Lessons learned from the implementation of a reflexive groupware system

Grégory Bourguin

Les leçons d'une expérience dans la réalisation d'un collecticiel réflexif

Grégory Bourguin

LIL
40, rue Ferdinand Buisson
62228, Calais Cédex, France
bourguin@lil.univ-littoral.fr

RESUME

Le domaine de recherche sur le Travail Coopératif Assisté par Ordinateur (TCAO) a aujourd'hui bien intégré le besoin de malléabilité dans les collecticiels. Cependant, même si les apports théoriques des sciences humaines permettent d'appréhender le «pourquoi» de cette malléabilité, il est toujours difficile de savoir comment la réaliser. Une piste suivie par certains chercheurs est celle des systèmes réflexifs. Nous avons travaillé sur le projet DARE, un collecticiel réflexif, ce qui nous a amené à définir le principe de co-évolution. Le projet DARE étant terminé, nous proposons dans cet article de tirer les leçons de sa réalisation. Ces leçons sont mises à profit dans la réalisation de la plate-forme naissante CooLDA, un environnement réflexif, global et intégré de TCAO destiné à mieux répondre au principe de co-évolution.

MOTS CLES : TCAO, co-évolution, réflexivité, approche par modèles, Théorie de l'Activité.

ABSTRACT

Strongly influenced by the Human Sciences, the CSCW research domain has well integrated the need for malleability in groupware systems. However, even if we understand why groupware need malleability, it is still difficult to know how to realise it. Some CSCW researchers try to implement reflective systems. We've been working for a few years on the DARE project, a reflective groupware. We then have defined the co-evolution principle. DARE has now ended. This paper presents some of the lessons that came from this work. Theses lessons are used to realise a new reflective groupware called CooLDA (Cooperative Layer supporting Distributed Activities) that should better support the co-evolution.

KEYWORDS : CSCW, co-evolution, reflectivity, model approach, Activity Theory.

INTRODUCTION

Le domaine de recherche du TCAO connaît depuis quelques années une mini-révolution qu'on pourrait intituler «A la recherche de la malléabilité». Il est certain que cette révolution n'a pas simplement touché le TCAO et que la malléabilité est aujourd'hui une propriété fortement recherchée dans toute application logicielle. Du fait de son approche pluridisciplinaire, le TCAO a été fortement influencé par les recherches en sciences humaines qui ont permis non seulement d'identifier ce besoin, mais aussi de saisir son «pourquoi». La malléabilité n'est plus simplement requise du fait d'un constat empirique. Elle possède un fondement théorique permettant de l'appréhender à partir de propriétés fondamentales de l'activité humaine. Ces fondements peuvent être trouvés dans les travaux de nombreux chercheurs comme L. Suchmann [17] ou, d'une manière plus globale, dans des théories comme l'Ethnométhodologie ou la Théorie de l'Activité (TA) [1], pour ne citer que celles qui ont le plus influencé notre domaine de recherche. Toutefois, force est de constater que même si le besoin de malléabilité et son pourquoi sont clairement identifiés, les questions subsistent quant au «comment la réaliser».

Même si des avancées significatives liées à la meilleure compréhension théorique de la malléabilité ont été faites, de nombreux problèmes subsistent. Les chercheurs ont aujourd'hui dépassé le stade des méta-collecticiels permettant de générer d'une manière malléable des systèmes [5]. Les travaux se sont tournés vers la création de collecticiels très flexibles, voire réflexifs faisant écho aux propriétés réflexives intrinsèques des activités humaines à supporter. Un exemple des plus emblématiques et précurseur est le projet Worlds [18]. La communauté française n'est pas en reste et nous pouvons citer les travaux sur les workflows [14] ou encore ceux sur la « régulation» [7]. Notre propre apport au domaine s'intitule la «co-évolution» [3]. Elle fut initiée dans le cadre du projet DARE [2] et pose les enjeux que nous voulons atteindre. Néanmoins, après l'expérience DARE, des

questions aussi bien techniques que conceptuelles perdurent. C'est pourquoi nous poursuivons aujourd'hui nos travaux sur la co-évolution dans le projet CooLDA (Cooperative Layer supporting Distributed Activities), qui vise à réaliser un environnement réflexif, global et intégré de TCAO.

Cet article présente les fondements de notre nouvelle plate-forme CooLDA, voulant supporter la co-évolution et tirant certaines leçons de notre expérience dans la réalisation du projet DARE. Dans une première partie, nous rappelons brièvement notre approche de la co-évolution. La seconde partie décrit les leçons que nous avons tirées de nos travaux de réalisation dans DARE. La troisième partie décrit la réponse proposée par CooLDA aux problèmes évoqués.

UN ERGI SUPPORTANT LA COEVOLUTION

Le projet CooLDA, se situant dans la suite directe du projet DARE, vise à réaliser les mêmes objectifs généraux que ce dernier. Il s'agit de fournir aux utilisateurs un environnement global et intégré (EGI) de TCAO. Un EGI de TCAO se distingue des collecticiels isolés dans le sens où il a pour vocation de fournir un contexte articulant l'utilisation de divers collecticiels. Ainsi, notre objectif principal n'est pas de développer un tableau blanc partagé, un outil de chat ou de mèl, mais de fournir un environnement qui intègre et articule l'utilisation de ces artéfacts. Il existe divers travaux de recherche sur la création de collecticiels malléables intégrateurs de composants [14][8]. Cependant, du fait de la granularité des composants que nous voulons articuler (allant de simples composants d'awareness jusqu'à des collecticiels à part entière), l'exemple le plus proche de nos travaux reste Orbit [12] qui utilise la théorie des «□Locales□» et mondes sociaux pour spécifier ses contextes d'intégration. CooLDA, comme DARE, est fortement inspiré des travaux liés à la Théorie de l'Activité, c'est pourquoi nous nommons ces contextes des «□supports d'activités□». Un support d'activité permet aux utilisateurs d'accéder à leurs ressources (coopératives ou non) en fonction de leur rôle. De plus, nos travaux sur la Théorie de l'Activité nous ont aussi permis d'identifier deux propriétés fondamentales de l'activité humaine qu'il est souhaitable de supporter au sein d'un EGI. Ces propriétés sont la réflexivité ainsi que la cristallisation de l'expérience.

La réflexivité est une propriété qui a été identifiée par de nombreux chercheurs dans le TCAO. Néanmoins, on distingue deux approches de la réflexivité. La plupart des travaux la considèrent du point de vue des développeurs [4]. Ainsi l'environnement est malléable car ses concepteurs peuvent accéder et modifier sa définition au cours de sa propre exécution. Au regard des apports des sciences humaines, nous avons déjà souligné [3] qu'il est préférable de s'inscrire dans une approche qui tente d'aller plus loin et veut fournir un accès à ces mécanismes réflexifs aux utilisateurs finaux. Ainsi, la malléabilité est directement accessible à ceux qui en ressentent le besoin, in situ. Dans le cas de DARE et de CooLDA, on peut alors parle d'ERGI (Environnement Réflexif Global et Intégré). Dans cette approche, le modèle d'exécution du collecticiel est réifié. Les utilisateurs peuvent ainsi par exemple redéfinir dynamiquement les jeux de rôles qui régissent leurs activités coopératives. Toutefois, DARE et CooLDA se distinguent encore des autres collecticiels réflexifs en voulant supporter la co-évolution [3], une approche plus globale et coopérative de la malléabilité.

Les systèmes de TCAO possédant des propriétés réflexives qui sont offertes aux utilisateurs finaux les intègrent souvent comme des fonctionnalités supplémentaires données à un type d'utilisateurs prédéfini du système [6]. Dans DARE, la redéfinition du système par ses utilisateurs est considérée comme une partie de l'activité coopérative nommée méta-activité. Toute activité contient une méta-activité sous forme d'un ajout de ressources permettant d'accéder au niveau méta du collecticiel. Les acteurs peuvent alors coopérativement redéfinir l'environnement de coopération en fonction de leurs rôles.

La propriété de cristallisation et réutilisation de l'expérience identifiée dans la TA est aussi un élément qui distingue notre approche. La TA souligne qu'un artéfact transformé au cours des activités par ses utilisateurs cristallise leur expérience. Au-delà de la mise en place de mécanismes réflexifs, nous mettons donc aussi l'accent sur ce mécanisme de cristallisation et voulons mettre à profit l'expérience des utilisateurs de notre ERGI. Une telle expérience est par exemple la spécification d'un jeu de rôles particulier pour un type d'activité particulière. Si ce jeu de rôle évolue au cours d'une activité, du fait de l'expérience des acteurs, nous voulons qu'il soit accessible à ceux qui désirent le mettre en œuvre et qui le feront peut-être eux-mêmes évoluer.

Voici donc, résumés très brièvement, les objectifs que nous nous sommes fixés dans la création de DARE. Cependant, nous montrerons dans la partie suivante que certains choix de conception de DARE ont influencé et limité notre vision de l'ERGI et de la co-évolution. Le but de la plate-forme CooLDA est de pallier ces problèmes.

LES LECONS DU PROJET DARE

Nécessité d'un modèle récursif sans décrochement

Dans DARE, une activité est définie comme un ensemble d'acteurs (ou sujets en référence au modèle basique d'activité d'Engeström largement repris dans la TA) utilisant un ensemble de ressources en fonction de leur rôle.

Une ressource correspond par exemple à l'interface cliente d'un tableau blanc partagé. Seuls les utilisateurs membres d'une activité peuvent utiliser ses ressources. Nous pensons aujourd'hui que cette approche crée un décrochement néfaste dans notre modèle conceptuel. Ce décrochement est encore plus néfaste au regard de la co-évolution.

Pour fournir un accès aux spécifications d'une activité de base, il faut fournir aux utilisateurs une ressource qui accède au niveau méta du système. Dans DARE, cette ressource est une des ressources de l'activité de base, ce qui en contraint l'utilisation. Par exemple, dans le contexte d'une activité «□cours de mathématique□», il peut être important pour un «□enseignant□» d'accéder à la redéfinition de sa séance. Il suffit alors de lui fournir une ressource pour laquelle il a des droits particuliers (au regard des étudiants par exemple). Le problème se pose si un «□administrateur□» de l'environnement doit aussi avoir accès à ce niveau méta. L'inscription de l'administrateur en tant que membre du cours serait artificielle. Il est possible de lui fournir l'accès à cette ressource en l'ajoutant à son activité d'administration. Cependant, du point de vue de la co-évolution, l'enseignant doit pouvoir faire évoluer son environnement en fonction du déroulement de son activité, ceci dans un processus coopératif impliquant potentiellement aussi bien ses élèves que l'administrateur. Nous sommes aujourd'hui convaincus que l'accès au niveau méta n'est pas le simple fait d'un ajout de ressource dans l'activité. La méta-activité n'est pas qu'une partie de l'activité de base, elle constitue une activité à part entière impliquant plusieurs acteurs provenant eux-mêmes d'activités différentes mais liées.

Une étude des travaux de Kuutti sur la TA et les organisations [11] nous conforte dans cette nouvelle approche. Kuutti souligne comment les sujets sont souvent impliqués dans différentes activités interconnectées. Le produit d'une activité peut constituer un des artéfacts d'une autre activité. De notre point de vue, une méta-activité a pour but la production (évolution) de tout ou partie des spécifications de son activité de base. Ainsi, les acteurs de la méta-activité peuvent aussi être impliqués dans l'activité de base, mais pas obligatoirement. Du fait de nos choix de conception, ce n'était pas le cas dans DARE. Il est donc nécessaire que nous fassions évoluer son modèle conceptuel pour que CooLDA possède des fondements plus aptes à supporter cette co-évolution.

Les bienfaits de l'approche par modèles ?

Pour réaliser DARE, nous nous sommes fortement inspirés du Meta-Object Protocol ou MOP [9]. D'autres chercheurs ont d'ailleurs eux aussi utilisé cette approche pour réaliser plus aisément les dimensions réflexives de leur collecticiels [4]. Adaptant l'approche MOP, la réflexivité dans DARE est rendue possible grâce à l'introduction d'un méta-modèle dont l'entité majeure est la tâche. Une tâche correspond à une spécification d'activité. Elle décrit les objectifs de l'activité, les ressources qui seront disponibles pour les réaliser ainsi que les rôles qui seront mis en œuvre pour la coopération des acteurs. Chaque tâche est un modèle d'activité et il est possible de créer différentes instances de ce modèle.

Il est important de souligner que nous utilisons la tâche d'une manière différente que dans l'approche classique fortement critiquée ces dernières années, la modélisation de la tâche étant jugée trop rigide et peu favorable à la malléabilité souhaitée. Notre tâche est bien un modèle qui décrit comment doit se dérouler l'activité coopérative, mais ce modèle est évolutif□ Il décrit la manière dont le système supporte l'activité à un instant donné. Du fait d'une relation causale entre une tâche et les activités qui la réalisent, toute modification de la tâche entraîne des répercussions directes sur les activités correspondantes. Ainsi notre approche de la tâche peut être comparée aux plans de Suchman [17]□une tâche est nécessaire (en particulier dans un environnement informatisé qui la réalise) mais révisable in situ.

L'approche par modèles est fort intéressante, en particulier du point de vue de la cristallisation de l'expérience. Lorsque les acteurs dans une activité accèdent à son niveau méta, ils modifient la tâche, c'est-à-dire un modèle dans lequel l'expérience alors cristallisée peut être directement mise à profit. DARE repose sur le MOP implémenté dans le langage Smalltalk. Il réalise la tâche en tant que méta-classe du système dont les méthodes définissent l'interface de méta-niveau qui permettent la redéfinition des activités. Chaque tâche est une classe et chaque activité est l'instance d'une tâche. Les mécanismes du langage sous-jacent ont donc été réutilisés pour implémenter ceux de notre collecticiel. Bien que cette démarche facilite grandement notre travail de développement, notre meilleure compréhension des besoins de la co-évolution la questionne aujourd'hui.

Une des questions majeures est celle de la stabilisation du système dans le cadre des approches réflexives. Pour qu'une expérience naisse d'une activité, il est nécessaire que le système passe par des phases stables. Les acteurs ne pourront correctement comprendre et faire évoluer leur environnement que si celui-ci ne cesse temporairement de changer. Nous espérons que les dimensions humaines et coopératives de notre approche permettront de pallier ce problème□ dans la co-évolution, toute évolution devrait être le résultat d'un processus coopératif. Ceci paraît réalisable dans le contexte d'activités fortement interconnectées. Le questionnement est plus grand pour l'évolution d'une tâche dont existent plusieurs activités disjointes, impliquant des utilisateurs différents. Puisqu'il existe une relation causale entre une tâche et

ses activités-instances, les acteurs qui accèdent aux propriétés réflexives du système et modifient leur tâche transforment aussi la spécification de toutes les autres activités issues de cette tâche. Se pose alors un dilemme : faut-il ou non faire profiter toutes les activités en cours ou futures d'une expérience développée dans une activité particulière similaire ?

L'utilisation des mécanismes du langage objet sous-jacent nous questionne aussi au niveau de la notion d'héritage. L'héritage est intéressant car il permet de définir une tâche à partir d'une autre. Comme dans le monde objet, c'est un moyen efficace pour réutiliser de l'expérience. Un autre point appréciable de l'héritage est qu'il facilite, d'un certain point de vue, la gestion de l'évolution des modèles : la modification d'un modèle entraîne des modifications automatiques dans tous les modèles qui en héritent et il n'est pas nécessaire de les modifier explicitement. Même si ce mécanisme est intéressant, il nous questionne pour nos modèles d'activités : une tâche définie à partir d'une autre doit-elle toujours subir les répercussions des transformations de celle dont elle est issue ? Bien qu'il paraisse intéressant, le mécanisme d'héritage nous questionne aussi sur les implications de la réflexivité du collecticiel et de l'approche par modèles.

Un dernier point est que l'approche par modèle telle que nous l'avons implémentée dans DARE implique toujours une démarche d'abstraction. Même si l'acteur a la possibilité de modifier dynamiquement le modèle d'exécution de son environnement, il doit accéder au niveau méta lui offrant un point de vue sur sa tâche, abstraire sa modification, la formaliser et l'appliquer. Ce processus peut s'avérer coûteux cognitivement pour les utilisateurs, ce qui introduirait un déséquilibre entre leur motivation pour réaliser leur activité et l'effort qu'ils doivent fournir pour adapter leur contexte. Dans ce cas, même si les mécanismes réflexifs sont disponibles, l'utilisateur peut refuser de les utiliser et abandonner son activité, ce qui correspond à un échec du système. Il existe différentes approches qui critiquent la notion de tâche de ce point de vue. On peut par exemple citer la méthode du « storytelling » [16] qui attire notre attention et tente de construire de l'expérience à partir d'histoires particulières rapportées. Néanmoins, cette approche qui semble fort prometteuse pour cristalliser l'expérience apparaît aujourd'hui difficilement utilisable pour nos problèmes. Même si nous accordons crédits à ces critiques de la modélisation par la tâche (même évolutive), nous pensons que notre approche par modèles reste fort intéressante pour la réflexivité, la cristallisation et la réutilisation de l'expérience. La tâche est un moyen d'abstraction puissant car générique et d'un niveau plus accessible que les entités d'un langage de programmation. Néanmoins, nous pensons aujourd'hui qu'il doit être possible et bénéfique de fournir aussi des moyens pour transformer la tâche à partir d'une activité sans forcer l'utilisateur à toujours entrer dans une description formelle de ses besoins.

Un nouveau départ

L'objectif principal de DARE était de vérifier la faisabilité d'un ERGI de TCAO. Comme nous venons de le montrer, cette problématique s'est enrichie de nouvelles questions. C'est dans le but d'y apporter des éléments de réponses que nous avons initié la réalisation de la plate-forme CooLDA.

LA PLATE-FORME COOLDA

La plate-forme CooLDA doit fournir un environnement générique permettant de spécifier et faire évoluer des ERGI supportant des activités coopératives particulières. Elle constitue, entre autres, le fondement d'un environnement plus spécialisé nommé CooLDev développé dans une ACI jeunes chercheurs du même nom. CooLDev a pour vocation le support d'activités coopératives d'équipes de développement de logiciels.

Notre objectif n'est pas d'exposer dans ce papier toutes les spécificités de CooLDA, en particulier son architecture, son mode de distribution, ses interfaces utilisateurs, etc. Nous voulons ici mettre en exergue les choix réalisés en fonction des questions posées dans la partie précédente suite à DARE. C'est pourquoi nous présenterons dans un premier temps son nouveau modèle qui pallie le problème de décrochement conceptuel. Nous présenterons ensuite nos travaux sur les mécanismes de son nouveau noyau réflexif.

Un nouveau modèle

Dans CooLDA et pour notre approche réflexive par modèle, toute activité est instance d'une tâche. Le modèle de tâche de CooLDA est présenté dans la *figure 1.*

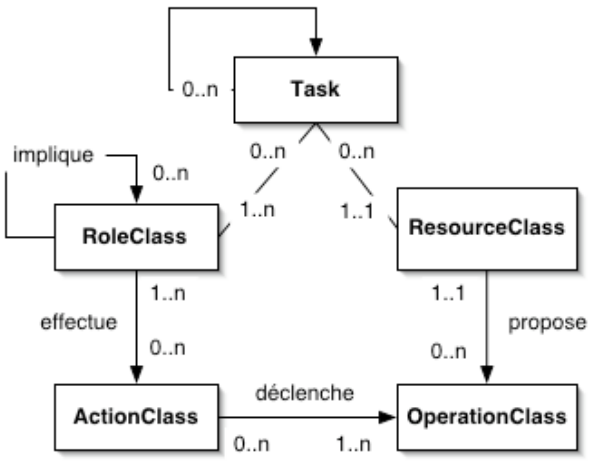

Figure 1 : Le modèle de tâche de CooLDA

La tâche est liée à un ensemble d'autres tâches (qui peut être vide), une ressource et un ensemble de rôles. Un rôle dans une tâche peut impliquer un autre rôle dans une autre tâche pour celui qui le joue. Chaque rôle pourra en fonction de l'état de l'activité effectuer des actions sur la ressource de sa tâche. Chaque action est décomposée en opérations proposées par la ressource. CooLDA étant un ERGI, les ressources qu'il intègre sont des composants java disponibles sur l'Internet. Les opérations permettent de déclencher dynamiquement des méthodes du composant tout en fournissant un niveau d'abstraction plus élevé que celui du langage Java : une opération possède un nom non contraint par la grammaire du langage sous-jacent, une description, etc. La majuscule au début de chaque terme indique qu'il s'agit d'entités qui peuvent être instanciées, ce modèle étant en réalité un méta-modèle. Ainsi, une tâche «Cours de mathématiques» sera instance de l'entité Task qui décrit le modèle de toutes les tâches. Le suffixe «Class» indique que les instances des entités de ce méta-modèle sont des modèles. Par exemple «Enseignant», instance de RoleClass décrit une classe de rôles. Une instance de «Enseignant» sera jouée par Greg dans son cours de mathématique. Ainsi, l'entité Task, pourrait aussi être dénommée Activity-Class.

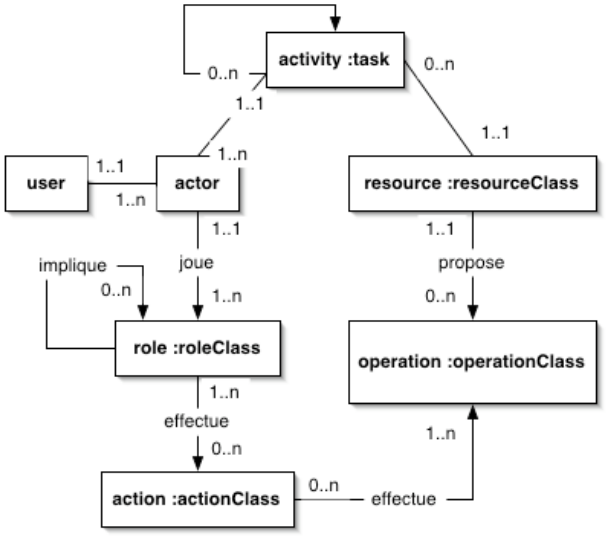

Figure 2 : Le modèle d'activité de CooLDA

La *figure 2* décrit le modèle d'activité de CooLDA. Une activité est instance d'une tâche elle-même instance de l'entité Task. Un acteur de l'activité est l'association d'un utilisateur du système et de rôles définis dans la tâche. Un utilisateur peut être impliqué dans diverses activités du système. Chaque activité peut être liée à d'autres activités, ce qui signifie certains de ses acteurs jouent un rôle qui les implique dans une autre activité. La *figure 1* et la *figure 2* forment un modèle qui définit l'ensemble des entités du noyau réflexif de CooLDA. On pourra noter que la terminologie activité/action/opération a été choisie en rapport au modèle de la dynamique entre les niveaux du même nom décrits dans la TA.

Une particularité majeure de ce modèle est que chaque tâche/activité ne possède qu'une unique ressource. Ce point ne pose pas de contradiction avec nos inspirations de la TA car cette dernière ne place pas de limite rigide entre les niveaux action et activité. Ainsi, une activité de CooLDA peut aussi être envisagée comme une action collective dans la TA. Par contre, nous levons ici le problème de décrochement de niveau qui existait dans DARE. Dans CooLDA, tout est activité. Nous abordons d'ailleurs le composant logiciel pointé par la ressource CooLDA lui-même comme une activité. En effet, un composant de mèl supporte d'une manière autonome une activité de mèl. Il définit implicitement au moyen de son interface et ses fonctionnalités le rôle de l'utilisateur dans l'activité qu'il supporte. Le rôle de CooLDA n'est autre que d'intégrer cette activité liée à une ressource particulière dans des activités plus larges. Par exemple, un acteur utilise le composant de mèl pour proposer une date d'examen à ses étudiants dans son activité «Cours de mathématiques». De la même manière, une méta-activité devient une activité à part entière qui peut elle-même intégrer des «sous» méta-activités et s'articuler avec son activité de base. Ce modèle permet donc d'envisager CooLDA comme un système qui supporte mieux le principe de co-évolution, l'unité basique étant réellement, et comme le suggère Kuutti [10], l'activité coopérative.

Enfin, il faut souligner que ce modèle ne répond pas au problème de séquencement des activités de type Workflow. Notre premier objectif est de permettre la description d'organisations et de fournir un environnement qui contextualise les ressources des utilisateurs en fonction de leurs rôles dans leurs activités.

Des mécanismes relâchés

À partir des entités décrites dans notre modèle, nous devons créer un noyau réflexif qui propose une solution aux problèmes liés à la répercussion des actions effectuées dans une méta-activité. Pour ce faire, nous avons implémenté totalement un nouveau noyau réflexif inspiré des mécanismes classiques trouvés dans les langages à objets, mais qui s'en démarquent par plusieurs aspects.

Le noyau de CooLDA a été implémenté dans le langage Java pour un souci de portabilité. Néanmoins, le langage Java, à la différence de Smalltalk n'est pas un langage totalement réflexif. Même s'il supporte les mécanismes d'introspection (accès en lecture aux éléments réifiés du langage, i.e un objet peut connaître dynamiquement sa classe, invoquer des méthodes, etc.), il ne supporte pas l'intercession (accès en modification aux éléments ré-

ifiés, i.e ajout de méthodes dans la classe d'un objet en cours d'exécution). Pourtant, notre collecticiel réflexif doit bien proposer des mécanismes aussi bien d'introspection (connaître la définition d'un rôle) que d'intercession (modifier la liste des actions disponibles). De ce fait, pour CooLDA, nous ne pouvons pas simplement utiliser les mécanismes réflexifs du langage sous-jacent pour implémenter son noyau, mais avons l'occasion d'implémenter des mécanismes sur les entités du modèle mieux adaptés à CooLDA que ceux directement disponibles dans les implémentations du MOP[1].

Dans CooLDA, chaque entité possède une représentation sous forme de classe Java ainsi que dans le langage IDL, le système étant distribué par CORBA grâce à l'ORB disponible dans le J2SE 1.4 de Sun. Il serait rébarbatif de présenter l'ensemble des classes qui représentent les entités de notre modèle. Nous préférons présenter ici les mécanismes que nous avons mis en place et qui créent les liens entre les instances et modèles de notre système.

Chaque objet représentant une entité du modèle de tâche de CooLDA possède une interface permettant d'accéder en lecture ou en modification à son contenu. Par exemple, un modèle de ressource « Tableau Blanc » peut fournir la définition de l'opération « Démarrer ». Il est aussi par exemple possible de définir dynamiquement de nouvelles actions pour un type de rôle, ou encore de redéfinir les actions existantes.

Chaque objet représentant une entité du modèle d'activité de CooLDA possède un état. Par exemple, une action sait si elle a déjà été déclenchée ou non. Il est de plus lié à l'objet qui représente son modèle. Ainsi, une activité « Cours de Greg » connaît sa tâche « Cours ». Il en va de même pour les ressources, les rôles... Par exemple, l'enseignant Greg peut accéder à sa définition, c'est-à-dire à un objet roleClass « Enseignant ».

Par l'implémentation de ces mécanismes, nous avons fourni à CooLDA les bases des dimensions réflexives qu'il requière pour supporter la co-évolution. Ces mécanismes permettent l'introspection et l'intercession au cours de l'exécution du système : un support d'activité est capable d'accéder à son modèle d'exécution et de le transformer si le besoin s'en fait sentir.

Une méthode pour réaliser la relation causale entre les modèles et leurs instances est d'obliger chaque instance à consulter son modèle avant d'agir. Avec cette liaison dynamique, l'exécution de l'instance reste en concordance avec son modèle. Néanmoins, nous avons choisi d'assouplir cette approche et de parfois relâcher la relation causale. Dans le cas général, l'instance consulte effectivement son modèle. Toutefois, certaines propriétés décrites dans le modèle peuvent être surchargées dans l'instance. Ceci permet à une activité particulière, ou à un de ses composants, de se comporter comme un prototype qui, une fois stabilisé, peut être abstrait et cristallisé dans le modèle. Un exemple simple est celui de l'arrangement des fenêtres dans le support d'activités d'un acteur. L'acteur interagit grâce à des ressources dont la disponibilité est spécifiée dans son rôle, ou plus précisément, dans la classe de son rôle. L'acteur arrange visuellement ses ressources pour parfaire son espace de travail. Cet espace de travail résulte de ses préférences personnelles, mais aussi du fait qu'il joue un rôle particulier dans l'activité. Si l'acteur en possède le droit, il peut décider, par exemple après concertation avec sa communauté, de faire remonter son point de vue sur l'activité au niveau du modèle de son rôle. Cette action aura pour effet de rendre accessible son expérience aux autres utilisateurs qui jouent le même rôle. CooLDA, en tant qu'artéfact, aura cristallisé l'expérience de l'acteur. C'est ce qui se passe par exemple dans le cockpit d'un avion qui a cristallisé l'expérience des pilotes, des ergonomes, etc. À l'inverse, si un modèle de rôle contient une organisation particulière, il est aussi possible de l'autoriser à arranger son espace à sa guise. Cet exemple expose l'effet bénéfique que peut avoir une relation causale relâchée.

Grâce à ce nouveau mécanisme, notre approche permet toujours de cristalliser l'expérience des utilisateurs, mais l'environnement local peut être stabilisé avant qu'on puisse parler d'expérience partageable au travers du modèle. L'introduction de ce type de mécanismes est un élément de réponse au problème de stabilisation évoqué précédemment. C'est aussi un exemple de réponse au problème de l'effort d'abstraction : ici, l'expérience est cristallisée dans le modèle sans que l'utilisateur ait eu à passer par une description formelle de ses modifications.

La partie la moins avancée de nos réponses concerne la répercussion des modifications d'une tâche sur ses activités disjointes. Nous proposons un mécanisme de clonage qui permet de dissocier, à la demande des acteurs, leur tâche de celle dont elle est issue. Une tâche clonée ne possède plus de lien avec la tâche originelle et de ce fait, ses instances peuvent transformer leur modèle sans influencer le reste du système. Nous envisageons de coupler ce mécanisme avec un mécanisme d'héritage, non encore implémenté dans la version actuelle de CooLDA. Le clonage d'une tâche qui hérite d'autres tâches fournira une tâche indépendante unique qui pourra évoluer de manière indépendante. La part et le choix de l'humain restent donc primordial dans notre approche.

[1] Même si ce n'est pas le choix que nous avons fait, préférant implémenter nous mêmes les mécanismes de notre noyau dans le langage Java, notons qu'il aurait aussi été possible d'adapter à nos besoin ceux du MOP d'un langage totalement réflexif tel que Smalltalk.

Ceci n'entre pas en contradiction avec nos objectifs□la co-évolution a pour but de supporter l'activité coopérative des utilisateurs, pas de tout automatiser. CooLDA n'est pas l'activité, il en fait partie.

PERSPECTIVES

Comme nous venons de le montrer, le modèle de tâche/activité de CooLDA et les mécanismes qui y sont associés apportent des éléments de réponses au problème du support de la co-évolution. Toutefois, ce modèle doit encore être développé. Même si CooLDA n'est pas envisagé d'emblée comme un Workflow, il est nécessaire que nous introduisions une gestion du produit des activités ainsi que les moyens qui permettront, à la demande des acteurs, d'en gérer la dimension temporelle. La notion de rôle doit elle aussi être développée plus avant et nous travaillons à l'heure actuelle sur la notion de jeux de rôles permettant de décrire leurs interactions intra et inter activités. En effet, il est nécessaire que nous puissions aussi donner un sens à l'action d'un rôle dans une activité par rapport aux rôles qui y sont liés dans d'autres activités. Enfin, la notion de contraintes (ex. sur la cohérence d'un jeu de rôle) en cas de modifications de modèles n'a pas encore été abordée.

Nous avons ici délibérément choisi de ne pas aborder le problème des interfaces utilisateurs. Il ne faut toutefois pas oublier qu'un noyau et un modèle qui nous semblent en bonne voie ne seraient pas utilisables et acceptables sans une réflexion à ce niveau. Les notions de présentation d'interface de niveau méta ou encore d'awareness entre méta-activités et activités de base nécessiteraient au moins un papier de recherche à elles seules. CooLDA étant la base du projet CooLDev qui prévoit d'expérimenter la co-évolution dans le domaine du développement coopératif de logiciel, nous travaillons déjà sur ces interfaces. Un composant de modélisation graphique permettant la définition et/ou la modification des modèles (tâches, rôles, etc.) est actuellement en cours de développement.

CONCLUSION

Dans une démarche radicalement pluridisciplinaire et s' inspirant des travaux de recherche sur la Théorie de l'Activité, nous avons réalisé le projet DARE. Ces travaux ont abouti à la définition du principe de co-évolution. Il nous semble logique que la co-évolution, apparue à posteriori, soit plus aboutie que la définition de DARE lui-même. Néanmoins, nous tirons dans ce papier les leçons de notre expérience avec une vision plus précise de ce que devrait être un ERGI de TCAO. Nous tentons de réaliser et de parfaire cette vision dans le projet CooLDA.

Une des leçons principales que nous avons tirées est la compréhension que toute activité coopérative est intimement liée à une méta-activité coopérative. Même si DARE se démarquait des autres réalisations en intégrant cette méta-activité au sein des activités qu'il supporte, l'erreur que nous avons commise est de ne pas avoir complètement considérée cette méta-activité comme une activité coopérative à part entière. Au moyen de la définition d'un nouveau modèle conceptuel minimal récursif, CooLDA pallie ce problème et envisage les activités des utilisateurs comme une organisation d'activités plus ou moins fortement connectées.

L'autre grande leçon que nous avons tirée est que l'approche par modèles pour supporter les propriétés réflexives et de cristallisation de l'expérience des activités humaines semble bien appropriée, mais notre culture d'informaticien a influencé notre vision des mécanismes à mettre en œuvre. Les mécanismes d'introspection, d'intercession et d'héritage paraissent forts intéressants au niveau du développement des supports d'activités, mais il s'avère bénéfique d'en envisager des variantes ou versions relâchées qui supportent mieux les besoins de CooLDA. CooLDA pouvant s'inscrire dans l'approche Model Driven Architecture ou MDA [13] fortement mise en avant depuis peu par l'OMG, il sera intéressant à l'avenir de voir en quelle mesure notre approche par modèles et méta-modèles pour supporter les activités coopératives humaines se démarque des travaux du génie logiciel.

BIBLIOGRAPHIE

1. Bedny G., Meister D. *The Russian theory of activity, Current Applications to Design and Learning*,. Lawrence Erlbaum Associates, Publishers, 1997.
2. Bourguin G. *Un support informatique à l'activité coopérative fondé sur la Théorie de l'Activité : le projet DARE*, Ph. D. Thesis, Informatique, n° 2753, Université des Sciences et Technologies de Lille, France, 210 p., 2000.
3. Bourguin G., Derycke A., Tarby J.C. Beyond the Interface: Co-evolution Inside Interactive Systems – A proposal Founded on Activity Theory, *Springer Verlag, People and Computer vol. 15 – Interaction without Frontiers, proceedings of Human Computer Interaction 2001 (HCI'2001)*, Blandford, Vanderdonckt, Gray (eds), pp. 297-310, 2001.
4. Dourish P., Using Metalevel Techniques in a Flexible Toolkit for CSCW Applications, ACM Transaction on Computer-Human Interaction, vol. 5, n°2, pp 109-155, 1998.
5. Ellis C., Wainer J., A conceptual model of Groupware, In *Proceedings of the ACM CSCW'94 conference*, ACM Press, pp 79-88, 1994.

6. Ferraris C., Brunier P., Martel C. Constructing collaborative pedagogical situations in classrooms : a scenario and role based approach", *Computer Support for Collaborative Learning 2002 (CSCL 2002)*, Boulder, Colorado, USA, 8 pages, 7-11, 2002.

7. Ferraris C., Martel C. Regulation in groupware: the example of a collaborative drawing tool for young children. *Proceedings of CRIWG'2000*, 6th international workshop on groupware, Madeira, Portugal, pp. 119-127. 2000.

8. Hummes J., Kohrs A., Merialdo B., Software Components for Co-operation: a Solution for the "Get help" Problem, *Proceeding of COOP'98 conference*, INRIA, 1998.

9. Kiczales G., Bobrow D.G., Des Rivieres J. *The Art of the Metaobject Protocol*, MIT Press, 335 p, 1991.

10. Kuutti K. The concept of activity as a basic unit of analysis for CSCW research, Proceeding of the second ECSCW'91 conference, Kluwers Academics Publishers, pp 249-264, 1991.

11. Kuutti K., Notes on systems supporting "Organisational context" – An activity theory viewpoint, COMIC European project, deliverable D1.1, pp 101-117, 1993.

12. Mansfield T., Kaplan S., Fitzpatrick G., Phelps T., Fitzpatrick M., Taylor R., Segall B., Herring C., Johnson P. and Berry A. Toward Locales : Supporting collaboration with Orbit. *In Proceedings of Group 97 Phoenix*, AZ, November 1997.

13. OMG, Model Driven Architecture, disponible à l'adresse http://www.omg.org/mda/

14. Roussev V., Prasun D., Jain V., Composable Collaboration Infrastructures Based on Programming Patterns. *In Proceedings of CSCW 2000I,* ACM Press, 2000.

15. Saikali K., David B. Vers l'usage du Workflow pour la coordination dans les collecticiels. *Dans Actes de IHM-HCI 2001*, Lille, France, septembre 2001.

16. Soulier E. La narration, technique avancée d'ingénierie de la connaissance. *Association Française des Sciences et Technologies de l'Information (ASTI) Hebso n°50*, 2001. http://www.asti.asso.fr/pages/Hebdo/h50/h50.htm.

17. Suchman, L. *Plans and Situated Actions*. Cambridge University Press, Cambridge, UK, 1987.

18. Tolone W. J., Kaplan S., Fitzpatrick G., Specifying Dynamic Support for Collaborative Work within wOrlds, *Proceedings of the 1995 ACM Conference on Organisational Computing Systems (COOCS'95)*, ACM Press, pp 55-65, 1995.