

Fig. 10-3 – Évaluation de la surveillance de l'application-cible

10.4. Le système SEPIA

Afin de mettre en œuvre la phase de spécification de l'assistance du processus d'adjonction d'un système d'assistance à une application existante, nous avons développé un éditeur d'assistance. Cet outil propose une interface pour permettre à des concepteurs d'assistance de définir des systèmes d'assistance constitués d'un ensemble de règles aLDEAS. Pour évaluer cette proposition opérationnelle, nous devons évaluer d'une part son utilité et d'autre part son utilisabilité par des concepteurs d'assistance, qu'ils soient ou non informaticiens.

Afin de mettre en œuvre la phase d'exécution de l'assistance du processus d'adjonction d'un système d'assistance à une application existante, nous avons développé un moteur générique d'assistance. Il s'appuie sur un ensemble d'épi-détecteurs, qui permettent la surveillance de l'application-cible, et d'un ensemble d'épi-assistants, capables de réaliser des actions d'assistance dans l'application-cible. Afin d'évaluer l'exécution de l'assistance dans SEPIA, nous devons tout d'abord montrer que les outils de SEPIA permettent l'exécution des systèmes d'assistance définis à l'aide de l'éditeur d'assistance de SEPIA. Nous devons ensuite évaluer la satisfaction des concepteurs d'assistance vis-à-vis de l'assistance exécutée par les outils de SEPIA (cf. Fig. 10-4).

Facettes à évaluer	Méthode d'évaluation	Critère de validation
Utilité de l'éditeur d'assistance de SEPIA	Expérimentation avec des concepteurs d'assistance : définition de systèmes d'assistance	Définition avec l'éditeur de SEPIA de systèmes d'assistance conformes aux désirs de leur concepteur
Utilisabilité de l'éditeur d'assistance de SEPIA	Évaluation ergonomique	Critères ergonomiques respectés
	Expérimentation avec des concepteurs d'assistance : définition de systèmes d'assistance	Utilisation avec succès de l'éditeur de SEPIA par des concepteurs d'assistance informaticiens et non-informaticiens
Exécution de l'assistance avec SEPIA	Expérimentation : exécution de systèmes d'assistance avec SEPIA	Exécution avec succès de systèmes d'assistance avec SEPIA
	Expérimentation avec des concepteurs d'assistance : exécution de systèmes d'assistance avec SEPIA	L'assistance exécutée correspond aux attentes des concepteurs d'assistance

Fig. 10-4 – Évaluation du système SEPIA

10.5. Assistance fournie aux utilisateurs finaux

Afin d'évaluer l'assistance fournie par les outils de SEPIA aux utilisateurs finaux d'une application-cible, deux principales questions se posent (cf. Fig. 10-5). Tout d'abord, cette assistance est-elle bien acceptée par les utilisateurs finaux d'une application-cible ? Ensuite, cette assistance est-elle efficace pour répondre aux besoins d'assistance de ces derniers ?

Facettes à évaluer	Méthode d'évaluation	Critère de validation
Acceptation de l'assistance fournie aux utilisateurs finaux	Expérimentation avec des utilisateurs finaux : réalisation assistée d'une tâche	Les utilisateurs finaux apprécient l'assistance fournie par SEPIA
Efficacité de l'assistance fournie aux utilisateurs finaux	Expérimentation avec des utilisateurs finaux	Les utilisateurs finaux jugent efficace l'assistance fournie par SEPIA
	Expérimentation avec des utilisateurs finaux : réalisation d'une tâche avec ou sans assistance	L'assistance fournie par SEPIA permet de réaliser une tâche plus rapidement
	Expérimentation avec des utilisateurs finaux : réalisation assistée d'une tâche pour laquelle un besoin d'assistance a été identifié	L'assistance fournie par SEPIA permet à des utilisateurs finaux de réaliser une tâche qu'ils ne parviennent pas à réaliser sans assistance

Fig. 10-5 – Évaluation de l'assistance fournie aux utilisateurs finaux

10.6. Personnalisation de l'assistance

Le langage aLDEAS et sa mise en œuvre à travers SEPIA permet une personnalisation de l'assistance en fonction des spécificités des utilisateurs finaux, exprimées dans leur profil. Pour évaluer ce point, il est nécessaire de déterminer si une telle personnalisation de l'assistance est pertinente du point de vue des utilisateurs finaux, comme de celui des concepteurs d'assistance (cf. Fig. 10-6).

Facettes à évaluer	Méthode d'évaluation	Critère de validation
Pertinence de la personnalisation pour les concepteurs d'assistance	Questionnaire auprès de concepteurs d'assistance	Des besoins de personnalisation d'assistance sont identifiés
Pertinence de la personnalisation pour les utilisateurs finaux	Questionnaire auprès d'utilisateurs finaux	Des besoins de personnalisation d'assistance sont identifiés
	Expérimentation avec des utilisateurs finaux : réalisation d'une tâche avec une assistance personnalisée	Une assistance personnalisée est plus appréciée par les utilisateurs finaux et/ou plus efficace

Fig. 10-6 – Évaluation de la personnalisation de l'assistance

Chapitre 11. Expérimentations

Objectif du chapitre

Évaluer nos contributions théoriques et leur mise en œuvre à travers le système SEPIA selon la méthodologie présentée dans le chapitre précédent.

Points clés

Nous présentons les différentes expérimentations que nous avons menées pour évaluer nos propositions théoriques et leur mise en œuvre.

Nous avons utilisé aLDEAS pour **représenter des assistances existantes**. Nous avons également évalué nos propositions relatives à la **surveillance d'une application-cible**.

Nous avons effectué plusieurs expérimentations avec des concepteurs d'assistance informaticiens ou non, dans lesquelles ces concepteurs devaient utiliser aLDEAS puis l'éditeur de SEPIA pour **spécifier un système d'assistance**. Ces expérimentations ont impliqués 71 concepteurs d'assistance extérieurs au projet AGATE, ainsi que 40 applications-cibles différentes.

Nous avons effectué deux expérimentations avec des **utilisateurs finaux** pour lesquels une assistance était fournie par le moteur générique de SEPIA. Ces expérimentations ont impliqué 252 utilisateurs finaux extérieurs au projet AGATE, pour 2 applications-cibles (PhotoScape et NetBeans) utilisées dans des contextes différents (un usage personnel pour PhotoScape et un usage en contexte éducatif pour NetBeans). Nous souhaitions en effet évaluer la capacité d'aLDEAS et de sa mise en œuvre à travers SEPIA à fournir une assistance efficace et bien acceptée par les utilisateurs finaux, dès lors que l'assistance a été judicieusement spécifiée par le concepteur. Cependant, l'évaluation de la pertinence d'une assistance sort du cadre de cette thèse.

Publications liées à ce chapitre

[Ginon et al., 2013c]

> Ginon B., Jean-Daubias S. et Champin P.-A., Mise en place d'un système d'assistance personnalisée dans une application existante, IC, Lille, France, 2013c.

[Ginon et al., 2014a]

> Ginon B., Jean-Daubias S., Champin P.-A. et Lefevre M., aLDEAS : un langage de définition de systèmes d'assistance épiphytes, IC, Clermont-Ferrand, France, pp. 137-148, 2014a.

[Ginon et al., 2014b]

> Ginon B., Thai V. L., jean-Daubias S., Lefevre M. et Champin P.-A., Adding epiphytic assistance systems in learning applications using the SEPIA system, Ec-Tel, Graz, Austria, 2014b.

[Thái et Ginon, 2014]

> Thái L. V. et Ginon B., Exploitation d'assistances épiphytes en contexte éducatif, RJC-EIAH, La Rochelle, France, 2014.

Afin d'évaluer les différents aspects de nos propositions que nous avons présentés dans le chapitre précédent, nous avons conduit plusieurs expérimentations de natures variées. Certaines expérimentations ont été menées par les membres du projet AGATE, dans les cas où il n'était pas pertinent d'impliquer des concepteurs d'assistance ni des utilisateurs finaux. Ainsi, nous avons utilisé aLDEAS puis SEPIA pour décrire des assistances existantes afin d'évaluer leur couverture. De même, nous avons réalisé une étude visant à évaluer nos épi-détecteurs qui permettent la surveillance d'une application-cible.

D'autres expérimentations impliquaient des personnes extérieures au projet AGATE : certaines personnes avaient le rôle de concepteur d'assistance et d'autres celui d'utilisateur final d'une application-cible pour laquelle un système d'assistance a été mis en place à l'aide du système SEPIA. Ainsi, nous avons réalisé plusieurs expérimentations dans lesquelles des concepteurs d'assistance ont utilisé SEPIA pour définir des systèmes d'assistance pour plusieurs applications-cibles. Nous avons également mené deux expérimentations plus complètes avec SEPIA, allant de la définition d'un système d'assistance pour une application-cible à l'exécution de cette assistance pour les utilisateurs finaux de cette application-cible.

11.1. Utilisation d'aLDEAS pour décrire des assistances existantes

Afin d'évaluer la couverture du langage aLDEAS, nous l'avons utilisé pour modéliser des assistances existantes variées, choisies pour être représentatives des types d'assistance fréquemment rencontrés (cf. Annexe A). Ainsi, notre langage permet de modéliser les assistances permettant de présenter une application ou une fonctionnalité (cf. Annexe A-1) : ces assistances sont souvent proposées à l'utilisateur lors de la première utilisation d'un logiciel, ou suite au développement d'une nouvelle version de ce logiciel. De plus, il permet de modéliser les assistances souvent utilisées dans les applications web contenant un formulaire (cf. Annexe A-2) : elles permettent par exemple d'expliquer à l'utilisateur où trouver les informations pour remplir un champ, ou le prévenir qu'un champ n'est pas rempli. Notre langage permet également de modéliser les assistances de type présentation guidée d'un logiciel ou d'une fonctionnalité. Par ailleurs, il existe de nombreux tutoriels proposés par des utilisateurs experts pour guider un utilisateur pas à pas pour réaliser une tâche, à l'aide de captures d'écran annotées et de messages. Notre langage permet de modéliser de tels systèmes d'assistance avec l'avantage de le faire de façon plus intégrée à l'application-cible. Ainsi, les captures d'écran annotées pourront être remplacées par des actions de mise en valeur intégrées à l'application-cible (cf. Annexe A-3).

Il existe néanmoins des systèmes d'assistance que notre langage ne permet pas de modéliser : c'est le cas des systèmes d'assistance très spécifiques à une application et requérant des informations non disponibles depuis l'extérieur de cette application. Par exemple, les moteurs de recommandations intégrés dans des applications de vente en ligne utilisent l'ensemble des informations sur les articles consultés ou achetés par tous les utilisateurs du site. Notre langage ne permet pas la consultation de l'ensemble de ces informations.

Afin de comparer notre approche avec d'autres approches de mise en place *a posteriori* d'assistances épiphytes, nous montrons ici comment aLDEAS peut être utilisé pour mettre en place des assistances équivalentes à celles proposées dans notre état de l'art.

Ainsi, le patron d'actions d'agents animés associé à aLDEAS permet de définir des actions d'assistance équivalentes à celles proposées dans iFrimousse [Carlier et Renault, 2010] : l'agent animé peut se déplacer à l'interface de l'application web, afficher ou lire des messages et jouer des animations. Avec notre approche, il n'est pas nécessaire d'ajouter des balises à l'application web pour permettre les actions de l'agent animé. Par exemple, pour indiquer à un agent animé de se placer à côté d'un composant donné, le concepteur de l'assistance doit compléter le paramètre *position* de l'action d'assistance, au lieu d'avoir recours à des balises de position comme c'est le cas dans iFrimousse.

Le langage aLDEAS permet également l'ajout d'un système conseiller dans une application web, de manière épiphyte, comme c'est le cas pour [Richard, 2008]. Le patron de règles aLDEAS permet de définir des règles avec comme événement déclencheur un clic sur un lien donné, comme éventuelle condition de déclenchement une consultation de l'historique de l'assistance ou de la navigation, et comme action d'assistance un message dans une fenêtre pop-up.

Concernant les assistances qui peuvent être mises en place dans les environnements ExploraGraph [Dufresne, 2001] et Telos [Paquette, 2012], le patron de règles d'assistance aLDEAS est comparable au modèle de règles de [Dufresne et al., 2003] : <événement déclencheur, condition optionnelle d'assistance, action d'assistance, événement optionnel de fin>. Le langage aLDEAS permet donc de définir des règles de formes équivalentes aux règles proposées dans ces deux environnements. Concernant les actions d'assistance, aLDEAS permet également la définition d'actions d'agents animés et de messages textuels et sonores équivalents à ceux proposés dans ces deux environnements. En revanche, dans l'environnement Telos, les actions d'assistance peuvent prendre la forme de notifications envoyées à certains acteurs du scénario, par exemple un mail envoyé à un enseignant. Ce type d'actions d'assistance n'est actuellement pas proposé par aLDEAS.

11.2. Évaluation de la surveillance d'une application-cible

Dans le cadre de cette thèse, nous avons proposé un processus de surveillance d'applications existantes de manière épiphyte (cf. Chapitre 4), qui s'appuie sur l'utilisation de bibliothèques d'accessibilité. Nous avons mis en œuvre ce processus à travers un ensemble d'épi-détecteurs (cf. Section 9.1), qui peuvent assurer la surveillance d'un grand nombre d'applications existantes, principalement sous Windows. Afin d'évaluer la couverture de nos épi-détecteurs, la Fig. 11-1 montre la compatibilité des applications bureau avec les bibliothèques d'accessibilité. Sur cette figure, la colonne Validation indique quelles catégories d'applications sont actuellement couvertes par nos épi-détecteurs. Plusieurs épi-détecteurs sont encore à développer pour fournir au système SEPIA une couverture maximale des applications bureau. Néanmoins, les épi-détecteurs actuellement développés démontrent la faisabilité de notre approche de surveillance épiphyte des applications bureau. Concernant la surveillance des applications web, nous avons proposé une technique

qui s'appuie sur des scripts utilisateurs, qui peuvent être intégrés à une application web par une extension de navigateur. Cette technique ne concerne néanmoins pas les applications Flash, qui ne permettent pas l'utilisation de scripts utilisateurs.

		Bibliothèque	Script utilisateur	Validation
Windows	Exécutables	UIAutomation		✓
	Applications Java	JavaAccessibility		✓
	Applications GTK / QT	ATK/AT-SPI		✗
	Applications web		Script de collecte	✓
Linux	Applications natives	ATK/AT-SPI		✓
	Applications Java	JavaAccessibility		✗
	Applications GTK / QT	ATK/AT-SPI		✓
	Applications web		Script de collecte	✓
Mac OS	Applications natives	Apple_API_Accessibility		✗
	Applications Java	JavaAccessibility		✗
	Applications GTK / QT	ATK/AT-SPI		✗
	Applications web		Script de collecte	✓

Fig. 11-1 – Compatibilité des applications avec les bibliothèques d'accessibilité

Afin d'évaluer l'efficacité de nos épi-détecteurs, nous les avons utilisés pour détecter les composants terminaux d'une cinquantaine d'applications variées (cf. Annexe C). Certaines étaient des applications professionnelles, comme la suite Microsoft Office ou l'IDE NetBeans, d'autres étaient des applications non professionnelles, comme des logiciels issus de travaux d'étudiants ou des freewares. Nous qualifions de *terminaux* les composants d'une application avec lesquels l'utilisateur peut interagir, ainsi que ceux qu'il peut identifier visuellement : les boutons, les menus, les cases à cocher, les barres de progressions, les images, etc. Le taux de détection est très proche de 100 %, à la fois dans les applications professionnelles et non professionnelles. Parmi les composants non détectés par nos épi-détecteurs, se trouvent certains composants développés spécifiquement pour une application et non compatibles avec une bibliothèque d'accessibilité, ainsi que certains composants Unity et Delphi (pour les versions de Delphi antérieures à 7).

Concernant la détection d'événements, la liste des types d'événements détectables par nos épi-détecteurs sur les composants auxquels ils accèdent est donnée en Fig. 9-1. Ces types d'événements sont très nombreux, nos épi-détecteurs peuvent être paramétrés pour choisir les types d'événements qui doivent être détectés. Certains types d'événements apportent des informations jugées non pertinentes par le concepteur de l'assistance, il peut donc être utile de ne pas les surveiller. En effet, nous avons constaté en testant nos épi-détecteurs que, lorsqu'ils surveillent tous les types d'événements possibles, le mouvement du pointeur de la souris peut être ralenti, selon les performances de l'ordinateur utilisé. Il n'y a en revanche aucune conséquence négative à l'utilisation de nos épi-détecteurs pour surveiller uniquement les principaux types d'événements comme les clics, les ouvertures et fermetures de menus, fenêtres ou infobulles, les saisies de texte…

11.3. Spécification de systèmes d'assistance avec SEPIA

Nous avons réalisé deux expérimentations avec le système SEPIA, afin d'évaluer la phase de spécification de l'assistance. Elles se sont déroulées à un an d'intervalle dans le contexte d'un cours de première année de master informatique de l'Université Lyon 1, sur les logiciels pédagogiques. Dans les deux cas, les étudiants ont développé en binômes dans le langage de leur choix un logiciel pédagogique simple, à destination d'élèves du primaire, dans le cadre du projet effectué chaque année dans ce cours. Ils ont ensuite utilisé SEPIA pour greffer un système d'assistance à leur logiciel.

Dans un tout autre contexte, nous avons également réalisé une première expérimentation qui visait à évaluer l'utilisabilité d'aLDEAS et de SEPIA par des non-informaticiens, pour mettre en place de l'assistance technique dans l'application grand public PhotoScape.

11.3.1. Spécification d'assistances par des informaticiens : printemps 2013

A. Description de l'expérimentation

Au semestre de printemps 2013, nous avons demandé à 37 étudiants, majoritairement en binômes, d'endosser le rôle de concepteurs d'assistance pour les 20 logiciels qu'ils avaient développés. Cette expérimentation avait pour objectif d'une part d'évaluer la capacité de l'éditeur d'assistance à permettre la spécification de systèmes d'assistance variés, et d'autre part d'évaluer l'utilisabilité de l'éditeur d'assistance.

B. Déroulement de l'expérimentation

Pendant une première séance de travail d'1 h, chaque binôme devait concevoir sur papier le système d'assistance qu'il souhaitait greffer à son logiciel pédagogique. Ensuite, pendant une seconde séance, nous avons présenté l'éditeur d'assistance aux étudiants, puis nous leur avons demandé de l'utiliser pendant 1h30 pour spécifier le système d'assistance qu'ils avaient conçu sur papier.

C. Résultats de l'expérimentation

Les 20 binômes de concepteurs d'assistance sont parvenus à spécifier 20 systèmes d'assistance en 1h30, sans connaissance préalable de l'éditeur d'assistance de SEPIA. Pour cela, ils ont créé 134 règles d'assistance, 81 conditions de déclenchement et 158 actions d'assistance, soit une moyenne par système d'assistance de 6,7 règles, 4,05 conditions et 7,9 actions d'assistance. Les Fig. 11-2 et Fig. 11-3 présentent respectivement la répartition des conditions de déclenchement et des actions d'assistance créées selon les types proposés par aLDEAS. L'expérimentation était suivie d'un questionnaire de satisfaction (cf. Fig. 39, Annexe E-3).

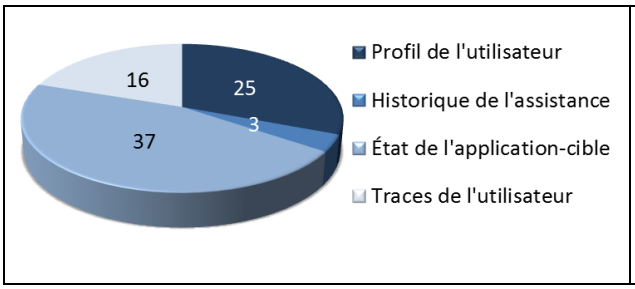


Fig. 11-2. Répartition des conditions de déclenchement

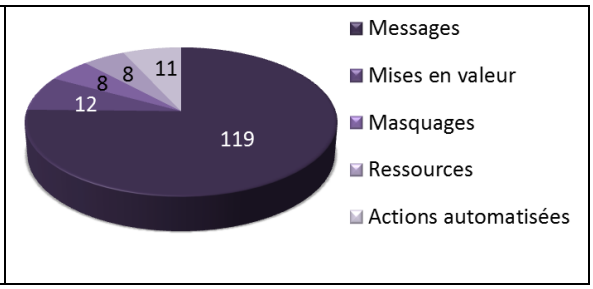


Fig. 11-3. Répartition des actions d'assistance

D. Analyse des résultats

Le nombre important de règles d'assistance contenant une condition de déclenchement (60,45 % des règles) montrent la volonté des concepteurs d'assistance de personnaliser l'assistance. Les concepteurs d'assistance ont ainsi créé 25 conditions sur le profil de l'utilisateur afin de personnaliser l'assistance en fonction des connaissances des apprenants, 3 conditions sur l'historique de l'assistance, 16 conditions sur les actions de l'utilisateur afin de faire évoluer l'assistance, et 37 conditions sur l'état de l'application-cible afin de contextualiser l'assistance (cf. Fig. 11-2). Il est à noter que le désir des concepteurs d'assistance de personnaliser l'assistance dans un contexte éducatif n'est pas nécessairement représentatif du désir de personnaliser l'assistance dans d'autres contextes.

Toutes les actions d'assistance imaginées par les concepteurs d'assistance ont pu être spécifiées à l'aide de l'éditeur d'assistance, à l'exception de trois actions. Une de ces trois actions non proposées dans notre approche consiste à modifier ponctuellement la forme du curseur, une autre propose une fonctionnalité de type loupe et une dernière permet de modifier le niveau de difficulté des activités pédagogiques proposées. En ce qui concerne l'adaptation des activités pédagogiques, cela n'était pas directement possible, puisque le logiciel-cible ne proposait qu'un seul niveau de difficulté. Néanmoins, il est possible de fournir à l'apprenant des indices ou des exemples afin d'adapter de manière progressive la difficulté de l'activité pédagogique proposée par le logiciel.

Sans avoir eu connaissance des types d'actions d'assistance proposés par aLDEAS et par l'éditeur d'assistance, les concepteurs d'assistance avaient prévu de définir des actions d'assistance correspondant à plusieurs types. Ils ont ainsi créé 119 actions de type *message*, 12 actions de type *mise en valeur*, 8 actions de type *masquage*, 8 actions de type *ressources* (sous forme de vidéos de démonstration), et 11 actions de type *action automatisée* (cf. Fig. 11-3). Cette forte dominance des actions de type message, qui représentent en effet 75 % des actions, peut être liée à la simplicité de ce type d'actions et au fait que les concepteurs d'assistance ont jugé les messages d'aide efficaces pour répondre aux besoins d'assistance de leurs logiciels pédagogiques. Par ailleurs, ils ne connaissaient pas la diversité des actions d'assistance proposées par aLDEAS et SEPIA et ont limité leur choix aux actions d'assistance les plus communément rencontrées dans les systèmes existants.

11.3.2. Spécification d'assistances par des informaticiens : printemps 2014

A. Description de l'expérimentation

Au semestre de printemps 2014, nous avons demandé à 29 étudiants du cours de logiciels éducatifs d'endosser le rôle de concepteurs d'assistance pour les 14 logiciels qu'ils avaient développés. Lors de la conception de leur logiciel, les étudiants avaient été informés qu'ils leur ajouteraient une assistance épiphyte par la suite. Les concepteurs d'assistance devaient dans un premier temps définir sur papier l'assistance qu'ils souhaitaient, en utilisant aLDEAS pour certains et sans formalisme pour les autres. Cette expérimentation avait pour objectif principal l'évaluation de l'utilisabilité et de la couverture d'aLDEAS.

B. Déroulement de l'expérimentation

La première séance de travail concernait la conception de l'assistance sur papier et a duré 3 heures. Après 5 minutes d'explications sur les objectifs de l'expérimentation, chacun des 13 binômes a été séparé dans deux salles, pour qu'il y ait un membre de chaque binôme dans chaque salle. Dans la première salle, 14 étudiants ont réfléchi librement à l'assistance qu'ils souhaitaient pour leur logiciel, avant de la définir sur papier. Dans l'autre salle, les 13 étudiants restants ont reçu des explications sur le langage aLDEAS pendant 10 minutes, avant de définir sur papier l'assistance qu'ils souhaitaient, sous la forme d'un ensemble de règles aLDEAS.

Après 90 minutes de travail dans des salles séparées, les binômes ont été reconstitués afin de mettre en commun leurs idées. Ils ont défini un système d'assistance pour leur logiciel sous la forme d'un ensemble de règles aLDEAS. Cette première séance de travail était suivie d'un questionnaire (cf. Annexe D).

La semaine suivante, la seconde séance de travail était consacrée à la spécification de ces systèmes d'assistance avec l'éditeur de SEPIA, en utilisant les systèmes d'assistance définis sur papier avec aLDEAS lors de la séance précédente. Cette séance a duré 3 heures et a débuté par une démonstration de 10 minutes de l'éditeur d'assistance. Chaque binôme a ensuite pu utiliser l'éditeur d'assistance de SEPIA pour définir des règles d'assistance, puis tester leur exécution dans leur logiciel en utilisant le moteur générique d'assistance de SEPIA. Cette seconde séance de travail était suivie d'un questionnaire (cf. Annexe D).

C. Résultats de l'expérimentation

Suite à la première séance de travail, les 29 concepteurs d'assistance ont réussi à définir sur papier 14 systèmes d'assistance sous la forme d'un ensemble de règles aLDEAS (cf. Fig. 28 à Fig. 31 en Annexe D). La Fig. 11-4 présente les résultats du questionnaire consécutif à cette première séance de travail (cf. Fig. 40, Annexe E-3).

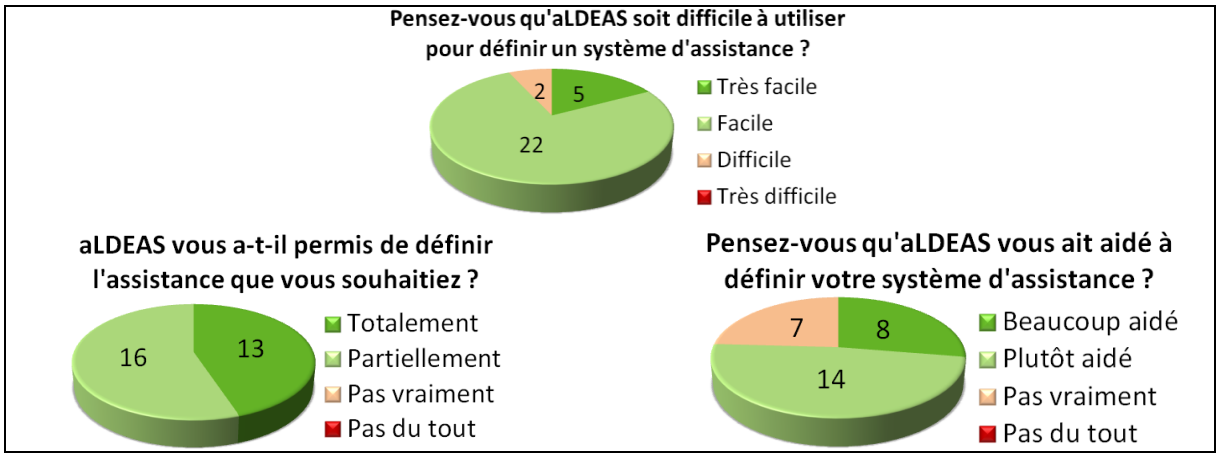


Fig. 11-4. Résultats de questions relatives à l'utilisation d'aLDEAS sur papier

Suite à la seconde séance de travail, les 27 concepteurs d'assistance ont réussi à définir avec l'éditeur d'assistance de SEPIA 14 systèmes d'assistance. Ils ont été confrontés à des difficultés dues à des *bugs* de SEPIA, qui ont été corrigés depuis cette expérimentation. Ceci ne les a toutefois été pas empêchés de créer des systèmes d'assistance intéressants. Les Fig. 11-5 à Fig. 11-7 présentent les résultats du questionnaire consécutif à cette seconde séance de travail (cf. Fig. 41, Annexe E-3).

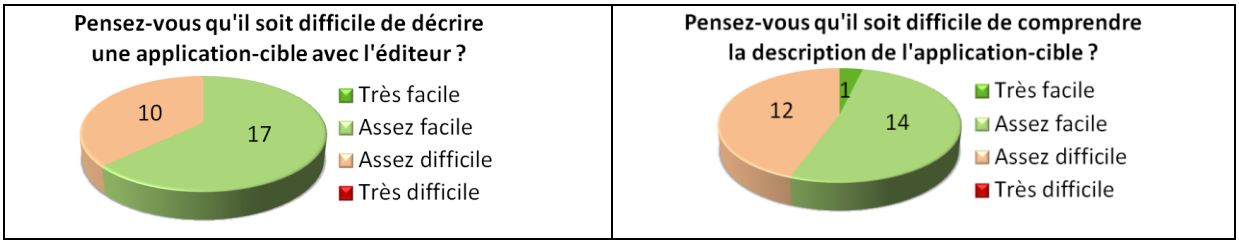


Fig. 11-5. Résultats de questions relatives à la description de l'application-cible avec l'éditeur

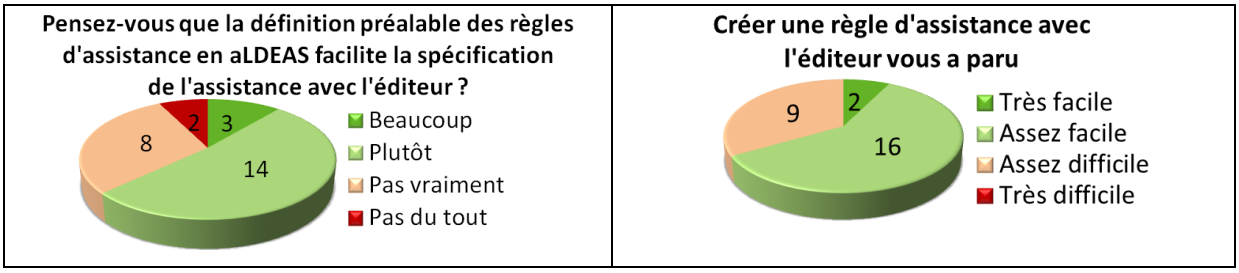


Fig. 11-6. Résultats de questions relatives à la définition de règles d'assistance avec l'éditeur

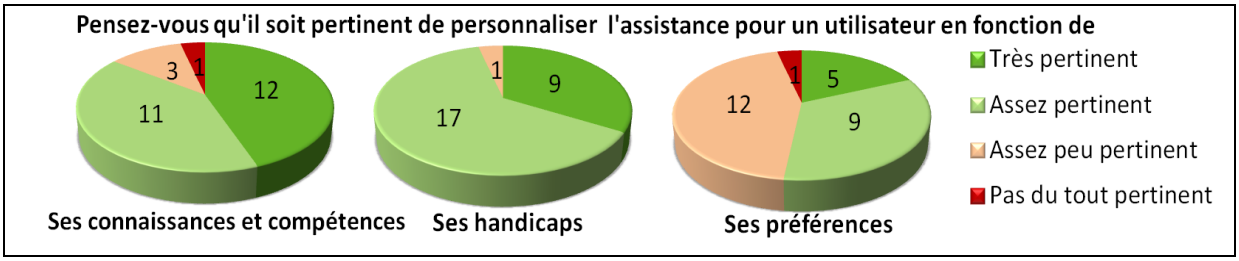


Fig. 11-7. Résultats de questions relatives à la personnalisation de l'assistance

D. Analyse des résultats

Cette expérimentation a montré qu'aLDEAS peut être utilisé pour définir sur papier des systèmes d'assistance par des concepteurs d'assistance ayant des connaissances en informatique. En effet, seuls 7 % des participants ont trouvé aLDEAS difficile à utiliser, 100 % des participants ont considéré qu'aLDEAS leur avait permis de définir l'assistance qu'ils souhaitaient au moins en partie, et 76% des participants ont pensé qu'aLDEAS les avait aidés à définir leur système d'assistance (cf. Fig. 11-4). De plus, 63 % des participants ont pensé que l'utilisation préalable d'aLDEAS facilite la spécification de l'assistance avec l'éditeur d'assistance de SEPIA (cf. Fig. 11-6). Pendant, le temps de mise en commun du travail sur la définition de l'assistance avec ou sans aLDEAS, les binômes ont en effet échangé sur leur expérience, puis ont travaillé ensemble à la définition en aLDEAS de leur système d'assistance, ce qui leur a permis à tous de se forger une opinion sur aLDEAS.

Cette expérimentation a également montré que l'éditeur d'assistance est utilisable par des concepteurs d'assistance ayant des connaissances en informatique, puisque 67 % des participants l'ont trouvé facile à utiliser pour définir des règles d'assistance (cf. Fig. 11-6). Les résultats sont plus mitigés concernant l'étape de description de l'interface de l'application-cible. En effet, 63 % des participants ont trouvé cette étape facile à réaliser avec l'éditeur de SEPIA, mais 44 % d'entre eux ont néanmoins considéré qu'il est assez difficile de comprendre le fichier de description de l'interface créé avec l'éditeur (cf. Fig. 11-5).

Concernant la personnalisation de l'assistance, 85 % des participants ont considéré qu'il est pertinent de personnaliser l'assistance en fonction des connaissances et compétences des utilisateurs et 97 % d'entre eux ont considéré qu'il est pertinent de personnaliser l'assistance en fonction des handicaps des utilisateurs. En revanche, seuls 59 % des participants ont jugé qu'il est pertinent de personnaliser l'assistance en fonction des préférences des utilisateurs. Il est cependant à noter que dans le cadre de ce cours, les étudiants avaient été sensibilisés à l'intérêt de la personnalisation en général dans le contexte éducatif, ce qui a certainement influencé leur opinion.

11.3.3. Spécification d'assistance par des non-informaticiens

A. Description de l'expérimentation

Nous avons demandé à cinq personnes non-informaticiennes d'utiliser aLDEAS pour définir un système d'assistance pour la correction des yeux rouges sur une photo avec PhotoScape. Ces concepteurs d'assistance avaient entre 11 et 68 ans et n'avaient aucune connaissance préalable ni d'aLDEAS ni de SEPIA. Les systèmes d'assistance qu'ils ont conçus sont équivalents à celui testé auprès de 151 utilisateurs finaux de PhotoScape (cf. Section 11.4.1).

B. Déroulement de l'expérimentation

Nos concepteurs d'assistance d'un jour ont reçu des explications sur aLDEAS pendant environ 5 minutes et nous leur avons fourni trois exemples de règles d'assistance aLDEAS provenant d'un système d'assistance pour une autre application-cible et qui pouvaient leur

servir de modèle. Dans un premier temps, les concepteurs d'assistance ont utilisé aLDEAS pour définir leur système d'assistance sur papier. Ce travail a duré entre 30 et 50 minutes selon les concepteurs, pour aboutir à la définition de 7 à 9 règles d'assistance respectant le patron de règles d'aLDEAS. Dans un second temps, après avoir reçu quelques explications sur le fonctionnement de l'éditeur de SEPIA, les concepteurs d'assistance ont utilisé le système SEPIA pour spécifier le système d'assistance conçu sur papier avec aLDEAS. Ils ont également pu tester l'exécution de leur système d'assistance tout au long de sa spécification avec SEPIA. Ce second travail a duré entre 55 et 95 minutes selon les concepteurs d'assistance.

C. Résultats de l'expérimentation

Chaque concepteur d'assistance a réussi à définir, en aLDEAS (cf. Fig. 27 en Annexe D), puis avec SEPIA, un système d'assistance pour la correction des yeux rouges dans PhotoScape, exécutable avec SEPIA. Quatre concepteurs d'assistance sur cinq ont trouvé aLDEAS facile à utiliser, et tous ont déclaré que l'utilisation préalable d'aLDEAS sur papier les avait beaucoup aidés à définir leur système d'assistance. Tous ont également trouvé l'éditeur de SEPIA facile à utiliser une fois que les règles d'assistance sont définies en aLDEAS.

D. Analyse des résultats

Ces premiers résultats encourageants semblent montrer qu'aLDEAS et SEPIA peuvent être utilisés par des non-informaticiens. Il est néanmoins indispensable de confirmer ces résultats par une expérimentation plus large avec des concepteurs d'assistance non-informaticiens et avec d'autres applications-cibles que PhotoScape.

11.4. Exécution de systèmes d'assistance avec SEPIA

Nous avons réalisé deux expérimentations avec SEPIA dans lesquelles un système d'assistance a été défini, puis exécuté pour des utilisateurs finaux de l'application-cible, afin d'évaluer auprès d'utilisateurs finaux l'utilité, l'efficacité et l'acceptation de l'assistance que nos propositions permettent de fournir.

11.4.1. Correction des yeux rouges sur une photo avec Photoscape

A. Description de l'expérimentation

Nous avons demandé à 200 personnes d'utiliser PhotoScape[16] pour corriger des yeux rouges sur une photo. Cette expérimentation avait pour objectif, d'une part d'évaluer l'efficacité de l'assistance fournie pour répondre à un besoin d'assistance, et d'autre part d'évaluer l'acceptation de cette assistance par les utilisateurs finaux.

Nous avons créé à l'aide de l'éditeur d'assistance de SEPIA un système d'assistance pour PhotoScape. Ce système d'assistance permet de guider les utilisateurs dans la réalisation de

[16] Vidéo de démonstration de l'assistance exécutée dans PhotoScape disponible à http://liris.cnrs.fr/blandine.ginon/PhDWork.html

chaque étape d'une tâche de correction des yeux rouges. La définition en aLDEAS de ce système d'assistance est donnée en annexe, ainsi que le fichier XML correspondant que nous avons produit à l'aide de l'éditeur de SEPIA. La définition à l'aide de l'éditeur de SEPIA nous a demandé environ 20 minutes, pour créer les 10 règles du système d'assistance, ainsi que 9 actions d'assistance (principalement de type messages et mises en valeur), 7 consultations de l'utilisateur et 2 attentes d'événements liées à l'action de l'utilisateur dans PhotoScape. La définition en aLDEAS de ce système d'assistance est donnée par la Fig. 20 en Annexe B, et son fichier de description xml généré par l'éditeur de SEPIA et exécuté par le moteur de SEPIA est donné par la Fig. 21 en Annexe B.

B. Déroulement de l'expérimentation

Nous avons demandé à 200 personnes de réaliser une tâche imposée : corriger avec PhotoScape les yeux rouges sur une photo donnée. Cette tâche devait être réalisée sans aide pour les 100 utilisateurs du groupe A, et avec l'aide exécutée par SEPIA pour les 100 utilisateurs du groupe B. Pour le groupe A, on distingue 2 sous-groupes : A1 pour les 49 utilisateurs qui ont réussi à réaliser la tâche demandée sans assistance, et A2 pour les 51 utilisateurs ayant abandonné et à qui nous avons ensuite demandé de réaliser la même tâche avec l'assistance conçue. La Fig. 11-8 présente la répartition des utilisateurs des groupes A1, A2 et B, en fonction de différents critères. Ces personnes proviennent en majorité de l'Université Lyon 1 : étudiants, personnels scientifiques et administratifs. L'âge des utilisateurs varie entre 15 et 68 ans, avec 64,5 % des utilisateurs entre 15 et 25 ans, 30,5 % des utilisateurs entre 25 et 55 ans et 5 % des utilisateurs ayant plus de 55 ans. Par ailleurs, parmi ces 200 utilisateurs, 72,5 % sont des hommes, 83 % sont des informaticiens ou des utilisateurs confirmés d'ordinateurs, et 65 % n'avaient jamais utilisé de logiciel pour corriger des yeux rouges sur une photo.

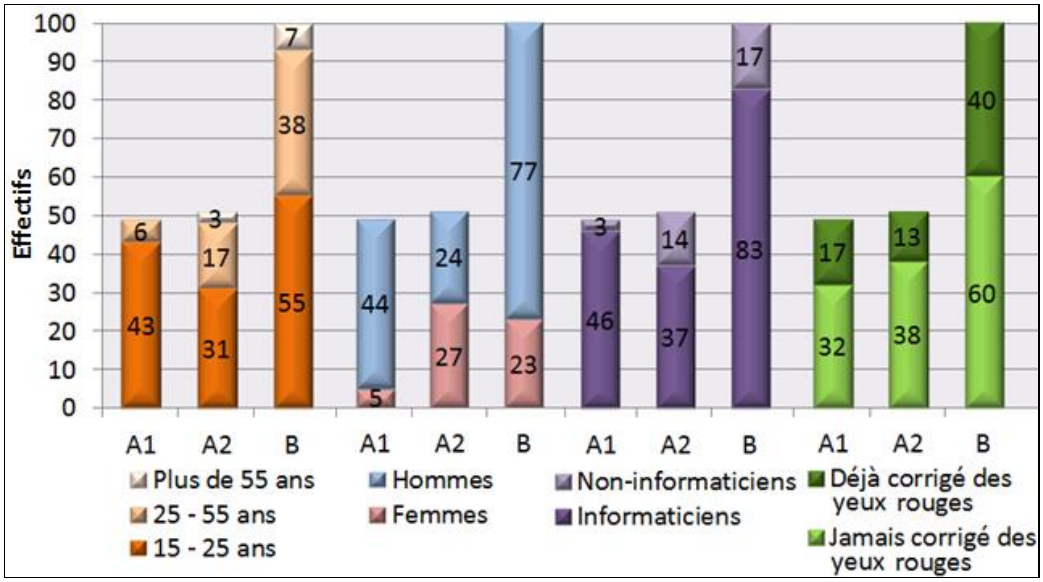


Fig. 11-8. Statistiques relatives aux utilisateurs de l'expérimentation avec PhotoScape

L'expérimentation était précédée d'un questionnaire (cf. Fig. 32 à Fig. 34 en Annexe E-1), visant à obtenir les informations qui ont permis d'établir les statistiques présentées en Fig.

11-8. L'expérimentation était également suivie d'un questionnaire (cf. Fig. 32 à Fig. 34 en Annexe E-1) visant à connaître la satisfaction des utilisateurs vis-à-vis de la tâche de correction des yeux rouges sur une photo avec PhotoScape et de l'assistance proposée le cas échéant.

C. Résultats de l'expérimentation

Les Fig. 11-9 à Fig. 11-12 présentent une partie des résultats de cette expérimentation.

	Effectif	Tâche sans assistance		Tâche avec assistance	
		Taux de réussite	Durée moyenne	Taux de réussite	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

Fig. 11-9. Résultats relatifs à l'expérimentation avec PhotoScape

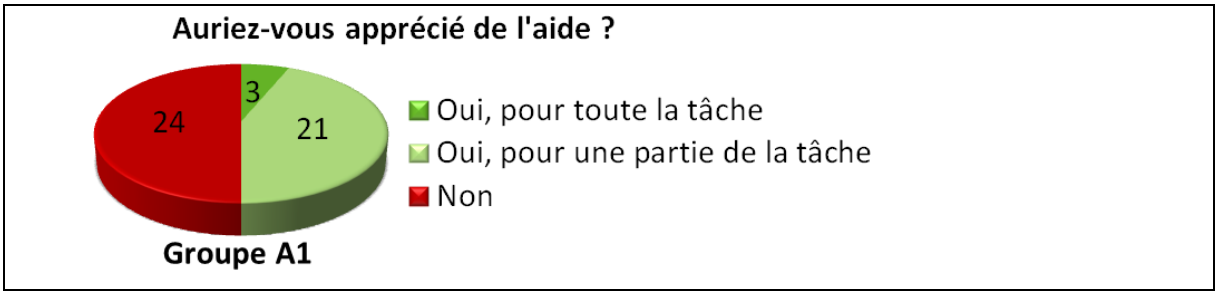


Fig. 11-10. Résultats d'une question consécutive à l'expérimentation avec PhotoScape, pour le groupe A1

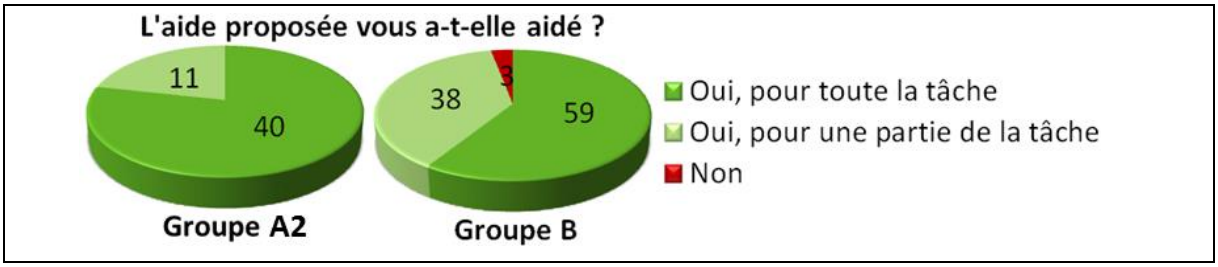


Fig. 11-11. Résultats d'une question consécutive à l'expérimentation avec PhotoScape, pour les groupes A2 et B

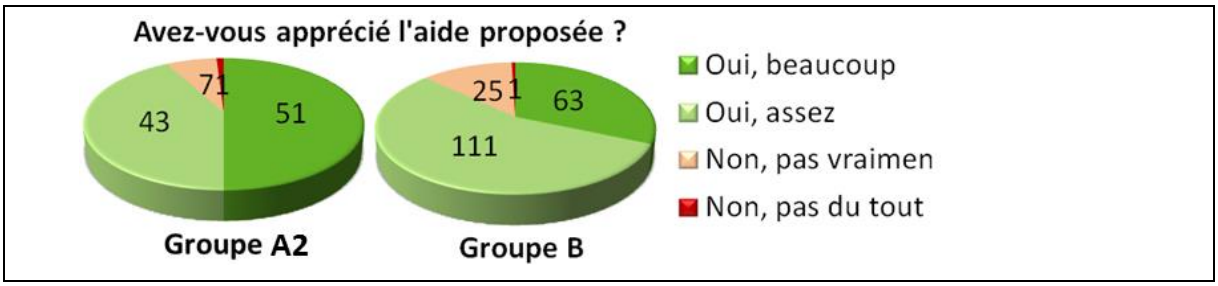


Fig. 11-12. Résultats d'une question consécutive à l'expérimentation avec PhotoScape, pour les groupes A2 et B

D. Analyse des résultats

Pour les utilisateurs du groupe A2, un besoin d'assistance a été démontré, puisque ces utilisateurs ont essayé sans succès de corriger les yeux rouges sur une photo et ont préféré abandonner. De plus, à la suite de leur abandon, tous les utilisateurs du groupe A2 ont réussi à réaliser la tâche demandée avec le système d'assistance que nous leur avons proposée. Ce succès démontre l'efficacité de l'assistance proposée pour répondre à un besoin d'assistance avéré pour les 51 utilisateurs du groupe A2. Tous ont d'ailleurs jugé utile l'assistance proposée, pour toute la tâche ou pour une partie de la tâche (cf. Fig. 11-11).

Les 100 utilisateurs du groupe B ont également tous réussi à corriger les yeux rouges sur la photo fournie en utilisant l'assistance proposée. Contrairement aux utilisateurs du groupe A2, les utilisateurs du groupe B n'ont pas essayé dans un premier temps de réaliser la tâche demandée sans assistance. En conséquence, le besoin d'assistance de ces utilisateurs n'est pas avéré. Néanmoins, dans le questionnaire consécutif à l'expérimentation, 97 % des utilisateurs du groupe B ont jugé que l'assistance proposée leur avait été utile pour toute la tâche ou pour une partie de la tâche (cf. Fig. 11-11).

Nous avons par ailleurs constaté que l'aide proposée permet de réaliser la tâche demandée de manière deux fois plus rapide en moyenne. En effet, les utilisateurs du groupe A1 ont réalisé la tâche demandée sans assistance en 141 secondes en moyenne, alors que les utilisateurs du groupe A2 et B l'ont réalisée respectivement en 72 secondes et 65 secondes en moyenne (cf. Fig. 11-9). Cette différence significative peut être une conséquence de l'interface de PhotoScape qui est peu intuitive. En effet, parmi les 49 utilisateurs du groupe A1, qui ont pourtant réussi sans assistance, 24 utilisateurs (soit environ 49 %) auraient souhaité bénéficier d'aide pour réaliser cette tâche (cf. Fig. 11-10). Ces utilisateurs ont précisé dans leurs commentaires qu'une assistance leur aurait été utile principalement pour trouver plus rapidement la zone de PhotoScape qui permet la correction des yeux rouges. Par ailleurs, les utilisateurs des groupes A2 et B ont jugé que le point fort de l'assistance proposée est qu'elle les a guidés efficacement pour ouvrir la photo fournie en mode édition, puis pour corriger les yeux rouges.

Par ailleurs, l'assistance a été appréciée par les utilisateurs l'ayant testée, on note en effet que 92 % des utilisateurs du groupe A2 et 87 % des utilisateurs du groupe B l'ont appréciée (cf. Fig. 11-12). Ces résultats très satisfaisants nous conduisent à considérer que le système SEPIA permet de fournir aux utilisateurs finaux d'applications-cibles une assistance à la fois

pertinente et efficace pour répondre aux besoins des utilisateurs, tout en étant bien acceptée par ces derniers.

11.4.2. Utilisation de NetBeans dans un contexte éducatif

A. Description de l'expérimentation

Nous avons réalisé une expérimentation en contexte éducatif, dans le cadre d'un cours sur l'ergonomie et les Interactions Homme-Machine, en 3ème année de licence d'informatique à l'Université Lyon 1, nécessitant l'utilisation de l'application NetBeans. Cette expérimentation avait pour objectif, d'une part d'évaluer la couverture d'aLDEAS et l'utilisabilité de l'éditeur d'assistance de SEPIA pour produire un système d'assistance qui réponde aux souhaits du concepteur d'assistance, et d'autre part d'évaluer l'efficacité et l'acceptabilité de l'assistance fournie aux utilisateurs finaux.

Dans ce cours, les étudiants doivent notamment réaliser des applications graphiques. Aucun langage de programmation ni IDE (Integrated Development Environnement) n'est imposé aux étudiants. Néanmoins, chaque semestre, les enseignants ont constaté que la quasi-totalité des étudiants choisissent d'utiliser l'IDE NetBeans et de programmer en langage Java. Bien que la programmation Java ne soit pas évaluée dans le cadre de ce cours d'IHM, les enseignants ont constaté que le niveau très hétérogène des étudiants, leurs lacunes en programmation Java et leur faible connaissance de NetBeans ralentissent la progression des étudiants, en particulier lors du premier TP.

Afin d'aider les étudiants qui en ont besoin à prendre en main NetBeans et à acquérir des bases de la programmation Java pour la création d'applications graphiques, les enseignants ont mis à disposition des étudiants des vidéos de démonstration dans lesquelles NetBeans est utilisé pour développer différents composants d'une interface. En complément de ces vidéos de démonstration, les enseignants ont souhaité mettre en place un tutoriel intégré à NetBeans. Ce tutoriel présenterait les mêmes notions pédagogiques que les vidéos de démonstration, mais il permettrait aux étudiants d'agir directement dans NetBeans pour créer une interface graphique simple.

B. Déroulement de l'expérimentation

Dans un premier temps, 4 concepteurs d'assistance ont utilisé l'éditeur d'assistance de SEPIA pour définir un tutoriel pour l'application-cible NetBeans. Ce tutoriel[17] est composé de 5 parties qui correspondent aux principaux besoins d'assistance des étudiants identifiés par les enseignants. Le première partie du tutoriel assiste la création avec NetBeans d'un projet Java contenant une fenêtre et permet une prise en main facilitée de l'IDE. La deuxième partie du tutoriel concerne les propriétés d'un projet NetBeans et permet de réaliser un exercice impliquant certaines propriétés du projet. La troisième partie du tutoriel présente les

[17] Vidéos de démonstration du tutoriel exécuté dans NetBeans disponibles à http://liris.cnrs.fr/blandine.ginon/PhDWork.html

fonctionnalités de NetBeans pour l'ajout de composants dans une fenêtre et propose un exercice aboutissant à la création d'un bouton respectant certaines spécificités. La quatrième partie du tutoriel concerne la création d'un menu comportant des raccourcis clavier. Enfin, la cinquième partie du tutoriel propose des exercices visant à apprendre aux étudiants à manipuler les événements ayant lieu à l'interface d'une application graphique.

Chaque partie du tutoriel contient en moyenne 14 règles et 23 actions d'assistance, principalement des messages contenant des explications ou des instructions, accompagnés de mises en valeur des composants de l'interface de NetBeans concernés par ces messages. Pour concevoir chaque partie du tutoriel, chacun des concepteurs d'assistance a utilisé l'éditeur d'assistance de SEPIA de manière autonome pendant environ 3 heures. Bien qu'il s'agisse d'un travail conséquent, ce travail pourra être réutilisé pour les prochains semestres.

Dans un deuxième temps, les enseignants ont proposé aux étudiants qui le souhaitaient de suivre le tutoriel au début du premier TP du semestre. Les étudiants étaient répartis en 5 salles avec un enseignant accompagné pour l'occasion d'un membre du projet AGATE, à titre d'observateur. Parmi les 85 étudiants présents, 64 ont été volontaires pour suivre le tutoriel. Par manque d'ordinateurs sur lesquels SEPIA était installé, seuls 52 étudiants ont pu effectivement suivre le tutoriel.

L'expérimentation était précédée par un questionnaire et un pré-test (cf. Fig. 35 en Annexe E-2) visant à connaître les connaissances préalables des étudiants en programmation Java et vis-à-vis de l'utilisation de NetBeans pour la création d'applications graphiques. L'expérimentation était également suivie d'un questionnaire de satisfaction et d'un post-test (cf. Fig. 37 en Annexe E-2) visant à évaluer la progression des étudiants suite à leur séance de travail sur le tutoriel. Ces questionnaires étaient anonymes et les étudiants avaient été assurés de la totale indépendance de ces questionnaires avec le cours.

Notons que 61,5 % des étudiants n'avaient jamais utilisé NetBeans précédemment (cf. Fig. 11-13). Pour les 20 étudiants qui avaient déjà utilisé NetBeans, seuls 12 avaient déjà utilisé NetBeans pour créer une interface graphique (cf. Fig. 11-13). Enfin, 84,6 % des étudiants avaient déjà suivi ou suivaient actuellement un cours de programmation Java (cf. Fig. 11-13). Malgré leurs connaissances préalables, ces étudiants ont souhaité suivre le tutoriel qui leur était proposé.

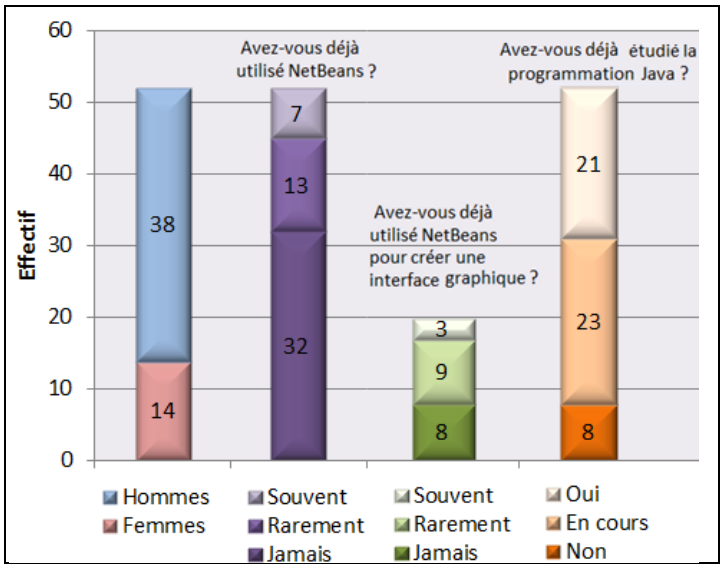


Fig. 11-13. Statistiques sur les participants à l'expérimentation avec NetBeans

La séance de travail avec le tutoriel a duré entre 30 et 90 minutes selon les étudiants. En effet, ils étaient libres de travailler avec le tutoriel de la manière qu'ils souhaitaient : ils pouvaient suivre les 5 parties, sauter les parties qui ne leur paraissaient pas utiles pour eux, passer autant de temps qu'ils le souhaitaient sur chaque partie et revenir sur une partie au besoin.

Pendant la séance de travail sur le tutoriel, les étudiants pouvaient échanger entre eux et demander des informations complémentaires à leur enseignant. Nous avons pu observer que certains étudiants ayant choisi de ne pas suivre une partie d'un tutoriel ont changé d'avis après en avoir parlé avec un étudiant ayant suivi cette partie. De plus, certains étudiants ont comparé entre eux les techniques apprises pendant le tutoriel avec les techniques qu'ils connaissaient ou qu'ils utilisaient précédemment. Deux étudiants ont ainsi précisé dans le questionnaire final que le tutoriel leur avait appris à utiliser la vue *design* de NetBeans pour la conception d'une interface graphique, ce qui leur semble plus simple que la technique qu'ils connaissaient et qui consiste à créer toute l'interface dans la vue *code* de NetBeans.

C. Résultats de l'expérimentation

Les Fig. 11-14 à Fig. 11-17 présentent les résultats de cette expérimentation.

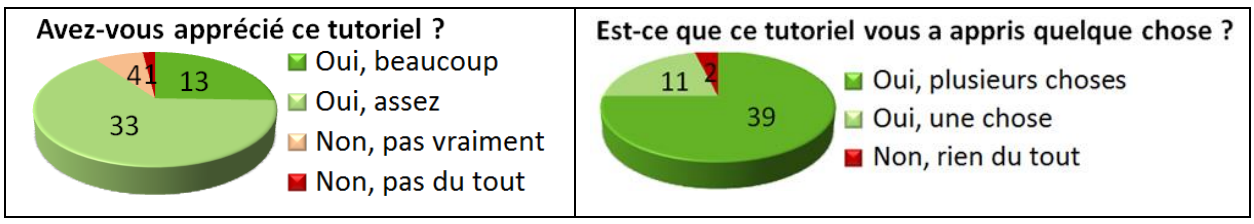


Fig. 11-14. Questionnaire final : satisfaction des étudiants

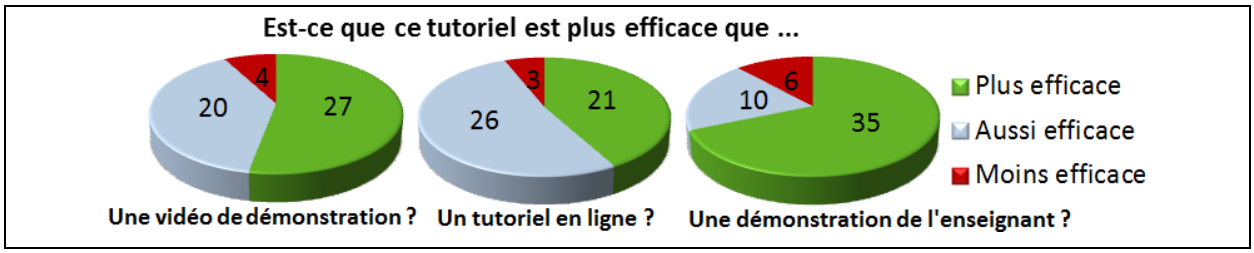


Fig. 11-15. Questionnaire final : efficacité du tutoriel perçue par les étudiants

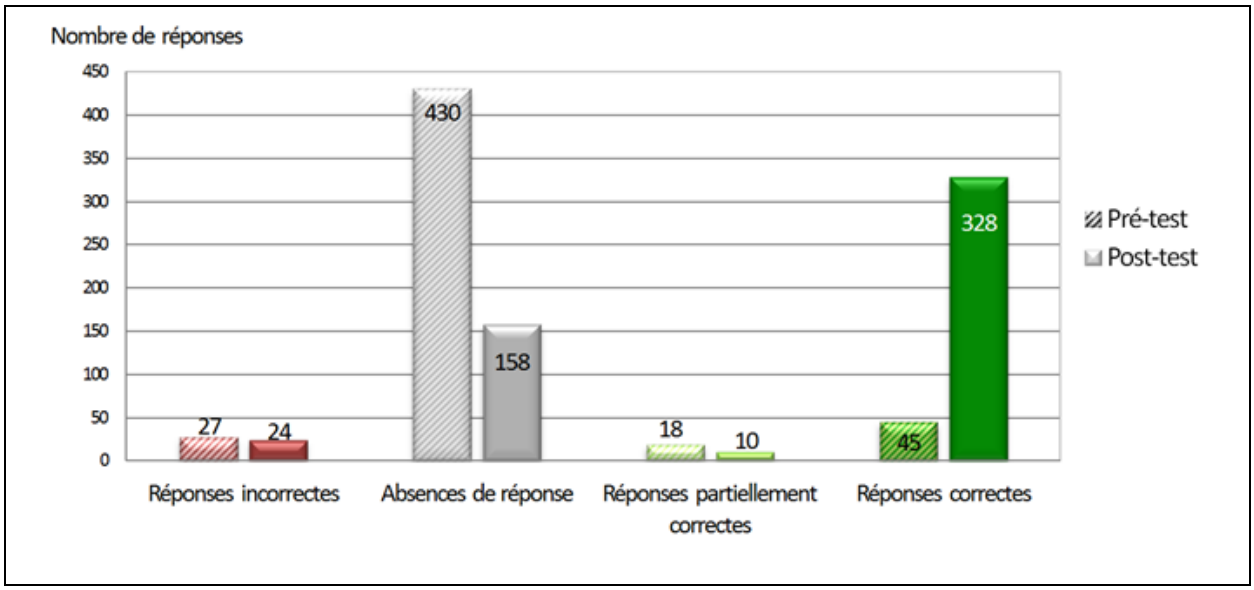


Fig. 11-16. Comparaison des résultats des 52 étudiants aux 10 questions communes au pré-test et au post-test

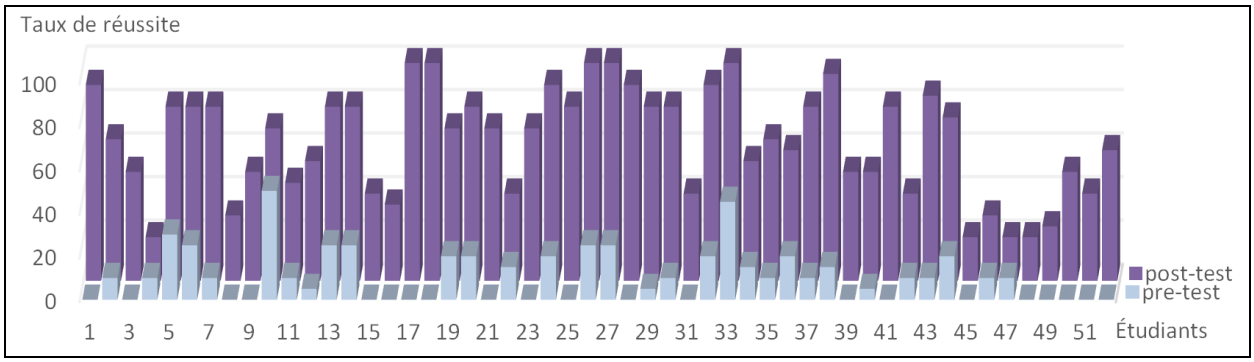


Fig. 11-17. Taux de succès des 52 étudiants aux 10 questions communes au pré-test et au post-test

D. Analyse des résultats

Les étudiants ont été très satisfaits et enthousiastes vis-à-vis du tutoriel : 90,4 % d'entre eux ont déclaré l'avoir apprécié (cf. partie gauche de la Fig. 11-14) et 62 % souhaitent suivre d'autres tutoriels de ce type intégrés à NetBeans. En particulier, dans leurs commentaires dans le questionnaire final, 62 % des étudiants ont demandé d'autres tutoriels intégrés à NetBeans, mais présentant des notions plus avancées. De plus, 32 % des étudiants ont également demandé à suivre des tutoriels de ce type intégrés à d'autres application-cible,

notamment les IDE utilisés dans d'autres cours, ou des outils qu'ils souhaitent utiliser dans leur vie courante, tels que Photoshop ou Gimp. Par ailleurs, 53 % des étudiants ont jugé ce tutoriel plus efficace qu'une vidéo de démonstration, et 69 % l'ont jugé plus efficace qu'une démonstration de l'enseignant (cf. Fig. 11-15). En revanche, 52 % des étudiants ont jugé l'efficacité du tutoriel équivalente à celle d'un tutoriel en ligne (cf. Fig. 11-15).

Le tutoriel semble avoir facilité l'acquisition de connaissances en jeu pour le cours d'IHM. En effet, 96,2 % des étudiants ont déclaré que le tutoriel leur avait appris quelque chose (cf. partie droite de la Fig. 11-14). Cet excellent résultat est confirmé par l'amélioration des connaissances des étudiants entre le pré-test et le post-test (cf. Fig. 11-17). Ces deux tests contenaient notamment 10 questions communes à réponse libre pour lesquelles certains résultats sont présentés en Fig. 11-16 et Fig. 11-17. Notons que le nombre de questions sans réponse a fortement baissé entre le pré-test et le post-test (- 63 %) et que le nombre de bonnes réponses a au contraire très fortement augmenté (+ 629%) (cf. Fig. 11-16).

Chaque étudiant a amélioré son taux de réussite entre le pré-test et le post-test, la plus faible progression étant de + 10% et la plus haute de + 100%, pour les étudiants qui n'avaient aucune réponse correcte au pré-test et toutes les réponses correctes au post-test. Il est à noter qu'aucun étudiant qui avait une réponse correcte au pré-test n'a eu de réponse incorrecte à la même question au post-test, ce qui tend à montrer que le tutoriel n'a pas altéré de connaissances des étudiants. Néanmoins, nous avons constaté quelques réponses incorrectes au post-test, pour des étudiants qui n'avaient pas proposé de réponse au pré-test pour cette même question. Certains étudiants ont en effet confondu dans le post-test la notion de *backgroung color* (la couleur de fond d'un composant) et la notion de *foreground color* (la couleur de la police d'un composant) : ces deux notions sont présentées dans le tutoriel. L'évolution des connaissances des étudiants est néanmoins très satisfaisante, avec une progression moyenne de + 53,65 %. Afin de vérifier si les étudiants étaient attentifs uniquement sur les parties du tutoriel traitant des questions incluses dans le pré-test, le post-test contenait également 3 nouvelles questions. Le taux de réussite moyen des étudiants pour ces questions est de 62 %, ce qui est équivalent à leur taux de réussite moyen pour les questions communes aux deux tests, qui est de 64 %.

Pour finir, les enseignants ont été particulièrement satisfaits de l'utilisation du tutoriel et prévoient de l'utiliser à nouveau lors des prochains semestres. Ils souhaitent également compléter le tutoriel en lui ajoutant des parties traitant de connaissances plus avancées, réclamées par les étudiants dans leurs commentaires au questionnaire final. Si ces connaissances ne sont pas nécessaires pour le premier TP d'IHM, elles peuvent l'être pour la suite du semestre et pour d'autres cours suivis par les étudiants qui nécessitent également des connaissances en programmation Java.

Chapitre 12. Synthèse des évaluations

Objectif du chapitre

Synthétiser les évaluations de nos contributions théoriques et leur mise en œuvre à travers le système SEPIA.

Points clés

Nous présentons la synthèse des évaluations de nos contributions, afin de faire le point sur la validation de nos travaux. Concernant la **pertinence de notre approche**, elle est validée d'une part par l'existence d'un besoin d'assistance pour les utilisateurs finaux d'applications-cibles, et d'autre part par l'existence, pour des concepteurs d'assistance potentiels, d'un désir de mettre en place de l'assistance dans des applications-cibles.

La **faisabilité** de cette approche est démontrée par sa mise en œuvre opérationnelle à travers le système SEPIA.

Concernant le langage **aLDEAS**, sa couverture est validée par la diversité des systèmes d'assistance qu'il permet de spécifier ; son utilisabilité par des concepteurs d'assistance et son intérêt pour faciliter la spécification de systèmes d'assistance sont démontrés par les expérimentations réalisées.

Concernant la **surveillance de l'application-cible**, la couverture de nos épi-détecteurs est partiellement validée : bien que de très nombreuses applications-cibles puissent d'ores et déjà être surveillées par nos épi-détecteurs, certaines catégories d'applications nécessitent le développement de nouveaux épi-détecteurs et quelques applications nécessitent le recours à une technique de surveillance alternative à celle que nous proposons.

Concernant le système **SEPIA**, l'utilité et l'utilisabilité de l'éditeur de SEPIA sont démontrées par les expérimentations menées, bien qu'une analyse ergonomique puisse permettre d'améliorer son interface. La capacité de SEPIA à exécuter une assistance spécifiée avec aLDEAS est validée par les expérimentations menées.

Concernant l'**assistance fournie aux utilisateurs finaux**, la capacité de nos contributions à mettre en place une assistance à la fois efficace et bien acceptée est démontrée par les expérimentations menées.

Concernant la **personnalisation de l'assistance**, un désir des concepteurs d'assistance de mettre en place une assistance personnalisée est démontré par les expérimentations menées, mais d'autres expérimentations doivent être réalisées pour démontrer la pertinence de la personnalisation pour les utilisateurs finaux.

Dans ce chapitre, nous revenons sur les critères d'évaluations exposés dans le Chapitre 10, afin de présenter l'évaluation de chacun de ces critères en fonction de l'analyse des expérimentations détaillées dans le Chapitre 11. Pour évaluer les différents critères, nous utilisons l'échelle présentée en Fig. 12-1.

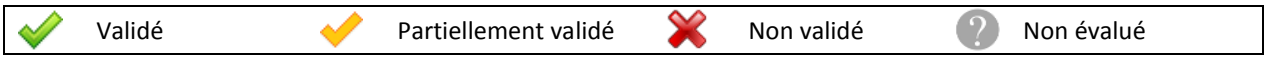


Fig. 12-1. Échelle d'évaluation utilisée

12.1. Pertinence de l'approche

Cette section présente l'évaluation des différentes facettes de la pertinence de l'approche proposée, présentées de manière synthétique en Fig. 12-2.

Facettes à évaluer	Évaluation menée	Validation
Existence d'un besoin d'assistance	Questionnaire auprès d'utilisateurs finaux durant les expérimentations avec PhotoScape et NetBeans	✓
	Expérimentation avec PhotoScape : réalisation sans assistance d'une tâche de correction des yeux rouges sur une photo	✓
Existence d'un désir de mettre en place de l'assistance	Questionnaire auprès de concepteurs d'assistance potentiels	✓
Faisabilité de l'approche	Implémentation de l'approche proposée	✓
Réutilisabilité des propositions	Exemples de réutilisations possibles	✓

Fig. 12-2 – Validation de la pertinence de l'approche

12.1.1. Existence d'un besoin d'assistance

L'existence d'un besoin d'assistance pour les utilisateurs finaux est démontrée de plusieurs manières. Tout d'abord, lors de l'expérimentation avec PhotoScape, présentée en section 11.4.1, nous avons demandé à 100 personnes de corriger des yeux rouges sur une photo avec PhotoScape sans assistance. Dans ce groupe, 51 personnes n'ont pas réussi à réaliser la tâche demandée, ce qui met en évidence un besoin d'assistance. De plus, pour les 49 personnes qui ont réussi à réaliser la tâche demandée sans assistance, 24 ont précisé qu'ils auraient tout de même souhaité recevoir de l'assistance.

Ensuite, dans les questionnaires consécutifs aux expérimentations avec PhotoScape et avec NetBeans, nous avons demandé aux participants s'ils aimeraient profiter d'assistance dans d'autres applications que PhotoScape ou NetBeans. Beaucoup de participants ont répondu positivement et ont suggéré des applications dans lesquelles ils souhaiteraient bénéficier d'assistances semblables à celle qu'ils ont reçue dans le cadre de l'expérimentation, comme Éclipse, Dr Racket ou CodeBlocks, trois applications également

utilisées dans le cadre de la formation des étudiants de licence d'informatique à l'Université Lyon 1.

12.1.2. Existence d'un désir de mettre en place de l'assistance

L'existence d'un désir de mettre en place de l'assistance dans des applications existantes est démontrée par les sollicitations de personnes souhaitant utiliser SEPIA. En effet, suite à l'expérimentation avec PhotoScape (cf. section 11.4.1), qui a fait connaître plus largement le système SEPIA, plusieurs personnes nous ont contactés pour mettre en place avec SEPIA des systèmes d'assistance dans diverses applications-cibles, allant de l'assistance pour l'utilisation de Skype par des personnes âgées à la découverte d'applications complexes issues de la recherche.

Ainsi, le projet pluridisciplinaire ONDES (Opportunités du Numérique pour le Développement pédagogique des Enseignants du Supérieur) qui débutera à l'automne 2014, vise à fournir des outils numériques pour soutenir la réussite des étudiants. Il exploitera les outils de SEPIA pour mettre en place de l'assistance à l'utilisation d'@LOES (Assistant en Ligne pour l'Opérationnalisation de l'Enseignement dans le Supérieur) [Dufour, 2007], un éditeur de plans de cours. SEPIA permettra également de mettre en place une assistance pluriculturelle pour prendre en compte les différents contextes dans lesquels cet outil est utilisé en France et au Québec, notamment au niveau de l'organisation et du vocabulaire employé.

De plus, l'expérimentation du tutoriel pour NetBeans utilisé dans le contexte d'un cours de licence (cf. section 11.4.2) a été mise en place en collaboration avec les enseignants responsables de ce cours. Ces enseignants ont été très satisfaits de ce tutoriel et prévoient de l'utiliser à nouveau lors des prochains semestres. Suite à cette expérimentation en contexte éducatif, d'autres enseignants responsables d'autres cours envisagent l'utilisation de SEPIA pour mettre en place des tutoriels pour leurs étudiants. Ainsi, un tutoriel sera mis en place pour faciliter la prise en main des outils de gestion de projet, tels que SVN (SubVersioN) et la forge utilisée au département informatique de l'Université Lyon 1. L'utilisation de ces outils est enseignée aux étudiants de deuxième année de licence. Cependant, certains étudiants de master 1 n'ayant pas suivi cette formation pourraient également bénéficier d'un tel tutoriel.

12.1.3. Faisabilité de l'approche

La faisabilité de notre approche est démontrée par l'implémentation qui en a été faite dans le système SEPIA, qui est pleinement opérationnel. De plus, le système SEPIA a été utilisé avec succès par de nombreux concepteurs d'assistance, pour mettre en place *a posteriori* des systèmes d'assistance épiphytes, dans des applications-cibles variées, sans les avoir modifiées et sans avoir eu recours à la programmation.

12.1.4. Réutilisabilité des propositions

Certaines propositions faites dans le cadre de cette thèse peuvent être réutilisées dans un contexte autre que celui du projet AGATE.

Tout d'abord, le langage aLDEAS peut être utilisé par des concepteurs d'assistance pour faciliter la définition de systèmes d'assistance, comme l'a démontré l'expérimentation de 2014 avec des concepteurs d'assistance informaticiens. En effet, 67 % des participants ont considéré qu'aLDEAS facilite la définition de systèmes d'assistance. Certains participants ont également précisé dans leurs commentaires consécutifs à cette expérimentation qu'ils trouvaient que l'utilisation d'aLDEAS permet de travailler plus facilement à plusieurs pour définir un système d'assistance.

Par ailleurs, le processus de surveillance d'une application-cible (cf. Chapitre 3) et sa mise en œuvre à travers les épi-détecteurs (cf. section 8.4) peuvent être utilisés avec un autre objectif que celui de mettre en place des systèmes d'assistance. De nombreuses utilisations des traces d'interactions entre un utilisateur et une application-cible sont en effet possibles. Tout d'abord, de telles traces d'interactions peuvent être utilisées par des applications de visualisation dans un but réflexif. Par exemple, PicTop [Terrat et Sagot, 2011 ; INS_HEA, 2014] est un logiciel à destination des enfants handicapés qui favorise l'apprentissage de la lecture. Il propose notamment un exercice dans lequel l'enfant visualise la trace laissée par une activité précédente, dans le but d'analyser cette activité. La visualisation de traces peut également concerner des personnes autres que celles à l'origine de ces traces. Ainsi, les traces peuvent être visualisées à des fins statistiques, ou pour la découverte de connaissances. Les épi-détecteurs que nous avons développés peuvent donc être utilisés indépendamment de SEPIA et du projet AGATE, afin de permettre la collecte de traces dans de nombreuses applications non conçues pour permettre une telle collecte.

12.2. Le langage aLDEAS

Cette section présente l'évaluation des différentes facettes du langage aLDEAS, présentées de manière synthétique en Fig. 12-3.

Facettes à évaluer	Évaluation menée	Validation
Couverture d'aLDEAS	Utilisation d'aLDEAS pour définir des assistances existantes	✓
	Expérimentation avec des concepteurs d'assistance : définition de systèmes d'assistance à l'aide d'aLDEAS	✓
Utilisabilité d'aLDEAS	Expérimentation avec des concepteurs d'assistance : définition de systèmes d'assistance avec aLDEAS	✓
Intérêt d'aLDEAS pour faciliter la spécification de systèmes d'assistance	Expérimentation avec des concepteurs d'assistance : définition de systèmes d'assistance avec ou sans aLDEAS	✓
Utilité des patrons d'aLDEAS	Utilisation de ces patrons pour définir des assistances existantes	✓

Fig. 12-3 – Validation du langage aLDEAS

12.2.1. Couverture d'aLDEAS

La capacité d'aLDEAS à permettre la définition de systèmes d'assistance variés est démontrée par les systèmes d'assistance définis avec aLDEAS (cf. Annexe A et Annexe C). Ces systèmes d'assistance concernent des applications-cibles variées, dans différents domaines. De plus, les systèmes définis avec aLDEAS peuvent être utilisés dans différents contextes et avec des objectifs divers. Par exemple, le tutoriel NetBeans a été expérimenté dans un contexte éducatif pour faciliter l'acquisition de connaissances pédagogiques, alors que le système d'assistance pour la correction des yeux rouges dans PhotoScape est destiné à un usage personnel de loisir. aLDEAS permet également la définition de systèmes d'assistance à destination d'utilisateurs finaux très divers : il peut s'agir d'enfants, d'adultes, qu'ils soient experts dans l'utilisation d'outils informatiques ou novices, de personnes âgées ou de personnes en situation de handicap.

12.2.2. Utilisabilité d'aLDEAS

L'utilisabilité d'aLDEAS est démontrée par l'expérimentation menée en 2014 avec des concepteurs d'assistance informaticiens (cf. Section 11.3.2). En effet, 93 % des participants ont jugé aLDEAS facile à utiliser et tous ont réussi à définir en aLDEAS un système d'assistance pour leur logiciel pédagogique. Cette expérimentation est complétée par de premiers tests avec des concepteurs d'assistance non-informaticiens (cf. Section 11.3.3), dont les résultats encourageants devront néanmoins être confirmés par une expérimentation plus vaste avec ce type d'utilisateurs. En effet, des concepteurs d'assistance non-informaticiens ont réussi à définir un système d'assistance avec aLDEAS, et une seule participante, une enfant de 11 ans, a jugé aLDEAS difficile à utiliser. De plus, pour la définition d'un système d'assistance contenant 8 à 9 règles, ces concepteurs d'assistance ont travaillé pendant 30 à 50 minutes, ce qui semble très raisonnable pour une prise en main

d'un langage inconnu et du concept de règle d'assistance. Par ailleurs, l'hétérogénéité de ces utilisateurs, avec notamment une enfant de 11 ans et un retraité ayant une très faible expérience de l'utilisation d'un ordinateur, montre l'étendue du spectre des utilisateurs potentiels d'aLDEAS et de SEPIA.

12.2.3. Intérêt d'aLDEAS pour faciliter la spécification de systèmes d'assistance

L'intérêt d'aLDEAS pour faciliter la spécification de systèmes d'assistance est démontré par l'expérimentation menée en 2014 avec des concepteurs d'assistance informaticiens (cf. Section 11.3.2). En effet, 76 % des participants ont jugé qu'aLDEAS les avait aidés à définir leur système d'assistance. De plus, les cinq premiers concepteurs d'assistance non-informaticiens auprès de qui nous avons testé aLDEAS ont également réussi à définir des systèmes d'assistance avec aLDEAS (cf. Section 11.3.3). L'expérimentation ne montre pas s'ils auraient également réussi sans aLDEAS, cependant, les participants ont jugé qu'aLDEAS les avait beaucoup aidés dans leur tâche de conception d'assistance.

12.2.4. Utilité des patrons associés à aLDEAS

L'utilité des patrons d'assistance qui complètent le langage aLDEAS est démontrée tout d'abord par l'utilisation qui en a été faite pour redéfinir des assistances existantes. Prenons l'exemple de l'assistance pour le jeu en ligne « Dora l'exploratrice : le spectacle de fin d'année » [Dora_Jeux, 2014], destiné à de jeunes enfants, dont la définition en aLDEAS est donnée en annexe (cf. Fig. 7, Annexe A-1). Pour définir cette assistance avec aLDEAS sans utiliser de patron, il est nécessaire de définir 26 règles d'assistance, alors qu'en utilisant le patron de présentation guidée, une seule règle d'assistance est nécessaire.

Par ailleurs, les concepteurs d'assistance ayant utilisé aLDEAS ont souligné l'intérêt de ces patrons pour faciliter la définition d'assistances répétitives. Plus particulièrement, quatre des cinq concepteurs d'assistance non-informaticiens ont spontanément déclaré que le patron de règles d'assistance facilite fortement la définition de l'assistance. Selon eux, en imposant un découpage de l'assistance étape par étape, le patron de règles d'assistance guiderait leur travail. L'un de ces concepteurs d'assistance a de plus précisé que le patron de règles rassurerait le concepteur de l'assistance, qui pourrait être effrayé par la richesse d'aLDEAS.

12.3. Surveillance de l'application-cible

Cette section présente l'évaluation des différentes facettes de la surveillance de l'application-cible, présentées de manière synthétique en Fig. 12-4.

Facettes à évaluer	Évaluation menée	Validation
Couverture de nos épi-détecteurs	Classification des applications-cibles et correspondance avec nos épi-détecteurs	
Détection des composants d'une application-cible	Vérification de l'arborescence des composants détectée par nos épi-détecteurs pour un ensemble d'applications-cibles variées	
Détection d'événements dans une application-cible	Expérimentation avec nos épi-détecteurs	
Identification de la source d'un événement	Analyse de notre technique d'identification de la source d'un événement	

Fig. 12-4 – Validation de la surveillance de l'application-cible

12.3.1. Couverture de nos épi-détecteurs

Nous avons développé un ensemble d'épi-détecteurs qui permettent la surveillance de la majorité des applications existantes. Néanmoins, il reste certaines catégories d'applications-cibles pour lesquelles nous devons développer d'autres épi-détecteurs, avec un mode de fonctionnement similaire aux épi-détecteurs actuellement développés, mais impliquant une autre bibliothèque d'accessibilité. C'est le cas pour les toutes les applications de type bureau sous Mac OS, pour les applications GTK ou Qt sous Windows et pour les applications Java sous Linux.

Par ailleurs, nous n'avons pas actuellement de solution pour permettre la surveillance des applications web en Flash. En effet, contrairement aux autres applications web, les applications Flash ne sont pas suffisamment ouvertes pour permettre le fonctionnement de scripts utilisateurs. Une solution pour pallier ce problème serait de recourir à des techniques de reconnaissances visuelles, telles que celles proposées par [Dixon et Fogarty, 2010] (cf. Section 3.3).

12.3.2. Détection des composants d'une application-cible

Pour les applications compatibles, nos épi-détecteurs accèdent généralement à la totalité de la hiérarchie des composants d'une application. Il existe cependant certains composants spécifiques qui ne sont pas détectés par nos épi-détecteurs, notamment ceux qui ont été conçus sans propriété d'accessibilité (cf. Section 11.2).

12.3.3. Détection d'événements dans une application-cible

Pour les composants auxquels ils accèdent, nos épi-détecteurs détectent une très large variété d'événements (cf. Fig. 9-1). Les différentes expérimentations que nous avons menées attestent de la fiabilité de nos épi-détecteurs, notamment l'exécution avec succès d'une assistance pour 151 utilisateurs finaux de PhotoScape (cf. Section 11.4.1) et 52 utilisateurs finaux de NetBeans (cf. Section 11.4.2).

Néanmoins, il est à noter que l'installation de SEPIA peut se révéler compliquée et longue pour assurer un bon fonctionnement de nos épi-détecteurs. Il est en effet indispensable d'installer ou d'activer les bibliothèques d'accessibilité impliquées, et d'effectuer différentes mises à jour du système sous Windows si celles-ci n'ont pas déjà été effectuées, avant que SEPIA ne soit pleinement opérationnel. Un gros travail de maintenance du système SEPIA sera également nécessaire pour assurer son bon fonctionnement avec les évolutions des systèmes d'exploitation.

12.3.4. Identification de la source d'un événement

L'analyse de notre technique d'identification a montré que nos épi-détecteurs sont capables d'identifier avec exactitude la source de tout événement détecté. Nos épi-détecteurs utilisent en effet la position d'un composant dans la hiérarchie des composants de l'interface d'une application : cette position est unique et constitue en conséquence un bon identifiant.

12.4. Le système SEPIA

Cette section présente l'évaluation des différentes facettes du système SEPIA, présentées de manière synthétique en Fig. 12-5.

Facettes à évaluer	Évaluation menée	Validation
Utilité de l'éditeur d'assistance de SEPIA	Expérimentation avec des concepteurs d'assistance : utilisation de l'éditeur de SEPIA	✓
Utilisabilité de l'éditeur d'assistance de SEPIA	Évaluation ergonomique	✓
	Expérimentation avec des concepteurs d'assistance : utilisation de l'éditeur de SEPIA	✓
Exécution de l'assistance avec SEPIA	Expérimentation : définition puis exécution de systèmes d'assistance avec SEPIA	✓
	Expérimentation avec des concepteurs d'assistance : définition puis exécution de systèmes d'assistance avec SEPIA	✓

Fig. 12-5 – Validation du système SEPIA

12.4.1. Utilité de l'éditeur d'assistance de SEPIA

L'utilité de l'éditeur d'assistance de SEPIA est démontrée par les différentes expérimentations que nous avons menées avec des concepteurs d'assistance. Les systèmes d'assistance générés par l'éditeur sont conformes aux attentes de leur concepteur.

12.4.2. Utilisabilité de l'éditeur d'assistance de SEPIA

En complément des évaluations ergonomiques qui ont été conduites, l'utilisabilité de l'éditeur d'assistance de SEPIA est démontrée par les différentes expérimentations que nous avons menées avec des concepteurs d'assistance. Lors de l'expérimentation menée avec des concepteurs d'assistance informaticiens en 2014, 67 % des participants ont jugé l'éditeur de SEPIA facile à utiliser pour définir des règles d'assistance et 63 % des participants ont jugé l'éditeur de SEPIA facile à utiliser pour décrire l'interface de l'application-cible. L'éditeur de SEPIA a également été utilisé avec succès par des concepteurs d'assistance non-informaticiens, qui l'ont globalement trouvé facile à utiliser.

12.4.3. Exécution de l'assistance dans SEPIA

La phase d'exécution de l'assistance avec SEPIA a été évaluée de plusieurs façons. Tout d'abord, nous avons défini puis exécuté avec succès différents systèmes d'assistance avec SEPIA (cf. Annexe C). En complément, nous avons expérimenté deux de ces systèmes d'assistance : nous avons exécuté avec SEPIA une assistance pour 151 utilisateurs finaux de PhotoScape et pour 52 utilisateurs finaux de NetBeans. Dans ces deux expérimentations, l'assistance a été exécutée conformément à l'assistance spécifiée par ses concepteurs, qui en ont été très satisfaits.

12.5. Assistance fournie aux utilisateurs finaux

Cette section présente l'évaluation des différentes facettes de l'assistance fournie aux utilisateurs finaux, présentées de manière synthétique en Fig. 12-6.

Facettes à évaluer	Évaluation menée	Validation
Acceptation de l'assistance fournie aux utilisateurs finaux	Expérimentations avec PhotoScape et NetBeans	✓
Efficacité de l'assistance fournie aux utilisateurs finaux	Expérimentations avec PhotoScape et NetBeans	✓
	Expérimentation avec PhotoScape : correction des yeux rouges avec ou sans assistance	✓
	Expérimentation avec PhotoScape : correction des yeux rouges avec assistance après un échec sans assistance	✓

Fig. 12-6 – Validation de l'assistance fournie aux utilisateurs finaux

Concernant l'assistance fournie aux utilisateurs finaux, il est important de distinguer l'évaluation de SEPIA, de l'évaluation de la pertinence de l'assistance fournie par SEPIA. Dans le cadre de cette thèse en informatique, il nous parait intéressant de démontrer que nos propositions théoriques mises en œuvre à travers le système SEPIA permettent la mise en place d'assistances pouvant être à la fois utiles et bien acceptées par les utilisateurs finaux, tout en répondant aux attentes des concepteurs d'assistance. Les expérimentations avec

PhotoScape et NetBeans montrent donc un aperçu des bénéfices que les utilisateurs finaux peuvent retirer de l'assistance fournie par SEPIA, lorsque celle-ci a été judicieusement spécifiée par le concepteur de l'assistance. En revanche, l'évaluation de la pertinence des différentes assistances qu'il est possible de mettre en place à l'aide de nos outils sort du cadre de cette thèse.

12.5.1. Acceptation de l'assistance fournie aux utilisateurs finaux

Les expérimentations menées avec PhotoScape et NetBeans ont montré que l'assistance fournie par les outils de SEPIA pouvait être très bien acceptée par les utilisateurs finaux. En effet, dans le contexte de l'expérimentation menée avec PhotoScape (cf. section 11.4.1), 98 % des 151 utilisateurs ayant reçu de l'assistance l'ont jugée utile et 79 % ont déclaré l'avoir appréciée. Dans le contexte de l'expérimentation avec NetBeans (cf. section 11.4.2), 90 % des 52 utilisateurs ayant suivi le tutoriel l'ont apprécié et 96 % considèrent que ce tutoriel leur a appris quelque chose. Beaucoup d'étudiants ont de plus demandé d'autres tutoriels de ce type.

12.5.2. Efficacité de l'assistance fournie aux utilisateurs finaux

L'efficacité de l'assistance fournie par les outils de SEPIA a été démontrée dans le cadre des expérimentations menées avec PhotoScape (cf. section 11.4.1). En effet, un besoin d'assistance a été démontré pour les 51 utilisateurs qui n'ont pas réussi à corriger des yeux rouges sur une photo. Or, 100 % de ces utilisateurs ont ensuite réussi à réaliser la tâche demandée en utilisant une assistance mise en place avec SEPIA. Tous les utilisateurs qui ont découvert PhotoScape avec l'assistance mise en place avec SEPIA ont également réussi à réaliser la tâche demandée. De plus, les utilisateurs ayant bénéficié de l'assistance ont réalisé la tâche demandée en moyenne deux fois plus vite que les autres utilisateurs.

Par ailleurs, l'expérimentation avec NetBeans semble avoir montré que l'assistance fournie par SEPIA pouvait faciliter l'acquisition de compétences par les utilisateurs finaux. En effet, 96 % des étudiants ont jugé que ce tutoriel leur avait appris quelque chose. De plus, la comparaison entre les réponses au pré-test et au post-test a montré que tous les étudiants avaient progressé, en moyenne de + 54 %. Si ces résultats sont très encourageants, nous devons toutefois noter l'absence de groupe témoin, constitué d'étudiants qui auraient également rempli les pré-test et post-test, mais n'auraient pas suivi le tutoriel proposé. Ce groupe aurait pu permettre de déterminer si l'utilisation de NetBeans sans assistance permet aux étudiants de progresser autant qu'une utilisation accompagnée du tutoriel proposé. Les enseignants ayant jugé très profitable la mise en place du tutoriel, par soucis d'équité entre les étudiants, ils ne nous ont pas permis de créer un tel groupe témoin. Notre entretien avec les enseignants à l'issue de l'expérimentation nous a toutefois donné quelques éléments. Les enseignants ont en effet trouvé que les étudiants ont été significativement plus à l'aise et plus autonomes pour ce premier TP par rapport aux séances correspondantes lors des semestres précédents.

12.6. Personnalisation de l'assistance

Cette section présente l'évaluation des différentes facettes de la personnalisation de l'assistance, présentées de manière synthétique en Fig. 12-7.

Facettes à évaluer	Évaluation menée	Validation
Pertinence de la personnalisation pour les concepteurs d'assistance	Questionnaire auprès de concepteurs d'assistance	
Pertinence de la personnalisation pour les utilisateurs finaux	Questionnaire auprès d'utilisateurs finaux	
	Expérimentation avec des utilisateurs finaux : réalisation d'une tâche avec une assistance personnalisée	

Fig. 12-7 – Validation de la personnalisation de l'assistance

12.6.1. Pertinence de la personnalisation de l'assistance pour les concepteurs d'assistance

La pertinence de la personnalisation de l'assistance du point de vue des concepteurs d'assistance a été démontrée par l'expérimentation avec des concepteurs d'assistance informaticiens en 2013 et 2014 (cf. section 11.3). En effet, 60 % des règles d'assistance définies dans ce contexte contenaient une condition de déclenchement visant à personnaliser l'assistance. De plus, ces concepteurs d'assistance ont déclaré qu'ils jugent pertinent de personnaliser l'assistance en fonction des connaissances (85 %), des handicaps (97 %) et des préférences (59 %) de l'utilisateur final. Il est néanmoins nécessaire de compléter ces résultats par une étude, d'une part auprès d'un public plus représentatif des concepteurs d'assistance, et d'autre part dans d'autres contextes.

12.6.2. Pertinence de la personnalisation de l'assistance pour les utilisateurs finaux

Nous n'avons actuellement pas réalisé d'expérimentation visant à évaluer la pertinence de la personnalisation de l'assistance pour les utilisateurs finaux. Nous faisons toutefois l'hypothèse qu'elle pourrait améliorer encore l'acceptabilité de l'assistance par les utilisateurs finaux. Ainsi, la prise en compte des connaissances des utilisateurs permettrait de leur proposer une assistance encore plus adaptée à leur besoin. Par exemple, dans le cadre de l'expérimentation avec PhotoScape, il aurait été possible de personnaliser l'assistance en fonction des connaissances des utilisateurs finaux. Nous avons en effet noté en analysant les résultats et l'historique de l'assistance que les informaticiens n'avaient pas besoin d'assistance pour repérer la photo à corriger dans l'arborescence des dossiers, alors que de nombreux non-informaticiens ont demandé plus d'aide à cette étape de la correction des yeux rouges.

De plus, nous faisons l'hypothèse que l'expression de préférences peut rendre plus attrayante l'assistance, en particulier auprès d'un jeune public : la définition de couleur, de

police ou d'agent animé préféré peut être source de motivation vis-à-vis de la tâche qui doit être réalisée avec l'assistance. Pour valider cette seconde hypothèse, nous envisageons de conduire une expérimentation auprès d'enfants, pour les assister dans la réalisation d'une activité pédagogique. Un premier groupe d'enfants bénéficierait d'une assistance personnalisée en fonction de leurs connaissances, compétences et préférences, alors que le second groupe d'enfants bénéficierait d'une assistance non personnalisée. Nous pourrions ensuite comparer les résultats et la satisfaction des utilisateurs ayant bénéficié d'une assistance personnalisée, et celle des utilisateurs n'ayant pas bénéficié d'une telle personnalisation.

12.7. Conclusion

Nous avons mené plusieurs expérimentations afin d'évaluer les différents aspects de nos contributions. Comme nous l'avons vu dans ce chapitre, la plupart des aspects à évaluer ont été validés.

Concernant les contributions théoriques, nous avons en particulier montré la faisabilité de notre approche générale de mise en place *a posteriori* d'assistance épiphyte. De plus, nous avons montré que le langage aLDEAS est utilisable pour définir des assistances variées et que son utilisation facilite le travail du concepteur de l'assistance.

Concernant la mise en œuvre de nos contributions à travers le système SEPIA, nous avons en particulier démontré que SEPIA est utilisable pour mettre en place de l'assistance dans des applications existantes variées. Nous avons par ailleurs montré que les modèles et outils génériques que nous proposons permettent à différents concepteurs d'assistance de mettre en place l'assistance spécifique qu'ils souhaitent pour une application-cible. De plus, si le concepteur de l'assistance définit avec SEPIA une assistance pertinente, celle-ci peut être à la fois efficace et appréciée par les utilisateurs finaux.

Concernant les aspects non encore évalués, nous envisageons de mettre en place de nouvelles expérimentations, complémentaires de celles d'ores et déjà réalisées.

Conclusion

Plan de la conclusion

Discussion

Cette thèse en informatique se situe plus particulièrement dans le domaine de l'ingénierie des connaissances. La problématique générale de nos travaux de recherche concernait la mise en place de systèmes d'assistance dans des applications existantes quelconques en adoptant une démarche générique. Elle soulevait trois principales questions de recherche : comment intégrer un système d'assistance à une application informatique, sans avoir à la redévelopper ou à la modifier et sans contrainte sur cette application ? Comment permettre à une personne de mettre en place un système d'assistance dans une application sans recourir à la programmation ? Comment proposer une assistance adaptée aux besoins de l'utilisateur ?

Afin de répondre à la première question de recherche, nous avons choisi d'adopter une démarche entièrement *épiphyte*. Nous avons proposé un processus d'adjonction d'un système d'assistance à une application-cible de manière épiphyte. Il est constitué de deux phases : la spécification de l'assistance et son exécution. La spécification de l'assistance concerne un expert de l'application-cible, le *concepteur de l'assistance*. Durant cette phase, le concepteur de l'assistance représente ses connaissances relatives à l'application-cible et à l'assistance qu'il souhaite mettre en place dans celle-ci. L'exécution de l'assistance concerne les utilisateurs finaux des applications-cibles qui vont recevoir de l'assistance. Notre approche permet ainsi la mise en place d'assistances *a posteriori* du développement de leur application-cible, mais ces assistances sont spécifiées *a priori* de leur exécution dans l'application-cible. Selon notre approche, aucune nouvelle règle ou action d'assistance n'est en effet générée automatiquement par le système d'assistance lors de l'exécution de l'assistance pour un utilisateur final. Les approches les plus proches de la nôtre sont celles qui permettent l'ajout de systèmes conseillers épiphytes dans les environnements Telos [Paquette, 2012] et ExploraGraph [Dufresne, 2001], ainsi que dans les applications web [Richard, 2008]. En effet, ces approches proposent également une phase de spécification de l'assistance par un concepteur, suivie d'une phase d'exécution de cette assistance pour les utilisateurs finaux de l'application-cible. Notre approche se distingue cependant par sa généricité : elle n'est ni spécifique à un environnement donné, ni aux applications web, mais concerne au contraire les applications les plus variées.

Pour répondre à notre deuxième question de recherche, qui concerne la mise en place d'assistance dans une application sans recourir à la programmation, nous avons proposé aLDEAS, un langage pivot entre la phase de spécification de l'assistance par un concepteur potentiellement non-informaticien et la phase d'exécution de cette assistance par un outil générique. Ce langage graphique permet de définir des systèmes d'assistance très variés sous forme d'un ensemble de règles. Le langage aLDEAS est complété par un ensemble de patrons qui peuvent aider les concepteurs d'assistance à définir avec aLDEAS des actions d'assistance complexes, fréquemment rencontrées dans les systèmes d'assistance existants et combinant de nombreux éléments d'aLDEAS. Lors de la phase de spécification de l'assistance, aLDEAS permet au concepteur de l'assistance de représenter ses connaissances relatives à l'application-cible et à l'assistance qu'il souhaite lui ajouter. Ces connaissances seront ensuite exploitées lors de la phase d'exécution de l'assistance pour permettre aux utilisateurs finaux

de l'application-cible de bénéficier de l'assistance définie par le concepteur. Ces connaissances peuvent non seulement concerner l'assistance à ajouter, mais également le fonctionnement de l'application-cible : une action d'assistance aLDEAS qui instancie le patron de présentation guidée est une manière de représenter une connaissance relative aux fonctionnalités de l'application-cible. De même, une action d'assistance aLDEAS qui instancie le patron de pas à pas est une manière de représenter une connaissance relative à l'enchaînement des actions à effectuer pour réaliser une tâche donnée dans l'application-cible. Pour répondre à cette question de recherche, nous avons également proposé aMEAS, le modèle d'exécution d'un système d'assistance aLDEAS qui détermine le fonctionnement d'un système d'assistance défini par un ensemble de règles aLDEAS. Les approches proposées dans Telos et ExploraGraph s'appuient également sur l'utilisation de règles d'assistance, or, nous avons montré que le langage aLDEAS permet de définir des règles et actions d'assistance équivalentes (cf. Section 2.2), à l'exception des actions de types envoi de mails. aLDEAS propose en outre des actions d'assistance non proposées par ces approches, telles que les actions automatisées, les pas à pas et les présentations guidées. Concernant l'approche d'assistance par ajout d'un assistant conversationnel mobile proposée par Carlier et Renault, notre patron d'action d'agent animé permet de définir des actions équivalentes à celles réalisées par l'assistant d'iFrimousse [Carlier et Renault, 2010], constituées de déplacements, de mises en valeur de composants de l'interface de l'application-cible et de messages. En revanche, les agents animés proposés dans notre approche ne sont actuellement pas capables de générer des réponses aux questions de l'utilisateur, à moins que celles-ci n'aient été définies avec aLDEAS par le concepteur de l'assistance. Concernant l'approche de Richard d'ajout de systèmes conseillers dans les applications web, nous avons montré que les associations entre les conseils et les liens de l'application-cible peuvent être représentés par des règles respectant le patron de règles d'assistance d'aLDEAS. Notre patron de règles présente également l'avantage de proposer de nombreux types d'événements déclencheurs en plus des clics de l'utilisateur sur des liens de l'application-cible, ce qui permet une contextualisation plus fine du déclenchement d'assistance. De plus, en complément des messages et des propositions de ressources (comme des liens vers des sites web), aLDEAS propose de nombreuses autres actions d'assistance.

Pour répondre à la troisième question de recherche, qui concerne l'adaptation de l'assistance aux besoins de l'utilisateur, nous avons proposé à travers le langage aLDEAS un large choix d'actions d'assistance, associées à des paramètres optionnels qui permettent la prise en compte des préférences des utilisateurs finaux. De plus, les attentes d'événements d'aLDEAS permettent de contextualiser finement le déclenchement de l'assistance. Enfin, à travers les consultations, aLDEAS permet la contextualisation et la personnalisation de l'assistance fournie aux utilisateurs finaux, en déclenchant des actions d'assistance uniquement dans certaines conditions. En particulier, les spécificités des utilisateurs finaux peuvent être prises en compte grâce à des consultations du profil. De même, grâce à des consultations de l'historique de l'assistance, aLDEAS permet de fournir aux utilisateurs finaux une assistance qui évolue en fonction de l'assistance déjà fournie et des choix antérieurs des utilisateurs finaux.

Nos différentes propositions théoriques ont été mises en œuvre de façon opérationnelle dans le système SEPIA, constitué de différents outils, principalement un éditeur d'assistance et un moteur générique d'assistance. L'éditeur d'assistance de SEPIA est destiné aux concepteurs d'assistance et met en œuvre la phase de spécification de l'assistance. Il leur fournit une interface à travers laquelle les concepteurs d'assistance manipulent les éléments du langage aLDEAS, afin de définir un système d'assistance sous la forme d'un ensemble de règles aLDEAS. Les systèmes d'assistance aLDEAS peuvent ensuite être exécutés par le moteur générique d'assistance qui met en œuvre la phase d'exécution de l'assistance. Le fonctionnement du moteur générique est décrit par aMEAS, le modèle d'exécution d'un système d'assistance aLDEAS. Il permet de fournir aux utilisateurs finaux des applications-cibles l'assistance spécifiée. Pour cela, le moteur d'assistance s'appuie sur différents outils. Les épi-détecteurs mettent en œuvre le processus de surveillance d'une application et informent le moteur d'assistance des interactions entre l'utilisateur final et l'interface de l'application-cible. Les épi-inspecteurs inspectent l'état de l'application-cible à la demande du moteur d'assistance. Enfin, les épi-assistants réalisent des actions d'assistance dans l'application-cible à la demande du moteur d'assistance, afin de fournir à l'utilisateur final l'assistance définie, sous une forme adaptée aux choix du concepteur de l'assistance et aux spécificités de l'utilisateur final. Le système SEPIA permet l'exécution d'actions d'assistance de différents types et sous des formes très variées. Il permet ainsi d'exécuter des actions d'assistance équivalentes à la majorité des actions d'assistance existantes. En revanche, le développement de nouveaux épi-assistants serait nécessaire pour exécuter des actions d'assistance visuellement identiques à certaines actions existantes (cf. Annexe A-2).

La surveillance des applications existantes est un point clé de notre approche. Nous avons développé un ensemble d'épi-détecteurs qui mettent en œuvre le processus de surveillance d'une application existante que nous avons proposé. Si ces épi-détecteurs permettent la surveillance de très nombreuses applications, il reste néanmoins nécessaire de développer de nouveaux épi-détecteurs pour certaines catégories d'applications. Par ailleurs, certaines catégories d'applications, comme les applications Flash, ne sont pas compatibles avec notre technique de surveillance. Plusieurs alternatives sont toutefois envisageables. Tout d'abord, les techniques de surveillance basées sur la reconnaissance visuelle telles que [Dixon et Fogarty, 2010] peuvent être utilisées en complément ou en remplacement de notre technique lorsque celle-ci n'est pas adaptée, en perdant néanmoins certaines informations non visibles à l'écran, comme les descriptions d'accessibilité et les infobulles. Par ailleurs, certaines applications ont été conçues pour collecter leurs traces, selon un formalisme qui leur est propre ou qui respecte une norme, telle que TinCan [Tincan, 2014]. Il serait intéressant de permettre la conversion de ces traces vers un système de gestion de traces tel que le kTBS, afin que celui-ci les stocke pour un usage futur et les relaie à notre système SEPIA pour l'exécution de l'assistance.

Afin de valider nos contributions théoriques et leur mise en œuvre, nous avons conduit plusieurs expérimentations. Concernant la phase de spécification de l'assistance, ces expérimentations ont montré que notre langage aLDEAS peut être utilisé pour concevoir des systèmes d'assistance très variés, par des concepteurs d'assistance informaticiens ou non-informaticiens. De même, l'éditeur d'assistance de SEPIA peut être utilisé par ces

concepteurs d'assistance pour spécifier les systèmes d'assistance aLDEAS. Si une utilisation préalable d'aLDEAS pour définir sur papier les règles d'assistance facilite l'utilisation de l'éditeur, cette étape n'est pas indispensable. En effet, bien qu'aLDEAS soit mis en œuvre à travers l'éditeur, le concepteur de l'assistance n'a pas besoin de connaître le langage pour l'utiliser via l'interface de l'éditeur pour définir un système d'assistance. Concernant la phase de spécification de l'assistance, les expérimentations que nous avons menées ont montré que le système SEPIA pouvait être utilisé pour mettre en place des systèmes d'assistance efficaces pour répondre aux besoins d'assistance des utilisateurs, à partir du moment où ces besoins ont été correctement identifiés par le concepteur de l'assistance. De plus, l'assistance fournie aux utilisateurs finaux peut être très bien acceptée par ces derniers, à condition toutefois que celle-ci ait été judicieusement définie par son concepteur. L'évaluation de la pertinence de l'assistance sort néanmoins du cadre de notre thèse.

Ainsi, les propositions théoriques effectuées dans le cadre de cette thèse et leur mise en œuvre opérationnelle, répondent pleinement à notre problématique générale. L'adoption d'une approche générique dans ce travail présente des avantages, illustrés par la Fig. C-1 [Jean-Daubias, 2011]. Dans notre contexte de mise en place *a posteriori* d'assistance, chaque personne souhaitant mettre en place de l'assistance dans une application-cible donnée doit la développer (cf. Ⓐ Fig. C-1), ce qui nécessite dans la plupart des cas de faire appel à un informaticien (cf. Ⓑ Fig. C-1). Lorsque le code source de l'application n'est pas disponible, une telle approche est impossible.

Nous avons proposé des modèles génériques, que nous avons mis en œuvre à travers le système SEPIA (cf. ©Fig. C-1). Ces modèles sont principalement utilisés via les outils qui les mettent en œuvre, par des concepteurs d'assistance, qu'ils soient ou non informaticiens, pour spécifier l'assistance qu'ils souhaitent pour des applications-cibles variées (cf. ⒹFig. C-1). Cette assistance peut ensuite être exécutée automatiquement par nos outils pour les utilisateurs finaux de ces applications-cibles (cf. ⒺFig. C-1). Nos propositions rendent ainsi la mise en place d'assistance, d'une part accessible à un grand nombre de concepteurs d'assistance, informaticiens ou non ; et d'autre part, possible à travers un outil unifié dans les applications-cibles les plus variées, sans besoin de les modifier.

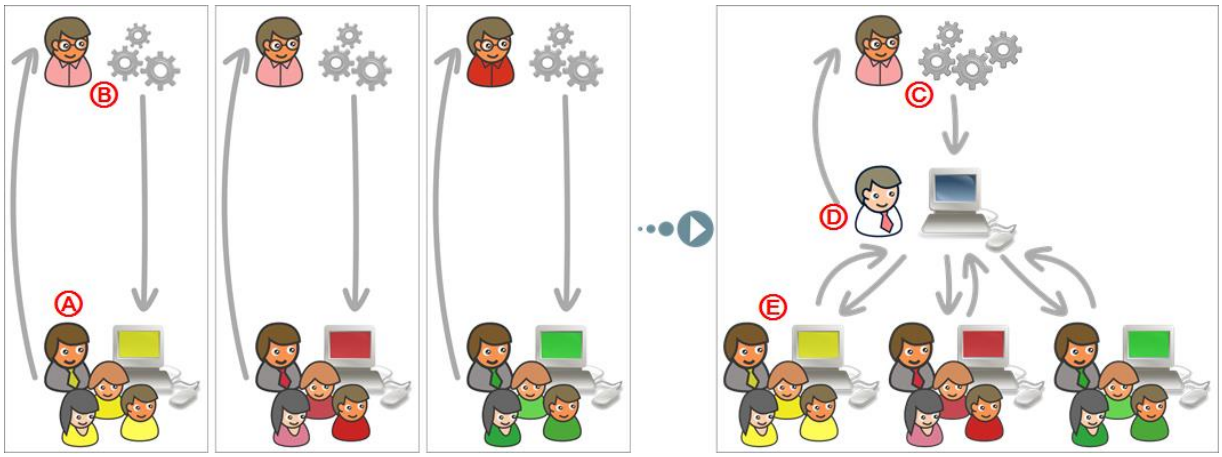


Fig. C-1. Intérêt d'une approche générique [Jean-Daubias, 2011]

Bien que mise en place à l'aide de modèles et outils génériques, l'assistance fournie aux utilisateurs finaux peut être pleinement adaptée aux spécificités des utilisateurs finaux, aux spécificités de l'application-cible et aux spécificités de son contexte d'utilisation. La Fig. C-2 illustre comment se fait cette articulation entre la généricité du langage aLDEAS et la spécificité de l'assistance fournie aux utilisateurs finaux. Selon notre approche, lorsqu'un concepteur souhaite mettre en place un système d'assistance pour une application-cible, il exploite le langage générique aLDEAS via les patrons qui lui sont associés, en particulier celui de règles d'assistance. En représentant ses connaissances à l'aide d'aLDEAS et en les intégrant à SEPIA, le concepteur d'assistance définit un système d'assistance spécifique à son application-cible. De plus, en fonction des choix faits par son concepteur, l'assistance fournie aux différents utilisateurs finaux peut être contextualisée et personnalisée pour chaque utilisateur final.

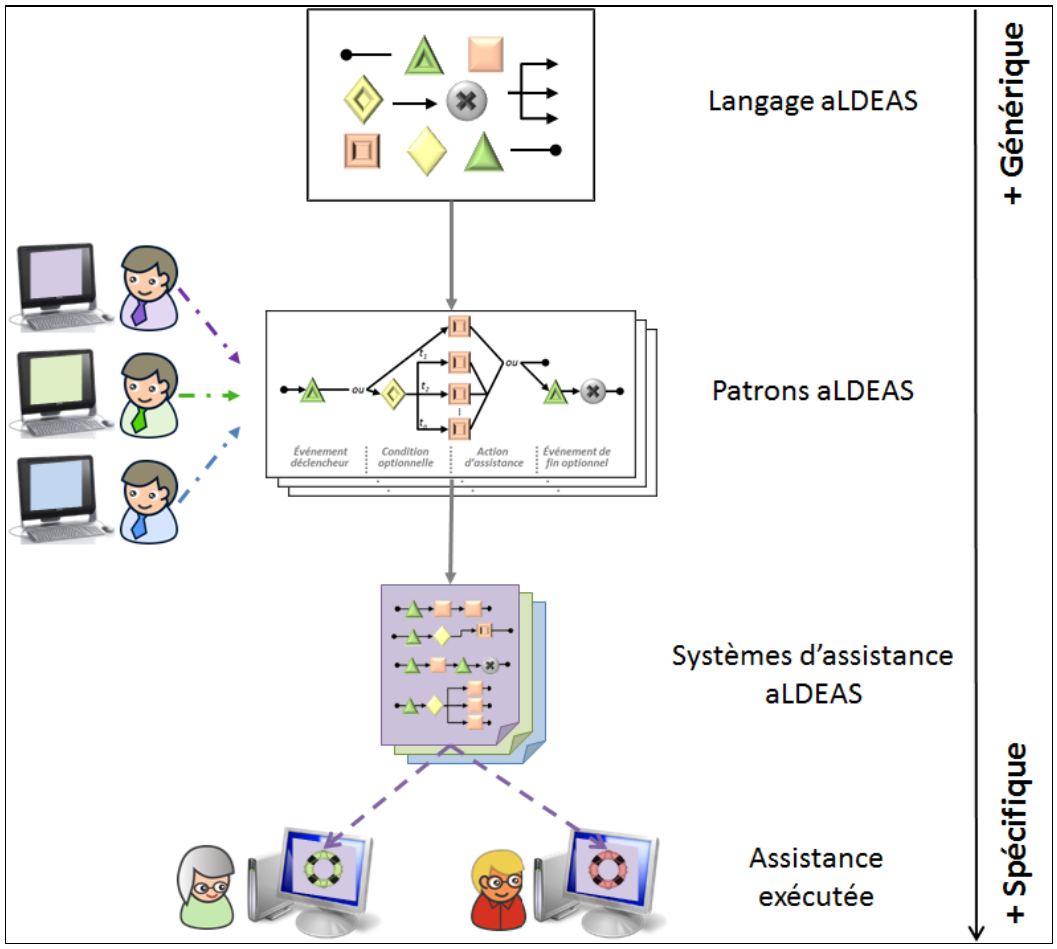


Fig. C-2. Articulation entre la généricité d'aLDEAS et la spécificité de l'assistance fournie aux utilisateurs finaux

Ceci montre comment le système SEPIA, à travers notre langage générique aLDEAS, permet la mise en place d'autant de systèmes d'assistance que le souhaitent les concepteurs, et l'exécution d'autant d'assistances que d'utilisateurs finaux et de contextes d'utilisation.

Retour sur les scénarios d'usage

Revenons maintenant sur les scénarios d'usage présentés en introduction, afin de montrer dans quelle mesure nos propositions permettent de répondre aux besoins des acteurs de ces scénarios.

Scénario 1 : une alternative au dépannage par téléphone pour PhotoScape

Stéphane souhaite permettre à sa mère Martine d'utiliser de manière autonome PhotoScape pour modifier ses photos, en particulier pour la correction des yeux rouges. En utilisant le système SEPIA, Stéphane peut définir un système d'assistance capable de guider Martine à travers les différentes étapes de la correction des yeux rouges sur une photo avec PhotoScape, ainsi que pour les autres manipulations de photos que Martine souhaite faire. Grâce à ce système d'assistance, Martine pourra être autonome pour utiliser PhotoScape.

L'assistance pour la correction des yeux rouges a été définie et expérimentée auprès de 151 utilisateurs finaux de PhotoScape (cf. Section 11.4.1). L'efficacité de cette assistance a été clairement démontrée auprès d'utilisateurs finaux très variés, notamment des personnes âgées, novices dans l'utilisation d'applications informatiques, ce qui correspond au profil de Martine. Des assistances destinées à d'autres manipulations de photos dans PhotoScape ont également été définies, bien qu'elles n'aient pas encore été expérimentées auprès d'utilisateurs finaux.

Scénario 2 : une prise en main facilitée de NetBeans

François désire permettre à ses étudiants de 3ème année de licence d'informatique d'être plus à l'aise avec la programmation Java avec NetBeans. En utilisant le système SEPIA, François peut définir un système d'assistance pour NetBeans, sous la forme d'un tutoriel contenant plusieurs parties indépendantes, chacune associée à des notions de programmation Java et à des compétences relatives à l'utilisation de NetBeans. Les étudiants pourront ainsi étudier à leur rythme et de manière souple en suivant les parties du tutoriel qu'ils jugent utiles. Contrairement à un tutoriel en ligne ou à une vidéo de démonstration, ce tutoriel parait intégré à NetBeans. En conséquence, les étudiants sont actifs pendant le tutoriel et peuvent pratiquer directement dans NetBeans les exercices proposés par le tutoriel.

Le tutoriel pour la découverte de NetBeans et la création d'applications graphiques avec NetBeans a été défini et expérimenté dans le cadre d'un cours d'IHM de licence d'informatique à l'Université Lyon 1 (cf. Section 11.4.2). Il a été utilisé par 52 étudiants lors du premier TP du semestre. En plus d'avoir été très apprécié par les étudiants comme par les enseignants, ce tutoriel semble avoir facilité l'acquisition par les étudiants des connaissances et compétences en jeu dans le cadre de ce cours.

Scénario 3 : une utilisation ludique de Géogébra

Mathilde, 12 ans, aimerait que son frère de 7 ans, Thomas, puisse jouer avec Géogébra pour créer simplement des rosaces colorées sans qu'il ait besoin de maîtriser les notions mathématiques en jeu. En utilisant le système SEPIA, Mathilde peut définir un système d'assistance pour guider pas à pas Thomas pour créer des rosaces avec Géogébra, à l'aide de messages simples et de mises en valeur des boutons sur lesquels Thomas doit cliquer. L'expérimentation que nous avons menée avec des concepteurs d'assistance non-informaticiens a notamment montré qu'une enfant pouvait utiliser SEPIA pour définir un système d'assistance simple (cf. Section 11.3.3). L'enfant ayant participé à cette expérimentation s'est d'ailleurs amusée à définir un système d'assistance pour Géogébra, de sa propre initiative. Ce système d'assistance n'a toutefois pas encore été mis en place avec l'éditeur de SEPIA, ni testé auprès d'utilisateurs finaux.

Scénario 4 : des tâches automatisées dans GIMP

Anthony souhaiterait créer un système d'assistance pour son propre usage, qui automatiserait certaines tâches répétitives dans Gimp, telles que l'ajout d'un fond transparent sur une image. En utilisant le système SEPIA, Anthony peut définir un système d'assistance pour GIMP, destiné à son propre usage. Ce système d'assistance peut

notamment contenir des actions automatisées conduisant à la réalisation des tâches qu'il souhaite automatiser. Pour une définition plus simple et plus rapide de ces ensembles d'actions d'assistance, Anthony peut instancier le patron de pas à pas automatisé via l'interface de l'éditeur. Grâce à la fonctionnalité d'enregistrement de séquences d'actions proposée par l'éditeur d'assistance de SEPIA, Anthony devra simplement effectuer dans GIMP la séquence d'actions élémentaires correspondant à la tâche qu'il souhaite automatiser. Cette séquence d'actions élémentaires pourra ensuite être rejouée automatiquement dans GIMP par le moteur d'assistance de SEPIA, afin par exemple d'ajouter un fond transparent à une image.

Perspectives

Ce travail de recherche ouvre plusieurs perspectives concernant l'assistance à l'utilisateur d'applications informatiques, en complément de l'actuel travail de diffusion et d'exploitation du système SEPIA pour mettre en place de l'assistance dans différents contextes, notamment avec le projet PEPS, ainsi qu'avec les plateformes du laboratoire LIRIS et avec les outils utilisés dans le cadre des enseignements réalisés à l'université Lyon 1 (cf. Section 12.1.2).

Assistance en contexte éducatif

Les propositions effectuées dans le cadre de cette thèse ne sont spécifiques à aucun domaine en particulier. Une perspective de recherche qui nous intéresse particulièrement consiste à étudier les spécificités de l'assistance dans le contexte éducatif, dans le but de faire évoluer les modèles et les outils proposés dans cette thèse afin de les adapter au contexte éducatif. Ce travail fait actuellement l'objet d'une thèse de doctorat, débutée en octobre 2013 par Lê Vinh Thái[18].

Dans un premier temps, deux catégories d'utilisateurs finaux ont été identifiées : les apprenants et les enseignants. Ensuite, plusieurs catégories d'applications ont été identifiées : les applications à destination des enseignants, tels que les outils auteurs ou les EIAH paramétrables, et les applications à destination des apprenants. Il peut s'agir d'applications spécifiquement dédiées à l'apprentissage, les EIAH, ou d'applications non spécifiquement dédiées à l'apprentissage. Ces dernières applications peuvent être utilisées dans le cadre d'une activité pédagogique pour favoriser l'acquisition de connaissances, par exemple Excel peut être utilisé comme support à l'apprentissage de l'algèbre [Ritter et Koedinger, 1995]. Par ailleurs, les applications non spécifiquement dédiées à l'apprentissage peuvent être utilisées dans le cadre d'une activité pédagogique qui vise la maîtrise de cette application. Par exemple, dans le cadre d'un cours de secrétariat, une activité pédagogique peut avoir pour but la maîtrise de Word en tant qu'outil professionnel.

En fonction du type d'application utilisée, ainsi que du type d'utilisateur final, plusieurs besoins d'assistance spécifiques ont ensuite été identifiés [Thái et Ginon, 2014]. Pour chacun

[18] http://liris.cnrs.fr/membres?idn=lvthai

de ces besoins, de premières réflexions et expérimentations ont été menées afin de déterminer dans quelle mesure les modèles et outils que nous avons proposés permettent d'ores et déjà d'y répondre [Ginon et al., 2014b]. Les propositions présentées dans ce travail de thèse, bien que non spécifiques à un domaine en particulier, permettent de répondre à de nombreux besoins d'assistance spécifiques au contexte éducatif, mais permettent difficilement de répondre à d'autres besoins. En particulier, pour mettre en place une assistance impliquant du diagnostic intermédiaire dans un EIAH qui en est dépourvu, le concepteur de l'assistance doit actuellement spécifier de nombreuses règles afin de permettre au système SEPIA de distinguer une réponse correcte d'une réponse incorrecte. Par exemple, la réponse « X – 3 » n'est pas équivalente à la réponse « -3 + X » pour SEPIA, qui ne peut actuellement que comparer des chaînes de caractères. L'exploitation de connaissances du domaine enseigné permettrait au système SEPIA de reconnaître ces deux réponses comme équivalentes et épargnerait ainsi un travail long et fastidieux au concepteur de l'assistance.

Par ailleurs, dans le contexte éducatif, le concepteur de l'assistance pourrait être un enseignant. Or, si le système SEPIA semble utilisable par des non-informaticiens dans le cas de systèmes d'assistance simples (cf. Section 11.3.3), ce n'est pas nécessairement le cas pour des systèmes d'assistance plus complexes. La thèse de Lê Vinh Thái prend donc principalement deux directions : l'acquisition de connaissances du domaine dans le but d'augmenter les capacités du système SEPIA et l'exploitation de patrons d'actions d'assistance spécifiques au contexte éducatif, dans le but de faciliter la tâche du concepteur de l'assistance.

Aide à la conception et à l'amélioration de systèmes d'assistance

La spécification de l'assistance est un travail conséquent et difficile pour le concepteur de l'assistance. En particulier, l'identification des besoins d'assistance est un travail délicat. Aider le concepteur de l'assistance à identifier les besoins d'assistance et à spécifier un système d'assistance qui répond à ces besoins d'assistance est donc une perspective intéressante. Une solution consisterait dans un premier temps à observer et à analyser les interactions entre l'application-cible et ses utilisateurs finaux (cf. Ⓐ Fig. C-3). Dans un deuxième temps, les informations relatives aux observations effectuées pourraient être fournies au concepteur de l'assistance sous la forme d'un bilan mis à disposition du concepteur lors de la spécification de l'assistance avec l'éditeur de SEPIA (cf. Ⓑ Fig. C-3). Ceci pourrait en effet attirer l'attention du concepteur d'assistance sur des besoins d'assistance potentiels et ainsi l'aider à concevoir le système d'assistance. Par exemple, il serait possible d'informer le concepteur de l'assistance de l'état de l'application-cible dans lequel les utilisateurs finaux quittent l'application. Parmi ces états, certains correspondront à des sorties « normales » de l'application à l'issue d'une tâche réalisée avec succès, et d'autres correspondront à des sorties « anormales » de l'application, à l'issue de l'abandon d'une tâche qui est la conséquence d'un problème rencontré par l'utilisateur final.

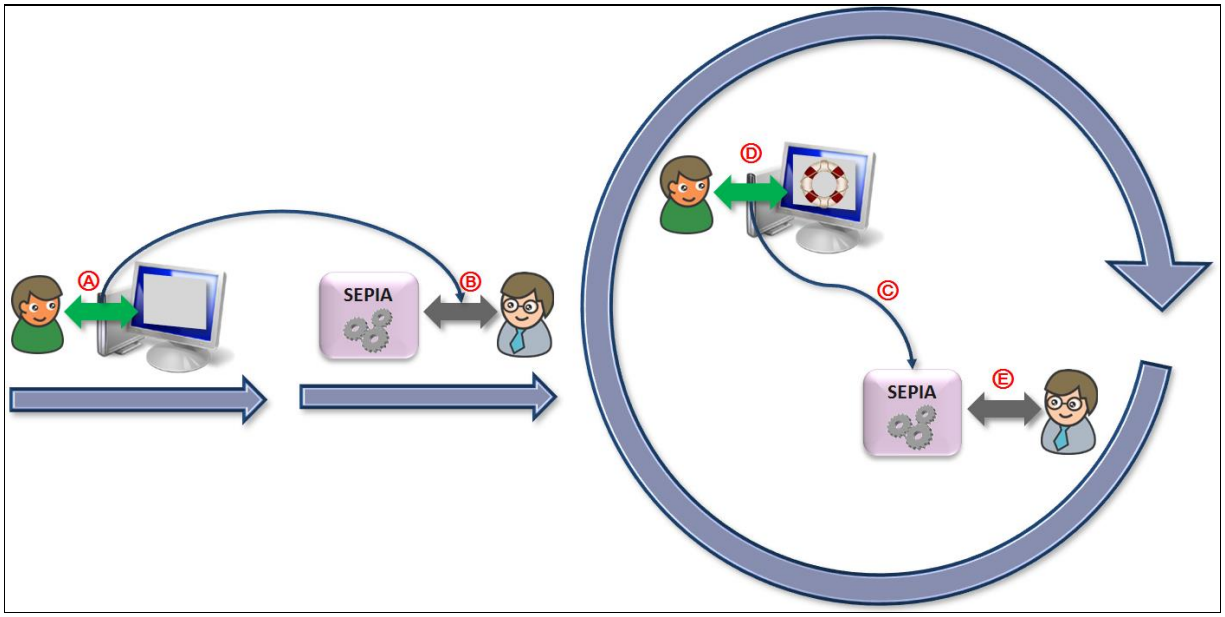


Fig. C-3. Cycle de vie d'un système d'assistance aLDEAS

Par ailleurs, il est également important de permettre au concepteur de l'assistance d'évaluer l'efficacité du système d'assistance qu'il a conçu et qui est exécuté par le moteur générique de SEPIA (cf. © Fig. C-3), ainsi que la satisfaction des utilisateurs finaux, afin de lui permettre de l'améliorer. Pour cela, une solution consisterait à observer les interactions entre l'application-cible, les utilisateurs finaux et le système d'assistance (cf. Ⓓ Fig. C-3). Il peut également être pertinent de permettre aux utilisateurs finaux d'exprimer leur opinion sur l'assistance qui leur a été proposée, sur celle qu'ils auraient aimé recevoir et sur leurs suggestions vis-à-vis de l'assistance. Ces retours des utilisateurs finaux, ainsi que des indicateurs calculés à partir des observations effectuées pourraient être présentés au concepteur de l'assistance à travers l'éditeur de SEPIA (cf. Ⓔ Fig. C-3), afin de l'aider à améliorer son système d'assistance. Par exemple, en analysant l'historique de l'assistance, il est possible de connaître les réponses des utilisateurs finaux aux consultations. En constatant que beaucoup d'utilisateurs ont demandé « plus d'aide » à une consultation, le concepteur de l'assistance peut en déduire que les utilisateurs finaux sont en difficulté dans ce contexte et que l'assistance proposée est insuffisante. En allant plus loin dans l'analyse, il est envisageable de trouver des points communs aux utilisateurs ayant demandé « plus d'aide » : ils peuvent par exemple être majoritairement novices dans la réalisation de la tâche correspondante. Dans ce cas, le concepteur de l'assistance pourrait juger pertinent de personnaliser l'assistance pour fournir systématiquement plus d'aide aux utilisateurs novices.

Stockage et exploitation des traces de l'utilisateur

Les traces informatiques peuvent avoir des formes très variées : historique de navigation pour le web, fichier de logs, profil d'utilisateur, etc. [Laflaquière et al., 2006]. Dans le cadre de cette thèse, nous nous sommes intéressée plus particulièrement aux traces d'interactions entre un utilisateur final et l'interface d'une application. Dans le système SEPIA, ces traces

sont utilisées pour contextualiser l'assistance, et plus précisément pour déclencher ou mettre fin à des actions d'assistance en fonction des actions passées de l'utilisateur dans l'application-cible. Elles sont donc utilisées immédiatement par le moteur générique lorsqu'il reçoit une notification des épi-détecteurs, sans être stockées pour un usage futur. Pourtant, les traces de l'activité passée des utilisateurs finaux peuvent également être utilisées pour adapter l'assistance. Ainsi, si un utilisateur a déjà réalisé avec succès une tâche par le passé, mais qu'il est actuellement en difficulté pour cette même tâche, une assistance possible consiste à lui proposer un pas à pas guidé à partir de ses propres traces. Ceci pourra aider l'utilisateur à réaliser la tâche, mais également à se souvenir plus facilement d'une méthode pour réaliser cette tâche qu'il a lui-même déjà utilisée.

Dans notre équipe de recherche, l'équipe Silex[19], les traces ont fait l'objet de nombreux travaux. Nous montrons dans cette section de quelle manière ces travaux peuvent être réutilisés et appliqués à SEPIA pour permettre le stockage et l'exploitation des traces pour adapter l'assistance à l'expérience de l'utilisateur final.

Selon la théorie de la trace modélisée de l'équipe Silex, une *trace modélisée* est composée d'éléments temporellement situés, appelés *obsels* (pour *observed elements*). Elle est associée à un modèle de traces décrivant des types d'obsels, et par conséquent un guide de construction et de manipulation [Champin et al., 2013]. Dans le langage aLDEAS, les événements élémentaires concernés par les éléments de type attente d'événement peuvent être considérés comme des obsels, alors que les séquences d'événements peuvent être considérées comme des traces modélisées. Une *transformation de trace* consiste à manipuler une trace modélisée pour obtenir une ou plusieurs nouvelles traces modélisées, associées à de nouveaux modèles [Champin et al., 2013]. Par exemple, une transformation fréquente consiste à filtrer certains types d'obsels.

Bien que le système SEPIA permette la consultation de l'historique de l'assistance, qui contient des traces relatives à l'exécution de l'assistance (cf. Section 6.2), dans le cadre de cette thèse nous n'avons pas traité la question de la consultation des traces de l'utilisateur relatives à ses actions passées dans l'application-cible. La Fig. C-4 montre cependant de quelle manière ces consultations peuvent être mises en œuvre à travers le système SEPIA en couplant ces travaux avec le système SEPIA.

[19] http://liris.cnrs.fr/equipes?id=44

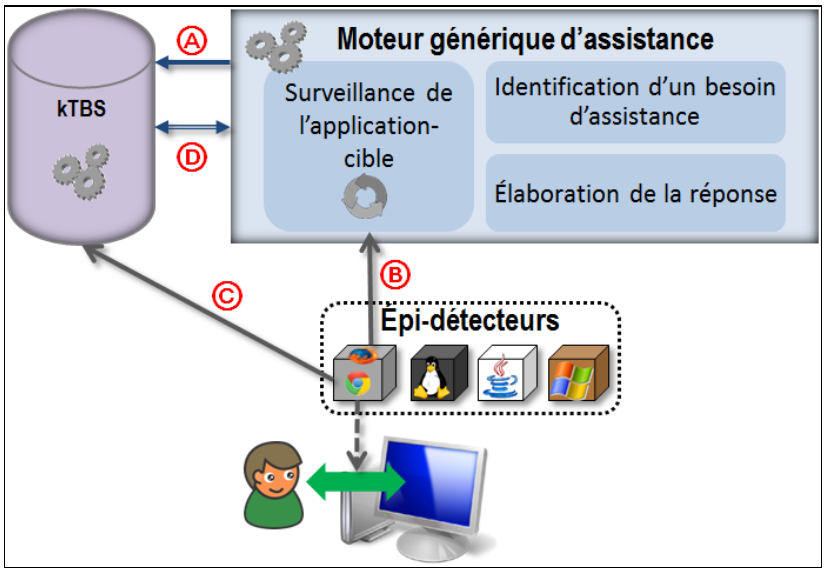


Fig. C-4. Intégration du kTBS au système SEPIA

Le stockage des traces ainsi que leurs transformations peuvent être délégués à un système de gestion de bases de traces, tel que le kTBS[20] (kernel for Trace Base System). Ainsi, au lancement de l'assistance, le moteur générique de SEPIA initialiserait une trace modélisée dans le kTBS (cf. Ⓐ Fig. C-4). Par la suite, au lieu de notifier uniquement le moteur d'assistance des événements détectés à l'interface de l'application-cible (cf. Ⓑ Fig. C-4), les épi-détecteurs notifieraient également ces événements au kTBS (cf. © Fig. C-4), qui les stockerait sous la forme d'obsels dans la trace modélisée.

Ainsi, le kTBS pourra par la suite être interrogé par le système d'assistance lorsqu'il sera nécessaire de réaliser une consultation des traces de l'utilisateur. Pour cela, le moteur d'assistance de SEPIA devra transmettre les consultations des traces de l'utilisateur au kTBS ; le kTBS se chargera ensuite d'effectuer la ou les transformations correspondantes sur la trace modélisée, puis communiquera le résultat au moteur d'assistance de SEPIA (cf. Ⓓ Fig. C-4). Par exemple, pour consulter le nombre de clics de l'utilisateur sur un bouton donné, le kTBS doit effectuer une transformation de type *suppression* de tous les obsels qui ne correspondent pas à ce clic. Le nombre de clics sur le bouton sera alors obtenu en comptant le nombre d'obsels de la trace transformée.

Ce travail d'ingénierie, qui exploitera des travaux de recherche réalisés dans l'équipe Silex, permettra d'augmenter encore la capacité du système SEPIA à adapter l'assistance fournie à l'utilisateur final.

[20] https://github.com/ktbs/ktbs

Assistance à base d'usages typiques

Actuellement, la spécification de l'assistance dans le système SEPIA se fait sous la forme d'un ensemble de règles respectant le patron de règles aLDEAS. Une façon complémentaire d'exploiter les outils de SEPIA pour fournir de l'assistance consisterait à s'appuyer sur des « usages typiques » de l'application-cible, en complément des règles d'assistance, afin de rendre plus efficace l'identification de besoins d'assistance. Un usage typique est défini par une séquence d'événements de bas niveau, tels que les clics sur des boutons ou les ouvertures de menus. Il représente une manière de réaliser une tâche donnée dans l'application-cible. Les usages typiques associés à une application-cible seraient définis par le concepteur de l'assistance au cours de l'étape de définition de l'assistance (cf. Ⓐ Fig. C-5).

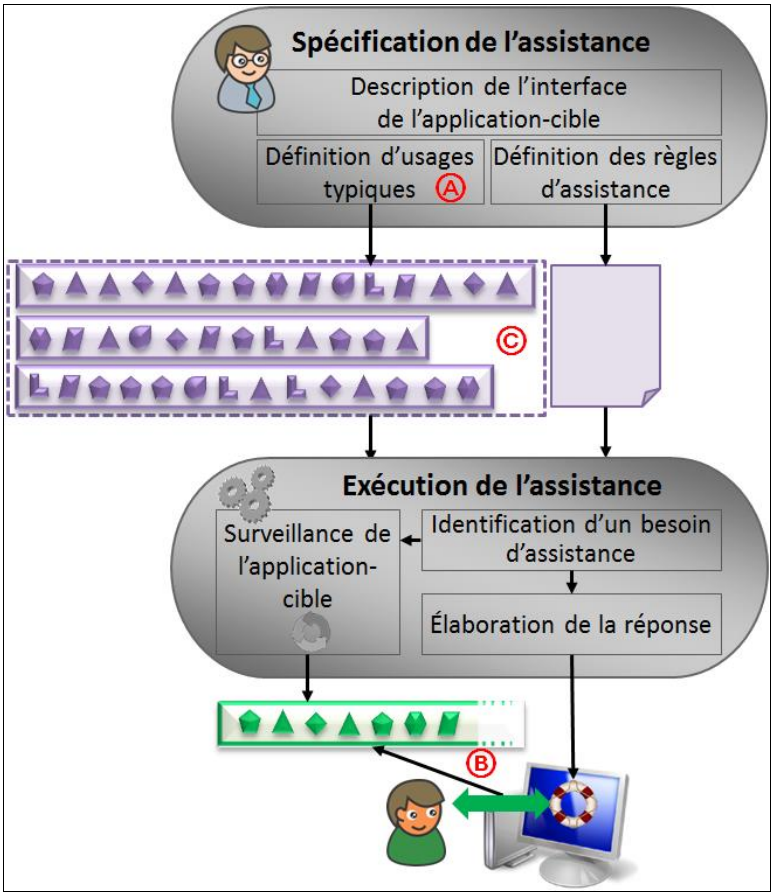


Fig. C-5. Exploitation d'usages typiques dans SEPIA

Par la suite, au cours de l'exécution de l'assistance, le moteur générique pourrait comparer les actions de l'utilisateur final (cf. Ⓑ Fig. C-5) avec les usages typiques définis par le concepteur de l'assistance (cf. Ⓒ Fig. C-5), afin de tenter d'identifier la tâche que l'utilisateur est en train de réaliser, par exemple grâce à une mesure de similarités comparant deux à deux des séquences d'événements bas niveau. Si l'utilisateur a une activité proche de l'un des usages typiques, le moteur générique pourra lui proposer une assistance spécifiquement définie pour cet usage.

Si le moteur détecte que l'utilisateur est en train de s'écarter de l'usage typique qu'il suivait jusque-là, deux principales interprétations sont possibles. Tout d'abord, l'utilisateur peut avoir changé d'objectif, ou l'objectif avait été mal identifié. Dans ce cas, le moteur d'assistance devra essayer à nouveau d'identifier l'objectif de l'utilisateur parmi les usages typiques définis. Au contraire, il est possible que l'objectif de l'utilisateur ait correctement été identifié, mais qu'il soit actuellement en difficulté pour atteindre cet objectif. Le moteur d'assistance pourra alors lui proposer une assistance visant à le guider pas à pas en s'appuyant sur l'usage typique, qui décrit les étapes à suivre pour réaliser la tâche correspondante.

Bien que nos propositions théoriques et leur mise en œuvre à travers le système SEPIA répondent d'ores et déjà pleinement à notre problématique de recherche, d'intéressantes perspectives restent ainsi à étudier.

Références

Références bibliographiques

[Alexander et al., 2008]

Alexander J., CoCkburn A. et Lobb R., appMonitor: a tool for recording user actions in unmodified Windows applications, 2008.

[Anacleto et al., 2007]

Anacleto J. C., De Souza Godoi M., Pinatti de Carvalho A. F. et Lieberman H., A Common Sense-Based On-Line Assistant for Training Employees, Human-computer interaction, Science L. N. i. C., vol 4662, pp. 243-254, 2007.

[Bailly et Malacria, 2013]

Bailly G. et Malacria S., MenuInspector: Outil pour l'analyse des menus et cas d'étude, IHM, Bordeaux, France, pp. 103, 2013.

[Bastien et Scapin, 2004]

Bastien C. et Scapin D., La conception de logiciels interactifs centrés sur l'utilisateur : étapes et méthodes, Ergonomie, Falzon P., PUF, pp. 451-462, 2004.

[Bohl et al., 2002]

Bohl O., Scheuhase J., Sengler R. et Winand U., The sharable content object reference model (SCORM) - a critical review, Computers in Education, Auckland, New Zealand, vol 2, pp. 950-951, 2002.

[Bouchet, 2006]

Bouchet F., Conception d'un langage de requêtes pour un agent conversationnel assistant, Rapport de stage de Master 2, LIMSI, Paris, France, 2006.

[Brahan et al., 1993]

Brahan J. W., Farley B., Orchard R. A., Parent A. et Phan T. T. H., A Designer's Consultant, Research and development in expert systems, Bramer M. A., Milne, R. W., Cambridge, UK, Press syndicate of the university of Cambridge, pp. 197-234, 1993.

[Brusilovsky, 2001]

Brusilovsky P., Adaptive Hypermedia, User Modeling and User-Adapted Interaction, vol. 11, pp. 87-110, 2001.

[Burke, 2002]

Burke R., Hybrid Recommender Systems: Survey and Experiments, User Modeling and User-Adapted Interaction, vol. 12 n°4, pp. 331-370, 2002.

[Capobianco et Carbonell, 2001]

Capobianco A. et Carbonell N., Contextual online help: elicitation of human experts' strategies, HCI'01 : 9th International Conference on Human-Computer Interaction, New Orleans, USA, vol 3, pp. 824–828, 2001.

[Carlier et Renault, 2010]

Carlier F. et Renault V., Educational webportals augmented by mobile devices with iFrimousse architecture, International Conference on Advanced Learning Technologies, Sousse, Tunisia, 2010.

[Champin et al., 2012]

Champin P.-A., Cordier A., Lavoué É., Lefevre M. et Skaf-Molli H., User Assistance for Collaborative Knowledge Construction, SWCS, Lyon, France, 2012.

[Champin et al., 2013]

Champin P.-A., Mille A. et Prié Y., Vers des traces numériques comme objets informatiques de premier niveau : une approche par les traces modélisées Intellectica, vol. 1 n°59, pp. 171-204, 2013.

[Chang et al., 2011]

Chang T.-H., Yeh T. et Miller R., Associating the Visual Representation of User Interfaces with their Internal Structures and Metadata, UIST, Santa Barbara, USA, pp. 245-256, 2011.

[Chikofsky et Cross, 1990]

Chikofsky E. J. et Cross J. H., Reverse Engineering and Design Recovery: A Taxonomy, Software IEEE, 1990.

[Choquet, 2007]

Choquet C., Ingénierie et réingénierie des EIAH : l'approche REDiM, Habilitation à Diriger des Recherches en Informatique, Laboratoire d'Informatique de l'Université du Maine (LIUM), 10 décembre 2007.

[Conejo et al., 2005]

Conejo R., Guzman E., Perez-De-La-Cruz J.-L. et Millán E., Introducing adaptive assistance in adaptive testing, AIED'05 - Artificial Intelligence in Education, vol. 125, pp. 777-779, 2005.

[Cordier et al., 2010]

Cordier A., Lefevre M., Jean-Daubias S. et Guin N., Concevoir des assistants intelligents pour des applications fortement orientées connaissances : problématiques, enjeux et étude de cas, IC 2010 - 21èmes Journées Francophones d'Ingénierie des Connaissances, Presses des Mines, pp. 119-130, 2010.

[Dixon et Fogarty, 2010]

Dixon M. et Fogarty J., Prefab: Implementing Advanced Behaviors Using Pixel-Based Reverse Engineering of Interface Structure, SIGCHI, New York, USA, pp. 1525-1534, 2010.

[Dong et al., 2012]

Dong R., McCarthy K., O'Mahony M. P., Schaal M. et Smyth B., Towards an Intelligent Reviewer's Assistant: Recommending Topics to Help Users to Write Better Product Reviews, IUI, Lisbon, Portugal, pp. 159-168, 2012.

[Dufour, 2007]

Dufour C., L'évaluation continue de programme comme stratégie d'ajustement aux environnements disciplinaire et professionnel pour les écoles de bibliothéconomie et des sciences de l'information, Congrès annuel de l'Association Canadienne des Sciences de l'Information, Montréal, Canada, 2007.

[Dufresne, 2001]

Dufresne A., Conception d'une interface adaptée aux activités de l'éducation à distance - ExploraGraph, Sciences et techniques éducatives, vol. 8 n°3, pp. 301-319, 2001.

[Dufresne et al., 2003]

Dufresne A., Basque J., Paquette G., Léonard M., Lundgren-Cayrol K. et PromTep S., Vers un modèle conceptuel générique de système d'assistance pour le téléapprentissage, Sciences et Technologies de l'Information et de la Communication pour l'Education et la Formation (STICEF), vol. 10, pp. 57-88, 2003.

[Dufresne et Promtep, 2006]

Dufresne A. et Promtep S., ExploraGraph et la Personnalisation des interactions pour l'apprentissage, IHM'06, Montréal, Canada, pp. 153-157, 2006.

[Eyssautier-Bavay, 2008]

Eyssautier-Bavay C., Modèles, langage et outils pour la réutilisation de profils d'apprenants,

[Ferguson et al., 2009]

Ferguson G., Allen J., Galescu L., Quinn J. et Swift M., CARDIAC: An Intelligent Conversational Assistant for Chronic Heart Failure Patient Heath Monitoring, Virtual Health Care Interaction (VHI 09), Arlington, USA, AAAI Press, 2009.

[Galluccio, 2006]

Galluccio R. G. P., Humanizing CALL: The use of pedagogical agents as language tutors, New England Regional Association for Language Learning Technology, 2006.

[Gapenne, 2006]

Gapenne O., Relation d'aide et transformation cognitive, Intellectica, vol. 2 n°44, pp. 7-16, 2006.

[Gilliot et al., 2013]

Gilliot J.-M., Garlatti S., Rebai I. et Belen-Sapia M., Le concept de iMOOC pour une ouverture maitrisée EIAH, Toulouse, France, 2013.

[Ginon et Jean-Daubias, 2011]

Ginon B. et Jean-Daubias S., Models and tools to personalize activities on learners profiles, ED-MEDIA 2011 - World Conference on Educational Multimedia, Hypermedia & Telecommunications, 2011.

[Ginon et al., 2011]

Ginon B., Jean-Daubias S. et Lefevre M., Evolutive learners profiles, ED-MEDIA 2011 - World Conference on Educational Multimedia, Hypermedia & Telecommunications, 2011.

[Ginon, 2012]

Ginon B., Towards a generic model for user assistance, Doctoral consortuim of User Modeling, Adaptation, and Personalization, Montréal, Canada, 2012.

[Ginon et al., 2012a]

Ginon B., Champin P.-A. et Jean-Daubias S., Taking into account users' knowledge, abilities and preferences to personalize animated assistant agents, Workshop PALE associated with UMAP, Montreal, Canada, pp. 29-34, 2012a.

[Ginon et Jean-Daubias, 2012]

Ginon B. et Jean-Daubias S., Prise en compte des connaissances, capacités et préférences pour une personnalisation multi-aspects des activités sur les profils d'apprenants, STICEF, vol. 19, 2012.

[Ginon et al., 2012b]

Ginon B., Jean-Daubias S. et Champin P.-A., Une assistance générique pour les utilisateurs d'applications fortement orientées connaissances, Ingénierie des connaissances, Paris, France, pp. 167-174, 2012b.

[Ginon et al., 2013a]

Ginon B., Champin P.-A. et Jean-Daubias S., Collecting fine-grained use traces in any application without modifying it, Worshop EXPPORT of ICCBR, 2013a.

[Ginon et al., 2013b]

Ginon B., Jean-Daubias S. et Champin P.-A., Adjonction de systèmes d'assistance personnalisée à des EIAH existants, poster pour EIAH, Toulouse, 2013b.

[Ginon et al., 2013c]

Ginon B., Jean-Daubias S. et Champin P.-A., Mise en place d'un système d'assistance personnalisée dans une application existante, IC, Lille, France, 2013c.

[Ginon et al., 2013d]

Ginon B., Jean-Daubias S. et Champin P.-A., Une typologie de l'assistance aux utilisateurs : exemple d'application aux EIAH, RR-LIRIS-2013-007, 2013d.

[Ginon et al., 2014a]

Ginon B., Jean-Daubias S., Champin P.-A. et Lefevre M., aLDEAS : un langage de définition de systèmes d'assistance épiphytes, IC, Clermont-Ferrand, France, pp. 137-148, 2014a.

[Ginon et al., 2014b]

Ginon B., Thai V. L., jean-Daubias S., Lefevre M. et Champin P.-A., Adding epiphytic assistance systems in learning applications using the SEPIA system, Ec-Tel, Graz, Austria, 2014b.

[Goodyear et al., 2004]

Goodyear P., Avgeriou P., Baggetun R., Bartoluzzi S., Retalis S., Ronteltap F. et Rusman E., Towards a Pattern Language for Networked Learning, Networked Learning 2004, Lancaster, UK, 2004.

[Guéraud et al., 2004]

Guéraud V., Adam J.-M., Pernin J.-P., Calvary G. et David J.-P., L'exploitation d'Objets Pédagogiques Interactifs à distance : le projet FORMID, STICEF, vol. 11, pp. 109-163, 2004.

[Grossman et Fitzmaurice, 2010]

Grossman T. et Fitzmaurice G., ToolClips: An Investigation of Contextual Video Assistance for Functionality Understanding, CHI, Atlanta, USA, pp. 1515-1524, 2010.

[Harper et al., 2005]

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

[Haverty, 2005]

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

[Hurst et al., 2010]

Hurst A., Hudson S. E. et Mankoff J., Automatically identifying targets users interact with during real world tasks, UIU, Hong Kong, China, pp. 11-20, 2010.

[Jean-Daubias et Guin, 2009]

Jean-Daubias S. et Guin N., AMBRE-teacher: a module helping teachers to generate problems, 2nd Workshop on Question Generation, International Conférence on Artificial Intelligence in Education (AIED 2009), Brighton, UK, pp. 43-47, 2009.

[Jean-Daubias et Ginon, 2010]

Jean-Daubias S. et Ginon B., Des profils d'apprenant évolutifs, TICE 2010 - Colloque Technologies de l'Information et de la Communication pour l'Enseignement, 2010.

[Jean-Daubias, 2011]

Jean-Daubias S., Ingénierie des profils d'apprenants, Habilitation à diriger les recherches, Université Claude Bernard Lyon 1, 2011.

[Jean-Daubias et al., 2011]

Jean-Daubias S., Ginon B. et Lefevre M., Modèles et outils pour prendre en compte l'évolutivité dans les profils d'apprenants, STICEF, vol. 18, pp. 99-134, 2011.

[Johnson et al., 2000]

Johnson W. L., Rickel J. W. et Lester J. C., Animated pedagogical agents: face-to-face interaction in interactive learning environments, International Journal of Artificial Intelligence in Education, vol. 11, pp. 47-78, 2000.

[Kantner et Rusinsky, 1998]

Kantner L. et Rusinsky L., Designing a WinHelp project for quick conversion to lowest-common-denominator HTML-based help: a case study, SIGDOC, Quebec, Canada, pp. 214 - 218, 1998.

[Kukreja et al., 2006]

Kukreja U., Stevenson W. E. et Ritter F. E., RUI—Recording User Input from Interfaces under Windows and Mac OS X, 2006.

[Kurataa et Hara, 2014]

Kurataa Y. et Hara T., CT-Planner4: Toward a More User-Friendly Interactive Day-Tour Planner, Information and Communication Technologies in Tourism, Dublin, Ireland, pp. 73-86, 2014.

[Labat, 2002]

Labat J.-M., EIAH : Quel retour d'informations pour le tuteur ? , TICE, Lyon, France, pp. 81-88, 2002.

[Laflaquière et al., 2006]

Laflaquière J., Settouti L. S., Prié Y. et Mille A., A trace-based System Framework for Experience Management and Engineering, International Workshop on Experience Management and Engineering, Bournemouth UK, 2006.

[Lave et Wenger, 1991]

Lave J. et Wenger E., Situated Learning. Legitimate Peripheral Participation, Cambridge, UK, Cambridge University Press, 1991.

[Lefevre, 2009]

Lefevre M., Processus unifié pour la personnalisation des activités pédagogiques : méta-modèle, modèles et outils, Thèse de doctorat en informatique, Université Claude Bernard Lyon 1, 1er décembre 2009.

[Leplat, 1998]

Leplat J., A propos des procédures, Performances humaines et techniques, Toulouse, France, vol 94, pp. 6-15, 1998.

[Leray et Sansonnet, 2007]

Leray D. et Sansonnet J.-P., Acquisition de connaissances perceptives pour un agent assistant, IC 2007 18es Journées Francophones d'Ingénierie des Connaissances, Cépadues-Éditions, Grenoble, France, vol 2, pp. 331-332, 2007.

[Lieberman, 1997]

Lieberman H., Autonomous interface agents, ACM SIGCHI Conference on Human factors in computing systems, New York, USA, pp. 67-74, 1997.

[Malacria et al., 2013]

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

[Marzin et al., 2005]

Marzin P., Ergun M., Girault I. et D'Ham C., La construction de protocole de travaux pratiques de chimie à l'aide d'un logiciel : quels apports pour les apprentissages ?, Bulletin de l'union des physiciens, vol. 99, pp. 991-1009, 2005.

[Matejka et al., 2013]

Matejka J., Grossman T. et Fitzmaurice G., Patina: Dynamic Heatmaps for Visualizing Application Usage, CHI, Paris, France, pp. 3227-3236, 2013.

[Mattews et al., 2000]

Mattews M., Pharr W., Biswas G. et Neelakandan H., USCSH: An Active Intelligent Assistance System, Artificial Intelligence Review, (eds.) S. J. H. e. a., Kluwer Academic Publishers, vol 14, pp. 121–141, 2000.

[Meinadier, 1991]

Meinadier J.-P., L'interface utilisateur pour une informatique conviviale, 1991.

[Mille et al., 2006]

Mille A., Caplat G. et Philippon M., Faciliter les activités des utilisateurs d'environnements informatiques : quoi, quand, comment?, Intellectica, vol. 2 n°44, pp. 121-143, 2006.

[Mille et Héraud, 2009]

Mille A. et Héraud J. M., Pixed: Projet d'Intégration de l'eXpérience en Enseignement à Distance, 2009.

[Ospina et Fougères, 2003]

Ospina V. E. et Fougères A.-J., Un système d'assistance dans un environnement coopératif d'apprentissage, Conférence nationale Coopération, Innovation et Technologies, Troyes, France, pp. 1-12, 2003.

[Pachet, 1995]

Pachet F., Néopus user guide., Laforia, 1995.

[Paquette et al., 1994]

Paquette G., Pachet F. et Giroux S., Épitalk, un outil générique pour la construction de systèmes conseillers, Sciences et Techniques Educatives, vol. 1 n°3, 1994.

[Paquette et al., 1996]

Paquette G., Pachet F., Giroux S. et Girard J., EpiTalk, a generic tool for the development of advisor systems, International Journal of Artificial Intelligence in Education, vol. 7, pp. 349-370, 1996.

[Paquette et Tchounikine, 2002]

Paquette G. et Tchounikine P., Contribution à l'ingénierie des systèmes conseillers : une approche méthodologique fondée sur l'analyse du modèle de la tâche, Sciences et Techniques Educatives, vol. 9 n°2-3, 2002.

[Paquette et al., 2008]

Paquette G., Léonard M. et Lundgren-Cayrol K., The MOT+ visual language for knowledge-based instructional design. , 2008.

[Paquette, 2012]

Paquette G., Référencement par compétence, recherche et assistance dans les environnements d'apprentissage et de travail, TICE, Lyon, France, pp. 190-199, 2012.

[Randall et Pedersen, 1998]

Randall N. et Pedersen I., Who exactly is trying to help us? the ethos of help systems in popular computer applications, international conference on Computer documentation, Quebec, Canada, pp. 63-69, 1998.

[Rech et al., 2007]

Rech J., Ras E. et Decker B., Intelligent Assistance in German Software Development: A Survey, Software IEEE, vol. 24 n°4, pp. 72-79, 2007.

[Réty et al., 2003]

Réty J.-H., Martin J.-C., Pelachaud C. et Bensimon N., Coopération entre un hypermédia adaptatif éducatif et un agent pédagogique H2PTM 03 : hypermedias hypertexts, products, tools and methods, Saint-Denis, France, Hermès, pp. 24-26, 2003.

[Richard et Tchounikine, 2004]

Richard B. et Tchounikine P., Une approche centrée modèle pour la construction d'un système conseiller pour un site web, IC 2004 - 15èmes Journées Francophones d'Ingénierie des Connaissances, Lyon, France, pp. 151-162, 2004.

[Richard, 2008]

Richard B., Une approche épiphyte pour la conception de systèmes conseillers, Université du Maine, 2 avril 2008.

[Ritter et Koedinger, 1995]

Ritter S. et Koedinger K. R., Toward lightweight tutoring agents, World Conference on Artificial Intelligence in Education (AIED'05), pp. 91-98, 1995.

[Rushing et al., 2009]

Rushing J., Berendes T., Lin H., Buntain C. et Graves S., Spyglass: A System for Ontology Based Document Retrieval and Visualization, FLAIRS, Florida, USA, 2009.

[Selker, 1994]

Selker T., COACH: a teaching agent that learns, Communication of the ACM, vol. 37 n°7, pp. 92-99, 1994.

[Simonin et Carbonell, 2007]

Simonin J. et Carbonell N., Interfaces Adaptatives - Adaptation dynamique à l'utilisateur courant, Paris, France, Hermès - Lavoisier, 2007.

[Stevens et Collins, 1977]

Stevens A. L. et Collins A., The goal structure of a Socratic Tutor, ACM '77 Proceedings of the 1977 annual conference, Seattle, USA, ACM pp. 256 - 263, 1977.

[Stuerzlinger et al., 2006]

Stuerzlinger W., Chapuis O., Phillips D. et Roussel N., User Interface Facades: Towards Fully Adaptable User Interfaces, UIST, Montreux, Switzerland, pp. 309-318, 2006.

[Swartz, 2003]

Swartz L., Why People Hate the Paperclip: Labels, Appearance, Behavior, and Social Responses to User Interface Agents, Master's report, Stanford University, Stanford, USA, 2003.

[Terrat et Sagot, 2011]

Terrat H. et Sagot J., Pictop, un outil informatique spécialisé pour accompagner la scolarisation des élèves handicapés dans la maîtrise de la langue, EIAH, Mons, Belgique, pp. 1-8, 2011.

[Thái et Ginon, 2014]

Thái L. V. et Ginon B., Exploitation d'assistances épiphytes en contexte éducatif, RJC-EIAH, La Rochelle, France, 2014.

[Thevenet et al., 2012]

Thevenet Q., Lefevre M., Cordier A. et Barnachon M., Intelligent Interactions: Articial Intelligence and Motion Capture for Negotiation of Gestural Interactions, Workshop TRUE, conference ICCBR, Lyon, France, 2012.

[Villiot-Leclercq, 2006]

Villiot-Leclercq E., Conception de scénarios pédagogiques : un dispositif d'assistance pour soutenir l'interaction entre l'enseignant et l'environnement ExploraGraph, colloque Scénarios, Pernin J-P. G. H., Lyon, France, pp. 83-88, 2006.

[Wenger et Snyder, 2000]

Wenger E. et Snyder W. M., Communities of practice: The organizational frontier, Harvard Business Review, vol. 78 n°1, pp. 139-145, 2000.

[Wolframe, 2014]

Wolframe S., http://blog.stephenwolfram.com/2012/03/the-personal-analytics-of-my-life/, (dernière visite: 2014).

[Zapata-Rivera et Greer, 2004]

Zapata-Rivera J. D. et Greer J., Interacting with Inspectable Bayesian Student Models, International Journal of Artificial Intelligence in Education, vol. 14, pp. 1-37, 2004.

[Zarka et al., 2010]

Zarka R., Cordier A., Corvaisier F. et Mille A., Providing assistance by reusing episodes stored in traces: a case study with SAP Business Objects Explorer, Atelier Français de Raisonnement à Partir de Cas, Florence Le Ber J. R. e., Strasbourg, France, pp. 91-103, 2010.

[Zarka et al., 2012]

Zarka R., Cordier A., Egyed-Zsigmond E. et Mille A., Contextual Trace-Based Video Recommendations., WWW-XperienceWeb, Lyon, France, pp. 751-754, 2012.

[Zarka, 2013]

Zarka R., Trace-Based Reasoning for User Assistance and Recommendations, Phd thesis, Lyon 1

Références nétographiques

[Accerciser, 2014]

Accerciser, Outil d'inspection des applications GTK pour GNOME, https://wiki.gnome.org/action/show/Apps/Accerciser?action=show&redirect=Accerciser, (dernière visite: 2014).

[AICC, 2014]

AICC, Standard AICC, http://www.aicc.org/joomla/dev/, (dernière visite: 2014).

[Apple_API_Accessibility, 2014]

Apple_API_Accessibility, Bibliothèque d'accessibilité pour Mac OS, https://developer.apple.com/library/mac/documentation/Accessibility/Conceptual/AccessibilityMacOSX/OSXAXIntro/OSXAXintro.html, (dernière visite: 2014).

[ATK/AT-SPI, 2014]

ATK/AT-SPI, Bibliothèque d'accessibilité pour Linux, http://www.linuxfoundation.org/collaborate/workgroups/accessibility/atk/at-spi, (dernière visite: 2014).

[AutoHotKey, 2014]

AutoHotKey, Outil de création de scripts, http://www.autohotkey.com/, (dernière visite: 2014).

[AutoIt, 2014]

AutoIt, Outil de création de scripts, http://www.autoitscript.com/site/autoit/, (dernière visite: 2014).

[Dora_Jeux, 2014]

Dora_Jeux, Jeu en ligne pour enfants, http://www.tfou.fr/dora-l-exploratrice/jeux/le-spectacle-de-fin-d-annee-7817399-739.html, (dernière visite: 2014).

[DoubleAgent, 2014]

DoubleAgent, Agents animés pour Windows, http://www.cinnamonsoftware.com/double_agent.htm, (dernière visite: 2014).

[EndNote, 2014]

EndNote, Outil de gestion bibliographique, http://endnote.com/, (dernière visite: 2014).

[FlyHelp, 2014]

FlyHelp, Outil de création de manuels d'aide en ligne, http://www.flyskysoft.com/, 2014.

[Google, 2014]

Google, Navigateur Chrome, http://www.google.com/intl/fr_fr/chrome/browser/, (dernière visite: 2014).

[Grease_monkey, 2014]

Grease_monkey, Extension de navigateur Grease monkey, https://addons.mozilla.org/fr/firefox/addon/greasemonkey/, (dernière visite: 2014).

[Gurgeh, 2014]

Gurgeh, Outil d'enregistrement de logs, https://github.com/gurgeh/selfspy, (dernière visite: 2014).

[HelpScribble, 2014]

HelpScribble, Outil de création de manuels d'aide, http://www.helpscribble.com/winhelp.html, (dernière visite: 2014).

[html-help-com-assistant, 2014]

html-help-com-assistant, Outil de création de documentation HTML, http://telecharger.logiciel.net/html-help-com-assistant/, (dernière visite: 2014).

[INS_HEA, 2014]

INS_HEA, Logiciel PicTop pour favoriser l'apprentissage de la lecture chez les enfants handicapés, http://laboutique.inshea.fr/pictop-2,fr,4,Lo2.cfm, (dernière visite: 2014).

[Inspect, 2014]

Inspect, Outil d'inspection des exécutables Windows, http://msdn.microsoft.com/en-us/library/windows/desktop/dd318521(v=vs.85).aspx, (dernière visite: 2014).

[It's_learning, 2014]

It's_learning, Environnement numérique de travail, http://www.itslearning.fr/, (dernière visite: 2014).

[Jacareto, 2014]

Jacareto, Outil d'enregistrement et de rejouage de séquences d'événements pour les applications Java, http://sourceforge.net/apps/mediawiki/jacareto/index.php?title=Main_Page, (dernière visite: 2014).

[Java_Ferret, 2014]

Java_Ferret, Outil de détection d'événements pour les applications Java, http://docs.oracle.com/cd/E17802_01/j2se/javase/technologies/accessibility/docs/jaccess-1.3/doc/Ferret.html, (dernière visite: 2014).

[Java_Monkey, 2014]

Java_Monkey, Outil d'inspection des applications Java, http://docs.oracle.com/cd/E17802_01/j2se/javase/technologies/accessibility/docs/jaccess-1.3/doc/Monkey.html, (dernière visite: 2014).

[Lancome, 2014]

Lancome, Site web commercial de Lancôme, http://www.lancome.fr/_fr/_fr/, 2014.

[Malacria, 2014]

Malacria, Outil d'inspection des menus pour les applications sous Mac OS, http://menuinspector.malacria.fr/, (dernière visite: 2014).

[Marathon, 2014]

Marathon, Outil d'automatisation pour les applications Java, http://marathontesting.com/, (dernière visite: 2014).

[Mozilla, 2014]

Mozilla, Navigateur Firefox, http://www.mozilla.org/fr/, (dernière visite: 2014).

[MsAgents, 2014]

MsAgents, Agents animés pour Windows, http://www.microsoft.com/products/msagent/main.aspx, (dernière visite: 2014).

[Office_2003, 2014]

Office_2003, Aide d'Office 2003, http://office.microsoft.com/fr-fr/onenote-help/presentation-de-microsoft-office-2003-HA001073117.aspx?CTT=1, (dernière visite: 2014).

[Office_2010, 2014]

Office_2010, Aide d'Office 2010, http://office.microsoft.com/fr-fr/support/premiers-pas-avec-microsoft-office-2010-FX100996114.aspx (dernière visite: 2014).

[PowerCHM, 2014]

PowerCHM, Compiled Windows HTML Help files, Format d'aide, http://www.dawningsoft.com/powerchm.html, 2014.

[Pronote, 2014]

Pronote, Logiciel de gestion de notes, http://www.index-education.com/fr/pronote-info191-pronote-net.php, (dernière visite: 2014).

[QuantifedSelf, 2014]

QuantifedSelf, Outil d'enregistrement de logs, http://quantifiedself.com/, (dernière visite: 2014).

[Scorm, 2014]

Scorm, Norme pour l'interopérabilité des activités pédagogiques, http://scorm.com/, (dernière visite: 2014).

[Scorm_driver, 2014]

Scorm_driver, Outil de conversion de contenus pédagogiques, http://scorm.com/scorm-solved/scorm-driver/, (dernière visite: 2014).

[Sésamath, 2012]

Sésamath, Communauté de pratiques destinée aux enseignants de mathématiques, http://www.sesamath.net/ (dernière visite: 2012).

[Sikuli, 2014]

Sikuli, Outil d'automatisation, http://www.sikuli.org/, (dernière visite: 2014).

[Tamper_monkey, 2014]

Tamper_monkey, Extension de navigateur, https://chrome.google.com/webstore/detail/tampermonkey/dhdgffkkebhmkfjojejmpbldmpobfkfo?hl=fr, (dernière visite: 2014).

[Tincan, 2014]

Tincan, Norme pour les activités pédagogiques, http://tincanapi.com/, (dernière visite: 2014).

[Tutoriels_animes, 2014]

Tutoriels_animes, site web proposant des tutoriels sous forme d'animations interactives, http://www.tutoriels-animes.com/, (dernière visite: 2014).

[UISpy, 2014]

UISpy, Outil d'inspection des exécutables Windows, http://msdn.microsoft.com/fr-fr/library/vstudio/ms727247(v=vs.100).aspx, (dernière visite: 2014).

[W3C, 2014]

W3C, Web Content Accessibility Guidelines, http://www.w3.org/TR/WCAG20/, (dernière visite: 2014).

[WinHelp, 2014]

WinHelp, Outil auteur pour la création de manuels d'aide, http://www.helpscribble.com/winhelp.html, (dernière visite: 2014).

[Wolframe, 2014]

Wolframe S., Outil d'enregistrement de logs, http://blog.stephenwolfram.com/2012/03/the-personal-analytics-of-my-life/, (dernière visite: 2014).

Annexes

Plan des annexes

Annexe A. Définition en aLDEAS d'assistances existantes

Afin d'évaluer la couverture d'aLDEAS, nous l'avons utilisé pour définir différentes assistances existantes. Nous détaillons ici plusieurs exemples d'assistances représentatives, fréquemment rencontrées dans les applications informatiques existantes : présentation d'une application ou d'une fonctionnalité, aide à la saisie de formulaire et tutoriel. Ces exemples ont été choisis selon la richesse de l'assistance proposée, et la variété de son contexte d'utilisation.

1. Présentation d'une application ou d'une fonctionnalité

De nombreuses applications informatiques proposent une assistance pour la première utilisation de l'application ou d'une fonctionnalité. De telles assistances peuvent être représentées en aLDEAS. À titre d'exemple, nous présentons la définition en aLDEAS d'assistances proposées par les applications CAD-KAS PDF Editor, YASGUI, Ikea et Dora l'exploratrice, qui proposent quatre types d'assistances typiques de ce qui peut être proposé au démarrage d'une application ou lors de la découverte d'une nouvelle fonctionnalité.

La Fig. 27 présente la définition en aLDEAS de l'assistance proposée par l'application CAD-KAS PDF Editor[21] lors de la première utilisation. Cette assistance prend la forme d'un message (cf. Fig. 1) proposant à l'utilisateur final de visionner une vidéo de démonstration.

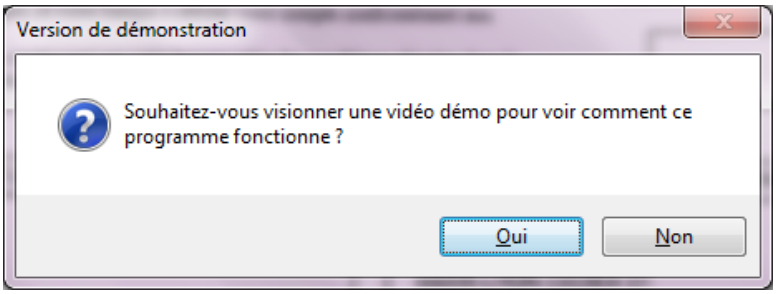


Fig. 1 – Assistance proposée lors de la première utilisation de CAD-KAS PDF Editor

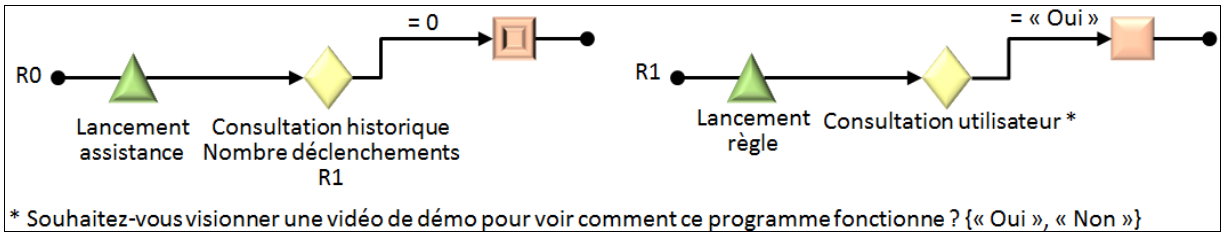


Fig. 2 – Définition en aLDEAS d'une assistance équivalente à celle proposée lors de la première utilisation de CAD-KAS PDF Editor

[21] Application téléchargeable sur http://cad-kas-pdf-editor.softonic.fr/

Lors d'une première utilisation, l'application Web YASGUI[22], qui fournit une interface graphique pour les requêtes SPARQL, propose une assistance sous la forme de messages et de mises en valeur (cf. Fig. 3). Cette assistance technique présente les principales fonctionnalités offertes par l'application, ainsi que quelques astuces pour faciliter l'utilisation de YASGUI. Une telle assistance peut être représentée avec aLDEAS, comme présentée en Fig. 4. En revanche, lors de l'exécution de cette assistance par le système SEPIA, l'aspect de l'assistance ne sera pas exactement le même que celui de l'assistance proposée par YASGUI. En effet, SEPIA ne propose pour l'instant pas d'épi-assistant capable d'afficher un message sous forme d'une bulle qui met en valeur le composant concerné par le message. Néanmoins, afin de respecter l'objectif de l'assistance de YASGUI, SEPIA peut faire appel à un épi-assistant pour afficher le message à proximité du composant, accompagné d'une mise en valeur de ce composant, réalisé par un second épi-assistant. SEPIA peut également faire appel à un épi-assistant impliquant un agent animé, qui est capable d'afficher un message tout en mettant en valeur le composant. Cette forme d'assistance semble néanmoins peu appropriée au public cible de YASGUI.

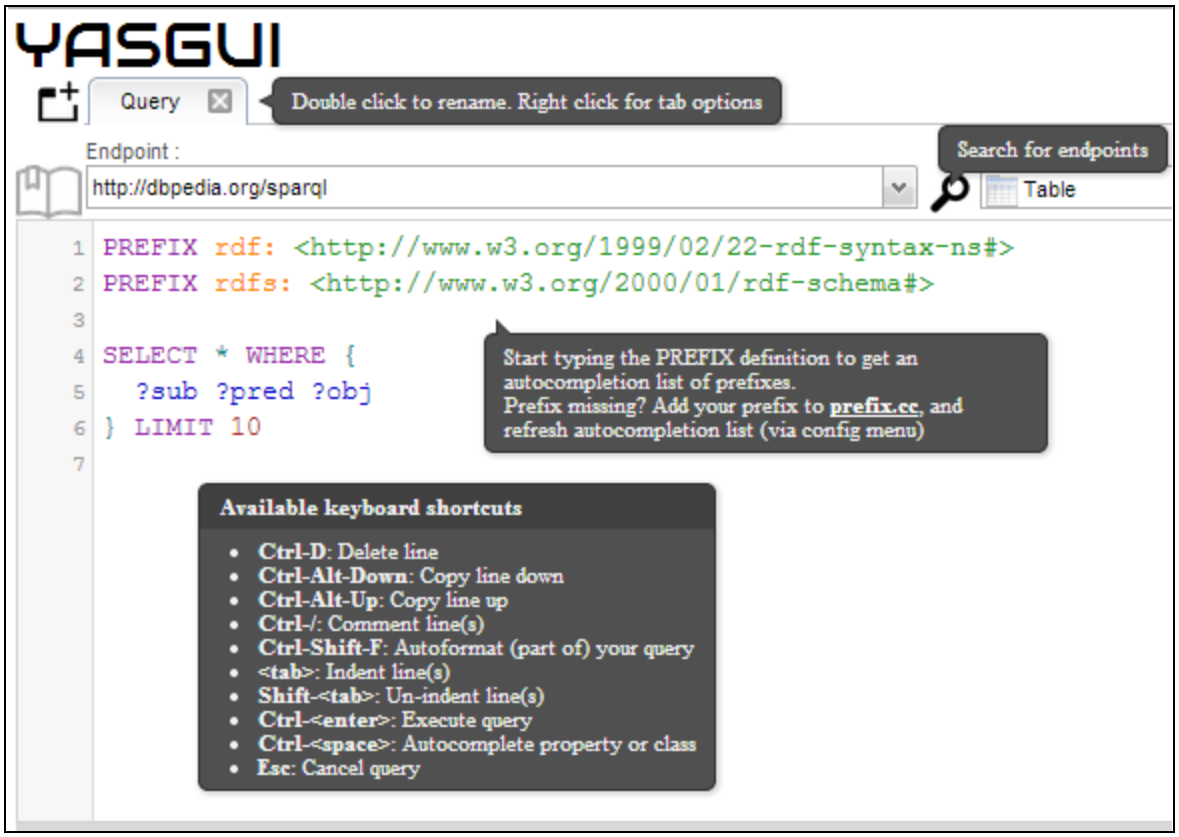


Fig. 3 – Assistance proposée lors de la première utilisation de YASGUI

[22] http://laurensrietveld.nl/yasgui/

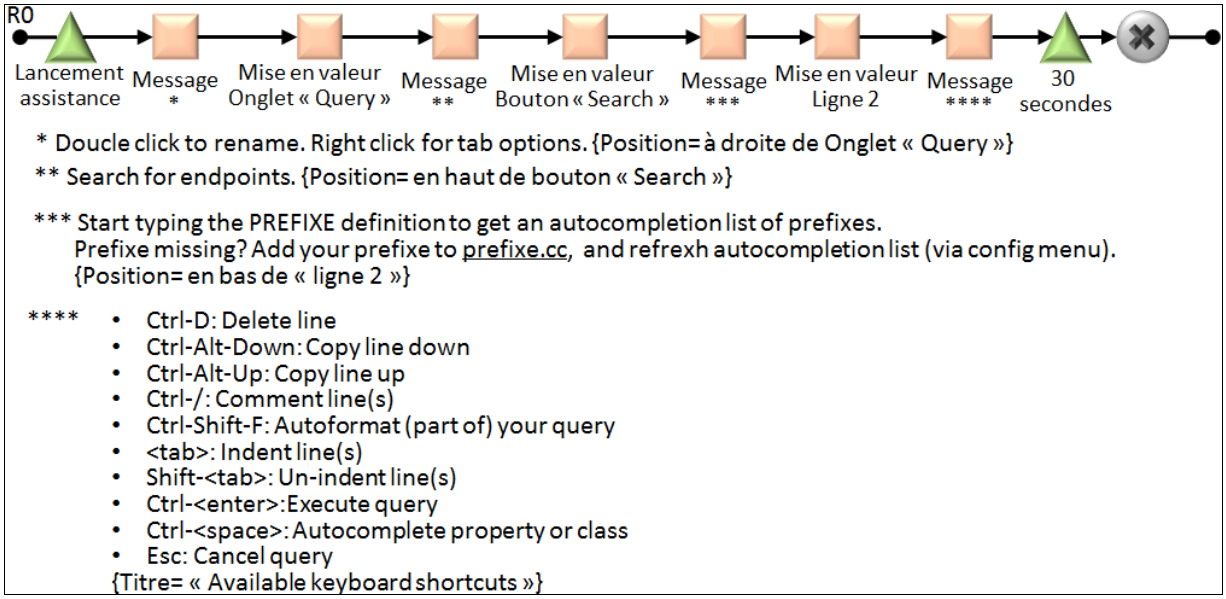


Fig. 4 – Définition en aLDEAS d'une assistance équivalente à celle proposée lors de la première utilisation de YASGUI

La Fig. 5 présente des captures d'écrans de trois étapes de l'assistance proposée par Ikea[23], pour faciliter l'utilisation d'un outil de conception de rangements. L'assistance se manifeste par des fenêtres pop-up qui guident l'utilisateur lors de la conception d'un placard aménagé. Lors de la première étape (cf. partie supérieure de la Fig. 5), pendant laquelle l'utilisateur doit sélectionner la gamme de rangements qu'il souhaite, l'assistance explique à l'utilisateur à quoi sert cet outil de conception et présente les trois principales étapes de la conception d'un placard aménagé. L'assistance Ikea réagit aux actions de l'utilisateur. Ainsi, lorsqu'il sélectionne une gamme de produits, l'utilisateur déclenche le passage à la seconde étape de l'assistance, qui consiste à choisir, positionner et personnaliser des éléments de rangement. Lors de cette étape, l'assistance présente à l'utilisateur les différents outils à sa disposition (cf. partie inférieure de la Fig. 5). Là encore, l'assistance réagit aux actions de l'utilisateur : lorsqu'il clique sur un outil, l'assistance le lui présente en détails.

[23] http://www.ikea.com/ms/fr_FR/rooms_ideas/planner_pax3d/index.html

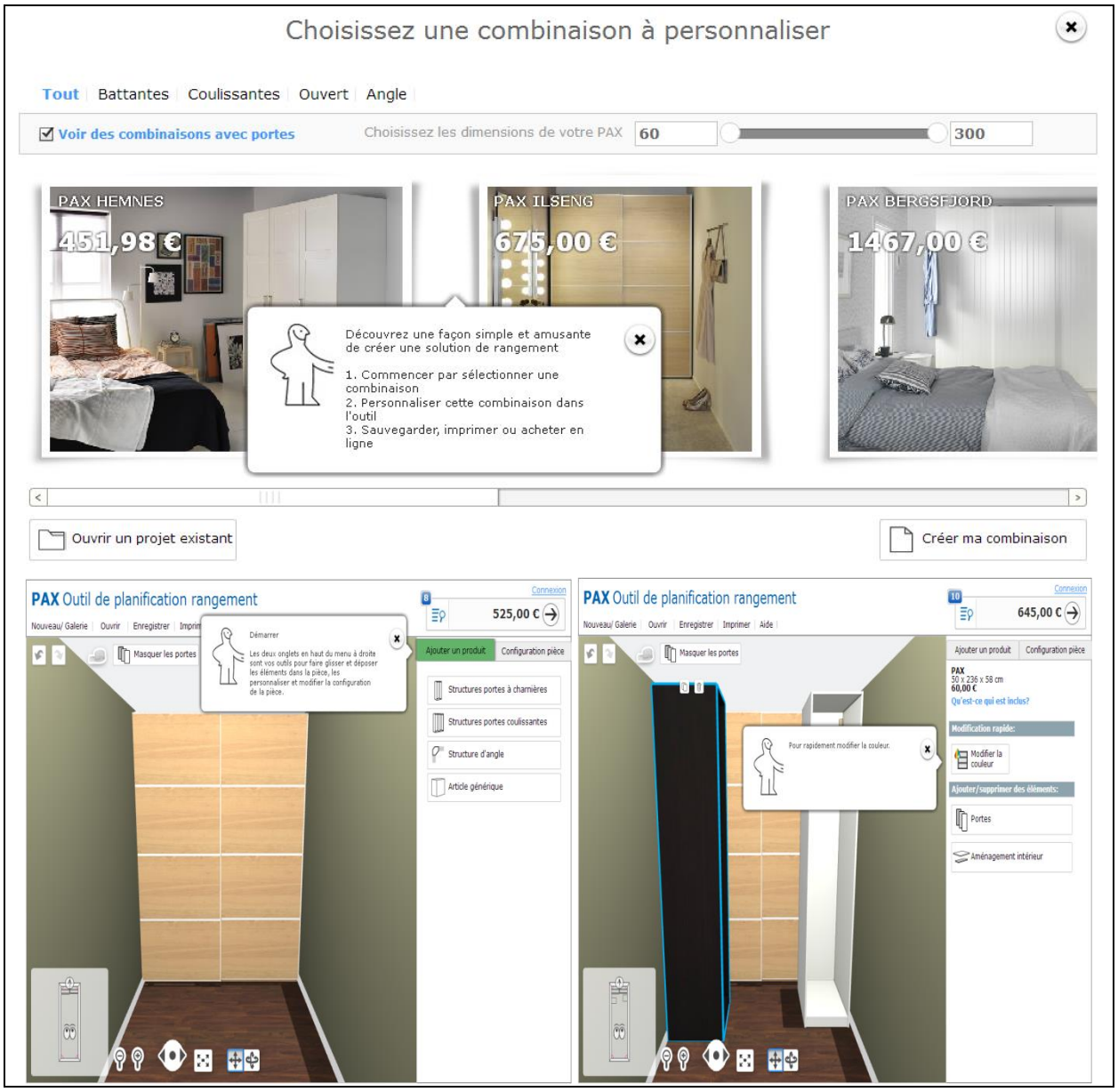


Fig. 5 – Assistance proposée par le site web Ikea

Une telle assistance peut être représentée avec aLDEAS : la Fig. 6 donne la définition en aLDEAS de trois règles d'assistance qui correspondent à l'assistance affichée en Fig. 5. Si aLDEAS permet de définir une assistance équivalente à celle proposée par Ikea, lors de son exécution par SEPIA, l'assistance définie en aLDEAS n'aura pas exactement la même forme que celle proposée par Ikea. En effet, tout comme celle de YASGUI (cf. Fig. 3), l'assistance d'Ikea propose une action d'assistance qui est un mélange entre les messages et les mises en valeur proposées par aLDEAS. L'assistance fournie par SEPIA devra donc faire appel à un épi-assistant pour afficher un message à proximité du composant à mettre en valeur, ainsi qu'à un épi-assistant pour le mettre en valeur, par exemple par l'affichage d'une flèche. SEPIA ne propose pas d'épi-assistant capable d'afficher des messages dans des fenêtres pop-up avec un contour contenant une flèche, comme c'est le cas des messages affichés par Ikea.

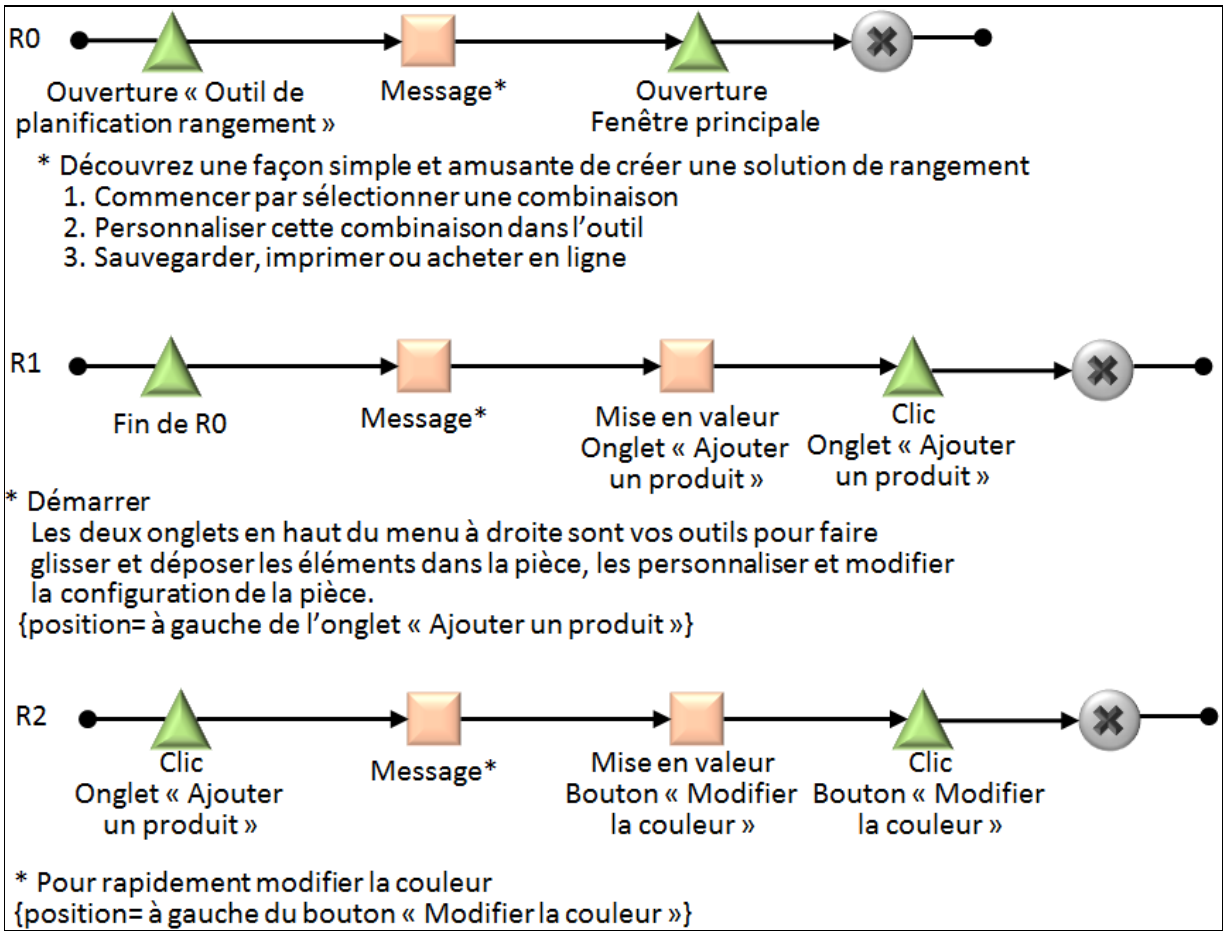


Fig. 6 – Trois règles d'assistance en aLDEAS pour représenter une partie de l'assistance proposée par Ikea

Le jeu en ligne Dora l'exploratrice « Le spectacle de fin d'année »[24] est destiné à de jeunes enfants. Il consiste à créer un spectacle en sélectionnant les mouvements qu'un ou deux personnages exécuteront ensuite. L'enfant peut également choisir le décor du spectacle. Ce jeu propose une assistance qui se déclenche au démarrage du jeu et à la suite d'une inactivité de l'utilisateur durant 5 secondes. Cette assistance est une présentation guidée : une flèche bleue désigne un à un les composants de l'interface pendant qu'une synthèse vocale explique leur fonction.

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

Fig. 7 – Assistance proposée par le jeu « Dora l'exploratrice : le spectacle de fin d'année ».

L'assistance proposée par le jeu Dora l'exploratrice peut être représentée avec aLDEAS, notamment en utilisant le patron de présentation guidée (cf. Fig. 8). L'étape 1 de la présentation guidée A0 est détaillée à titre d'exemple, chaque étape est similaire mais concerne un composant différent. L'étape 1 correspond à la Fig. 7 : cette étape concerne le bouton *Mouvement 1*, qui est mis en valeur par une flèche bleue.

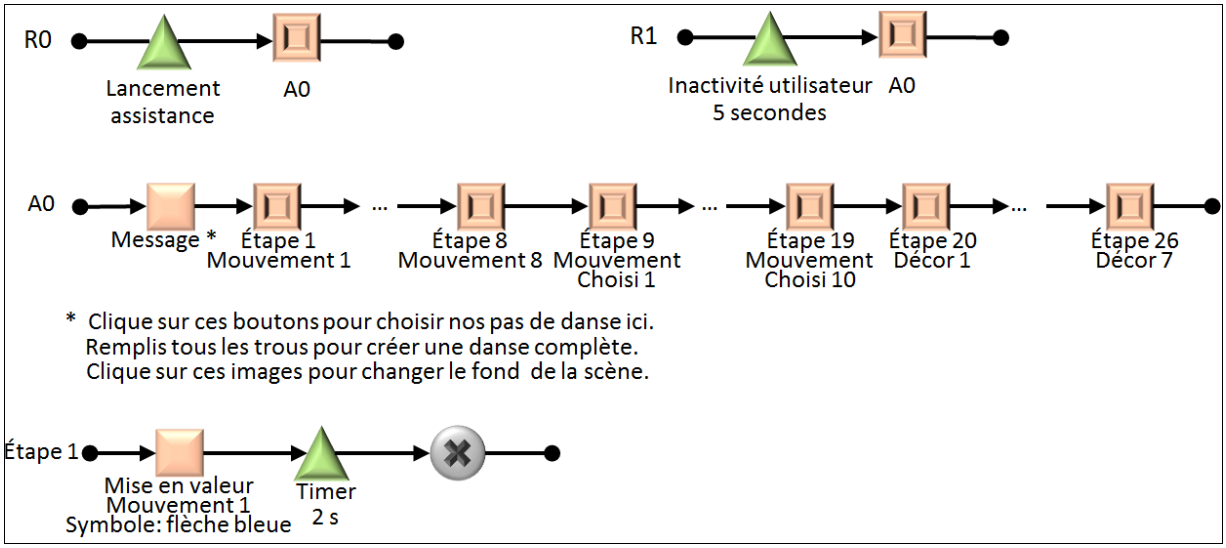


Fig. 8 – Représentation en aLDEAS de l'assistance proposée par le jeu « Dora l'exploratrice : le spectacle de fin d'année »

2. Aide à la saisie dans un formulaire

De nombreuses applications informatiques, en particulier les applications web, proposent de l'assistance pour remplir des formulaires. Cette assistance peut notamment consister à indiquer à l'utilisateur qu'un champ du formulaire n'est pas rempli, ou est incorrectement rempli. Elle peut prendre la forme de messages d'aide comme sur le site de simulation de prêts immobiliers du Crédit Mutuel[25] (cf. Fig. 9) ou sur le site des allocations familiales[26] (cf. Fig. 11), et/ou sous la forme de mises en valeur des erreurs, comme sur le site des impôts[27] (cf. Fig. 12). Ces messages d'aide peuvent être simplement indicatifs, par exemple « veuillez saisir un taux » (cf. partie gauche de la Fig. 9), ou explicatifs, par exemple « Saisissez votre numéro fiscal à 13 chiffres figurant en haut de la première page de votre déclaration de revenus » (cf. Fig. 12). Dans un dernier exemple, l'assistance indique à l'utilisateur que ce champ doit être rempli, mais lui explique également où il peut trouver cette information.

De telles assistances à la saisie de formulaires peuvent généralement être représentées en aLDEAS. À titre d'exemple, la Fig. 10 présente la représentation en aLDEAS de l'assistance proposée par le site du Crédit Mutuel pour aider l'utilisation de son simulateur de prêts immobiliers, dont deux exemples de messages d'aide sont donnés en Fig. 9. La Fig. 13 présente quant à elle la représentation en aLDEAS de l'assistance proposée par le site des impôts, affichée en Fig. 12.

[25] https://www.creditmutuel.fr/cmcee/fr/banques/particuliers/quotidien/financer-vos-projets/simulation-credit.html
[26] https://wwwd.caf.fr/wps/portal/caffr/login/
[27] https://cfspart.impots.gouv.fr/Login

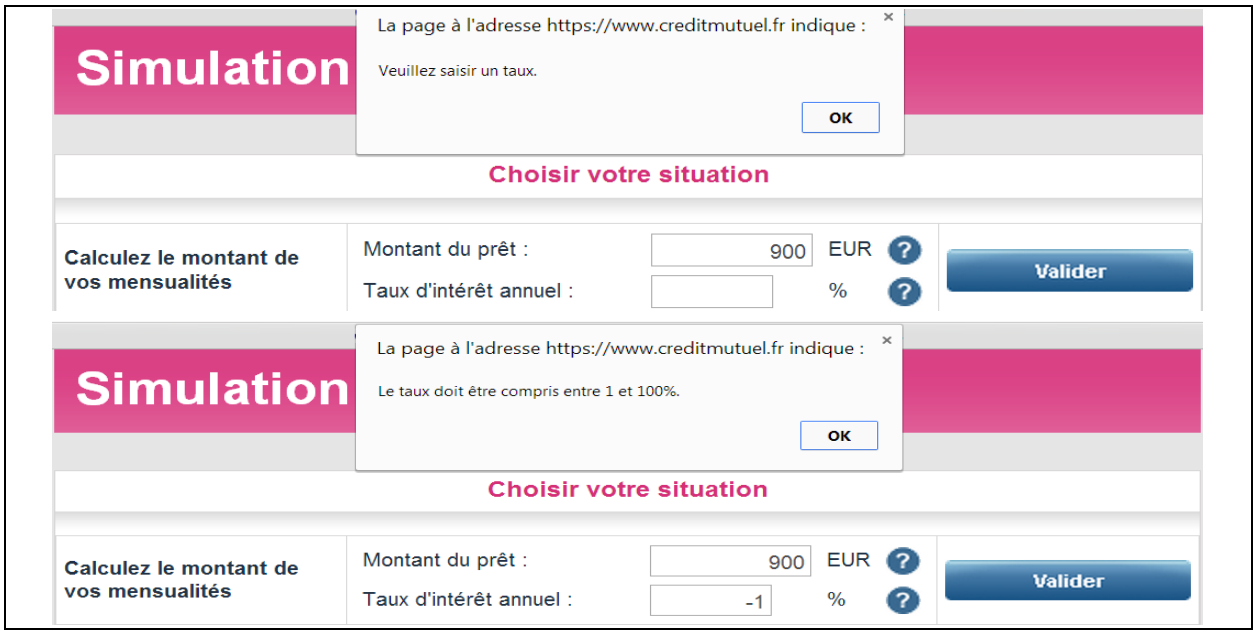


Fig. 9 – Assistance proposée par le site Web du Crédit Mutuel

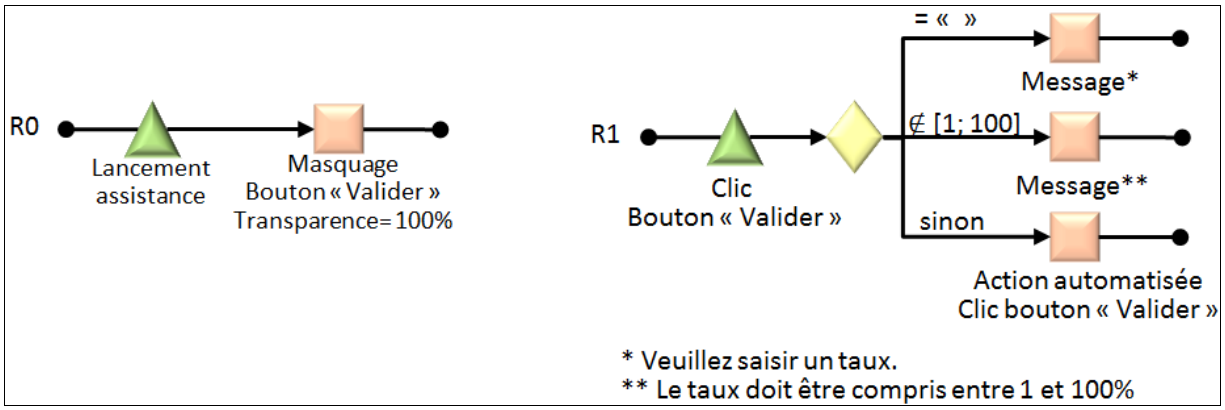


Fig. 10 – Représentation en aLDEAS de l'assistance proposée par le Crédit Mutuel

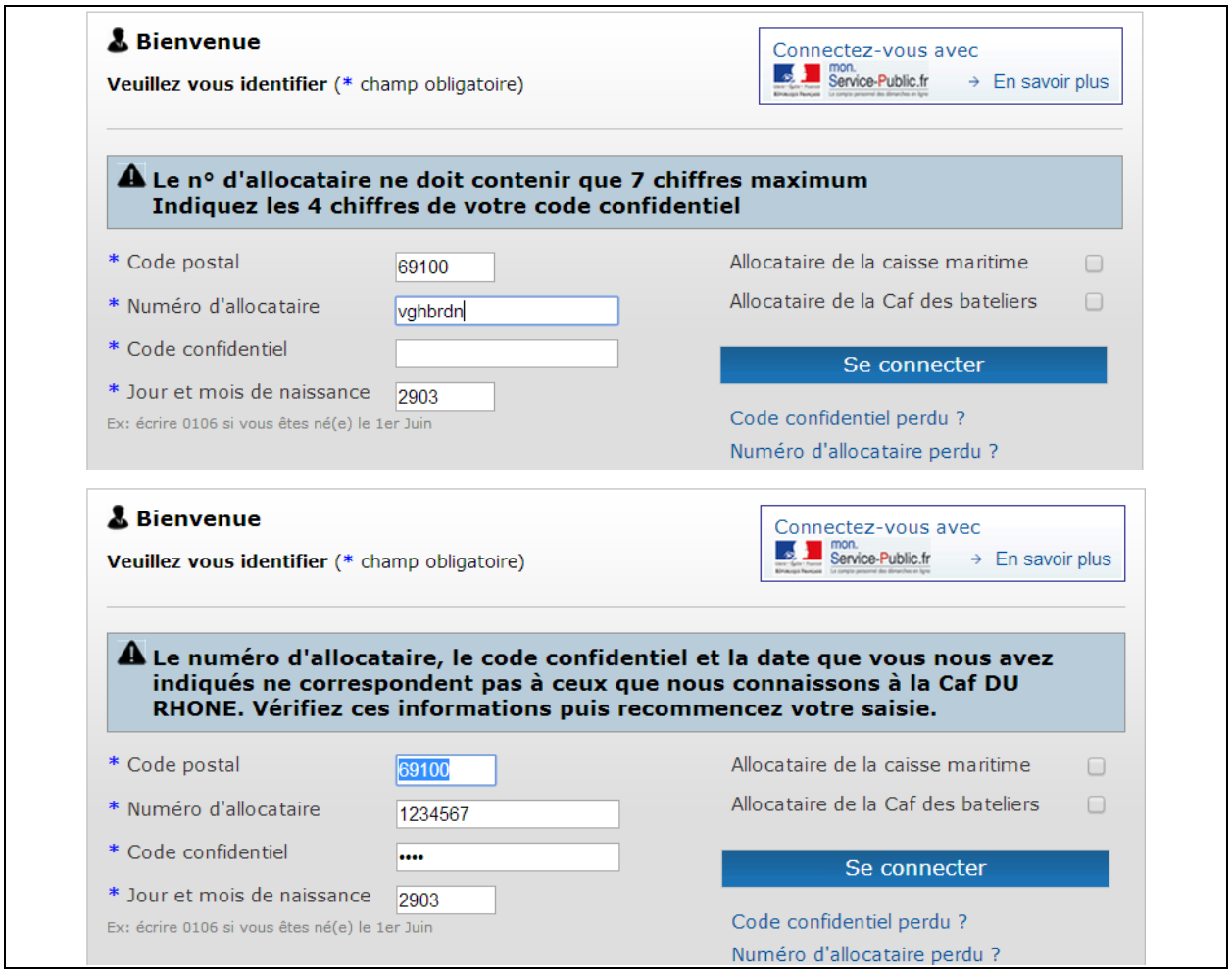


Fig. 11 – Assistance proposée par le site de la CAF sous la forme de messages d'aide

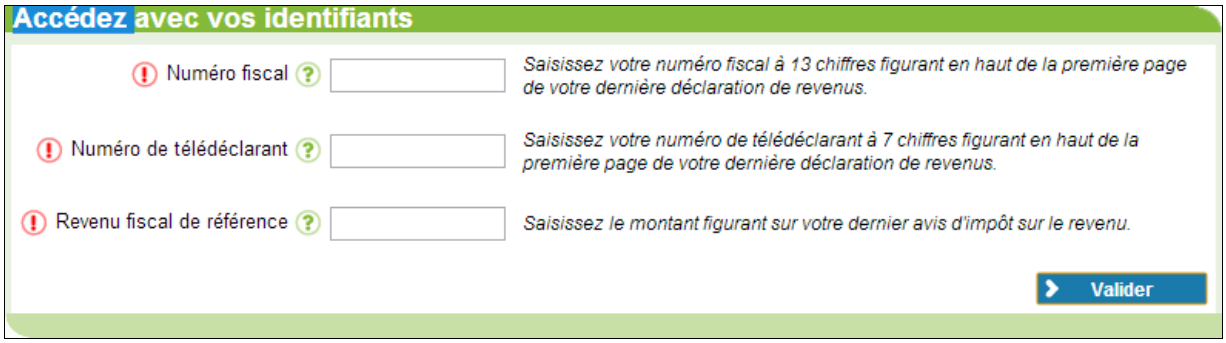


Fig. 12 – Assistance proposée par le site des impôts sous la forme de messages d'aide et de mises en valeur

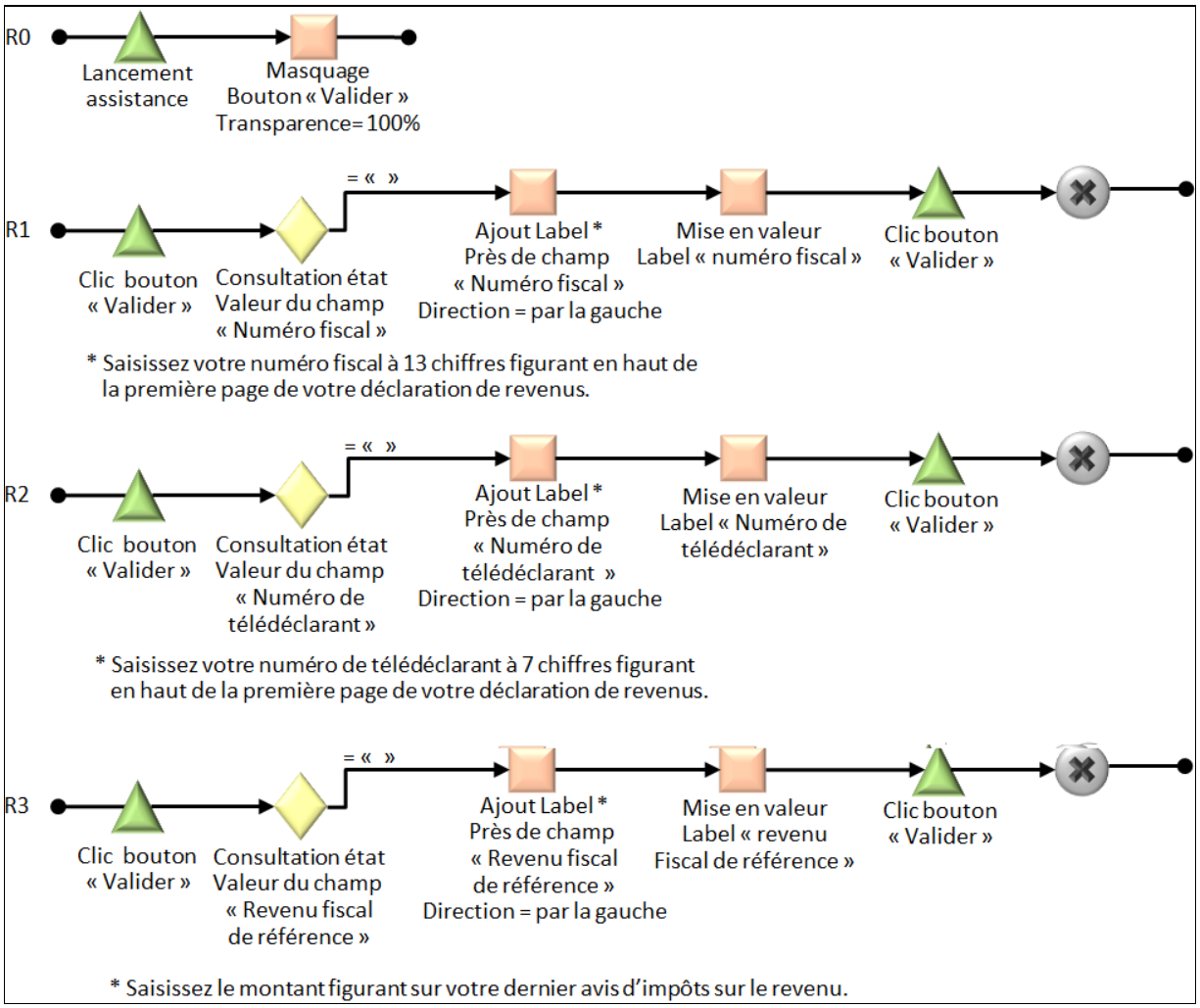


Fig. 13 – Représentation en aLDEAS de l'assistance proposée par le site des impôts

Dans le cas d'une assistance à la saisie de formulaires, le système d'assistance doit intercepter le clic de l'utilisateur sur le composant qui permet de le valider. Le système d'assistance doit ensuite analyser les champs du formulaire avant de retransmettre le clic de l'utilisateur à l'application-cible, ce qui aura pour effet de valider le formulaire, ou de fournir de l'assistance à l'utilisateur si besoin. Avec aLDEAS et sa mise en œuvre à travers SEPIA, une manière d'intercepter un clic consiste à réaliser un masquage transparent sur le composant. Ce masquage sera invisible pour les utilisateurs finaux mais il interceptera les clics de l'utilisateur : celui-ci cliquera en effet sur le masquage et non directement sur le composant.

Néanmoins, le langage aLDEAS ne permet pas de reproduire toutes les assistances à la saisie de formulaires. En effet, s'il est facile de vérifier qu'un champ a été rempli, il est plus difficile de juger de la validité du contenu de ce champ ; des connaissances supplémentaires sont alors nécessaires. Parfois, ces connaissances sont faciles à acquérir pour le système d'assistance à la condition que le concepteur de l'assistance accepte de les représenter avec aLDEAS. Prenons l'exemple du simulateur de prêts immobiliers du Crédit Mutuel. De manière très classique, le système d'assistance vérifie dans un premier temps que tous les champs obligatoires ont été remplis. Le système d'assistance n'ayant aucune connaissance sur

l'application-cible, c'est au concepteur de l'assistance d'indiquer quels sont les champs obligatoires. Ceci peut être fait par une règle d'assistance consultant le contenu d'un champ obligatoire puis le comparant avec une chaîne de caractères vide, comme dans l'exemple de la règle R1 de la Fig. 27. Ensuite, pour vérifier la validité du contenu du champ, le concepteur de l'assistance doit indiquer quelles sont les valeurs valides, en les définissant par un intervalle numérique, comme c'est le cas pour le taux d'intérêt du prêt qui doit appartenir à [1 ; 100], ou par une expression régulière. Ce travail peut être long et répétitif pour le concepteur de l'assistance, il est néanmoins possible avec aLDEAS de permettre au système d'assistance d'acquérir certaines connaissances, comme le fait que pour le simulateur de prêts du Crédit Mutuel, un taux d'intérêt doit être inclus dans l'intervalle [1 ; 100]. En revanche, dans certains cas, aLDEAS ne permet pas de consulter ces informations. Par exemple, pour les sites de la CAF et des impôts, aLDEAS et sa mise en œuvre à travers SEPIA ne permettent pas de vérifier les informations confidentielles des utilisateurs, qui sont connues de l'application-cible, mais qui ne sont heureusement pas accessibles de l'extérieur. Il est cependant envisageable de proposer une assistance dans laquelle le concepteur de l'assistance représenterait avec aLDEAS des informations confidentielles, pour l'usage exclusif d'une personne. Par exemple, quelqu'un pourrait concevoir un système d'assistance qui pré-remplirait les informations personnelles d'une personne âgée ayant de grosses difficultés à réaliser des démarches administratives en ligne. Ce genre de pratiques n'est pas sécurisé, mais elles ne le sont finalement pas moins que des pratiques consistant à inscrire des informations confidentielles sur un post-it collé sur l'ordinateur.

3. Tutoriels en ligne

Sur internet, il existe de très nombreux tutoriels pour des applications-cibles et des tâches très variées, des plus simples, principalement destinées au grand public, aux plus complexes, destinées à un public averti. Ces tutoriels peuvent prendre différentes formes. Tout d'abord, il existe des tutoriels textuels, avec une série d'instructions courtes et claires, souvent complétées par des captures d'écrans illustrant une instruction et contenant éventuellement des mises en valeur. Les Fig. 14 et Fig. 15 sont deux exemples de tels tutoriels, respectivement pour la modification de l'arrière-plan sur une image avec Paint[28] et pour exporter un montage au format MP3 avec l'éditeur audio Audacity[29].

Le langage aLDEAS propose des actions de type *lancement de ressources*, qui peuvent lancer pour l'utilisateur final de tels tutoriels, qu'il s'agisse de fichiers (pdf, ppt…) ou de pages web. De plus, aLDEAS et sa mise en œuvre à travers SEPIA permettent de fournir à l'utilisateur final une assistance inspirée de tels tutoriels, mais plus intégrée dans l'application-cible. À titre d'exemple, la Fig. 16 présente la représentation en aLDEAS d'une assistance inspirée d'une partie du tutoriel pour Audacity (cf. Fig. 15).

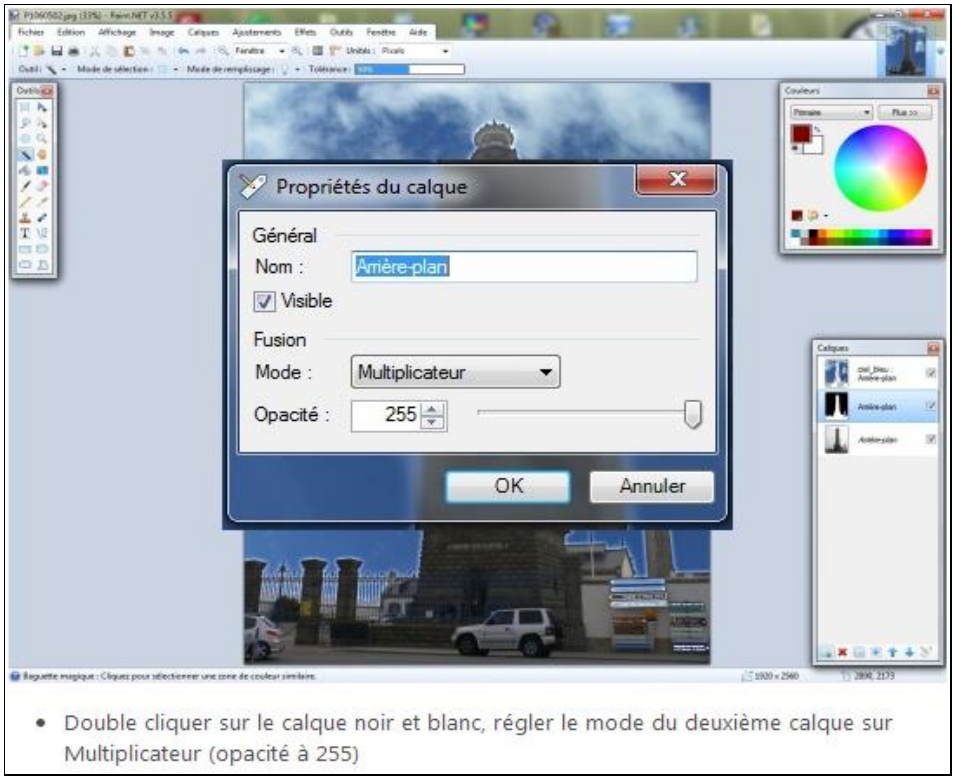


Fig. 14 – Extrait d'un tutoriel en ligne[30] pour Paint : modification de l'arrière-plan

[28] http://paintnet.fr/
[29] http://audacity.fr/
[30] http://www.zapwallpaper.fr/blog/paint-net-changer-la-couleur-du-ciel-dune-photo/

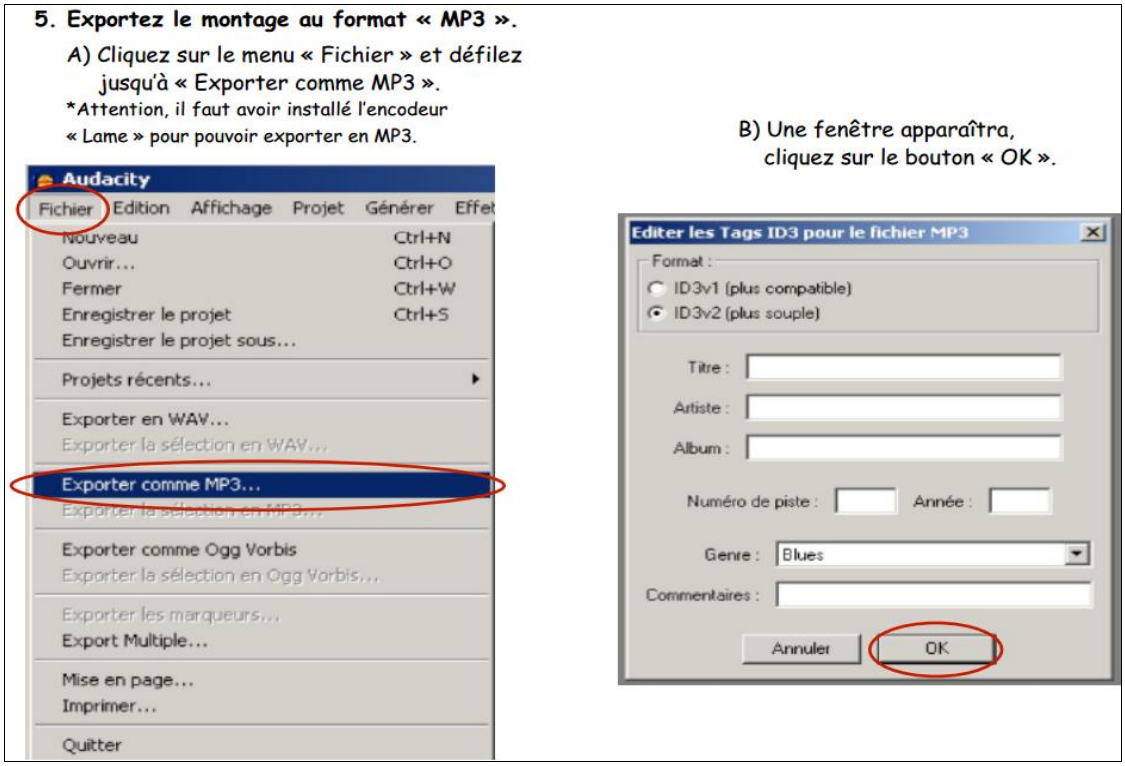


Fig. 15 – Extrait d'un tutoriel en ligne[31] pour Audacity : exporter un montage au format MP3

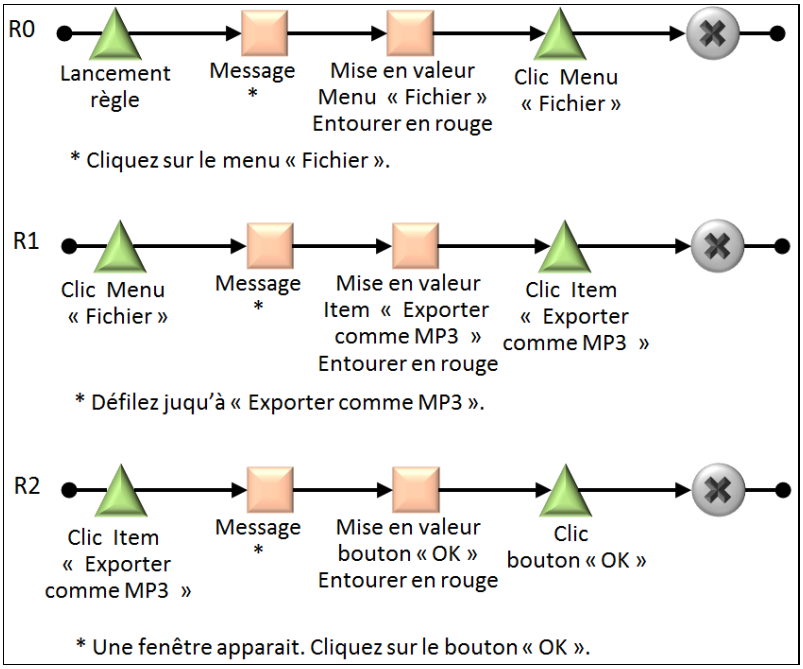


Fig. 16 – Représentation en aLDEAS d'un partie de l'assistance proposée pour Audacity

[31] http://www.educlasse.ch/activites/coursinfo/documents/Audacity_tutoriel.pdf

Beaucoup de tutoriels ont la forme d'une vidéo, qui donne des explications tout en montrant la réalisation d'une tâche donnée. Le langage aLDEAS et sa mise en œuvre à travers SEPIA permettent bien sûr le lancement d'une vidéo de démonstration existante. De plus, ils permettent de fournir à l'utilisateur final une assistance équivalente à celle proposée dans ces vidéos de démonstration, mais avec l'avantage d'être intégrée à l'interface de l'application-cible.

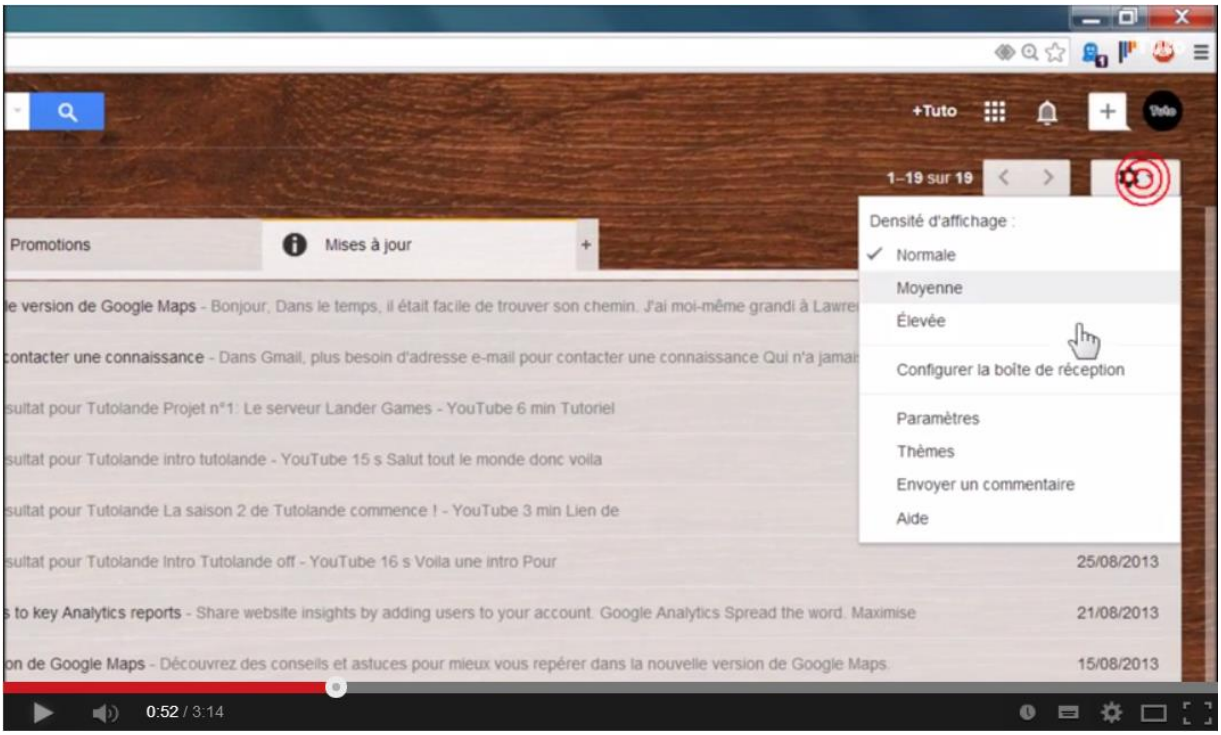


Fig. 17 – Capture d'écran d'un tutoriel vidéo en ligne[32] pour Gmail : ajout d'un volet d'aperçu pour le courriel ouvert

Fig. 18 – Captures d'écrans de deux étapes du tutoriel vidéo en ligne pour Gmail

[32] http://www.educlasse.ch/activites/coursinfo/documents/Audacity_tutoriel.pdf

À titre d'exemple, prenons la première partie de la vidéo de démonstration[33] qui explique comment paramétrer Gmail[34] pour ajouter un volet d'aperçu du courriel ouvert à côté du traditionnel volet d'affichage de la liste des courriels (cf. Fig. 17). Nous avons défini avec aLDEAS un système d'assistance qui guide l'utilisateur pour l'ajout d'un volet d'aperçu, suivant les mêmes étapes et explications que dans cette vidéo (cf. Fig. 19). Par exemple, la règle d'assistance R3 (cf. Fig. 19) correspond à l'étape au cours de laquelle l'utilisateur doit activer l'option « volet aperçu » ; la partie gauche de la Fig. 18 présente une capture d'écran de la vidéo de démonstration pendant cette même étape. La règle d'assistance R5 (cf. Fig. 19) correspond à l'étape au cours de laquelle l'utilisateur peut faire afficher à l'interface de Gmail ce volet d'aperçu ; la partie droite de la Fig. 18 présente une capture d'écran de la vidéo de démonstration pendant cette même étape.

[33] https://www.youtube.com/watch?v=GUDxPI92gHM
[34] https://mail.google.com/

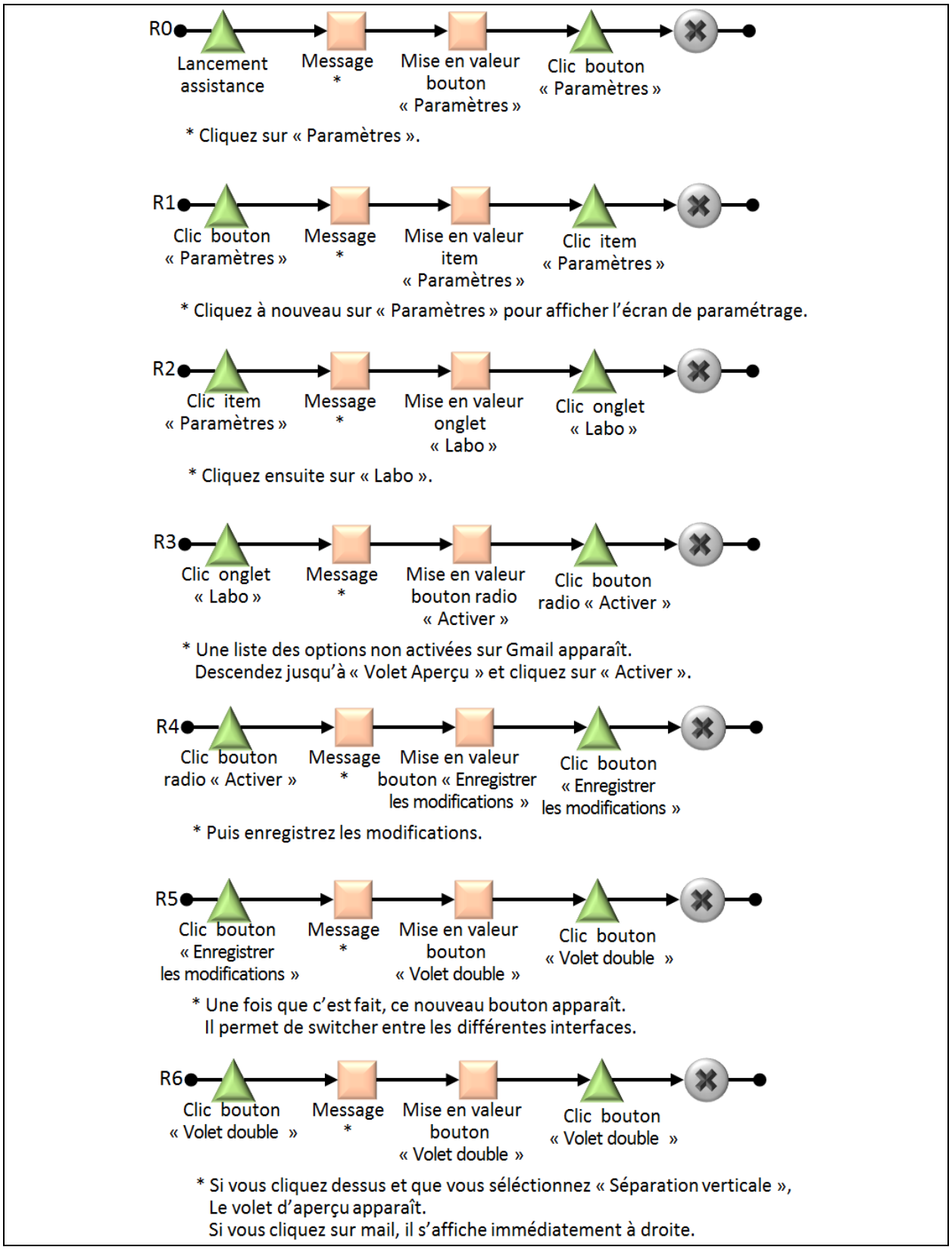
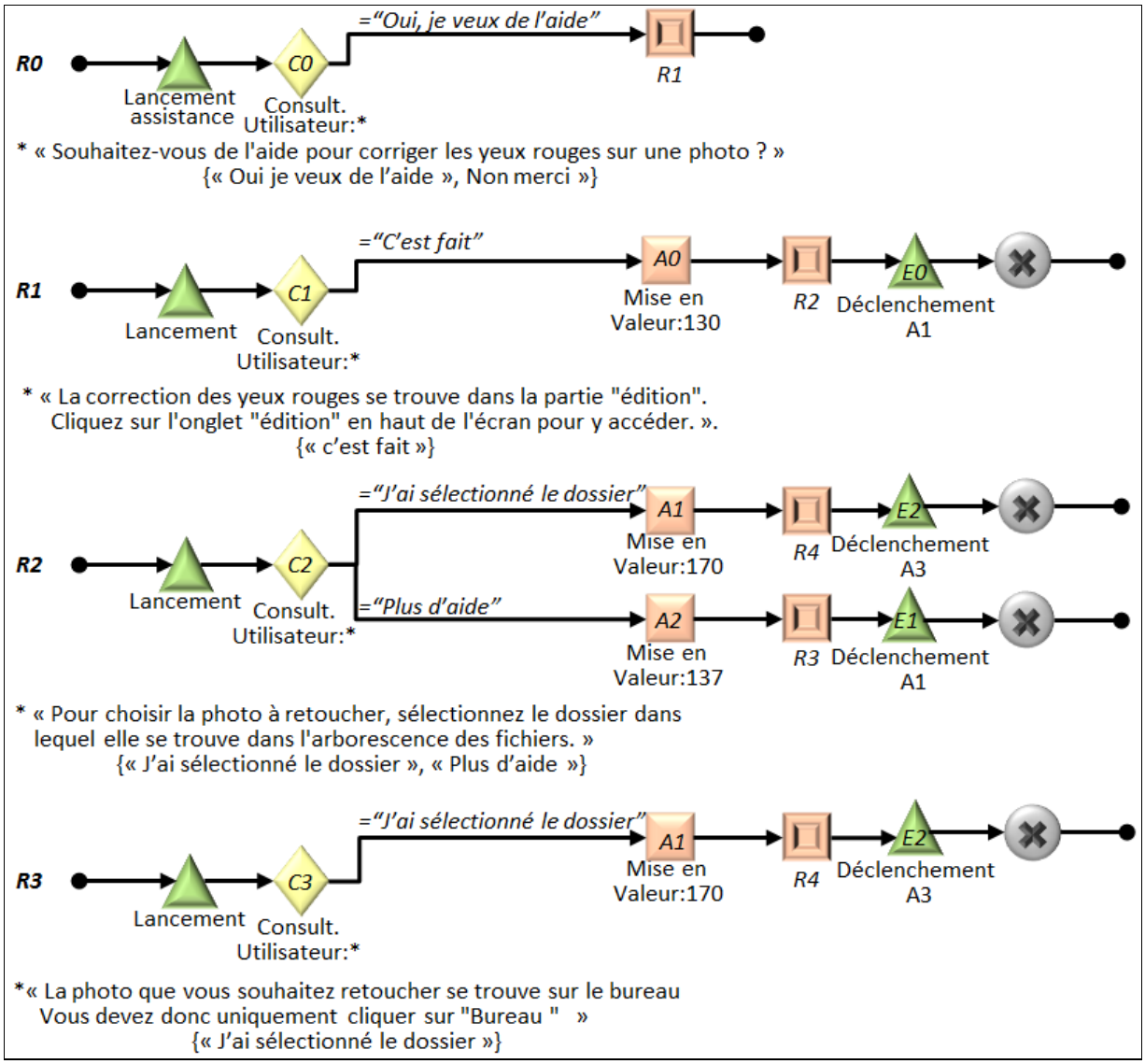


Fig. 19 – Représentation en aLDEAS d'une assistance pour l'ajout d'un volet d'aperçu de courriel Gmail

Annexe B. Assistance pour PhotoScape

1. Définition de l'assistance pour PhotoScape

La Fig. 20 présente la définition en aLDEAS du système d'assistance pour la correction des yeux rouges sur une photo avec PhotoScape. La Fig. 21 présente son équivalent en XML, créé avec l'éditeur de SEPIA et exécutable par le moteur générique de SEPIA. Ce système d'assistance a été expérimenté avec 151 utilisateurs finaux de PhotoScape (cf. Section 11.4.1).



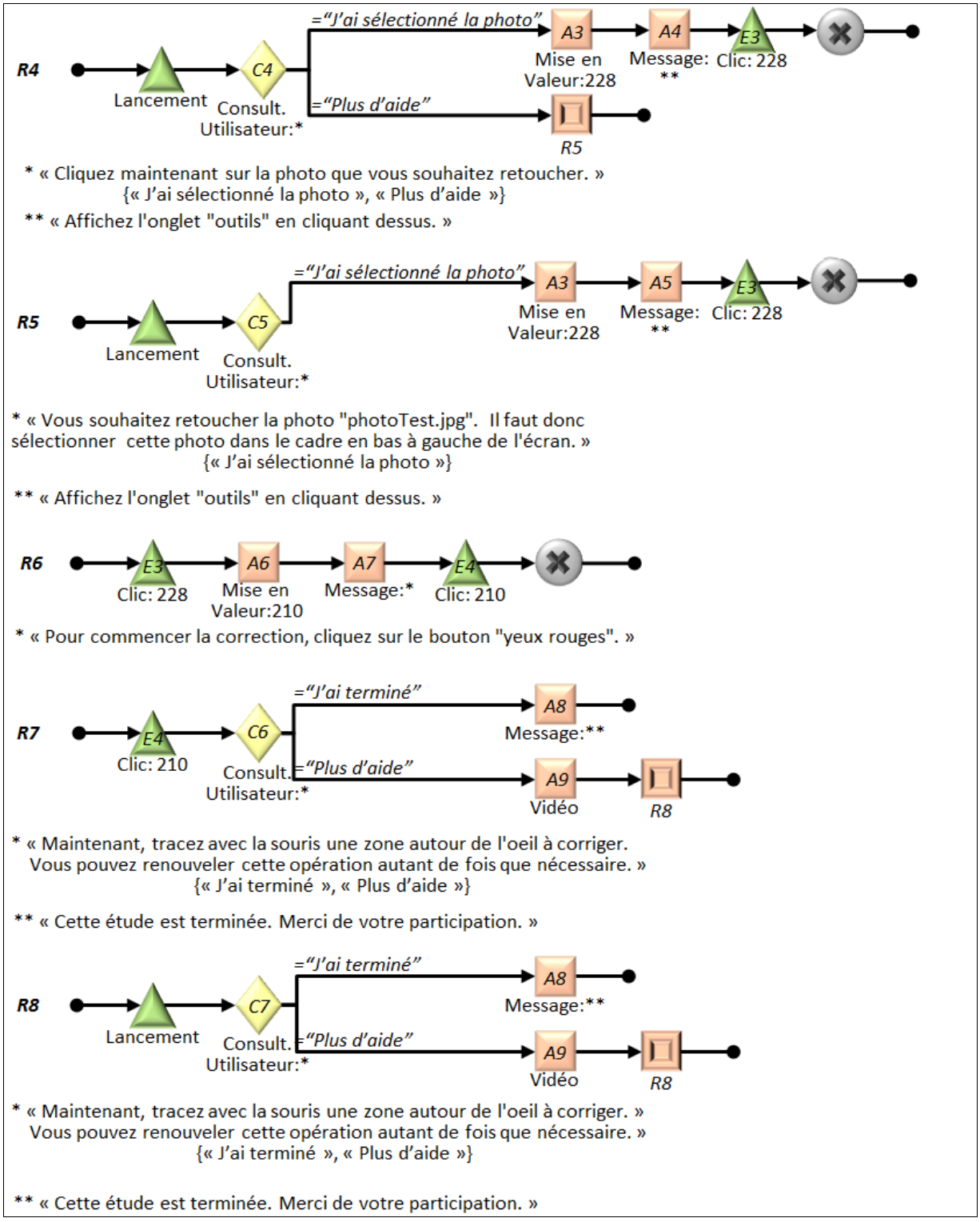


Fig. 20 – Définition en aLDEAS du système d'assistance pour la correction des yeux rouges avec PhotoScape

```
<metadonnees>
  <assistance nom="AssistancePhotoScape" logiciel="PhotoScape"
              type="Exécutable Windows" interface="photoscape.interface" />
  <creation date_creation="26/11/2013" date_modification="12/05/2014" />
  <commentaires />
</metadonnees>
<regles>
  <regle id="R0">
    <evenement_declencheur type="lancementAssistance" idComp="" />
    <alternatives condition="C0">
      <alternative valeur="oui je veux de l'aide">
        <action id="R1" />
      </alternative>
    </alternatives>
  </regle>
  <regle id="R1">
    <evenement_declencheur type="lancementRegle" idComp="R1" />
    <alternatives condition="C1">
      <alternative valeur="C'est fait">
        <action id="R2" />
      </alternative>
    </alternatives>
  </regle>
  <regle id="R2">
    <evenement_declencheur type="lancementRegle" idComp="R2" />
    <alternatives condition="">
      <alternative>
        <action id="A0" />
        <action id="R3" />
      </alternative>
    </alternatives>
    <evenement_de_fin idEve="E2" type="declenchement" idComp="A1" />
  </regle>
  <regle id="R3">
    <evenement_declencheur type="lancementRegle" idComp="R3" />
    <alternatives condition="C2">
      <alternative valeur="J'ai sélectionné le dossier">
        <action id="A1" />
        <action id="R5" />
      </alternative>
      <alternative valeur="Plus d'aide">
        <action id="A2" />
        <action id="R4" />
      </alternative>
    </alternatives>
    <evenement_de_fin idEve="E3" type="declenchement" idComp="A3" />
  </regle>
```

```
<regle id="R4">
  <evenement_declencheur type="lancementRegle" idComp="R4" />
  <alternatives condition="C3">
    <alternative valeur="J'ai sélectionné le dossier">
      <action id="A1" />
      <action id="R5" />
    </alternative>
  </alternatives>
</regle>
<regle id="R5">
  <evenement_declencheur type="lancementRegle" idComp="R5" />
  <alternatives condition="C4">
    <alternative valeur="J'ai sélectionné la photo">
      <action id="A3" />
      <action id="A4" />
    </alternative>
    <alternative valeur="Plus d'aide">
      <action id="R6" />
    </alternative>
  </alternatives>
  <evenement_de_fin idEve="E0" type="mouseClicked" idComp="228" />
</regle>
<regle id="R6">
  <evenement_declencheur type="lancementRegle" idComp="R6" />
  <alternatives condition="C5">
    <alternative valeur="J'ai sélectionné la photo">
      <action id="A3" />
      <action id="A4" />
    </alternative>
  </alternatives>
  <evenement_de_fin idEve="E0" type="mouseClicked" idComp="228" />
</regle>
<regle id="R7">
  <evenement_declencheur idEve="E0" type="mouseClicked" idComp="228" />
  <alternatives condition="">
    <alternative>
      <action id="A5" />
      <action id="A6" />
    </alternative>
  </alternatives>
  <evenement_de_fin idEve="E1" type="mouseClicked" idComp="210" />
</regle>
<regle id="R8">
  <evenement_declencheur idEve="E1" type="mouseClicked" idComp="210" />
  <alternatives condition="C6">
    <alternative valeur="J'ai terminé">
      <action id="A8" />
    </alternative>
```

```
      <alternative valeur="Plus d'aide">
        <action id="A7" />
        <action id="R9" />
      </alternative>
    </alternatives>
  </regle>
  <regle id="R9">
    <evenement_declencheur type="lancementRegle" idComp="R9" />
    <alternatives condition="C6">
      <alternative valeur="J'ai terminé">
        <action id="A8" />
      </alternative>
      <alternative valeur="Plus d'aide">
        <action id="A7" />
        <action id="R9" />
      </alternative>
    </alternatives>
  </regle>
</regles>
<actions>
  <action id="A0" type="mise en valeur">
    <composant id="130" type="entourer" couleur="[r=75,g=125,b=185]"
               arrondi="5" eloignement="5" epaisseur="5" />
  </action>
  <action id="A1" type="mise en valeur">
    <composant id="170" type="entourer" couleur="[r=75,g=125,b=185]"
               arrondi="5" eloignement="5" epaisseur="5" />
  </action>
  <action id="A2" type="mise en valeur" timer="5000">
    <composant id="137" type="symbole" direction="par la droite"
               symbole="fleche.png" />
  </action>
  <action id="A3" type="mise en valeur">
    <composant id="228" type="symbole" direction="par le haut"
               symbole="fleche.png" />
  </action>
  <action id="A4" type="message" sous-type="textuel" assistant="messager">
    <textuel>
      <texte>Affichez l'onglet "outils" en cliquant dessus.</texte>
    </textuel>
  </action>
  <action id="A5" type="mise en valeur">
    <composant id="210" type="symbole" direction="par le haut"
               symbole="fleche.png" />
  </action>
  <action id="A6" type="message" sous-type="textuel" assistant="messager">
    <textuel>
      <texte>Pour commencer la correction,
             cliquez sur le bouton "yeux rouges".</texte>
    </textuel>
  </action>
```

```
    <action id="A7" type="fichier" chemin="demoPhotoScape.avi" />
    <action id="A8" type="message" sous-type="textuel" assistant="messager">
      <textuel>
        <texte>Cette étude est terminée. Merci de votre participation.</texte>
      </textuel>
    </action>
  </actions>
  <conditions>
    <condition id="C0" type="consultation">
      <texte>Souhaitez-vous de l'aide pour corriger
          les yeux rouges sur une photo ?</texte>
    </condition>
    <condition id="C1" type="consultation">
      <texte>La correction des yeux rouges se trouve dans la partie "édition".
      Cliquez sur l'onglet "édition" en haut de l'écran pour y accéder.</texte>
    </condition>
    <condition id="C2" type="consultation">
      <texte>Pour choisir la photo à retoucher, sélectionnez le dossier
      dans lequel elle se trouve dans l'arborescence des fichiers</texte>
    </condition>
    <condition id="C3" type="consultation">
      <texte>La photo que vous souhaitez retoucher se trouve sur le bureau.
      Vous devez donc uniquement cliquer sur "Bureau".</texte>
    </condition>
    <condition id="C4" type="consultation">
      <texte>Cliquez maintenant sur la photo que vous souhaitez retoucher</texte>
    </condition>
    <condition id="C5" type="consultation">
      <texte>Vous souhaitez retoucher la photo "photoTest.jpg".
      Il faut donc sélectionner cette photo
      dans le cadre en bas à gauche de l'écran.</texte>
    </condition>
    <condition id="C6" type="consultation">
      <texte>Maintenant, tracez avec la souris une zone
          autour de l'oeil à corriger. Vous pouvez renouveler
          cette opération autant de fois que nécessaire.</texte>
    </condition>
  </conditions>
  <evenements>
    <evenement typeEve="action utilisateur" id="E0"
          type="mouseClicked" idComp="228" />
    <evenement typeEve="action utilisateur" id="E1"
          type="mouseClicked" idComp="210" />
    <evenement typeEve="evenement assistance" id="E2"
          type="declenchement" objet="A1" />
    <evenement typeEve="evenement assistance" id="E3"
          type="declenchement" objet="A3" />
  </evenements>
  <sequences />
</description>
```

Fig. 21 – Définition en XML du système d'assistance pour PhotoScape

2. Utilisation de SEPIA pour spécifier l'assistance pour PhotoScape

La Fig. 22 montre un zoom sur une capture d'écran de définition des règles d'assistance de l'éditeur de SEPIA. Elle correspond à la définition de la règle R0 du système d'assistance pour la correction des yeux rouges sur une photo avec PhotoScape ; sa définition en aLDEAS est donnée en Fig. 20, et son fichier xml généré par l'éditeur de SEPIA est donné en Fig. 21. La règle d'assistance R0 contient la condition de déclenchement C0 : la définition avec l'éditeur de SEPIA de la condition C0 est donnée en Fig. 23. Lors de l'exécution de l'assistance par le moteur générique de SEPIA, le déclenchement de la règle R0 et donc la vérification de la condition C0 est illustrée par la Fig. 24.

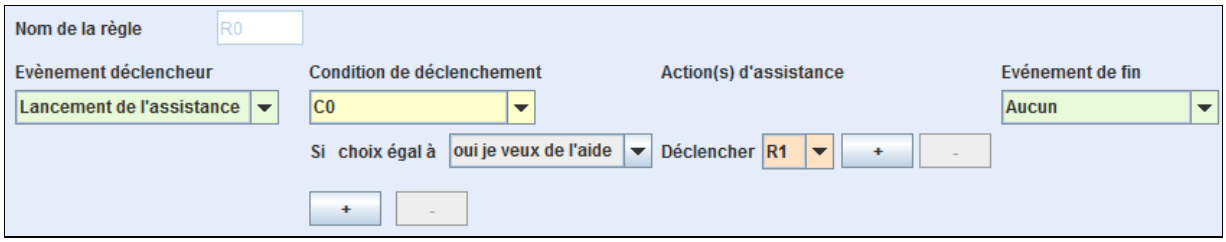


Fig. 22 – Capture d'écran de l'éditeur de SEPIA : définition de la règle R0

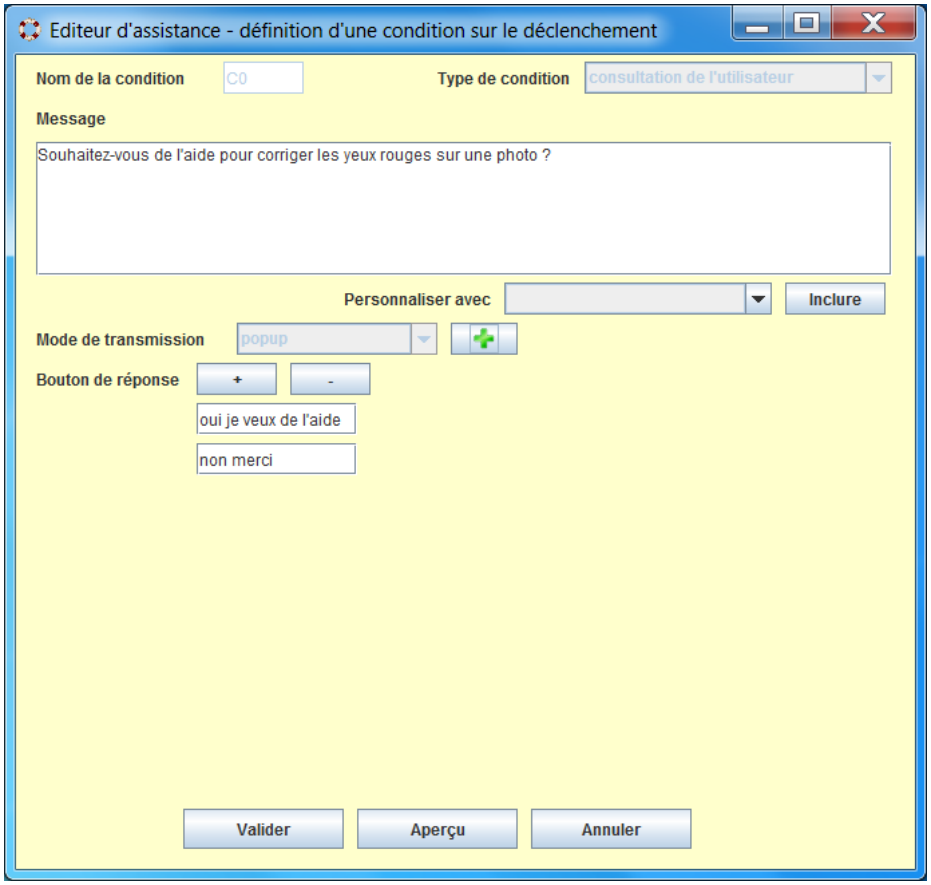


Fig. 23 – Capture d'écran de l'éditeur de SEPIA : définition de la condition C0

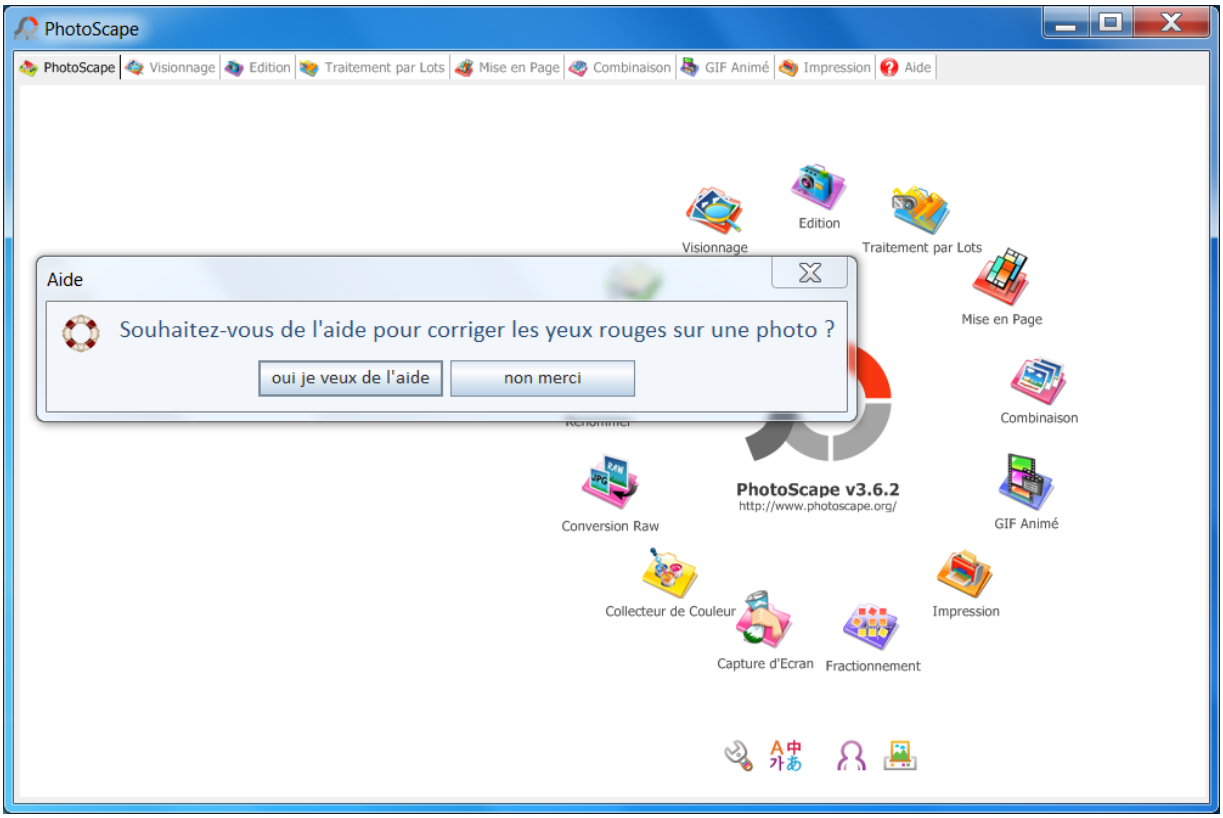


Fig. 24 – Déclenchement dans PhotoScape de la règle R0 : vérification de la condition C0

3. Historique de l'assistance pour PhotoScape

La Fig. 25 donne l'exemple de l'historique de l'assistance pour la correction des yeux rouges sur une photo avec PhotoScape, exécutée pour Lucas Dupuis.

```
<historique date_creation="10/07/2014" date_modif="10/07/2014"
            nom_assistance="Correction_YeuxRouges" logiciel="PhotoScape">
  <utilisateur nom="LucasDupuis.xml">
    <seance date="10/07/2014">
      <element nom="R0">
        <execution declenchement="1404978962549" fin="1404978971001" />
      </element>
      <element nom="R1">
        <execution declenchement="1404978971001" fin="1404978971789" />
      </element>
      <element nom="R2">
        <execution declenchement="1404978971789" fin="1404978979590" />
      </element>
      <element nom="R3">
        <execution declenchement="1404978979590" fin="1404978985069" />
      </element>
      <element nom="R5">
        <execution declenchement="1404978985069" fin="1404978990464" />
      </element>
      <element nom="R7">
        <execution declenchement="1404978990464" fin="1404979000676" />
      </element>
      <element nom="R8">
        <execution declenchement="1404979000676" fin="1404979010364" />
      </element>
      <element nom="R9">
        <execution declenchement="1404979010364" />
      </element>
      <element nom="A0">
        <execution declenchement="1404978979590" fin="10/07/2014" />
      </element>
      <element nom="A2">
        <execution declenchement="1404978985070" fin="10/07/2014" />
      </element>
      <element nom="A3">
        <execution declenchement="1404978985649" fin="10/07/2014" />
      </element>
```

```
        <element nom="A4">
          <execution declenchement="1404978990465" fin="10/07/2014" />
        </element>
        <element nom="A5">
          <execution declenchement="1404978990980" fin="10/07/2014" />
        </element>
        <element nom="A6">
          <execution declenchement="1404979000676" fin="10/07/2014" />
        </element>
        <element nom="A7">
          <execution declenchement="1404979010365" fin="10/07/2014" />
        </element>
        <element nom="A8">
          <execution declenchement="1404978971789" />
        </element>
        <element nom="C0">
          <execution declenchement="1404978962549" fin="1404978971001">
          oui je veux de l'aide</execution>
        </element>
        <element nom="C1">
          <execution declenchement="1404978971001" fin="1404978971789">
          C'est fait</execution>
        </element>
        <element nom="C2">
          <execution declenchement="1404978971789" fin="1404978979590">
          J'ai sélectionné le dossier</execution>
        </element>
        <element nom="C4">
          <execution declenchement="1404978985069" fin="1404978986592">
          J'ai sélectionné la photo</execution>
        </element>
        <element nom="C6">
          <execution declenchement="1404979002811" fin="10/07/2014">
          Plus d'aide</execution>
          <execution declenchement="1404979012918" fin="10/07/2014">
          J'ai terminé</execution>
        </element>
    </seance>
  </utilisateur>
</historique>
```

Fig. 25 – Historique de l'assistance de PhotoScape : séance d'utilisation de Lucas Dupuis

Annexe C. Assistances définies

La Fig. 26 présente une liste non exhaustive des applications pour lesquelles une assistance a été définie dans le cadre du projet AGATE. Pour ces applications, la définition de l'assistance, avec aLDEAS et/ou avec SEPIA peut avoir été effectuée par un membre du projet AGATE (représenté par un ✔ dans la Fig. 26), ou par des personnes extérieures au projet (représenté par un ✔ dans la Fig. 26).

Application	Type	Assistance		
		Définie en aLDEAS	Mise en place avec SEPIA	Expérimentée avec des utilisateurs finaux
PhotoScape	exe	✔	✔	✔
NetBeans	java	✔	✔	✔
Visual Studio	exe	✔	✔	
Code Blocks	exe		✔	
Skype	exe		✔	
MOOC Fovéa	web		✔	
Recherche avancée d'images dans Google	web		✔	
Flickr	web		✔	
Forge de l'Université Lyon 1	web	✔	✔	
Xmind	exe		✔	
Caf	web	✔	✔	
Site web du LIRIS	web	✔	✔	
Page web de P-A Champin	web	✔	✔	
Microsoft Word 2010	exe	✔	✔	
Microsoft PowerPoint 2010	exe	✔	✔	
Microsoft Excel 2010	exe	✔	✔	
Accessoires Windows 7 (calculatrice, jeux, Paint…)	exe	✔	✔	
CamStudio	exe		✔	

EndNote	exe		✓	
Aplusix	exe		✓	
Géogébra	java	✓		
Audacity	exe	✓	✓	
7 gratuiciels divers	divers	✓	✓	
10 logiciels développés par des étudiants (avant 2013)	divers		✓	
20 logiciels développés par des étudiants (en 2013)	divers		✓	
14 logiciels développés par des étudiants (en 2014)	divers	✓	✓	
Gmail	web	✓		
Cad-Kas PDF editor	?	✓		
Yasgui	web	✓		
Outil de planification PAX Ikea	?	✓		
Jeu « Dora l'exploratrice : le spectacle de fin d'année »	?	✓		
Simulateur de prêts du Crédit Mutuel	web	✓		
Site des impôts	web	✓		

Fig. 26 – Applications pour lesquelles une assistance a été définie

Annexe D. Utilisation d'aLDEAS par des concepteurs d'assistance

Nous présentons ici cinq exemples de systèmes d'assistance définis en aLDEAS par des concepteurs d'assistance extérieurs au projet AGATE. Ces systèmes d'assistance ont été définis sur papier dans le cadre des expérimentations que nous avons menées avec des concepteurs d'assistance informaticiens et non informaticiens, et concernent différentes applications-cibles.

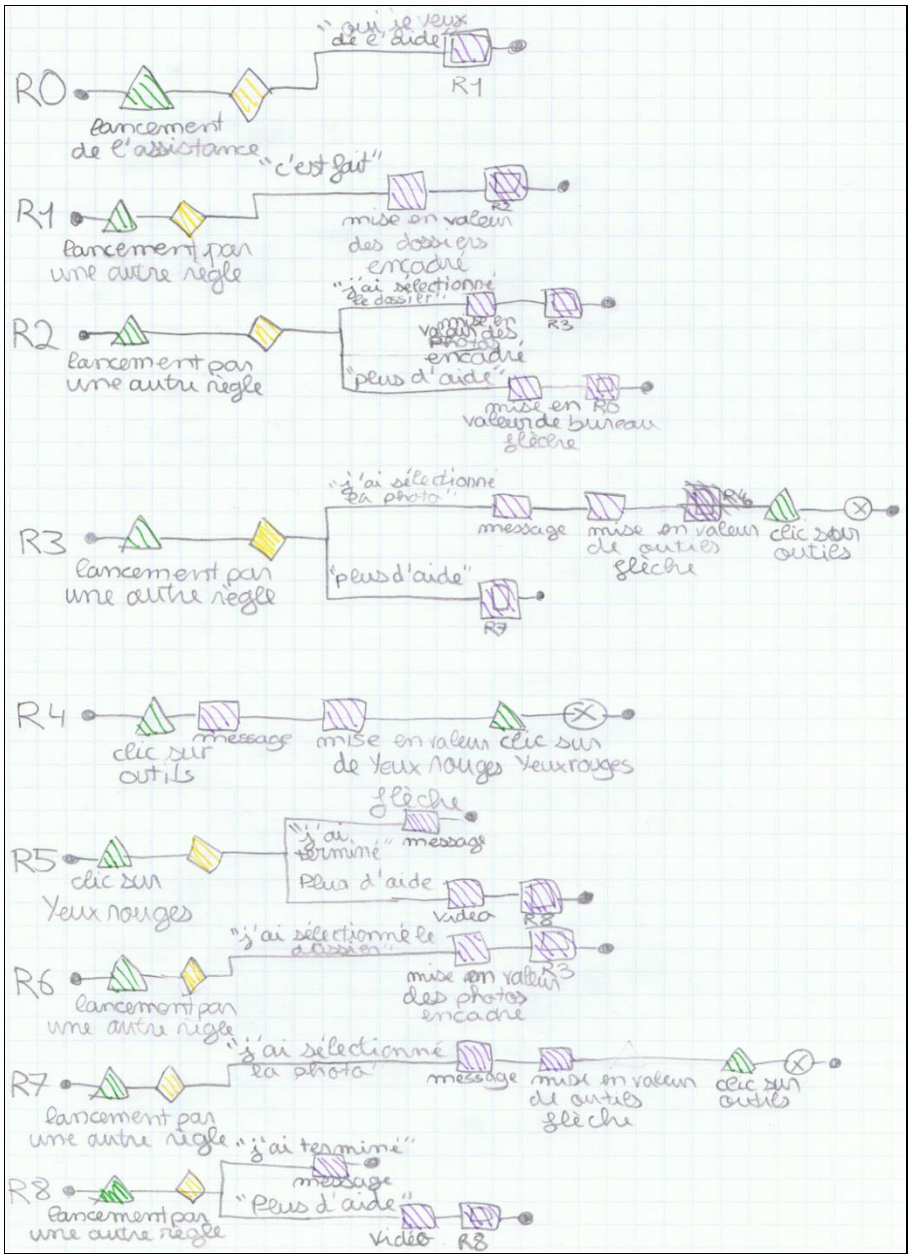


Fig. 27 – Assistance pour PhotoScape définie en aLDEAS par Maëlys, une enfant de 11 ans

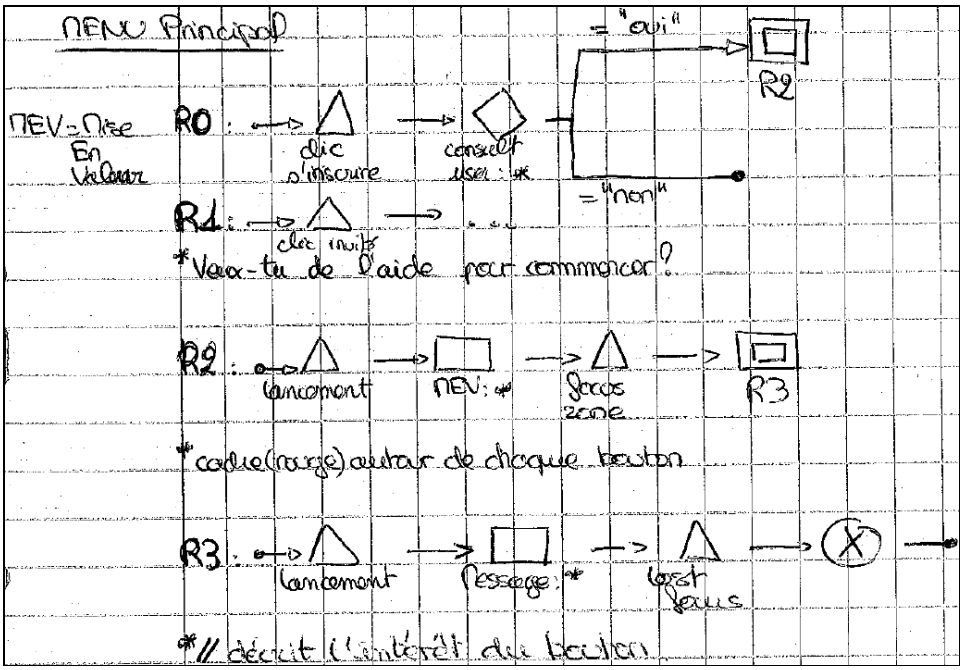


Fig. 28 – Extrait d'une assistance pour un logiciel éducatif définie en aLDEAS par Laurenn et Martin, deux étudiants en master informatique

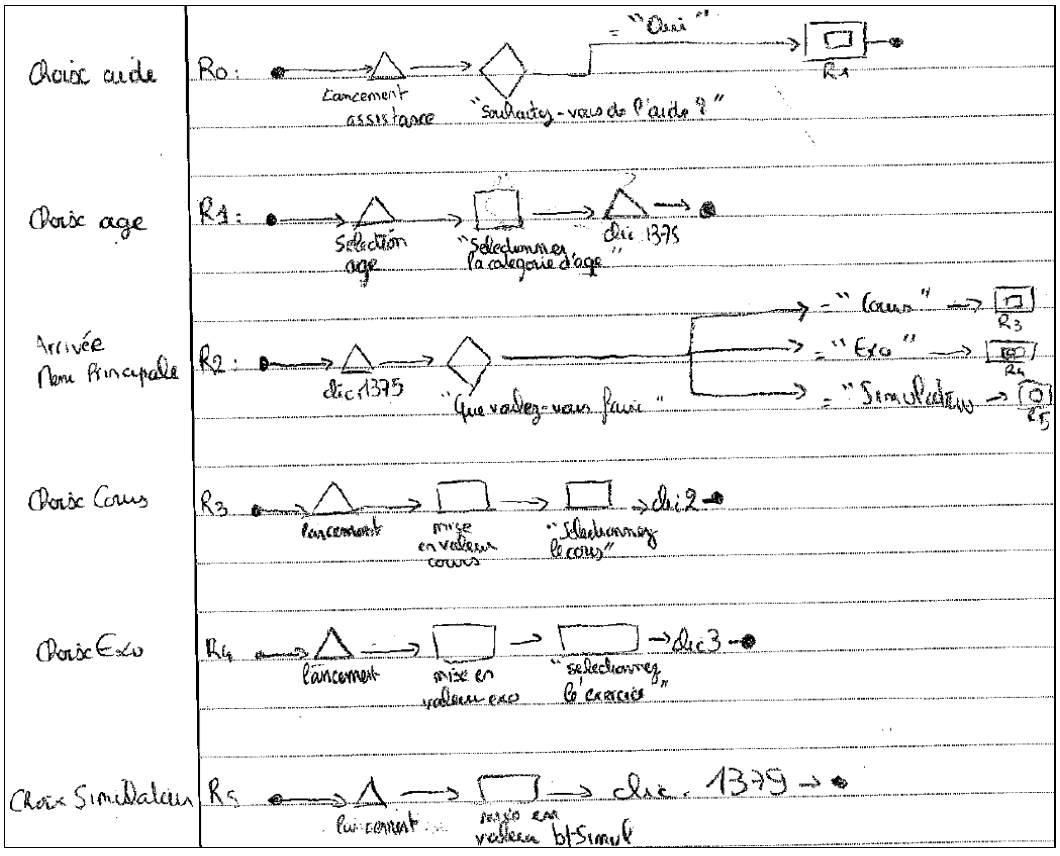


Fig. 29 – Extrait d'une assistance pour un logiciel éducatif définie en aLDEAS par Sébastien et Romain, deux étudiants en master informatique

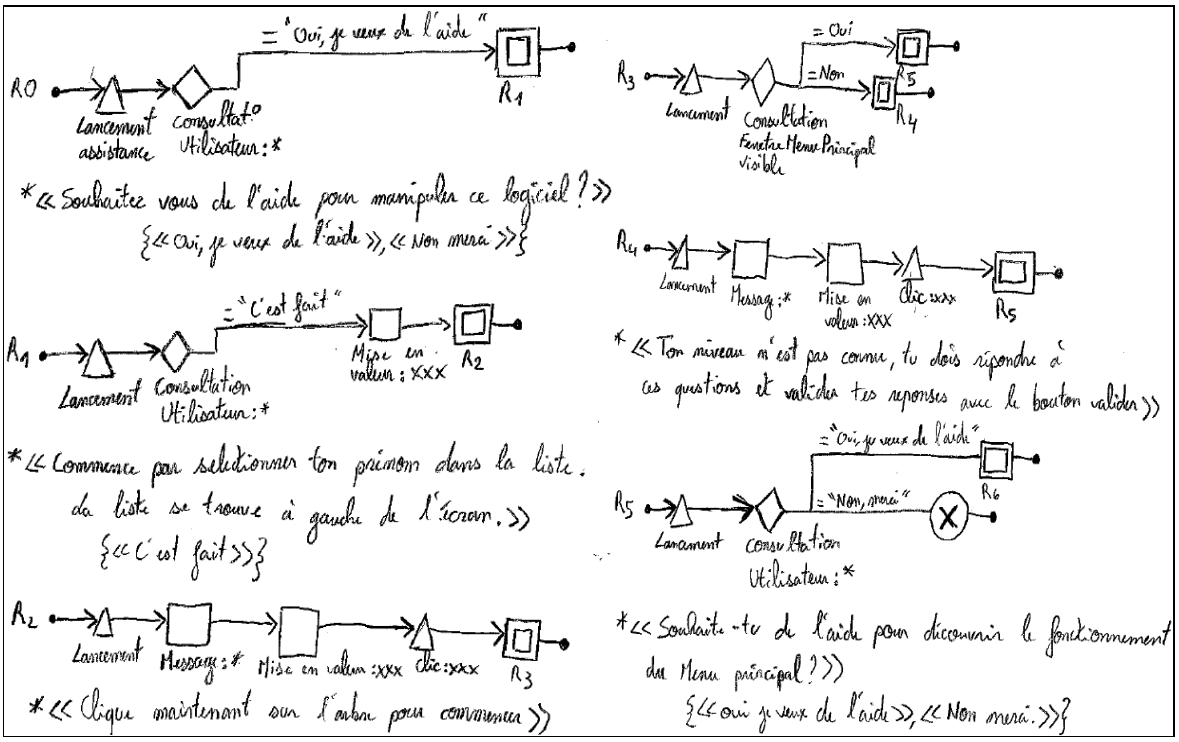


Fig. 30 – Extrait d'une assistance pour un logiciel éducatif définie en aLDEAS par Maxime, un étudiant en master informatique

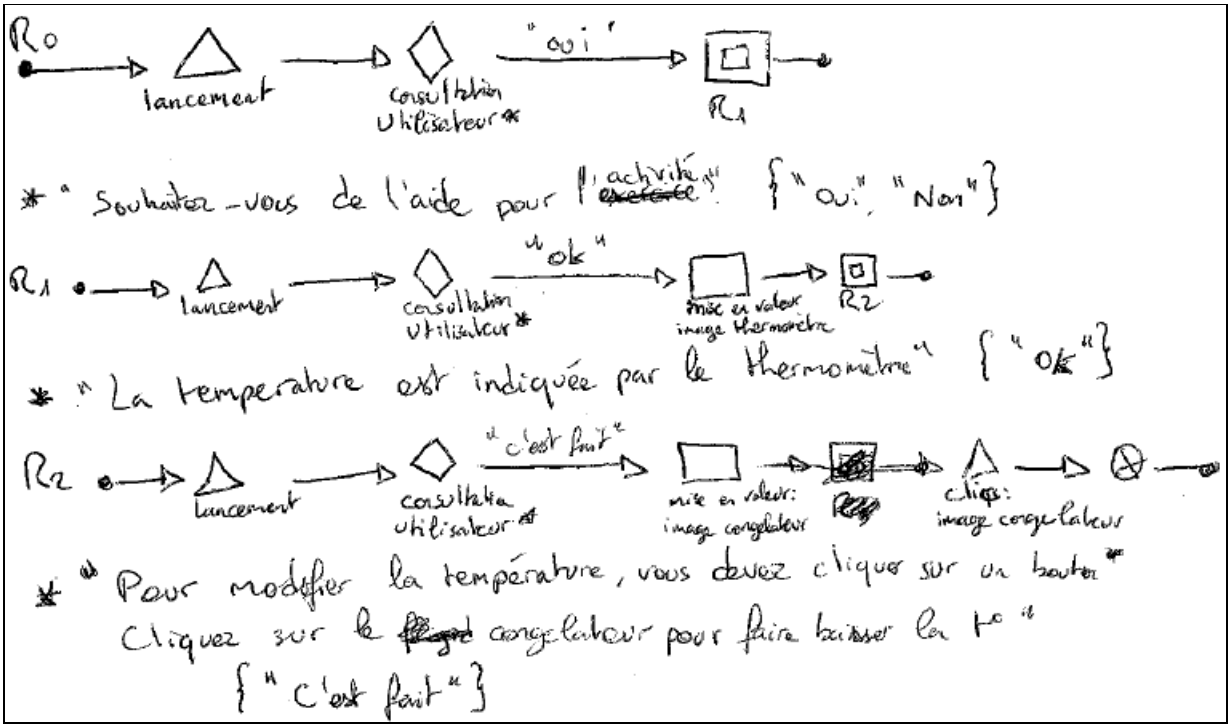


Fig. 31 – Extrait d'une assistance pour un logiciel éducatif définie en aLDEAS par Vin-Nam, un étudiant en master informatique

Annexe E. Questionnaires des expérimentations

1. Expérimentation avec PhotoScape

Utilisation de PhotoScape pour la correction des yeux rouges sur une photo numérique

A

Informations personnelles

Date : _____

Sexe : ❍ féminin ❍ masculin
5 + 27 44 + 24

Prénom : _____ Nom : _____

Âge : ❍ - de 10 ans ❍ 10 - 15 ans ❍ 15 - 25 ans ❍ 25 – 55 ans ❍ + de 55 ans
0 + 0 0 + 0 43 + 31 6 + 17 0 + 3

Questions préliminaires

En informatique, êtes-vous ?
❍ novice ❍ utilisateur occasionnel ❍ utilisateur confirmé ❍ informaticien
0 + 4 1 + 5 2 + 5 46 + 37

Avez-vous déjà utilisé le logiciel PhotoScape ?
❍ jamais ❍ quelques fois ❍ souvent
48 + 48 0 + 3 1 + 0

Avez-vous déjà utilisé PhotoScape pour corriger des yeux rouges ?
❍ jamais ❍ quelques fois ❍ souvent
48 + 50 0 + 1 1 + 0

Avez-vous déjà utilisé un outil autre que PhotoScape pour corriger des yeux rouges ?
❍ Non ❍ Oui Si oui, le(s)quel(s)_____
32 + 38 17 + 13

Expérimentation

Vous allez maintenant utiliser PhotoScape afin d'essayer de corriger les yeux rouges sur une photo. Pour cela, suivez les instructions suivantes :

- Notez l'heure à laquelle vous débutez ___h___min

- Corrigez les yeux rouges sur la photo **PhotoTest.jpg** qui se trouve sur le **bureau**

- Si vous avez réussi, notez l'heure ___h___min,
puis remplissez le questionnaire final au dos de cette feuille

- Si vous abandonnez, notez l'heure ___h___min,
puis appelez l'observateur

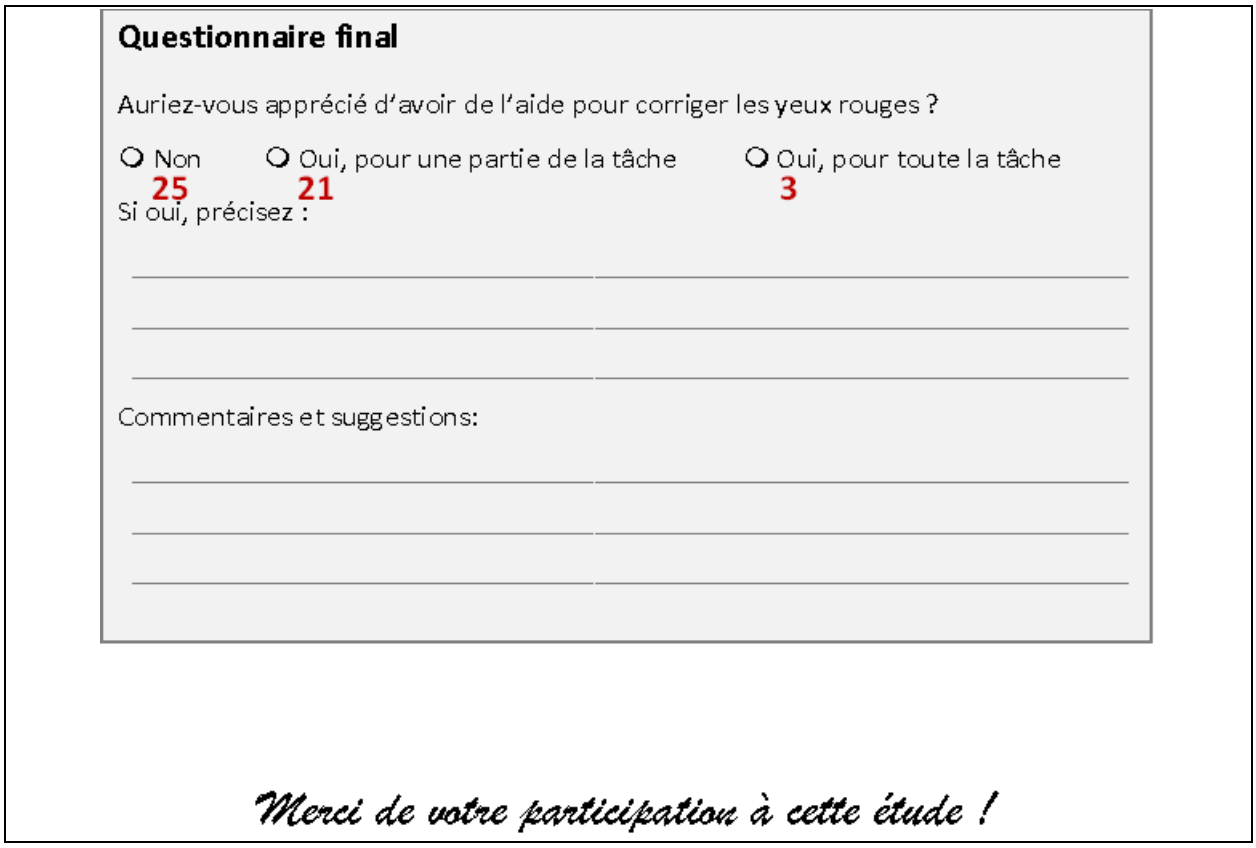


Fig. 32 – Questionnaire distribué aux participants du groupe A : les utilisateurs ont essayé de réaliser la correction des yeux rouges sur une photo sans assistance dans un premier temps

Sur la Fig. 32, la synthèse des réponses des participants du groupe A est notée en rouge. Nous distinguons les réponses des utilisateurs du groupe A1, ayant réussi à réaliser la tâche, de celles des utilisateurs du groupe A2, ayant abandonné. Ainsi, l'annotation « 5 + 27 » signifie qu'une réponse a été cochée par 5 utilisateurs du groupe A1 et par 27 utilisateurs du groupe A2. Notons que seuls les utilisateurs du groupe A1 ont rempli le questionnaire final, les utilisateurs du groupe A2 ont rempli celui de la Fig. 33, après avoir réalisé la tâche demandée en suivant l'assistance fournie par SEPIA.

Utilisation de PhotoScape pour la correction des yeux rouges sur une photo numérique

A2

Expérimentation

Vous allez maintenant disposer d'une aide pour utiliser PhotoScape afin d'essayer de corriger les yeux rouges sur une photo. Pour cela, suivez les instructions suivantes :

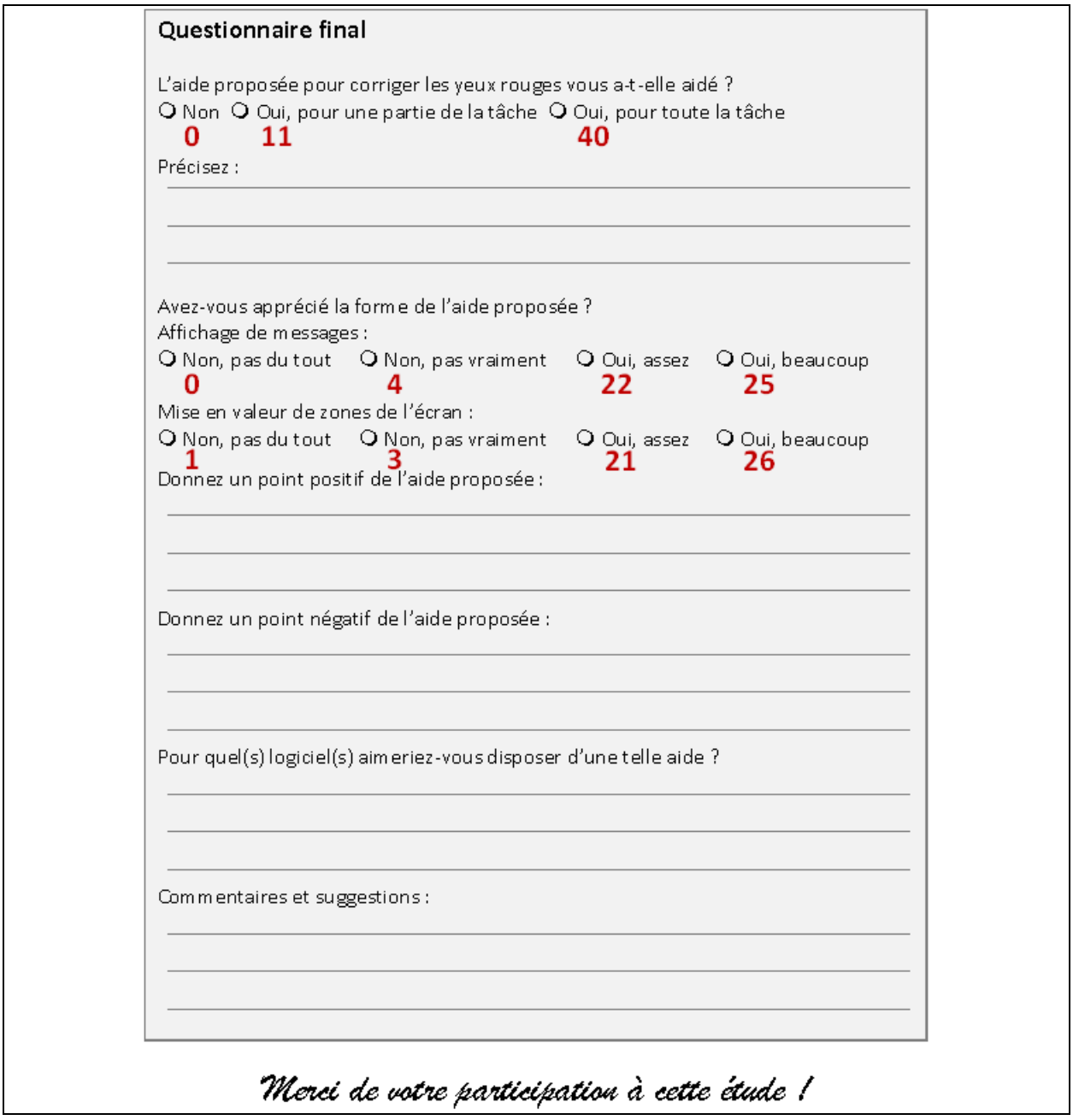
- Notez l'heure à laquelle vous débutez ___h___min

- Suivez les conseils de l'assistant pour corriger les yeux rouges sur la photo **PhotoTest.jpg** qui se trouve sur le **bureau**

- Si vous avez réussi, notez l'heure ___h___min

- Si vous abandonnez, notez l'heure ___h___min

- Remplissez le questionnaire final au dos de cette feuille



Questionnaire final

L'aide proposée pour corriger les yeux rouges vous a-t-elle aidé ?

❍ Non ❍ Oui, pour une partie de la tâche ❍ Oui, pour toute la tâche

0 11 40

Précisez :

Avez-vous apprécié la forme de l'aide proposée ?

Affichage de messages :

❍ Non, pas du tout ❍ Non, pas vraiment ❍ Oui, assez ❍ Oui, beaucoup

0 4 22 25

Mise en valeur de zones de l'écran :

❍ Non, pas du tout ❍ Non, pas vraiment ❍ Oui, assez ❍ Oui, beaucoup

1 3 21 26

Donnez un point positif de l'aide proposée :

Donnez un point négatif de l'aide proposée :

Pour quel(s) logiciel(s) aimeriez-vous disposer d'une telle aide ?

Commentaires et suggestions :

Merci de votre participation à cette étude !

Fig. 33 – Questionnaire distribué aux participants du groupe A2 : les utilisateurs du groupe A n'ayant pas réussi à réaliser la correction des yeux rouges sur une photo sans assistance (la synthèse des réponses des participants est indiquée en rouge)

Utilisation de PhotoScape pour la correction des yeux rouges sur une photo numérique

B

Informations personnelles

Date : _____ Sexe : ❍ féminin ❍ masculin

Prénom : _____ Nom : _____

Âge : ❍ - de 10 ans ❍ 10 - 15 ans ❍ 15 - 25 ans ❍ 25 – 55 ans ❍ + de 55 ans
1 2 52 38 7

Questions préliminaires

En informatique, êtes-vous ?
❍ novice ❍ utilisateur occasionnel ❍ utilisateur confirmé ❍ informaticien
0 2 15 83

Avez-vous déjà utilisé le logiciel PhotoScape ?
❍ jamais ❍ quelques fois ❍ souvent
98 2 0

Avez-vous déjà utilisé PhotoScape pour corriger des yeux rouges ?
❍ jamais ❍ quelques fois ❍ souvent
100 0 0

Avez-vous déjà utilisé un outil autre que PhotoScape pour corriger des yeux rouges ?
❍ Non ❍ Oui Si oui, le(s)quel(s)_____
60 40

Expérimentation

Vous allez maintenant utiliser PhotoScape afin d'essayer de corriger les yeux rouges sur une photo. Pour cela, suivez les instructions suivantes :

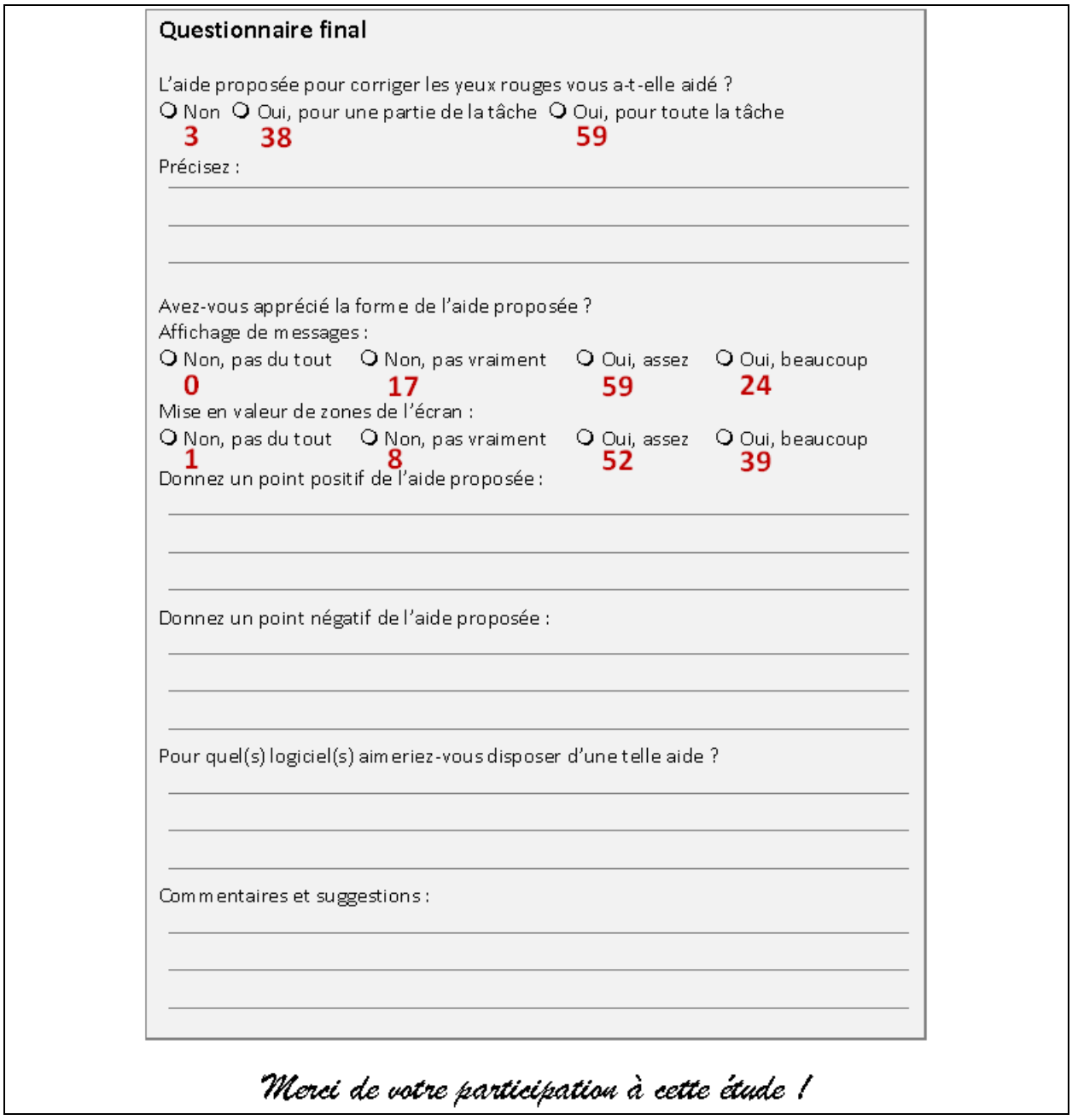
- Notez l'heure à laquelle vous débutez ___h___min

- Suivez les conseils de l'assistant pour corriger les yeux rouges sur la photo **PhotoTest.jpg** qui se trouve sur le **bureau**

- Si vous avez réussi, notez l'heure ___h___min

- Si vous abandonnez, notez l'heure ___h___min

- Remplissez le questionnaire final au dos de cette feuille



Questionnaire final

L'aide proposée pour corriger les yeux rouges vous a-t-elle aidé ?

❍ Non ❍ Oui, pour une partie de la tâche ❍ Oui, pour toute la tâche

3 38 59

Précisez :

Avez-vous apprécié la forme de l'aide proposée ?

Affichage de messages :

❍ Non, pas du tout ❍ Non, pas vraiment ❍ Oui, assez ❍ Oui, beaucoup

0 17 59 24

Mise en valeur de zones de l'écran :

❍ Non, pas du tout ❍ Non, pas vraiment ❍ Oui, assez ❍ Oui, beaucoup

1 8 52 39

Donnez un point positif de l'aide proposée :

Donnez un point négatif de l'aide proposée :

Pour quel(s) logiciel(s) aimeriez-vous disposer d'une telle aide ?

Commentaires et suggestions :

Merci de votre participation à cette étude !

Fig. 34 – Questionnaire distribué aux participants du groupe B : les utilisateurs qui ont essayé de réaliser de la correction des yeux rouges sur une photo directement avec assistance (la synthèse des réponses des participants est indiquée en rouge)

2. Expérimentations avec NetBeans

TP du ❍ jeudi ❍ vendredi N° d'ordinateur : ……

Utilisation de NetBeans dans le cadre du cours de LIF14 – IHM

Remarque : ce questionnaire est **anonyme**, il ne sera en aucune façon utilisé pour vous noter.
Il sera uniquement utilisé dans le cadre de nos recherches.

Questions préliminaires

Avez-vous déjà utilisé NetBeans :
❍ jamais 32 ❍ rarement 13 ❍ souvent 7

Si oui, avez-vous déjà utilisé NetBeans pour créer une interface graphique :
❍ jamais 8 ❍ rarement 9 ❍ souvent 3

Avez-vous suivi un cours de programmation Java :
❍ aucun 8 ❍ LIF13 23 ❍ autre : ….21……

Avez-vous visionné les vidéos de démonstration de NetBeans disponibles sur le site de LIF14 ?
❍ non 42 ❍ oui, « démo-event » 6 ❍ oui, « démo-GUI » 3
Pourquoi : ……

Apprécieriez-vous de suivre un tutoriel concernant l'utilisation de NetBeans dans le cadre de LIF14 ?
❍ non 0 ❍ oui 52
Pourquoi : ……

Quizz Java préliminaire

- Quelle instruction permet de rendre visible une fenêtre Java de type JFrame nommée maFenetre ? ……
- Quelle propriété d'un composant de type JButton permet de modifier la couleur de sa police ? ……
- Quelle propriété d'un composant de type JTextField permet de modifier son texte ? ……
- Quelle instruction permet de quitter une application Java

-normalement ? ……
-en signalant un problème? ……

- Est-il possible d'associer une icône et un texte à un composant de type JButton ?

-Si oui comment ? ……
-Si non, pourquoi ? ……

- Quelle propriété d'un composant de type MenuItem doit-on modifier pour lui associer un raccourci clavier ? ……
- Où peut-on sélectionner la « Main Class » d'un projet NetBeans? ……
- Quelle instruction permet de d'afficher « toto » sur la sortie Standard d'une application Java ? ……
- Dans NetBeans, quel bouton permet de générer un jar exécutable ? ……
- Dans quel dossier d'un projet NetBeans trouve-t-on le jar exécutable lorsqu'il a été généré ? ……

Pour vous aider dans ce 1er TP de création d'une interface graphique avec NetBeans, nous vous proposons d'utiliser un tutoriel que nous avons conçu dans le cadre de nos recherches. Si vous souhaitez suivre ce tutoriel NetBeans, contactez l'enseignant et retournez cette feuille. Sinon, commencez directement le TP « création d'une interface simple ».

Fig. 35 – Questionnaire et pré-test distribués aux participants au début du TP
(la synthèse des réponses des participants est indiquée en rouge)

Utilisation d'un tutoriel NetBeans

Consignes : afin de préparer le TP de création d'une interface simple, nous vous proposons de suivre un tutoriel de NetBeans qui vous permettra de (re-)découvrir cet IDE et certaines fonctionnalités qui vous seront utiles dans le cadre du cours de LIF14.

Ce tutoriel est intégré à NetBeans. Il est constitué de plusieurs petits tutoriels décrits ci-après. Vous êtes libres de suivre les tutoriels qui vous intéressent autant de fois que vous le souhaitez.

0_CreationProjet	Création d'une application Java, d'un package et d'une fenêtre.
1_ProprietesProjet	Présentation de propriétés d'un projet NetBeans. Exploitation des arguments d'un projet.
2_Bouton	Ajout d'un bouton à une fenêtre, manipulation de ses propriétés (texte, icone, événement…)
3_Menu	Création d'un menu permettant notamment de quitter l'application
4_Evenements	Gestion des événements : quitter l'application, choix d'une couleur dans une palette avec JColorChooser

Lorsque vous avez terminé les tutoriels, contactez l'enseignant avant de commencer le TP « création d'une interface simple ».

Fig. 36 – Consignes distribuées aux participants au début du TP

TP du ❍ jeudi ❍ vendredi N° d'ordinateur :……

Utilisation d'un tutoriel NetBeans

Remarque : ce questionnaire est **anonyme**, il ne sera en aucune façon utilisé pour vous noter. Il sera uniquement utilisé dans le cadre de nos recherches.

Quizz Java final à remplir à la fin du tutoriel

- Quelle propriété d'un composant de type JButton permet de modifier la couleur de sa police ? ……
- Quelle propriété d'un composant de type JTextField permet de modifier son texte ? ……
- Quelle instruction permet de quitter une application Java

-normalement ? ……

-en signalant un problème? ……

- Est-il possible d'associer une icône ET un texte à un composant de type JButton ?

-Si oui comment ? ……

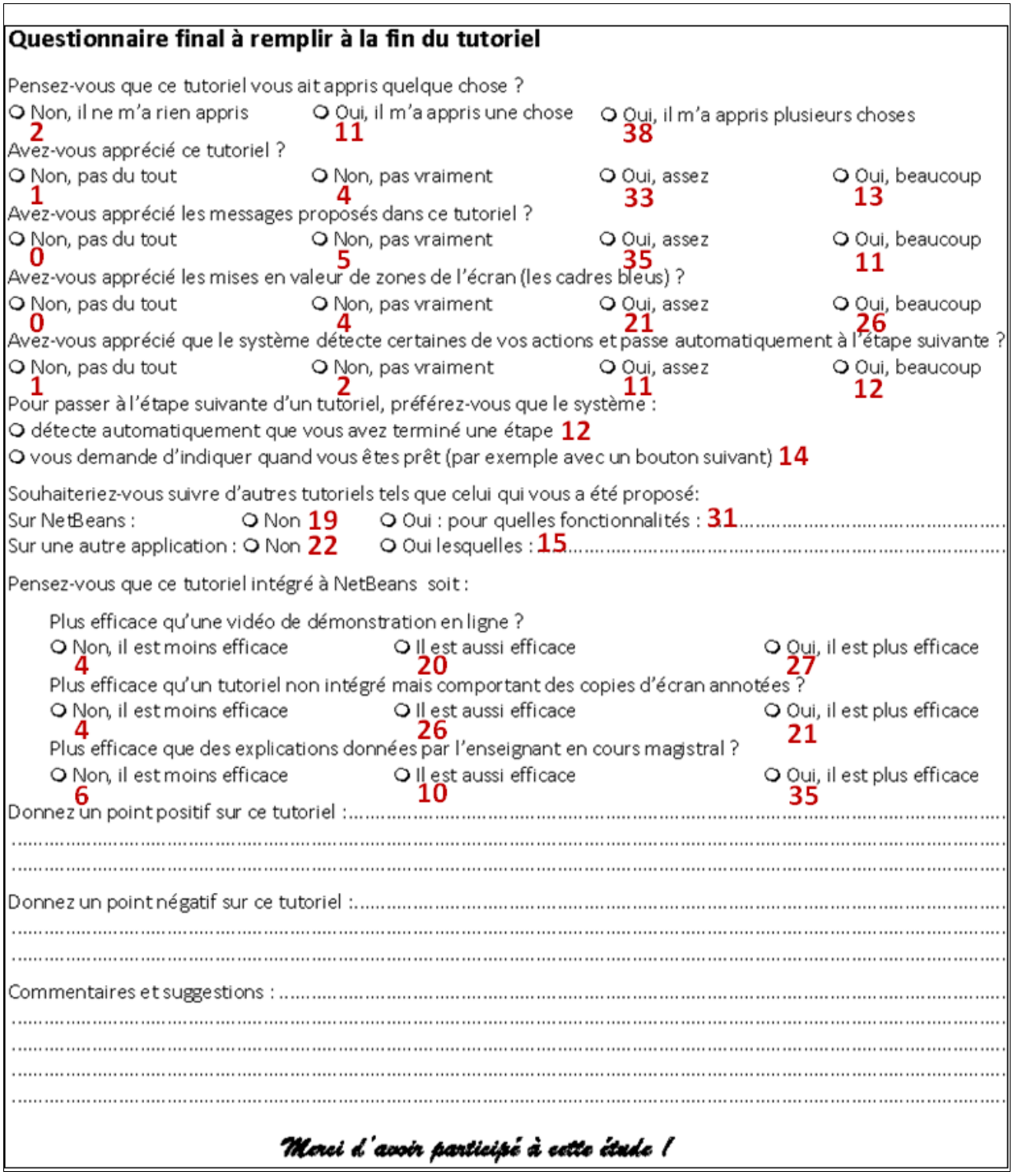
-Si non, pourquoi ? ……

- Quelle propriété d'un composant de type JMenuItem doit-on modifier pour lui associer un raccourci clavier ? ..
- Où peut-on sélectionner la « Main Class » d'un projet NetBeans? ……
- Quelle instruction permet d'afficher « toto » sur la sortie Standard d'une application Java ? ……
- Dans NetBeans, quel bouton permet de générer un jar exécutable ? ……
- Dans quel dossier d'un projet NetBeans trouve-t-on le jar exécutable lorsqu'il a été généré ? ……
- Quelle propriété d'un composant de type JButton permet de lui associer une infobulle ?……
- Comment-on peut insérer un sous menu dans un menu de type JMenu ?……

……

Retournez cette feuille svp

Fig. 37 – Post-test distribué aux participants suite à l'utilisation du tutoriel



Questionnaire final à remplir à la fin du tutoriel

Pensez-vous que ce tutoriel vous ait appris quelque chose ?

❍ Non, il ne m'a rien appris 2 ❍ Oui, il m'a appris une chose 11 ❍ Oui, il m'a appris plusieurs choses 38

Avez-vous apprécié ce tutoriel ?

❍ Non, pas du tout 1 ❍ Non, pas vraiment 4 ❍ Oui, assez 33 ❍ Oui, beaucoup 13

Avez-vous apprécié les messages proposés dans ce tutoriel ?

❍ Non, pas du tout 0 ❍ Non, pas vraiment 5 ❍ Oui, assez 35 ❍ Oui, beaucoup 11

Avez-vous apprécié les mises en valeur de zones de l'écran (les cadres bleus) ?

❍ Non, pas du tout 0 ❍ Non, pas vraiment 4 ❍ Oui, assez 21 ❍ Oui, beaucoup 26

Avez-vous apprécié que le système détecte certaines de vos actions et passe automatiquement à l'étape suivante ?

❍ Non, pas du tout 1 ❍ Non, pas vraiment 2 ❍ Oui, assez 11 ❍ Oui, beaucoup 12

Pour passer à l'étape suivante d'un tutoriel, préférez-vous que le système :

❍ détecte automatiquement que vous avez terminé une étape 12

❍ vous demande d'indiquer quand vous êtes prêt (par exemple avec un bouton suivant) 14

Souhaiteriez-vous suivre d'autres tutoriels tels que celui qui vous a été proposé:

Sur NetBeans : ❍ Non 19 ❍ Oui : pour quelles fonctionnalités : 31.....

Sur une autre application : ❍ Non 22 ❍ Oui lesquelles : 15.....

Pensez-vous que ce tutoriel intégré à NetBeans soit :

Plus efficace qu'une vidéo de démonstration en ligne ?

❍ Non, il est moins efficace 4 ❍ Il est aussi efficace 20 ❍ Oui, il est plus efficace 27

Plus efficace qu'un tutoriel non intégré mais comportant des copies d'écran annotées ?

❍ Non, il est moins efficace 4 ❍ Il est aussi efficace 26 ❍ Oui, il est plus efficace 21

Plus efficace que des explications données par l'enseignant en cours magistral ?

❍ Non, il est moins efficace 6 ❍ Il est aussi efficace 10 ❍ Oui, il est plus efficace 35

Donnez un point positif sur ce tutoriel :.....

.....

.....

Donnez un point négatif sur ce tutoriel :.....

.....

.....

Commentaires et suggestions : .....

.....

.....

.....

.....

Merci d'avoir participé à cette étude !

Fig. 38 – Questionnaire final de satisfaction distribué aux participants suite à l'utilisation du tutoriel (la synthèse des réponses des participants est indiquée en rouge)

3. Expérimentations avec des concepteurs d'assistance informaticiens

M1if22 – logiciels éducatifs –2012-2013		
Thème	Ajouter un système d'assistance dans un EIAH en utilisant un éditeur d'assistance	
Type de logiciel	EIAH	
Public cible	primaire - collège	
Composition du binôme		
Étudiant 1 :		Étudiant 2 :
Application-cible		
Besoins d'assistance identifiés		
Description des règles d'assistance créées pour répondre aux besoins identifiés		
Description de l'assistance que vous n'avez pas pu créer avec l'éditeur		
Remarques pour l'amélioration de l'éditeur d'assistance		

Fig. 39 – Questionnaire final distribué aux participants suite à l'utilisation de SEPIA en 2013

Mise en place d'un système d'assistance épiphyte dans votre EIAH

L'objectif des prochaines séances est de mettre en place un système d'assistance pour l'EIAH que vous avez développé. Pour cela, vous allez utiliser le système SEPIA qui permet l'ajout de systèmes d'assistance à des applications-cibles existantes (ici votre projet) sans recourir à la programmation. Pour cela, SEPIA propose un éditeur d'assistance que vous utiliserez lors de la prochaine séance. L'assistance que vous aurez définie pourra ensuite être exécutée sur votre projet de manière épiphyte[1] par le moteur générique d'assistance de SEPIA. Les systèmes d'assistance mis en place avec SEPIA ne sont donc pas intégrés à l'application-cible, mais greffés sur elle. Pourtant, aux yeux de l'utilisateur final, qui reçoit l'assistance, ils semblent intégrés à l'application-cible.

Organisation :

Le TP du **12 mai** est consacré à la définition théorique sur papier d'un système d'assistance pour l'EIAH que vous avez développé. Vous devrez tout d'abord identifier les besoins d'assistance que les utilisateurs de votre EIAH peuvent rencontrer. Vous déterminerez de quelle manière vous souhaitez répondre à ces besoins d'assistance.

Pour ce premier TP, chaque binôme sera séparé pour constituer 2 groupes, et vous vous retrouverez en fin de séance pour mette en commun vos idées.

Le TP du **19 mai** est consacré à la mise en œuvre du système d'assistance que vous aurez conçu. Pour cela, vous utiliserez l'éditeur d'assistance de SEPIA en tant que concepteurs d'assistance. Vous pourrez ensuite tester l'exécution de ce système d'assistance en utilisant le moteur d'assistance de SEPIA en tant qu'utilisateurs finaux.

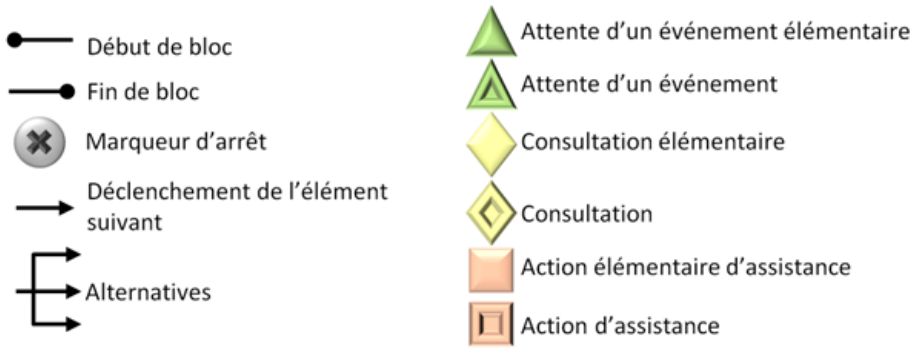
---

[1] Le terme épiphyte vient de la biologie : une plante est dite épiphyte si elle pousse sur une autre plante sans perturber le développement de cette autre plante. En informatique, un assistant épiphyte est une application capable de réaliser des actions d'assistance dans une application-cible sans perturber son fonctionnement et sans nécessiter un redéveloppement de cette application.

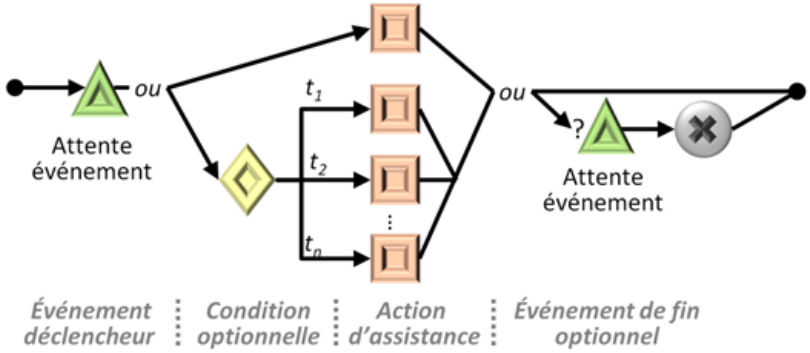
Fig. 40 – Consignes distribuées aux participants au début de la séance de conception d'assistances sur papier en 2014

Groupe aLDEAS

Le système SEPIA met en œuvre aLDEAS, un langage graphique qui a pour but la définition de systèmes d'assistance sous la forme d'un ensemble de règles d'assistance. aLDEAS est composé de différents éléments présentés ci-après.

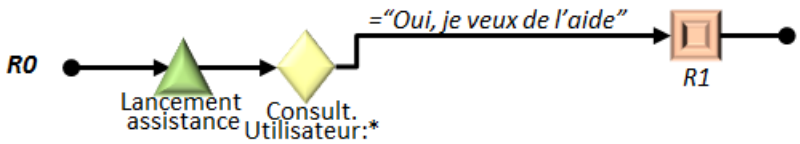


Les éléments d'aLDEAS peuvent être combinés pour former des règles d'assistance. Pour que votre système d'assistance soit compatible avec les outils du système SEPIA, il doit être composé d'un ensemble de règles respectant le patron de règles d'assistance suivant :

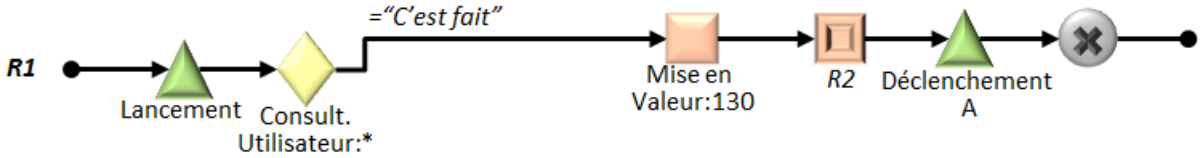


Définissez les règles d'assistance aLDEAS qui définiront le comportement de l'assistance lors de son exécution, en fonction des besoins d'assistance que vous avez identifiés.

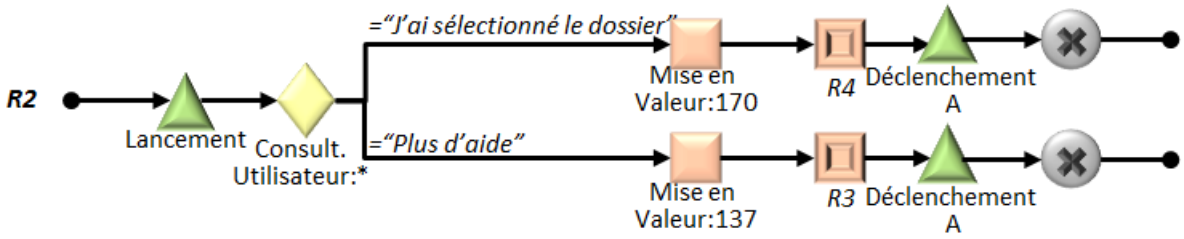
À titre d'exemple, voici le système d'assistance défini pour la correction des yeux rouges sur une photo avec le logiciel de retouche d'images PhotoScape. Ce système d'assistance est composé de 10 règles respectant le patron de règles d'assistance aLDEAS. Une vidéo de l'exécution de ce système d'assistance est disponible en http://liris.cnrs.fr/blandine.ginon/PhDWork.html.



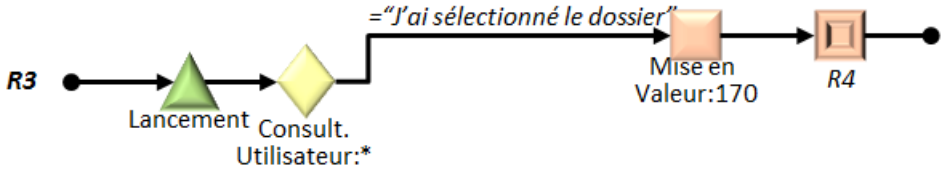
* « Souhaitez-vous de l'aide pour corriger les yeux rouges sur une photo ? »
{« Oui je veux de l'aide », Non merci »}



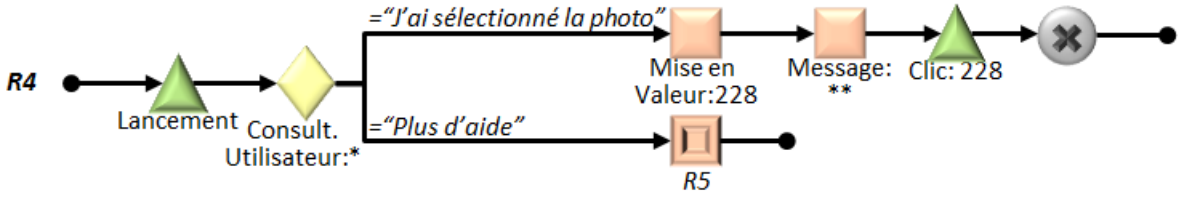
* « La correction des yeux rouges se trouve dans la partie "édition".
Cliquez sur l'onglet "édition" en haut de l'écran pour y accéder. ».
{« c'est fait »}



* « Pour choisir la photo à retoucher, sélectionnez le dossier dans lequel elle se trouve dans l'arborescence des fichiers. »
{« J'ai sélectionné le dossier », « Plus d'aide »}



*« La photo que vous souhaitez retoucher se trouve sur le bureau
Vous devez donc uniquement cliquer sur "Bureau " »
{« J'ai sélectionné le dossier »}



* « Cliquez maintenant sur la photo que vous souhaitez retoucher. »
{« J'ai sélectionné la photo », « Plus d'aide »}
** « Affichez l'onglet "outils" en cliquant dessus. »

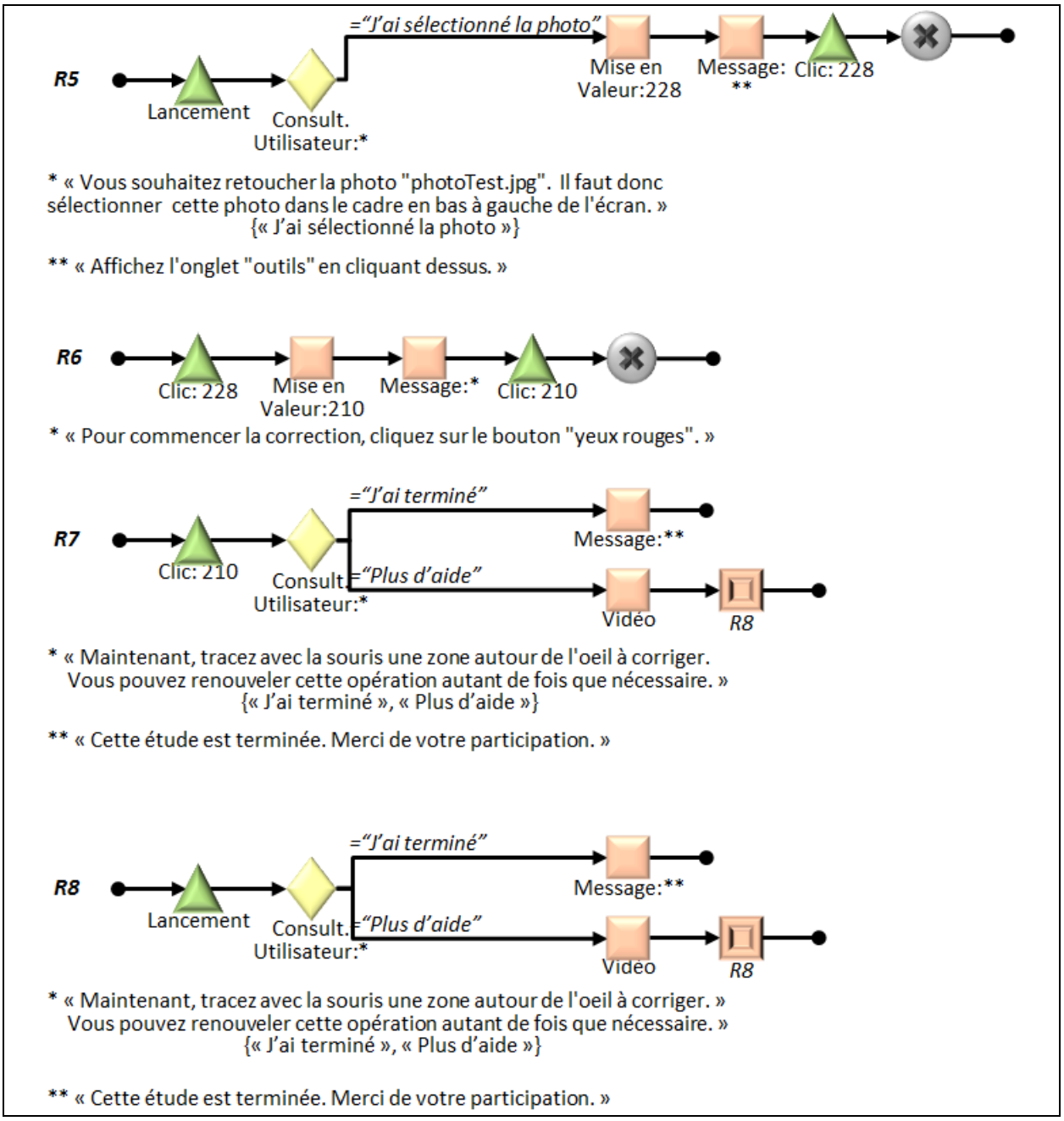
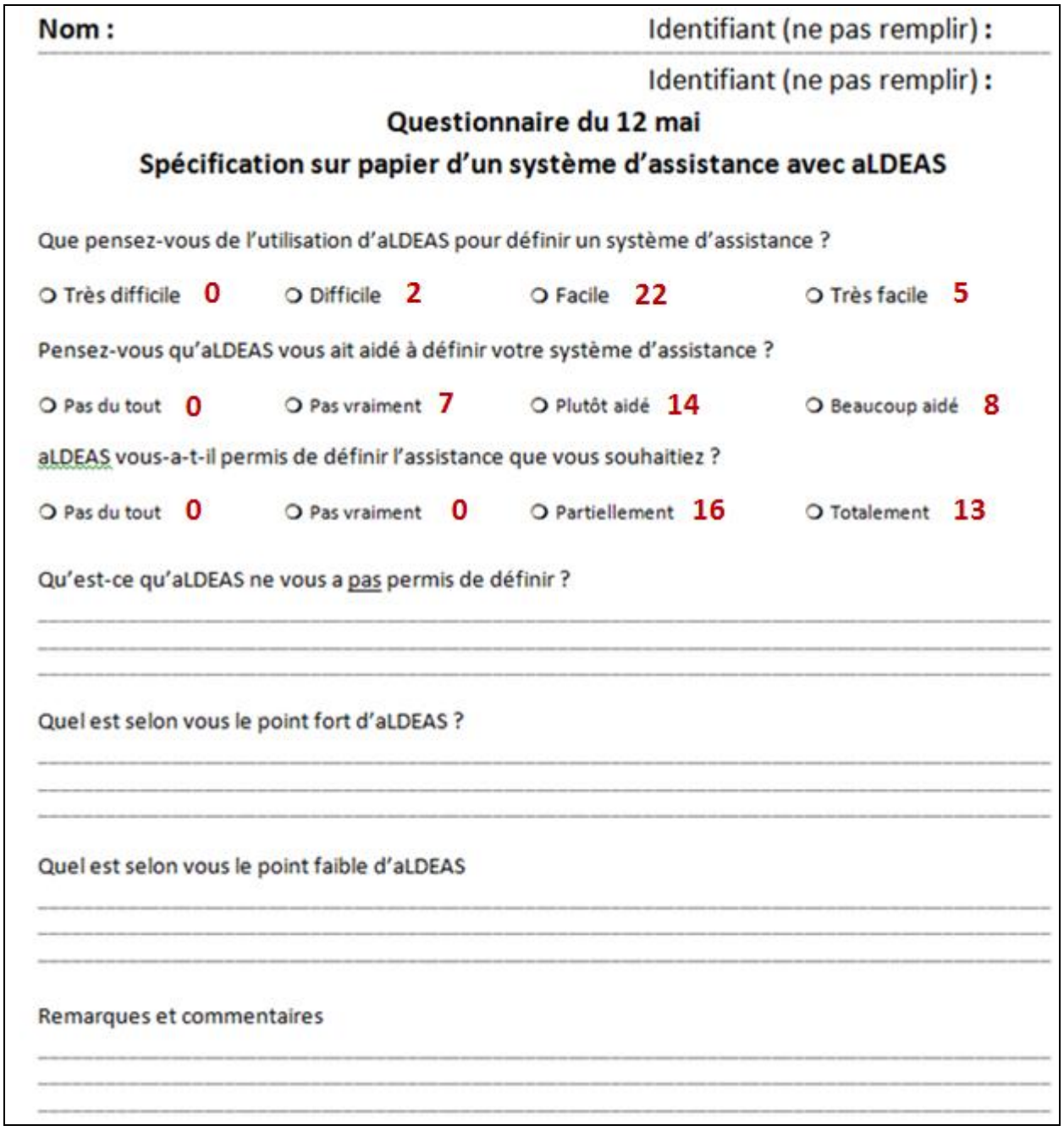


Fig. 41 – Consignes distribuées aux participants dans le groupe qui utilisait aLDEAS lors la séance de conception d'assistances sur papier en 2014



Nom : Identifiant (ne pas remplir) :

Identifiant (ne pas remplir) :

Questionnaire du 12 mai

Spécification sur papier d'un système d'assistance avec aLDEAS

Que pensez-vous de l'utilisation d'aLDEAS pour définir un système d'assistance ?

❍ Très difficile **0** ❍ Difficile **2** ❍ Facile **22** ❍ Très facile **5**

Pensez-vous qu'aLDEAS vous ait aidé à définir votre système d'assistance ?

❍ Pas du tout **0** ❍ Pas vraiment **7** ❍ Plutôt aidé **14** ❍ Beaucoup aidé **8**

aLDEAS vous-a-t-il permis de définir l'assistance que vous souhaitiez ?

❍ Pas du tout **0** ❍ Pas vraiment **0** ❍ Partiellement **16** ❍ Totalement **13**

Qu'est-ce qu'aLDEAS ne vous a pas permis de définir ?

Quel est selon vous le point fort d'aLDEAS ?

Quel est selon vous le point faible d'aLDEAS

Remarques et commentaires

Fig. 42 – Questionnaire de satisfaction distribué aux participants à l'issue de la séance de conception d'assistances sur papier en 2014 (la synthèse des réponses des participants est indiquée en rouge)

Nom : Identifiant (ne pas remplir) :

Identifiant (ne pas remplir) :

Questionnaire du 19 mai
Utilisation de SEPIA pour mettre en place un système d'assistance

Spécification de l'assistance avec l'éditeur

Partie description de l'application-cible

Pensez-vous qu'il est difficile de décrire une application-cible avec l'éditeur d'assistance ?

❍ Très difficile **0** ❍ Assez difficile **10** ❍ Assez facile **17** ❍ Très facile **0**

Pensez-vous qu'il est difficile de comprendre la description de l'application-cible créée avec l'éditeur d'assistance ?

❍ Très difficile **0** ❍ Assez difficile **12** ❍ Assez facile **14** ❍ Très facile **1**

Remarques et commentaires

Partie définition des règles d'assistance

Pensez-vous que la définition préalable des règles d'assistance en aLDEAS facilite la spécification de l'assistance avec l'éditeur ?

❍ Pas du tout **2** ❍ Pas vraiment **8** ❍ Plutôt **14** ❍ Beaucoup **3**

Créer une règle d'assistance avec l'éditeur vous a paru :

❍ Très difficile **2** ❍ Assez difficile **7** ❍ Assez facile **18** ❍ Très facile **0**

Créer une attente d'événement avec l'éditeur vous a paru :

❍ Très difficile **0** ❍ Assez difficile **9** ❍ Assez facile **16** ❍ Très facile **2**

Créer une condition avec l'éditeur vous a paru :

❍ Très difficile **1** ❍ Assez difficile **8** ❍ Assez facile **18** ❍ Très facile **1**

Créer une action d'assistance avec l'éditeur vous a paru :

❍ Très difficile **1** ❍ Assez difficile **8** ❍ Assez facile **17** ❍ Très facile **1**

Il y a-t-il des règles d'assistance aLDEAS que l'éditeur ne vous a pas permis de définir ?

❍ Non **20** ❍ Oui **7**

Si oui lesquelles ?

Remarques et commentaires

Personnalisation de l'assistance

Pensez-vous qu'il soit pertinent de personnaliser l'assistance :

En fonction des connaissances et compétences des utilisateurs ?

❍ Pas du tout pertinent **1** ❍ Assez peu pertinent **3** ❍ Assez pertinent **11** ❍ Très pertinent **12**

En fonction des handicaps des utilisateurs ?

❍ Pas du tout pertinent **0** ❍ Assez peu pertinent **1** ❍ Assez pertinent **17** ❍ Très pertinent **9**

En fonction des préférences des utilisateurs ?

❍ Pas du tout pertinent **1** ❍ Assez peu pertinent **12** ❍ Assez pertinent **9** ❍ Très pertinent **5**

Remarques et commentaires

Quel est selon vous le point fort de SEPIA ?

Quel est selon vous le point faible de SEPIA ?

Fig. 43 – Questionnaire de satisfaction distribué aux participants à l'issue de la séance de spécification et exécution d'assistances avec aLDEAS en 2014 (la synthèse des réponses des participants est indiquée en rouge)

Annexe F. Exemple de profil défini en PMDLe

La Fig. 44 donne l'exemple d'un profil défini en PMDLe, dans sa représentation en xml. Il s'agit du profil de Lucas Dupuis, pour un cours de programmation orientée objet. La structure de ce profil est commune à tous les étudiants de ce cours. Elle est composée de trois *briques* qui correspondent aux principaux concepts que les étudiants doivent maîtriser à la fin du semestre : « Modélisation » (cf. Ⓐ Fig. 44), « Design patterns » (cf. Ⓑ Fig. 44) et « Outils » (cf. © Fig. 44). Ces *briques* sont elles-mêmes décomposées en *composantes* et *sous-composantes*. Les éléments terminaux, c'est-à-dire ne contenant pas de *sous-composantes*, peuvent être associés à des *évaluations*, dont la valeur doit respecter l'*échelle* de la *brique*. Une évaluation est associée à une *date* et éventuellement à une *source*. Par exemple, lors du Contrôle Continu du 2013/10/17, Lucas a obtenu la valeur « partiellement maîtrisé » pour la composante « Diagrammes UML » de la brique « Modélisation » (cf. Ⓓ Fig. 44).

```
<structure id="142" nom="POO" createur="Blandine" creation="06/07/2014"
   derniere_modif="2013/11/06" nom_eleve="Dupuis" prenom_eleve="Lucas">
 Ⓐ <brique id="1" type="0" nom="Modélisation" indice="0">
      <infos_echelle><echelle id="3"/></infos_echelle>
      <arbre_des_composantes niveaux="1" ponderation="faux">
         <composante nom="Diagrammes UML">
            <valeur id_echelle="3" num="0">
               Ⓓ <evaluation date="2013/10/17" source="CC1">
                  partiellement maîtrisé</evaluation>
                  <evaluation date="2013/11/06" source="CC2">
                  partiellement maîtrisé</evaluation>
            </valeur>
         </composante>
         <composante nom="Diagrammes de classes">
            <valeur id_echelle="3" num="0">
                  <evaluation date="2013/10/17" source="CC1">
                  non maîtrisé</evaluation>
                  <evaluation date="2013/11/06" source="CC2">
                  maîtrisé</evaluation>
            </valeur>
         </composante>
         <composante nom="Diagrammes d'objets">
            <valeur id_echelle="3" num="0">
                  <evaluation date="2013/10/17" source="CC1">
                  partiellement maîtrisé</evaluation>
                  <evaluation date="2013/11/06" source="CC2">
                  non maîtrisé</evaluation>
            </valeur>
         </composante>
```

```
            <composante nom="Diagrammes de séquences">
                <valeur id_echelle="3" num="0">
                    <evaluation date="2013/10/17" source="CC1">
                    non maîtrisé</evaluation>
                    <evaluation date="2013/11/06" source="CC2">
                    partiellement maîtrisé</evaluation>
                </valeur>
            </composante>
        </arbre_des_composantes>
    </brique>
Ⓑ <brique id="2" type="0" nom="Design patterns" indice="1">
        <infos_echelle><echelle id="3"/></infos_echelle>
        <arbre_des_composantes niveaux="1" ponderation="faux">
            <composante nom="MVC">
                <valeur id_echelle="3" num="0">
                    <evaluation date="2013/10/17" source="CC1">
                    partiellement maîtrisé</evaluation>
                    <evaluation date="2013/11/06" source="CC2">
                    partiellement maîtrisé</evaluation>
                </valeur>
            </composante>
            <composante nom="Observer">
                <valeur id_echelle="3" num="0">
                    <evaluation date="2013/10/17" source="CC1">
                    partiellement maîtrisé</evaluation>
                    <evaluation date="2013/11/06" source="CC2">
                    non maîtrisé</evaluation>
                </valeur>
            </composante>
            <composante nom="Factory">
                <valeur id_echelle="3" num="0">
                    <evaluation date="2013/10/17" source="CC1">
                    partiellement maîtrisé</evaluation>
                    <evaluation date="2013/11/06" source="CC2">
                    maîtrisé</evaluation>
                </valeur>
            </composante>
        </arbre_des_composantes>
    </brique>
```

```
<brique id="3" type="0" nom="Outils" indice="2">
    <infos_echelle><echelle id="37"/></infos_echelle>
    <arbre_des_composantes niveaux="2" ponderation="faux">
        <composante nom="IDE">
            <sous_composante nom="NetBeans">
                <valeur id_echelle="37" num="0">
                    <evaluation date="2013/10/17" source="CC1">
                    Standart</evaluation>
                    <evaluation date="2013/11/06" source="CC2">
                    Expert</evaluation>
                </valeur>
            </sous_composante>
            <sous_composante nom="Eclipse">
                <valeur id_echelle="37" num="0">
                    <evaluation date="2013/10/17" source="CC1">
                    Standart</evaluation>
                    <evaluation date="2013/11/06" source="CC2">
                    Standart</evaluation>
                </valeur>
            </sous_composante>
        </composante>
        <composante nom="Gestion de projet">
            <sous_composante nom="SVN">
                <valeur id_echelle="37" num="0">
                    <evaluation date="2013/10/17" source="CC1">
                    Standart</evaluation>
                    <evaluation date="2013/11/06" source="CC2">
                    Standart</evaluation>
                </valeur>
            </sous_composante>
            <sous_composante nom="Forge">
                <valeur id_echelle="37" num="0">
                    <evaluation date="2013/10/17" source="CC1">
                    Novice</evaluation>
                    <evaluation date="2013/10/17" source="CC1">
                    Standart</evaluation>
                </valeur>
            </sous_composante>
        </composante>
    </arbre_des_composantes>
</brique>
</structure>
```

Fig. 44 – Exemple du profil de Lucas Dupuis, défini en PMDLe, pour un cours de programmation orientée objet

Annexe G. Principaux types de composants

Nous avons identifié les principaux composants de l'interface d'applications graphiques. Afin de faciliter la compréhension du lecteur, nous les présentons brièvement par la suite, en nous appuyant sur différentes illustrations.

Un **bouton** (cf. Fig. 45) est un composant sur lequel l'utilisateur peut cliquer, ce qui produit généralement un effet. Les boutons peuvent avoir des apparences très variées, ils peuvent contenir un texte et/ou une image. Un bouton peut notamment être associé à une infobulle.

Une **infobulle** (cf. Fig. 46) est un texte qui vise généralement à donner une explication sur le composant auquel elle est associée. Si un composant est associé à une infobulle, celle-ci apparait lors du survol de la souris sur le composant.

Fig. 45 – Exemples de composants de type bouton

Fig. 46 – Exemples d'infobulles affichées au survol d'un composant

Un **bouton radio** (cf. Fig. 47) est un composant constitué d'un petit cercle associé à un texte. Le cercle est rempli lorsque l'option qui correspond au bouton radio est sélectionnée, et vide sinon. Dans un groupe de boutons radio, un seul bouton radio peut être sélectionné à la fois. La sélection d'un bouton radio d'un groupe entraîne la désélection de tous les autres.

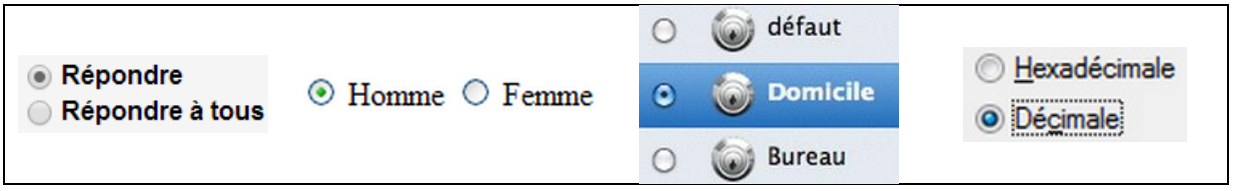


Fig. 47 – Exemples de composants de type bouton radio

Une **case à cocher** (cf. Fig. 48) est un composant constitué d'un carré associé à un texte. Le carré contient une croix ou un tick lorsque la case est cochée.

Fig. 48 – Exemples de composants de type case à cocher

Une **zone de saisie** (cf. Fig. 49) est un composant pouvant contenir du texte. Une zone de saisie est dite *éditable* lorsque l'utilisateur peut saisir ou modifier ce texte. Une zone de saisie peut contenir une seule ligne ou plusieurs. Lorsque le texte saisi est plus large ou plus haut que la taille de la zone de saisie, celle-ci peut être associée à une **barre de défilement** (cf.

partie droite de la Fig. 49), horizontale ou verticale, qui permet de visualiser tout le texte en faisant défiler le contenu de la zone de saisie.

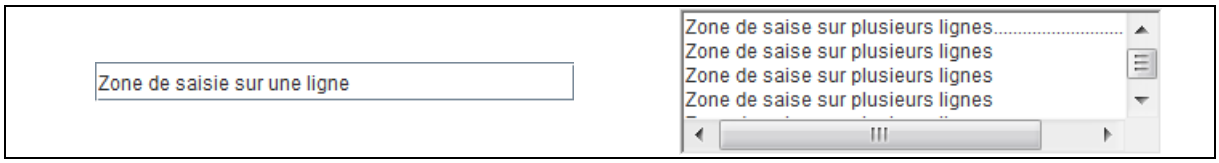


Fig. 49 – Exemples de composants de type zone de saisie

Une **liste déroulante** (cf. Fig. 50) est un composant constitué d'une partie principale qui contient une liste d'**items**, et d'un bouton permettant d'afficher cette liste pour sélectionner un item.

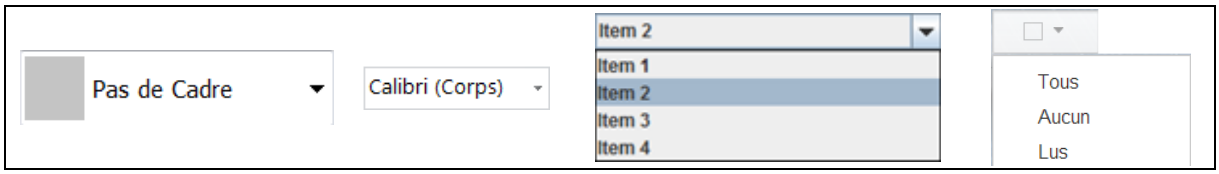


Fig. 50 – Exemples de composants de type liste déroulante :
les deux premières ne sont pas déroulées

Un **label** est une étiquette de texte affichée à l'écran, l'utilisateur ne peut pas interagir avec ce type de composants. Au contraire, un lien est un texte associé à un effet lorsque l'on clique dessus, comme l'ouverture d'une page.

Un **slider** (cf. Fig. 51) est un composant qui contient un curseur que l'on peut faire glisser d'une extrémité à l'autre du slider, pour définir une valeur comprise entre deux bornes selon un pas.

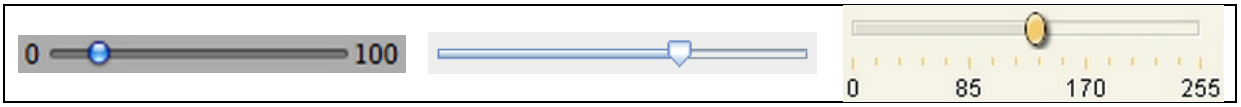


Fig. 51 – Exemples de composants de type slider

Un **spinner** est une molette d'incrément, constituée d'une zone de saisie et de deux boutons pour incrémenter ou décrémenter sa valeur. La valeur d'un spinner respecte un modèle, par exemple elle doit appartenir à un intervalle donné, comme [0, 100], ou à une liste de valeurs possibles, comme les mois de l'année. La Fig. 52 donne plusieurs exemples de composants de type spinner, associés à différents modèles de valeurs.

Fig. 52 – Exemples de composants de type spinner

Un composant de type **arbre** (cf. Fig. 53) contient un ensemble de nœuds hiérarchisés : les nœuds qui ne contiennent pas d'autres nœuds sont appelés « feuilles » par opposition aux « branches ». Un nœud peut être développé pour faire apparaître les nœuds qu'il contient, généralement grâce à un bouton « + ».

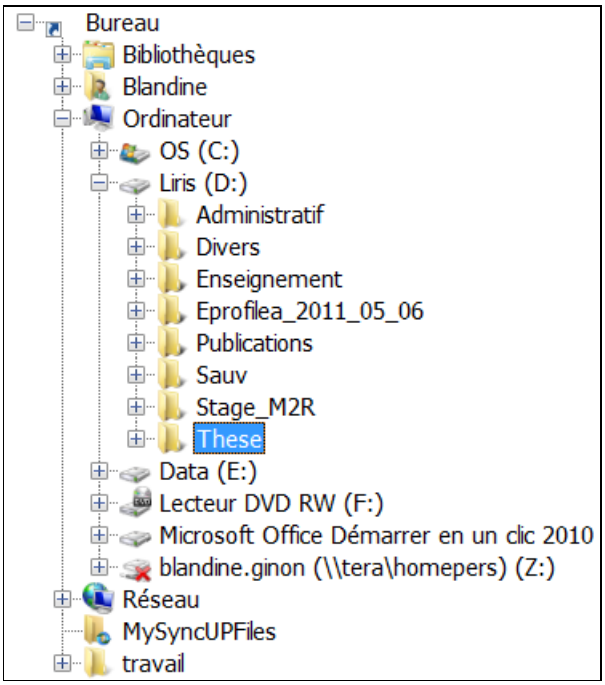


Fig. 53 – Exemple d'un composant de type arbre

Une **liste** contient un ensemble d'items. Un ou plusieurs de ces items peuvent être sélectionnés, selon les propriétés de la liste. La Fig. 54 donne l'exemple de plusieurs composants de type liste, avec des items de types variés.

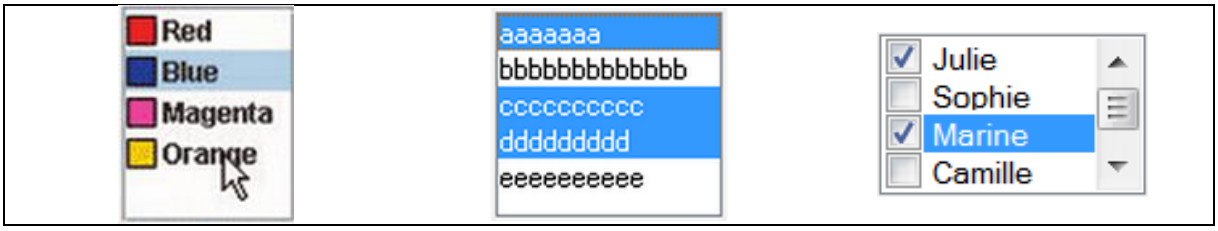


Fig. 54 – Exemples de composants de type liste

Un **menu** est un ensemble d'éléments, rassemblés sous une même étiquette, dont l'activation exécute des commandes associées. Un menu peut contenir plusieurs items, éventuellement associés à des raccourcis clavier. La Fig. 55 donne l'exemple d'une barre de menus : le menu « fichier » est ouvert. Une **fenêtre** ne contient qu'une seule barre de menus.

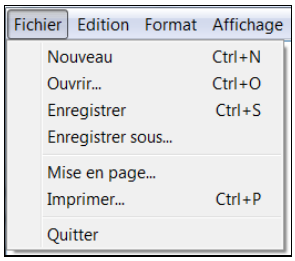
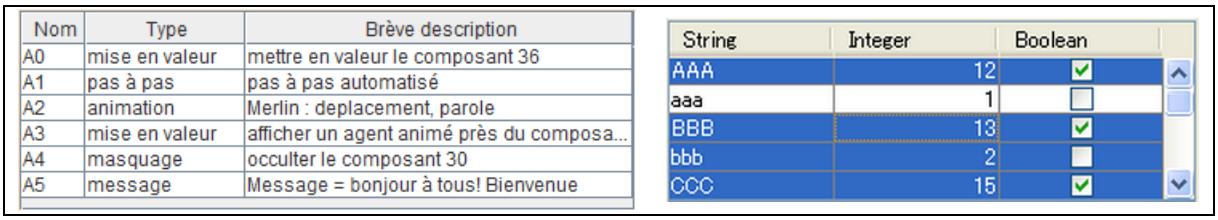


Fig. 55 – Exemple d'un composant de type menu

Une **table** (cf. Fig. 56) contient plusieurs lignes et colonnes, les cellules d'une table peuvent contenir différents types de composants.



Nom	Type	Brève description
A0	mise en valeur	mettre en valeur le composant 36
A1	pas à pas	pas à pas automatisé
A2	animation	Merlin : deplacement, parole
A3	mise en valeur	afficher un agent animé près du composa...
A4	masquage	occulter le composant 30
A5	message	Message = bonjour à tous! Bienvenue

String	Integer	Boolean
AAA	12	☑
aaa	1	☐
BBB	13	☑
bbb	2	☐
CCC	15	☑

Fig. 56 – Exemples de composants de type table

Un **panel** est un conteneur qui englobe d'autres composants : il peut être matérialisé à l'interface, par exemple par un encadré ou un effet 3D, ou être non perceptible par les utilisateurs. La Fig. 57 présente des exemples de composants de type panel, matérialisés de différentes manières.

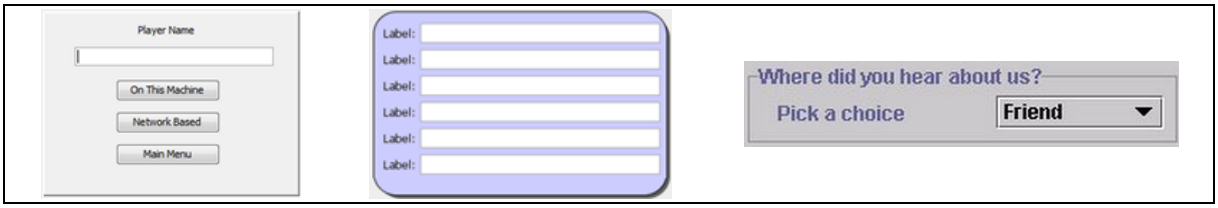


Fig. 57 – Exemples de composants de type panel

Annexe H. Glossaire

Action composée. Cf. action d'assistance aLDEAS.

Action d'agent animé. Action d'assistance permettant de fournir de l'assistance à l'utilisateur final en combinant plusieurs actions élémentaires d'un même personnage : messages, mises en valeur de composants, animations (montrer un composant, applaudir, saluer…), et déplacements à l'écran (cf. Section 5.2.2).

Action d'assistance aLDEAS. Élément d'aLDEAS permettant de fournir de l'assistance à l'utilisateur final. Une action d'assistance peut être élémentaire (cf. Section 5.1.3), ou être un bloc aLDEAS composé de plusieurs éléments du langage (cf. Section 5.1.4).

AGATE (Approche Générique d'Assistance aux Tâches complexEs). Projet de recherche visant à proposer des modèles et des outils pour permettre la mise en place *a posteriori* de systèmes d'assistance dans des applications existantes en adoptant une démarche entièrement générique.

aLDEAS (a Language to Define Epi-Assistance Systems). Langage grfaphique permettant de définir des systèmes d'assistance sous la forme d'un ensemble de règles (cf. Chapitre 5).

Alternative à n branches aLDEAS. Élément d'aLDEAS prenant en entrée la valeur renvoyée par l'élément qui précède l'alternative dans le bloc aLDEAS. Elle permet de conditionner la poursuite de l'exécution du bloc dans une ou plusieurs branches en associant à chaque branche un test sur cette valeur (cf. Section 5.1.2).

aMEAS (a Model to Execute aLDEAS Assistance Systems). Modèle expliquant le fonctionnement de la phase d'exécution d'un système d'assistance aLDEAS (cf. Chapitre 7).

Application-cible. Application à laquelle un système d'assistance est associé et dans laquelle l'assistance est exécutée.

Assistance à l'utilisateur d'applications informatiques. Ensemble des moyens permettant de faciliter la prise en main et l'utilisation d'une application, de manière adaptée à l'utilisateur final et au contexte d'utilisation. L'assistance vise à permettre à l'utilisateur final d'exploiter pleinement toutes les possibilités d'une application. Elle facilite l'appropriation des connaissances et compétences nécessaires à l'utilisation de cette application.

Attente d'événement aLDEAS. Élément d'aLDEAS permettant de suspendre l'exécution d'un bloc aLDEAS jusqu'à ce que l'événement concerné ait lieu. Une attente d'événement peut être élémentaire ou combiner un ensemble d'attentes d'événement élémentaire (cf. Section 0).

Bibliothèque d'accessibilité. Bibliothèque visant notamment à permettre l'utilisation de dispositifs d'accessibilité destinés aux personnes en situation de handicap (tels que les barrettes braille et les lecteurs d'écran), dans les applications compatibles. Dans le cadre de nos travaux, ces bibliothèques sont utilisées pour accéder à la hiérarchie des composants d'une application, pour inspecter les propriétés de ces composants, pour

détecter les événements ayant lieu à l'interface d'une application et pour automatiser des actions sur l'interface.

Bloc aLDEAS. Cf. action d'assistance aLDEAS.

Bounding rectangle. Propriété d'un composant de l'interface d'une application graphique correspondant aux coordonnées du rectangle englobant le composant à l'écran. Dans le système SEPIA, cette propriété est centrale pour les mises en valeur de composants.

Composant de l'interface d'une application graphique. Cf. Annexe G.

Concepteur d'assistance. Personne qui spécifie un système d'assistance. Il peut s'agir d'un expert de l'application-cible, qui peut éventuellement être le développeur de cette application. Ce terme peut également désigner un ensemble de personnes.

Condition de déclenchement aLDEAS. Dans une règle d'assistance aLDEAS, la condition de déclenchement est une consultation suivie d'une alternative à n branches, chacune associée à un test sur la valeur de retour de la consultation. La validation d'un tel test permet l'exécution de l'action d'assistance associée (cf. Section 5.2.1).

Consultation aLDEAS. Élément d'aLDEAS permettant d'obtenir une information sous la forme d'une valeur renvoyée par la consultation. Les consultations sont suivies par des alternatives à n branches, chacune associée à un test sur cette valeur. Ce test permet de conditionner la poursuite de l'exécution de la branche correspondante. Une consultation peut être élémentaire ou combiner un ensemble de consultations élémentaires (cf. Section 5.1.2).

cPMDLe. Modèle de contraintes sur profil associé à PMDLe (cf. Section 6.1.2).

Éditeur d'assistance. Outil permettant la spécification de l'assistance par un concepteur d'assistance. L'éditeur d'assistance de SEPIA permet ainsi la génération de fichiers XML de description de systèmes d'assistance aLDEAS, qui peuvent ensuite être exécutés automatiquement par le moteur de SEPIA.

EIAH (Environnement Informatique pour l'Apprentissage Humain).

Épi-assistant. Application capable de réaliser de manière épiphyte des actions d'assistance sur l'application-cible d'un système d'assistance. Dans le système SEPIA, les épi-assistants agissent à la demande du moteur d'assistance (cf. Section 9.3).

Épi-descripteur. Application informatique capable de générer une description de l'interface d'une application-cible. Les épi-descripteurs de SEPIA sont utilisés lors de la spécification de l'assistance avec l'éditeur d'assistance (cf. Section 8.2).

Épi-détecteur. Application informatique capable de surveiller les interactions entre un utilisateur final et l'interface d'une application-cible. Lors de l'exécution de l'assistance, les épi-détecteurs de SEPIA notifient le moteur d'assistance de tous les événements détectés (cf. Section 9.1).

Épi-inspecteur. Application informatique capable d'inspecter l'état d'une application-cible, et plus précisément de connaître la valeur des propriétés de composants de son interface. Les épi-inspecteurs de SEPIA sont appelés par le moteur d'assistance pour consulter l'état

de l'application-cible et pour obtenir les coordonnées d'un composant à mettre en valeur (cf. Section 9.2).

Épiphyte. Terme provenant du grec et signifiant littéralement « sur (*épi*) une plante (*phyte*) ». Il est utilisé en biologie pour désigner les plantes qui grandissent avec comme support une autre plante, l'hôte. Suivant la métaphore biologique de la plante épiphyte, en informatique, une application épiphyte [Paquette et al., 1994] est une application pouvant être greffée sur une application-cible.

Événement de fin aLDEAS. Association d'une attente d'événement suivie immédiatement du marqueur d'arrêt. Il met fin à toutes les actions d'assistance et consultations de l'utilisateur déclenchées depuis le début d'un bloc aLDEAS lorsque cet événement a lieu (cf. Section 5.1.4).

Événement déclencheur aLDEAS. Attente d'événement placée au début d'un bloc aLDEAS. Il permet de déclencher l'exécution de ce bloc lorsque l'événement a lieu (cf. Section 5.1.4).

Générique. Non spécifique à une application, un environnement ou un domaine.

Hiérarchie des composants. Représentation sous forme hiérarchique des composants d'une application graphique dans laquelle les fenêtres sont les composants de plus haut niveau.

IDE (Integrated Development Environnement). Outil de développement d'applications informatiques.

IHM (Interactions Homme-Machine).

kTBS (kernel for Trace Base System). Système de gestion de bases de traces développé par l'équipe Silex du laboratoire LIRIS.

Marqueur d'arrêt aLDEAS. Élément d'aLDEAS provoquant la fin de toutes les actions d'assistance déclenchées depuis le début du bloc aLDEAS, ainsi que des consultations de l'utilisateur.

Moteur générique d'assistance. Outil permettant l'exécution automatisée de l'assistance. Le moteur générique de SEPIA permet l'exécution automatisée des systèmes d'assistance aLDEAS.

Pas à pas automatisé. Action d'assistance de type pas à pas dans lequel le système d'assistance va réaliser les actions à la place de l'utilisateur (cf. Section 5.2.2).

Pas à pas guidé. Action d'assistance de type pas à pas dans lequel le système va demander à l'utilisateur de réaliser lui-même les actions. (cf. Section 5.2.2).

Pas à pas. Action d'assistance visant à faciliter la réalisation d'une tâche en la détaillant sous forme de plusieurs étapes. Chaque étape correspond à une action à réaliser sur un composant de l'interface de l'application-cible (cf. Section 5.2.2).

Patron d'action d'assistance aLDEAS. Structure visant à faciliter la définition avec aLDEAS d'actions composées (cf. Section 5.2).

PMDLe (Profiles MoDeling Language). Modèle de représentation de profils (cf. Section 6.1.1).

Présentation guidée. Action d'assistance permettant de fournir de l'assistance à l'utilisateur final. Elle comporte plusieurs étapes successives, dans lesquelles un composant de l'interface de l'application-cible est mis en valeur et éventuellement présenté par un message (cf. Section 5.2.2).

Profil d'utilisateur. Ensemble d'informations relatives à un utilisateur rassemblées dans un fichier pouvant concerner notamment ses connaissances et compétences, ses capacités et éventuels handicaps, ainsi que ses préférences.

Règle d'assistance. Bloc aLDEAS instanciant le patron de règles d'assistance associé à aLDEAS. Elle permet de fournir de l'assistance à l'utilisateur final. (cf. Section 5.2.1).

SEPIA (Specification and Execution of Personnalised Intelligent Assistance). Système mettant en œuvre nos propositions théoriques. Il est composé de deux principaux outils : l'éditeur d'assistance, qui permet la spécification de l'assistance par un concepteur, et le moteur générique d'assistance, qui exécute l'assistance pour les utilisateurs finaux.

Système d'assistance aLDEAS. Système d'assistance défini par un ensemble de règles respectant le patron de règles d'assistance aLDEAS.

Utilisateur final. Utilisateur d'une application informatique, qui bénéficie d'assistance lorsqu'un système d'assistance a été mis en place pour cette application-cible.

Résumé

Cette thèse en informatique se situe plus particulièrement dans le domaine de l'ingénierie des connaissances. Elle concerne la mise en place *a posteriori* de systèmes d'assistance dans des applications existantes, en adoptant une démarche générique.

Afin de permettre la mise en place de systèmes d'assistance dans des applications existantes sans avoir à les redévelopper ni à accéder à leur code source, nous avons choisi d'adopter une démarche entièrement épiphyte. Nous avons proposé un processus d'adjonction d'un système d'assistance à une application-cible de manière épiphyte. Il est constitué de deux phases : la spécification et l'exécution de l'assistance. La phase de spécification de l'assistance permet à un expert, le concepteur de l'assistance, de représenter ses connaissances relatives à l'application-cible et à l'assistance qu'il souhaite mettre en place dans celle-ci. La phase d'exécution de l'assistance exploite ces connaissances pour fournir aux utilisateurs finaux l'assistance souhaitée par le concepteur. Pour permettre d'une part la spécification de l'assistance par un concepteur potentiellement non-informaticien, et d'autre part l'exécution automatique de l'assistance spécifiée, nous avons proposé un langage pivot : aLDEAS. Ce langage graphique permet de définir des systèmes d'assistance très variés sous la forme d'un ensemble de règles.

Nos propositions théoriques ont été mises en œuvre de façon opérationnelle à travers le système SEPIA, constitué de différents outils. L'éditeur d'assistance de SEPIA est destiné aux concepteurs d'assistance et met en œuvre la phase de spécification de l'assistance. Il fournit aux concepteurs d'assistance une interface pour manipuler les éléments du langage aLDEAS, afin de définir un système d'assistance sous la forme d'un ensemble de règles aLDEAS. Les systèmes d'assistance aLDEAS peuvent ensuite être exécutés par le moteur générique d'assistance qui met en œuvre la phase d'exécution de l'assistance. Il permet de fournir l'assistance ainsi définie aux utilisateurs finaux des applications-cibles. Pour cela, le moteur d'assistance s'appuie sur différents outils épiphytes, pour surveiller et inspecter l'application-cible, ainsi que pour réaliser les actions d'assistance pour l'utilisateur final. Bien que mettant en œuvre des modèles génériques, le système SEPIA permet de mettre en place de l'assistance finement contextualisée et adaptée aux spécificités, d'une part de l'application-cible, et d'autre part des utilisateurs finaux.

Mots-clés

Assistance à l'utilisateur, approche épiphyte, approche générique, langage de définition d'assistance, ingénierie des connaissances, surveillance et inspection d'applications.