Modèle d'articulation entre les règles définissant un système d'assistance aLDEAS

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

Université de Lyon, CNRS, France
Université Lyon 1, LIRIS, UMR5205, F-69622, France
Bât. Nautibus - 23-25 av. Pierre de Coubertin, 69 100 Villeurbanne - France
le-vinh.thai@liris.cnrs.fr

Résumé

Le projet AGATE a donné naissance au système SEPIA qui permet à un concepteur d'assistance de définir des systèmes d'assistance greffés dans des applications-cibles (Windows, Java, Web…), sous forme d'un ensemble de règles aLDEAS. L'articulation entre ces règles était jusqu'à maintenant exprimée implicitement dans le langage aLDEAS. L'étude de systèmes d'assistance existants montre que cette articulation entre les règles d'assistance peut prendre des formes variées. Nous proposons dans cet article d'une part un modèle d'articulation entre règles comportant les cinq modes d'articulation que nous avons identifiés, et d'autre part l'implémentation de ce modèle au sein du système SEPIA. Ce modèle explicite et facilite la définition d'articulations entre les règles d'un système d'assistance.

Mots Clef

Système d'assistance, assistance à l'utilisateur, système à base de règles, langage.

1 Introduction

De nos jours, les applications informatiques sont très nombreuses, variées et indispensables. Cependant, beaucoup d'utilisateurs renoncent à utiliser certaines applications en raison de difficultés de prise en main et d'utilisation [1]. De même, les utilisateurs peuvent sous-exploiter l'application et ignorer certaines fonctionnalités à cause de leur complexité réelle ou supposée. L'assistance aux utilisateurs est l'une des solutions pour pallier ces difficultés. Cette assistance comporte différents aspects, chacun porteur d'une grande variété. Gapenne [2] a classifié l'assistance en fonction du degré d'intervention du système d'assistance : substitution, suppléance, assistance et aide. Ginon et al. [5] ont classifié les objectifs de l'assistance (compréhension du système, découverte d'une fonctionnalité, suivi de la tâche…), les techniques d'assistance (message, exemple, modification de l'interface, création automatisée…), ainsi que les approches d'assistance (manuel d'aide, aide contextualisée…). Dans cet article, nous nous intéressons à un autre aspect de l'assistance : l'articulation entre les différents éléments qui constituent un système d'assistance. Si faciliter la compréhension d'une application est l'objectif de l'assistance, et si la technique d'assistance utilisée pour atteindre cet objectif est l'affichage de messages, alors l'articulation dans l'assistance concerne l'ordre d'affichage de ces messages. Par exemple, les messages d'aide peuvent être affichés en même temps, l'un après l'autre selon un ordre préétabli ou en fonction des actions de l'utilisateur.

Pour présenter nos travaux sur la notion d'articulation au sein des systèmes d'assistance, nous présentons tout d'abord le projet AGATE, ainsi que le langage aLDEAS et le système SEPIA qui constituent le contexte de notre travail. Pour savoir comment décrire l'articulation dans l'assistance avec aLDEAS, nous présentons ensuite notre proposition de modèle d'articulation entre règles d'assistance qui s'appuie sur une étude de l'existant concernant les systèmes d'assistance d'applications variées. Nous décrivons notamment les cinq modes d'articulation que nous avons identifiés. Puis, nous présentons l'implémentation de ce modèle dans le système SEPIA, ainsi que l'évaluation que nous avons effectuée, avant de conclure.

2 Le projet AGATE

Le projet AGATE (Approche Générique d'Assistance aux Tâches complexEs) vise à proposer des modèles génériques et des outils unifiés pour permettre la mise en place de systèmes d'assistance dans des applications existantes, que nous appelons applications-cibles. Les modèles et outils proposés dans le cadre du projet AGATE ne sont spécifiques ni à une application ni à un domaine, mais peuvent au contraire être exploités pour ajouter une assistance à des applications les plus variées, sans que celles-ci aient été spécifiquement conçues pour permettre l'intégration d'une assistance. Pour permettre la mise en place de telles assistances, nous avons précédemment proposé un processus d'adjonction d'un système d'assistance épiphyte sur une application-cible [3]. Un système d'assistance épiphyte est un système qui peut être greffé sur une application-cible sans perturber son fonctionnement [7]. Ce processus comporte deux phases : la spécification de l'assistance, puis l'exécution de cette assistance de manière épiphyte.

La première phase est effectuée par un expert de l'application-cible, appelé par la suite concepteur d'assistance. Cette phase préparatoire permet au concepteur 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 spécifiée par le concepteur. Cette phase a lieu à chaque utilisation de l'application-cible par son utilisateur. Des épi-détecteurs rendent possible la surveillance de l'application-cible. Ils exploitent des bibliothèques d'accessibilité, chaque bibliothèque étant compatible avec un type d'application (applications Windows natives, applications Java, applications Web…) pour pouvoir détecter les événements liés aux interactions entre l'utilisateur et leurs interfaces. Enfin, des épi-assistants rendent possible l'élaboration de la réponse qui permet de fournir de l'assistance à l'utilisateur final, sous la forme d'une action d'assistance. Cette multiplicité des assistants épiphytes permet de diversifier l'assistance fournie. Ainsi, un message d'aide peut notamment être transmis à l'utilisateur final *via* une fenêtre pop-up, une synthèse vocale ou un agent animé. De même, la mise en valeur d'un composant, qui permet de guider l'utilisateur, peut prendre la forme d'un symbole affiché à côté du composant, d'un entourage de ce composant ou encore d'une modification de sa couleur.

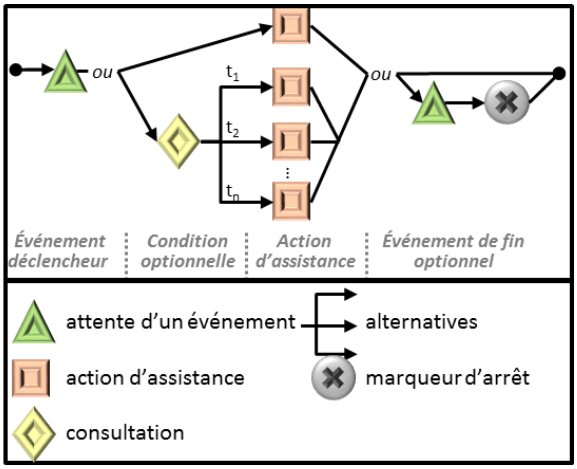

Figure 1 – Patron de règles aLDEAS

Le lien entre les deux phases du processus d'adjonction se fait grâce au langage pivot aLDEAS (a Language to Define Epi-Assistance Systems) [3]. Ce langage graphique se compose de 3 principaux éléments : attente d'événements (un clic sur un bouton…), consultation (du profil, de l'état de l'application…) et actions d'assistance (message, mise en valeur d'un élément de l'interface…). Ce langage est complété d'une part par un patron de règles (cf. Figure 1) qui permet de spécifier un système d'assistance par un ensemble de règles, et d'autre part par des patrons qui facilitent la définition d'actions d'assistance (patron pas-à-pas, présentation guidée…). Une règle débute par l'attente d'un événement nommé événement déclencheur. Dès que cet événement a lieu, le lancement de la ou des actions d'assistance est soit immédiat (chemin du haut sur la Figure 1), soit contraint par une condition (chemin du bas). Cette condition prend la forme d'une consultation avec des alternatives associées chacune à une de ces actions. Enfin, la règle peut se terminer par un événement de fin qui met fin à toutes les actions élémentaires déclenchées par cette règle.

aLDEAS et les patrons qui le complètent sont implémentés dans le système SEPIA [4] qui se compose de deux principaux outils : l'éditeur d'assistance et le moteur d'assistance. L'éditeur opérationnalise la phrase de spécification d'assistance. Il fournit une interface qui permet au concepteur de l'assistance de définir le système d'assistance en créant un ensemble de règles d'assistance en aLDEAS. Le moteur d'assistance opérationnalise la phase d'exécution d'assistance. Il permet d'exécuter le système d'assistance créé lors de la phase précédente en exécutant l'ensemble des règles d'assistance en aLDEAS définies avec l'éditeur d'assistance.

3 Modèle d'articulation entre les règles d'un système d'assistance

Si aLDEAS et sa mise en œuvre dans SEPIA permettent déjà de décrire des systèmes d'assistance comportant une articulation entre les éléments d'assistance, l'expression de l'articulation entre les règles aLDEAS est implicite et peut être complexe à définir pour le concepteur de l'assistance. Pour opérationnaliser plus efficacement ces modes d'articulation dans nos propositions, il est nécessaire de permettre l'explicitation de l'articulation entre les règles d'un système d'assistance aLDEAS.

Ainsi, un système d'assistance était précédemment défini dans le projet AGATE par un ensemble de règles aLDEAS de même niveau. Dans le patron de règles aLDEAS (cf. Figure 1), l'événement déclencheur, l'événement de fin et la condition de déclenchement sont des éléments centraux pour former l'articulation entre règles. Par exemple, pour définir l'articulation entre deux règles R_1 et R_2, telle que R_2 soit déclenchée lors de la fin de R_1, l'événement déclencheur de R_2 doit être l'événement « fin de R_1». Cette articulation entre règles est exprimée implicitement. En plus de se concentrer sur le contenu de chaque règle d'assistance, le concepteur d'assistance doit définir rigoureusement les événements et les conditions afin qu'ils assurent une articulation correcte entre les règles. Ces différents points entraînent une complexité dans la définition d'un système d'assistance.

Pour ces raisons, nous proposons de compléter notre langage par un modèle explicite d'articulation entre les règles d'assistance. Pour simplifier ici la représentation du modèle, nous ne notons que les règles entre lesquelles nous voulons étudier l'articulation. Ces règles sont nommées R_i, avec $i \in [1, n]$ ($n \geq 2$). La représentation globale de notre modèle est donnée en Figure 2. Elle donne une vue d'ensemble des cinq modes d'articulation que nous avons identifiés dans les différents travaux que nous avons étudiés: *indépendant*, *successif*, *simultané*, *progressif* et *interactif*. Le premier mode d'articulation,

indépendant (cf. partie haute de la Figure 2) exprime une absence d'articulation explicite entre règles, même s'il est toujours possible de trouver un lien, même faible entre elles. Ainsi, le mode d'articulation *indépendant* reflète la définition par défaut d'un système d'assistance en règles aLDEAS : les règles sont au même niveau et l'articulation entre elles est implicite.

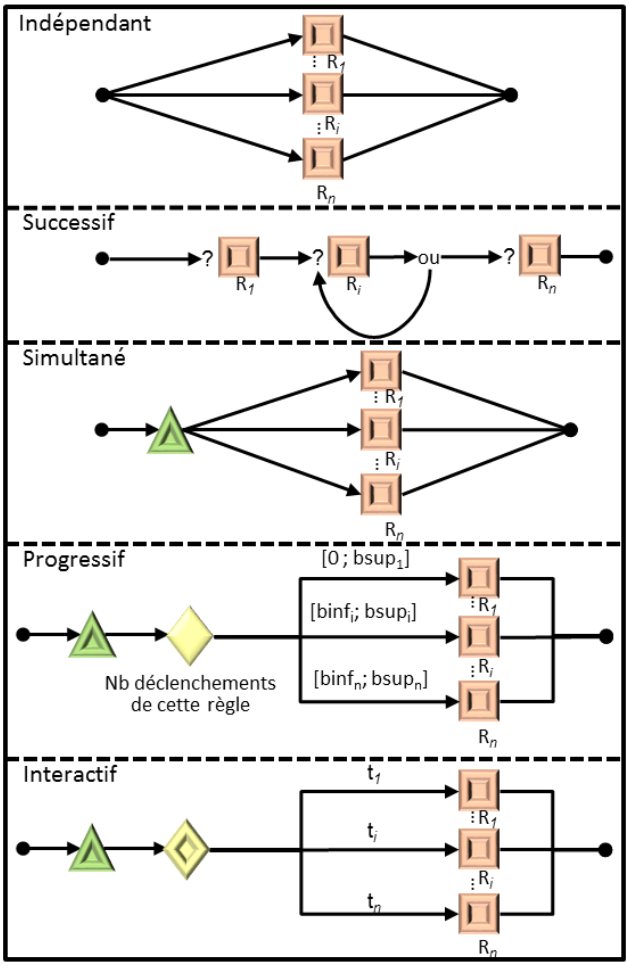

Figure 2 – Modèle d'articulation entre des règles d'assistance aLDEAS.

Dans chaque mode d'articulation, il existe des contraintes que des règles doivent respecter afin d'assurer l'articulation entre elles (par exemple, en mode *successif*, chaque règle doit être déclenchée par la fin de la règle précédente). Les contraintes propres à chaque mode d'articulation sont présentées dans la section suivante. De plus, ces modes peuvent être combinés pour fournir une assistance complète. La notion de combinaison de modes d'articulation est ensuite abordée.

3.1 Mode d'articulation successif

Dans le mode d'articulation *successif* (cf. Figure 2), les règles sont déclenchées les unes après les autres, c'est-à-dire qu'à la fin d'une règle R_i, la règle R_{i+1} est lancée. Ce mode d'articulation correspond à des assistances « étape par étape » telle que le tutoriel de Connectify[1].

Dans la définition détaillé de ce mode d'articulation (cf. ① Figure 3), nous pouvons observer que la règle R_i est contrainte à posséder un événement de fin et la règle R_{i+1} à posséder comme événement déclencheur « fin de R_i ». Cette contrainte est valable pour toutes les règles excepté la première et la dernière : la première règle R_1 peut débuter par n'importe quel(s) événement(s) déclencheur(s) et la dernière règle R_n peut se terminer par aucun, un ou plusieurs événements de fin. Dans ce mode d'articulation, la règle R_1 est un point d'entrée pour lancer l'ensemble des règles R_i. Les événements déclencheurs de cette règle R_1 sont donc également ceux qui lancent cet ensemble de règles.

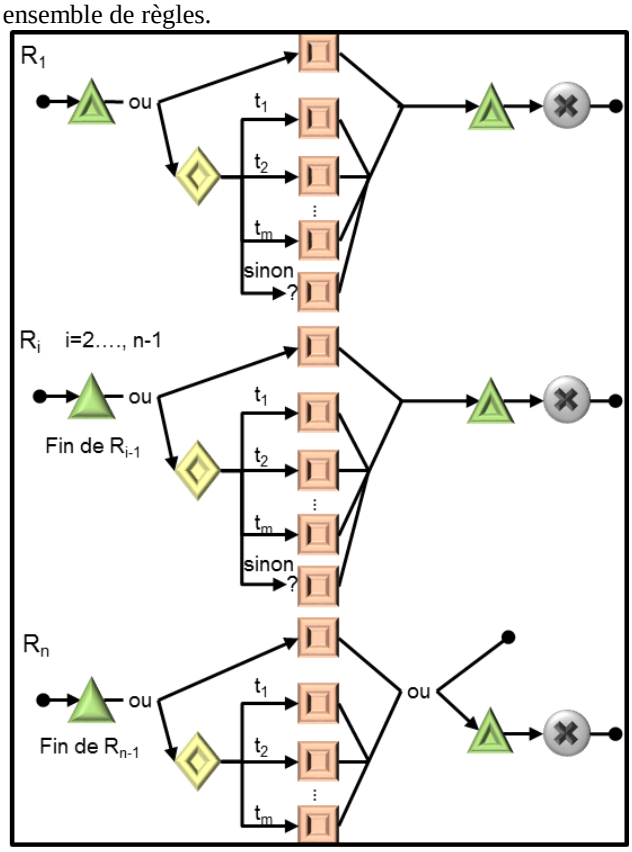

Figure 3 – Représentation en aLDEAS de n règles en mode d'articulation successif.

De plus, une règle R_{i+1} ne peut être lancée qu'après la fin de la règle R_i qui la précède. En conséquence, si R_i possède une condition de déclenchement qui n'est pas validée au moment de l'exécution de l'assistance, la règle ne sera pas exécutée jusqu'à son événement de fin, et la règle suivante ne sera donc pas déclenchée. Ainsi, en mode *successif*, si une condition de déclenchement n'est pas validée, toute la succession des règles est interrompue. Cela contraint une règle R_i contenant une condition à

[1]http://www.connectify.me/

posséder une alternative « sinon ». Cette alternative assure que la condition est toujours validée.

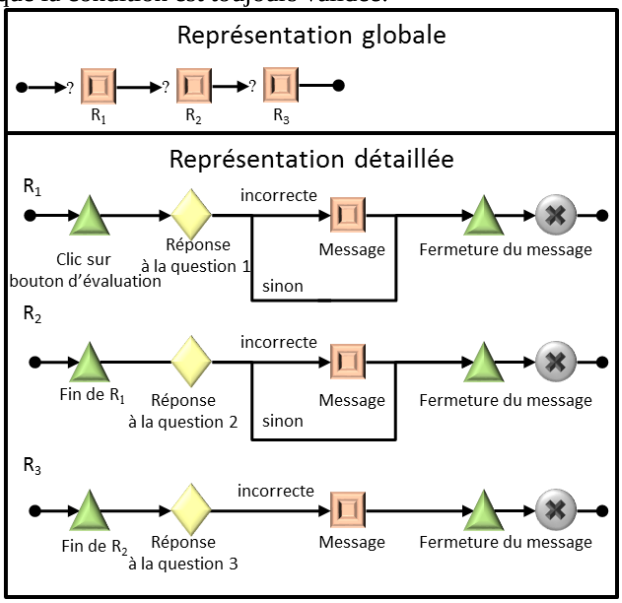

Figure 4 – Exemple de trois règles aLDEAS articulées

Prenons l'exemple d'une assistance pédagogique dans un logiciel éducatif (cf. Figure 4). Lorsque de l'utilisateur valide un exercice, l'assistance vérifie successivement les réponses de l'utilisateur pour les trois questions de l'exercice : en cas de réponse incorrecte, l'assistance fournit à l'utilisateur une information sur cette erreur, puis poursuit son exécution avec la question suivante jusqu'à la dernière question de l'exercice. Dans cet exemple, l'assistance est définie par un ensemble de trois règles articulées en mode *successif*, dans lequel chaque règle correspond à une question de l'exercice. Chaque règle contient une condition de déclenchement qui vérifie si la réponse donnée par l'utilisateur à la question est correcte et dans le cas contraire une explication est affichée à l'utilisateur. Dans ce cas, il peut être important de permettre à l'assistance de ne pas s'interrompre à la première réponse correcte de l'utilisateur, c'est-à-dire à la première règle dont la condition de déclenchement n'est pas validée. Pour cette raison, il faut ajouter une branche « sinon » dans toutes les règles (sauf la dernière). Lorsque la condition de déclenchement n'est pas validée, c'est-à-dire si aucun des tests des alternatives n'est vérifié, alors la règle prend fin dans la branche « sinon ». Les règles R_1, R_2 doivent obligatoirement posséder une branche « sinon », et des événements de fin. Les règles R_2 et R_3 doivent obligatoirement posséder un événement déclencheur, respectivement « fin de R_1 » et « fin de R_2 ».

3.2 Mode d'articulation simultané

Le mode d'articulation *simultané* (cf. Figure 2) permet de lancer plusieurs règles d'assistance en même temps, comme c'est le cas avec l'assistance du formulaire d'inscription de Google[2] qui met en valeur simultanément toutes les erreurs de saisies de l'utilisateur.

Dans le mode d'articulation *successif* (cf. section 3.1), la règle R_1 était le point d'entrée du système d'assistance pour lancer les règles R_i successivement, alors que dans ce mode d'articulation *simultané*, il faut une règle supplémentaire comme point d'entrée, afin de lancer toutes les règles R_i simultanément. Nous appelons cette règle R'. R' doit comporter un ou plusieurs événements déclencheurs quelconques, et les règles R_i doivent débuter par l'événement déclencheur « lancement par une règle (R') ».

3.3 Mode d'articulation progressif

Le mode d'articulation *progressif* (cf. Figure 2) permet de déclencher des règles d'assistance qui diffèrent selon le nombre de fois que l'utilisateur se retrouve dans une même situation problématique. En particulier, ce mode d'articulation permet de proposer progressivement à l'utilisateur une assistance de plus en plus importante pour répondre à un besoin d'assistance qui se répète. C'est le cas avec l'assistance pédagogique d'ActiveMaths [6] qui donne dans un premier temps un indice, puis la solution lorsque l'utilisateur donne plusieurs fois de suite une mauvaise réponse à une même question.

Dans ce mode d'articulation *progressif*, comme dans le mode d'articulation *simultané*, une règle R' sert de point d'entrée pour le lancement de l'assistance. Cette règle lance une seule règle parmi les règles R_i, en fonction du nombre de lancements de la règle R'. Le nombre de déclenchements d'une règle R_i dépend de l'intervalle $[binf_i, bsup_i]$: R_i sera déclenchée un nombre de fois compris entre la borne inférieure $binf_i$ et la borne supérieure $bsup_i$. Lors de ses $bsup_1$ premiers déclenchements, R' déclenche R_1, puis lors de ses déclenchements suivants, R' déclenchera R_2, puis R_3, etc. Dans ce mode d'articulation, les règles R_i doivent débuter par l'événement déclencheur « lancement par une règle (R') ».

3.4 Mode d'articulation interactif

Dans le mode d'articulation *interactif* (cf. Figure 2), une règle parmi les règles R_i est lancée en fonction d'une consultation du profil, de l'état de l'application, de l'historique de l'assistance, des traces et/ou de l'utilisateur. Par exemple, ce mode d'articulation permet de consulter l'utilisateur pour lui demander de choisir entre plusieurs assistances correspondant à différentes fonctionnalités du logiciel comme l'assistance dans le centre d'aide de Google[3]. Dans d'autres systèmes, l'assistance pour un utilisateur peut différer de celle proposée à d'autres utilisateurs [8] grâce à la consultation

[2] https://accounts.google.com/SignUp?
[3] https://support.google.com/chrome/

du profil de l'utilisateur contenant des informations relatives à ses préférences ou à son expérience.

À nouveau, une règle R' sert de point d'entrée pour le lancement de l'assistance. Chaque règle R_i est associée à une alternative de la condition de déclenchement de la règle R'. Les règles R_i doivent débuter par l'événement déclencheur « lancement par une règle (R') ». Le mode d'articulation *progressif* (cf. section 3.3) est un cas particulier du mode d'articulation *interactif*, fréquemment rencontré dans les systèmes d'assistance existants et dans lequel la condition de déclenchement de R' porte exclusivement sur un nombre de déclenchement.

3.5 Combinaison de modes d'articulation

Un système d'assistance donné n'utilise pas forcément un seul mode d'articulation entre règles. La combinaison de modes d'articulation peut concerner tous les modes que nous avons identifiés. Par exemple, les tutoriels fournissent une assistance étape par étape, mais dans une étape, ils peuvent simultanément afficher un message textuel et effectuer la mise en valeur d'un bouton.

4 Mise en œuvre des modes d'articulation entre règles d'un système d'assistance

Nous avons implémenté le modèle d'articulation que nous avons proposé (cf. section 3) dans SEPIA. Jusqu'à présent, SEPIA n'intégrait pas ce modèle d'articulation et un système d'assistance était composé d'un ensemble de règles aLDEAS sans articulation explicite entre elles, ce qui correspond au mode d'articulation que nous appelons *indépendant*. Nous avons donc enrichi l'éditeur d'assistance de SEPIA pour permettre au concepteur de l'assistance de définir explicitement des modes d'articulation entre les règles de son système d'assistance.

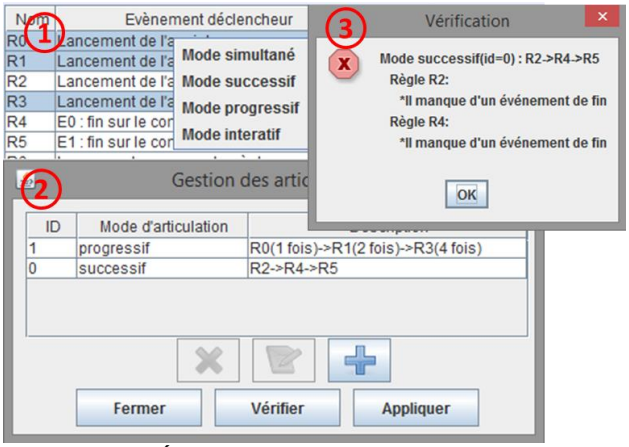

Figure 5 – Écrans de gestion des articulations entre règles dans le système SEPIA

Dans un premier temps, le concepteur d'assistance peut définir (cf. ① dans la Figure 5) et visualiser (cf. ② dans la Figure 5) les modes d'articulation entre les règles de son système d'assistance, qui sont par défaut articulées en mode indépendant. L'éditeur d'assistance complète alors si besoin les règles d'assistance concernées pour qu'elles respectent les contraintes imposées par le mode d'articulation choisi. L'éditeur génère automatiquement les événements manquants, qu'ils soient déclencheurs ou de fin. Par exemple, pour une articulation en mode *successif*, chaque règle doit avoir pour événement déclencheur la fin de la règle précédente. En cas de conflit avec la définition d'une règle d'assistance pour laquelle le concepteur de l'assistance avait précédemment défini un événement déclencheur ou de fin qui ne respecte pas les contraintes imposées par le mode d'articulation choisi, l'éditeur d'assistance informe le concepteur du problème (cf. ③ dans la Figure 5) afin qu'il choisisse entre accepter les modifications nécessaires et conserver la règle telle qu'elle en la retirant de l'ensemble des règles concernées par le mode d'articulation. La vérification du respect des contraintes imposées par un mode d'articulation peut être faite à tout moment.

5 Évaluation de nos propositions

Notre modèle d'articulation a été implémenté dans le système SEPIA de façon opérationnelle, ce qui montre la faisabilité de notre approche. Afin de tester cette mise en œuvre, nous avons utilisé cette nouvelle version de notre outil pour créer un système d'assistance assez riche pour un exercice de l'application web Mathématiques Faciles[4]. Cet exercice permet aux élèves de pratiquer l'addition en colonne sur des nombres à trois chiffres. L'élève doit saisir le résultat de l'addition en complétant les trois champs correspondant aux chiffres des unités, dizaines et centaines (cf. Figure 6).

Le système d'assistance, que nous avons créé avec SEPIA, affiche d'abord un bouton d'aide dès le lancement de l'assistance (cf. ① dans la Figure 6). Pendant la réalisation du premier exercice, l'élève peut cliquer sur ce bouton d'aide. Si sa réponse est mauvaise, le système d'assistance fournit des explications sur les étapes lors d'une première intervention, et un diagnostic lors des interventions suivantes. Le mode d'articulation est ici *progressif*. Toujours dans ce système d'assistance, les explications sur les différentes étapes de résolution de l'exercice sont affichées les unes après les autres (cf. ② dans la Figure 6). Le mode d'articulation est donc *successif* pour cette partie. Enfin, le diagnostic sur les différentes valeurs saisies est effectué simultanément, afin de mettre un symbole « erreur » à côté de chaque champ contenant une saisie incorrecte (cf. ③ dans la Figure 6). Le mode d'articulation de cette partie est donc *simultané*.

[4] http://www.mathematiquesfaciles.com/additions-a-trous-1-nombres-entiers_2_48182.htm

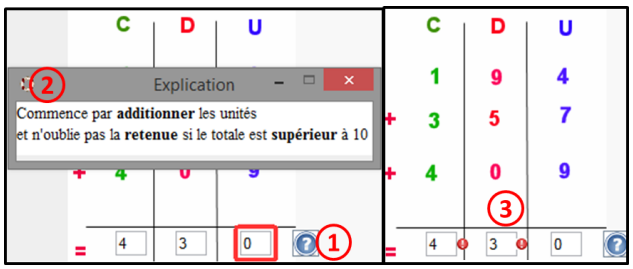

Figure 6 – Capture d'écrans des étapes d'assistance pour l'application Mathématiques Faciles.

Ce premier test a montré la pertinence de nos propositions et mis en lumière trois points forts. Tout d'abord, la version de SEPIA qui intègre notre modèle d'articulation entre règles permet de créer et visualiser explicitement des déroulements variés d'assistance. Ensuite, cette version assure que les règles respectent les contraintes liées aux articulations, ce qui réduit les erreurs pendant la définition de systèmes d'assistance par le concepteur d'assistance. Enfin, la nouvelle version de SEPIA décharge le concepteur d'une partie de ses tâches, grâce à la génération automatique de certains éléments aLDEAS nécessaires à la définition d'une assistance. En plus de la mise en œuvre des cinq modes d'articulation que nous avons identifiés, la combinaison de modes d'articulation est également possible. Toutefois, le concepteur doit actuellement établir manuellement cette combinaison, car l'interface du système n'intègre pas pour l'instant de définition graphique de cette combinaison.

Cette première évaluation de nos propositions nécessite bien sûr d'être complétée. Nous allons en effet créer d'autres systèmes d'assistance dans des contextes variés (reproduction de systèmes d'assistance existants ou créations) pour tester les différents modes d'articulation entre règles et leur combinaison. Nous prévoyons également de conduire des expérimentations confrontant un groupe de concepteurs d'assistance au modèle d'articulation et à la nouvelle version de SEPIA afin d'évaluer l'utilisabilité et l'utilité du modèle et de sa mise en œuvre. Nous demanderons pour cela à des concepteurs d'assistance de définir des systèmes d'assistance sur papier avec le langage aLDEAS sans connaissance du modèle d'articulation, et en ayant connaissance de ce modèle. Par ailleurs, ces concepteurs définiront des systèmes d'assistance tout d'abord avec la version de SEPIA n'intégrant pas le modèle d'articulation, puis avec la version intégrant ce modèle. Ces comparaisons devraient nous permettre d'évaluer la pertinence de notre modèle d'articulation et de sa mise en œuvre.

6 Conclusion

Dans cet article, nous avons présenté le modèle d'articulation entre les règles d'un système d'assistance que nous proposons pour compléter le langage aLDEAS. Ce modèle explicite la notion d'articulation entre règles d'un système d'assistance. Il propose pour cela cinq modes d'articulation correspondant aux articulations que nous avons identifiées en nous appuyant notamment sur une étude de systèmes d'assistance existants : *indépendant*, *successif*, *simultané*, *progressif*, *interactif*. Grâce à l'introduction de ce modèle dans notre approche, un système d'assistance n'est plus seulement défini par un ensemble de règles, mais peut également contenir des informations qui explicitent l'articulation entre ces règles. Nous avons implémenté ce modèle dans le système SEPIA, en ajoutant trois fonctionnalités principales à notre système : la spécification des modes d'articulation entre règles, l'application des contraintes sur les règles et la vérification des contraintes sur les règles. Notre utilisation de l'outil ainsi enrichi pour créer un système d'assistance combinant plusieurs modes d'articulations entre règles a démontré la faisabilité notre proposition.

Nous travaillons actuellement pour enrichir enrichir l'éditeur d'assistance de SEPIA afin de proposer au concepteur une visualisation graphique du système d'assistance qu'il crée. Cette interface permettra notamment de visualiser graphiquement l'ensemble des règles d'un système d'assistance et plus particulièrement les modes d'articulations entre elles.

Bibliographie

[1] Cordier, A., Lefevre, M., Jean-Daubias, S. & Guin, N., Concevoir des assistants intelligents pour des applications fortement orientées connaissances : problématiques, enjeux et étude de cas, *IC*, p. 119-131, 2010.

[2] Gapenne, O., Relation d'aide et transformation cognitive, *Intellectica*, p 7-16, 2006.

[3] Ginon, B., Jean-Daubias, S., Champin, P.-A. & Lefevre, M., aLDEAS : un langage de définition de systèmes d'assistance épiphytes, *IC*, p. 137-148, 2014.

[4] Ginon, B., Jean-Daubias, S., Champin, P.-A. & Lefevre, M., Setup of epiphytic assistance systems with SEPIA, *Ec-Tel*, p. 138-151, 2014.

[5] Ginon, B., Jean-Daubias, S., & Champin, P.-A.s Une typologie de l'assistance aux utilisateurs: exemple d'application aux EIAH, *RR-LIRIS-2013-007*, 2013.

[6] Melis, E., Andres, E., Budenbender, J., Frischauf, A., Goduadze, G., Libbrecht, P., Pollet, M. & Ullrich, C., ActiveMath: A Generic and Adaptive Web-Based Learning Environment, *IJAIED*, 12, p. 385–407, 2001.

[7] Paquette, G., Pachet, F., Giroux, S., Girard, J., Epitalk, a generic tool for the development of advisor systems, *IJAIED*, p. 349-370, 1996.

[8] Simonin, J., & Carbonell, N., Interfaces adaptatives Adaptation dynamique à l'utilisateur courant, *Interface numérique*, p. 37-54, 2007.