Intégration d'Outils Logiciels par le Contrôle des Echanges et Vérification de Données

P.F. Tiako, O. Perrin, J.-C. Derniame
CRIN - Bâtiment Loria BP 239
54506 Vandoeuvre-les-Nancy, France
Email: {tiako, operrin, derniame}@loria.fr

Résumé

Ajouter des outils logiciels dans un environnement nécessite de leur permettre de communiquer avec les autres outils en échangeant et/ou partageant des données (intégration par les données) et en échangeant des informations de contrôle (requêtes, notifications). Ces deux dimensions de l'intégration ne sont pas toujours offertes dans le même environnement. Cette proposition vise à faciliter une intégration à posteriori d'outils par un contrôle qui prenne en compte la sémantique des messages échangés.

Mots clés: Génie logiciel, Intégration par le Contrôle, communication, interopérabilité, échange de données.

Abstract

To add software tools in an environment needs to offer them support for communicating with other tools by data exchange and/or data sharing (data integration) and by exchanging control informations (requests, notifications). These dimensions of integration are not always offered simultaneously in an environment. This proposal aims to easier an a posteriori tools integration by the control taking into account the exchanged messages semantics.

KeyWords: Software engineering, control integration, communication, interoperability, data exchange.

1 Introduction

L'intégration d'outils dans les environnements de développement de logiciels est un problème important, tant pour les utilisateurs de ces environnements que pour les constructeurs d'outils. Les environnements actuels souffrent bien souvent du fait que chaque outil a été pensé, réalisé et implanté sans tenir compte des services offerts par les autres outils.

Notre travail s'intéresse à ces problèmes. Nous proposons de compléter l'approche traditionnelle d'intégration par le contrôle par la prise en compte de la sémantique des messages échangés. Ceci permet de vérifier les propriétés de messages dans le but de faciliter l'interopérabilité entre outils.

Cet article est organisée comme suit. Le paragraphe 2 met en évidence l'importance de l'intégration par le contrôle et le paragraphe 3 dresse un panorama des différents mécanismes d'intégration par le contrôle dans les environnements existants, et le rôle incontournable que le contrôle peut jouer dans l'intégration des données. Dans le paragraphe 4, nous proposons une approche d'intégration par le contrôle d'outils basée sur la sémantique des messages échangés. La conclusion donne l'état d'avancement de nos travaux, et les perspectives entrevues.

2 L'intégration d'outils

2.1 Les différentes dimensions de l'intégration

Plusieurs travaux [Wass90, ThNe92] ont contribué à classer les différents types d'intégration qui peuvent être analysés selon cinq points de vues distincts : la présentation, les données, le contrôle, la plateforme et le procédé.

Dans l'intégration de la présentation, le critère essentiel est donc celui de l'apparence et du comportement, i.e le "*look and feel*". L'intégration par les données requiert à la fois le partage des données entre outils et la gestion des données produites par les différents outils. L'intégration par le contrôle consiste à permettre à un ensemble d'outils développés indépendamment de pouvoir se notifier des événements et déclencher des activations de fonctions. L'intégration par la plateforme peut être vue comme l'ensemble des services systèmes permettant aux applications d'être déchargées de la gestion du fonctionnement du système d'exploitation hôte et de l'architecture du réseau. L'intégration par le procédé a pour objectif de permettre à des outils de réaliser un procédé en suivant chacun de leur côté et de façon coordonnée leurs procédés respectifs. La dimension procédé a été mise en œuvre dans le cadre de projets tels que ALF [DeGr94].

2.2 L'importance de l'intégration par le contrôle

Le contrôle peut avoir des conséquences sur les autres dimensions [ArMe92]. La dimension **présentation** se sert des activités de la dimension contrôle par exemple pour l'addition, la modification et la suppression des objets de l'interface utilisateur. Le contrôle peut augmenter les performances de la dimension **données** en facilitant le partage et les échanges d'informations entre outils. Le lien entre le contrôle et les données peut être illustré par les concepts de l'approche orientée objet, dans la mesure où un objet peut être considéré comme une entité qui encapsule données et comportements. Le comportement peut être vu comme le contrôle qui est imposé à l'objet. La dimension **plateforme** requiert de la part du mécanisme d'intégration par le contrôle l'envoi ou la réception d'événements, d'où qu'ils proviennent. Enfin, l'intégration par le **procédé** utilise le mécanisme du contrôle pour l'automatisation et la représentation du procédé. Pour rendre le procédé explicite, l'intégration par le contrôle s'avère importante, dans la mesure où pour mieux représenter le procédé, elle doit assurer que seuls certains outils doivent pouvoir réagir à des messages particuliers, dans des circonstances précises.

3 Les problèmes du contrôle

3.1 La description du problème

Une intégration par le contrôle intègre *faiblement* ou *fortement* les outils. Dans le cas de l'intégration faible, les outils sont intégrés avec une connaissance minimale des autres outils. Celle-ci est souvent remplacée par une couche supérieure basée sur un ensemble d'interfaces et de contraintes sur ces outils. Par contre, dans le cas d'intégration forte, chaque outil sait exactement comment interagir à la fois au niveau données et contrôle avec les autres outils. L'intégration forte concerne l'intégration *a priori* d'un ensemble d'outils. Elle a moins d'intérêt pour l'intégration *a posteriori* qui est caractérisée par des environnements évolutifs. Il faut alors offrir les mécanismes suivants :

- gestion de la coopération entre plusieurs outils développés indépendamment afin de parvenir à une utilisation homogène. Cela regroupe les possibilités qu'a un outil de contrôler un autre outil, et leur capacité à se communiquer des informations (données et/ou événements) ;
- gestion des relations qui existent entre les différents outils de l'environnement. En particulier, on doit concevoir qu'un outil puisse être en mesure de fournir des services à d'autres outils, et qu'il soit en droit de leur réclamer des services. Il s'agit d'une ouverture vers l'extérieur, vers les autres outils et les services de l'environnement ;

- utilisation d'un service standard de communication et de coopération pour les liaisons inter-outils.

Notre approche basée sur la sémantique des messages et illustrée par la combinaison du contrôle et des données nécessite de mettre en place deux types de mécanismes:

- La vérification des propriétés de messages : (a) messages sans service de données (par exemple, un message de notification qui informe un outil de la destruction d'un objet) ; (b) messages avec service de données transiantes (par exemple, dans un message envoyé à un outil pour lui donner une date, la donnée date est transiante en ce sens que sa mise à jour n'est conservée que dans le message) ; (c) messages de service de données persistantes (par exemple, un message de mise à jour envoyé à un outil pour mettre automatiquement à jour l'objet concerné).
- L'invocation implicite d'outils. C'est un mécanisme qui permet de déterminer quand et comment les messages doivent être échangés entre les outils.

Comme le rappelle [Reis95c] on ne peut pas parler de données sans contrôle et d'après la section 2.2, le contrôle intervient dans toutes les autres dimensions de l'intégration. Plusieurs solutions ont été proposées pour la dimension données : l'utilisation d'un convertisseur générique de données [BoPe94, Perr94], l'utilisation d'un mécanisme d'intégration de données à base de fragments [Reis95c] et l'interopérabilité orientée objet [Kons93].

La solution basée sur un convertisseur générique de données a l'avantage de permettre des échanges dynamiques de données entre outils. Notre approche du contrôle sera basée sur ces propositions et nous nous limiterons ici à l'aspect vérification des propriétés de messages d'échanges dynamique de données entre outils.

3.2 Les différents mécanismes d'intégration par le contrôle

On peut regrouper les propositions en trois familles. Il s'agit d'une part des différentes implantations issues des concepts proposées par FIELD [Reis90a], des propositions basées sur un langage de définition d'interface d'autre part [OMG92, OSF92, IBM93], et de l'approche traitant de l'interopérabilité orienté objet (OOI) [Kons93].

3.2.1 Les concepts de FIELD

Le système FIELD est à l'origine des travaux sur l'intégration par passage de messages et a servi de base à d'autres environnements comme FUSE [DEC93a], Softbench [Caga90], FOREST [Gall90] ou ToolTalk [Fran91a]. L'approche consiste à faire dialoguer les outils avec un serveur de messages qui gère et distribue les messages que les outils lui envoient. Le modèle canonique de message utilisé est basé sur l'utilisation d'une simple chaîne de caractères.

Avantages

- l'approche par passage de messages insiste plus sur l'*indépendance de l'implantation*, autrement dit, les outils sont indépendants et offrent des services sans avoir besoin de savoir ceux qui vont l'utiliser ;
- ce type d'intégration utilise des techniques d'abstraction des messages dans le serveur, ce qui permet d'analyser leur sémantique.

Inconvénients

- un inconvénient de cette indépendance peut être l'obligation de plusieurs outils à réagir au moindre message diffusé (i.e pas de sélection des messages) ;

- l'absence de module permettant d'interpréter les messages provenant du serveur ;
- l'approche ne permet pas aux outils de s'échanger les données ;
- la réécriture des outils est nécessaire la plupart du temps.

3.2.2 Les approches basées sur les IDLs

Cette catégorie regroupe les propositions basées sur l'intégration par un langage de définition d'interfaces (IDL), à savoir CORBA [OMG92], SOM/DSOM [IBM93] et DCE [OSF92]. Les approches basées sur les IDLs ont tendance à vouloir interconnecter des outils directement via une approche *point-à-point* ou *appel de procédure à distance*, cela permet à un outil de faire des appels directs aux interfaces des autres outils. Dans le cas où l'accès au code source de l'outil existe (l'implémentation de l'environnement ayant prévu cette intégration), l'outil est obligé d'utiliser ces interfaces. A défaut, une traduction et un filtrage de routines doivent être utilisés pour implémenter l'intégration, politique utilisée dans l'Object Request Broker (ORB) de CORBA. Contrairement aux concepts à la FIELD, la sélection est effectuée grâce à l'IDL décrivant les services.

Avantages

- comme précédemment, un client ne sait pas à l'avance qui va répondre à sa requête ;
- la définition d'un modèle de donnée typé à travers l'IDL permet de capturer plus de sémantique.

Inconvénients

- l'approche est limitée à la connexion *point-à-point* et aux outils utilisant ce type d'interface. Pour étendre l'utilisation d'un outil à d'autres modes de connexion, il faut introduire les configurations différentes, ce qui ne rend pas la maintenance du produit plus facile [Brow92] ;
- les approches basées sur un IDL commun, externe aux outils, sont bien adaptées à l'intégration a priori. Pour l'intégration a posteriori, il serait nécessaire de les compléter par un convertisseur de données.

3.2.3 Les approches OOI

La proposition d'interopérabilité orientée objet (OOI) est une approche qui prend en compte la sémantique des messages échangés entre outils et peut aller jusqu'à transformer les messages et les données afin qu'un autre outil puisse les manipuler. Le modèle canonique de message utilisé ici repose sur un langage de définition et de transformation d'interfaces (TMSL-*Type Matching Specification Language).*

Avantages

- les mêmes que ceux de l'approche précédente ;
- pas d'hypothèses quant à l'environnement dans lequel le prototype va être utilisé quand on connaît les transformations à effectuer sur les paramètres de messages.

Inconvénients

- pas de traduction dynamique des messages échangés entre outils, ce qui signifie que les informations et les fonctions de conversion sont définies de manière statique et chargés lors de la phase d'initialisation ;
- la résolution statique et à priori de certains conflits liées à l'identification et au renommage de

certaines entités.

3.3 Discussion

Dans ce paragraphe, les points forts de certains environnements sont récapitulés sur le tableau (Tableau 1) qui présente le mode d'intégration, la communication, le type de connexion et la prise en compte de la dimension données.

L'intégration. Les différentes propositions sont étudiées sur les niveaux syntaxique *(synt)* et sémantique *(sema)* du contrôle. Le niveau syntaxique autorise non seulement l'échange de messages, mais également l'échange de données afin que l'outil qui a reçu un message puisse éventuellement traiter ces données. Le niveau sémantique donne la possibilité à un outil de prendre le contrôle d'un autre. Ensuite, une comparaison est faite sur le mode d'adaptabilité des outils. Un outil peut être adapté soit par une enveloppe *(enve)*, soit par réécriture *(rééc)*.

	Intégration		**Communication**		**Message**		**Connection**	**Données**
	Niveau	Adaptabilité	mode	type	Nature	Format		
FIELD [Reis90a,Brow92]	synt	rééc	sync/asyn	cmd/not	chaîne	sans dest	oto	non
FOREST [Gall90]	synt	rééc	sync/asyn	cmd/not	chaîne	sans dest	oto	non
FUSE [DEC93a,ZaBr94]	sema	enve	asyn	req/not	chaîne	sans dest	otm	non
Softbench [Caga90,Perr94]	synt	enve	asyn	req/not	chaîne	sans dest	otm	non
ToolTalk [Fran91a]	synt	rééc	sync/sync	req/not	chaîne	comp	oto	non
CORBA [OMG92]	sema	rééc	asyn	req	proc	comp	ptp	non
DCE [OSF92]	sema/synt	rééc	sync	req/not	proc	avec para	RPC	non
SOM [IBM93]	sema	rééc	asyn	req/not	proc	comp	ptp	non
OOI [Kons93]	sem	rééc	sync	req/not	proc	avec para	RPC	oui

Tableau 1: Tableau comparatif des propositions d' intégration par le contrôle

La communication. Le premier critère de comparaison est le mode de communication, qui peut être synchrone *(sync)* et/ou asynchrone *(asyn)* et le second est le type de communication qui peut être une commande *(com)*, une requête *(req)* ou une notification *(not)*.

Les messages. La comparaison porte sur la nature et le format des messages. Selon les environnements, les messages peuvent se présenter sous forme de chaînes de caractères *(chaîne)* ou de procédures *(proc)* à exécuter. Les formats de messages sont très variés, nous distinguons ceux qui ont un format complexe *(comp)* avec un ensemble de règles sur la structure des messages de ceux qui sont simples *(simp)*. Certains environnements gèrent des messages sans destinataires *(sans dest)* connus à l'avance et d'autres des messages avec paramètres *(para)*.

Les connexions. On peut distinguer les connexions un-à-un *(oto)* et un-à-plusieurs *(otm)* relatives à l'approche passage de messages, les connexions point-à-point *(ptp)* de type ORB de CORBA et les connexions par appels de procédures distantes *RPC* (Remote Procedure Call) de type DCE.

Les données. Dans l'intégration d'outils, on ne peut pas dissocier l'intégration par le contrôle de l'intégration par les données. Comme l'indique le tableau (Tableau 1), très peu de ces environnements s'adressent à ces deux dimensions à la fois.

4 Notre Contribution

L'approche par passage de messages contourne certaines de ces difficultés (identification et renommage des entités, migration des paramètres de messages, contraintes sur les types d'interfa-

ces,...) par généralisation et abstraction des services d'interconnections dans le serveur de messages, ce qui rend la communication entre outils plus **explicite, visible et contrôlable**. Cette approche à la base de plusieurs propositions montre ses limites dans la prise en compte de la sémantique des messages échangées. Nous proposons d'abord une architecture d'intégration par le contrôle basée sur l'approche par passage de messages et conçue autour d'un certain nombre de composants qui permettent de prendre en compte la sémantique de messages échangés entre outils, puis, nous illustrons nos propos par la combinaison du contrôle et des données.

4.1 Composants principaux de l'approche

Dans notre approche, un environnement ouvert, permettant l'intégration d'outils à priori et à posteriori est composé :

- des outils classiques. Ils représentent les outils que l'on peut intégrer ou ôter de façon dynamique dans le système et que l'on désire faire communiquer ;
- d'un gestionnaire de données. Il utilise le convertisseur de données pour permettre aux différents outils classiques de s'échanger les données;
- d'un outil d'aide à la décision. Il intercepte et analyse les messages qui sont émis dans le système, et selon le type de message de service, invoque l'outil approprié, notamment le gestionnaire de données dans le cas des messages de services de données;
- d'un gestionnaire de messages. Il enregistre les signatures de messages et est au centre de la communication entre toutes les autres entités de l'environnement.

Ces différents composants définissent une architecture d'intégration par le contrôle de type "*message passing*" avec la particularité de pouvoir vérifier les propriétés de messages de services échangés entre outils. Cette proposition permet de combiner le contrôle et les données.

4.2 Combinaison contrôle-données et propriétés de messages

Les interactions de communication et de contrôle doivent permettre les échanges de données entre outils. Seule la sémantique des paramètres de messages permet d'évaluer le niveau de transformation à effectuer dans les échanges de données. De façon générale, et d'après les propositions de ([Kons93], [Perr94]), un message ne peut être porteur que de trois propriétés ou niveaux d'informations de transformation, discernables par la sémantique des paramètres du message. Ces trois propriétés sont : 1) texte *sans donnée* ou sans référence de donnée, 2) texte avec *donnée émise* et 3) texte avec *référence d'objet émise*. Les deux dernieres propriétés sont qualifiées de type service de données. Dans la suite de cette section, nous allons faire des propositions relatives à ces propriétés des messages, et proposer des relations de vérification de ces différentes propriétés.

4.2.1 Les différentes propriétés de messages

1. **sans donnée :** les paramètres du message ne contiennent aucune donnée ou référence de donnée. Le message est traité de façon classique sans invocation du convertisseur. Par exemple, un outil qui envoie une notification pour informer les autres outils qu'il est désactivé. Cette propriété peut s'illustrer sur la figure (Fig. 1) par le scénario : 1) l'outil *A* envoie un message au gestionnaire de messages ; 2) l'outil d'aide à la décision analyse les paramètres du message ; 3) les paramètres ne sont pas de type service de données ; 4) le message est immédiatement transmis à l'outil *B* destinataire.
2. **donnée émise :** la donnée est contenue dans les paramètres du message et en cas d'incompatibilité, l'échange n'implique qu'une mise à jour du message. Par exemple, l'outil *A* demande à l'outil *B* la date de dernière mise à jour d'un produit logiciel. Les deux outils n'ayant pas né-

cessairement le même format de représentation de la date, une conversion du type de représentation de l'un vers l'autre s'impose. C'est le gestionnaire de données qui se charge alors de cette tâche. Cette propriété peut s'illustrer sur la figure (Fig. 2) par le scénario : 1) l'outil *A* envoie un message au gestionnaire de messages ; 2) l'outil d'aide à la décision analyse les paramètres du message ; 3) les paramètres sont de type service de données avec donnée émise ; 4) transmission des références de l'outil *A* et de l'objet au convertisseur; 5) et 6) transformation de l'objet par le convertisseur ; 7) notification au gestionnaire de messages de la fin de la transformation ; 8)transmission de l'objet transformé à l'outil *B*, destinataire.

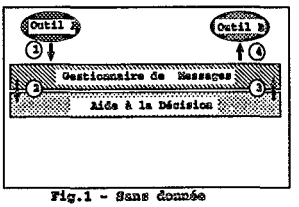

Fig.1 - Sans donnée

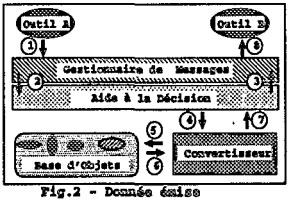

Fig.2 - Donnée émise

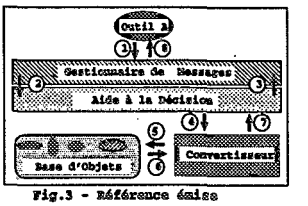

Fig.3 - Référence émise

3. **référence émise :** un outil demande à utiliser un objet produit par un autre outil. L'objet est référencé dans les paramètres du message. Comme dans la propriété donnée émise et en cas d'incompatibilité seulement, le gestionnaire de données s'occupe d'adapter l'objet au format du demandeur du service. Il y a alors mise à jour effective de l'objet. Par exemple, cette situation peut arriver lorsqu'un nouvel outil est introduit dans l'environnement pour remplacer un autre ou pour offrir de nouvelles fonctionnalités. Cette propriété peut s'illustrer sur la figure (Fig. 3) par le scénario : 1) l'outil *A* envoie un message au gestionnaire de messages ; 2) l'outil d'aide à la décision analyse les paramètres du message ; 3) les paramètres sont de type service de données avec référence d'objet émise ; 4) transmission des références des outils *A* et de l'objet au convertisseur ; 5) et 6) transformation de l'objet par le convertisseur ; 7) notification au gestionnaire de messages de la fin de la transformation ; 8)réception de la notification par l'outil *A*.

4.2.2 Les ensembles de base

Dans une telle architecture, les outils d'un environnement communiquent entre eux par envoi et réception de messages. Pour établir un bon dialogue entre ces outils, il faut définir un protocole d'interaction et de communication. C'est un ensemble de règles et de conventions que les outils doivent respecter pour communiquer ou se synchroniser les uns avec les autres. Ce paragraphe présente brièvement les ensembles de base de la modélisation des interactions de communication détaillés dans [Tiak95] et issus des composants principaux :

- Les ensembles provenant des outils classiques sont constitués des **Outils** pour représenter tous les outils possibles de l'environnement ; des **Classes** pour représenter toutes les classes d'outils possibles de l'environnement ; des **Objets** comme ensemble de tous les objets possibles (données, interfaces,...) ; des **Messages** pour représenter les messages qui peuvent être échangés entre outils.

- Les ensembles provenant du gestionnaire de messages sont constitués des **Echantillons** de messages pour représenter les signatures des messages échangés dans l'environnement, des différents **Types** de messages (requête, notification, commande,...) et des **Champs** du message (expéditeur du message, opération à exécuter, paramètres du message,...).

- Les ensembles provenant de l'outil d'aide à la décision (OAD) sont constitués des catégories d'**Usagers** de l'environnement, des **Priorités** des usagers, de l'ensemble **Test** pour représenter les conditions élémentaires qui peuvent être évaluées par l'outil d'aide à la décision et des **Ac-**

tions pour représenter les actions élémentaires qui peuvent être déclenchées si une condition est vérifiée.

4.2.3 Les relations entre ensembles de base

Les relations qui suivent permettent d'établir des liens entre les ensembles de base décrits précédemment.

- **Classe d'outils**: cette relation décrit pour chaque outil, la classe à laquelle il appartient ;
- **Fonctionnalités d'outil**: cette relation décrit pour chaque fonctionnalité, l'outil susceptible de l'offrir ou de la demander ;
- **Clause de décision**: elle décrit l'ensemble des actions à exécuter séquentiellement (sous forme de triggers) si la condition portant sur un usager, un outil et un échantillon de messages est vérifiée ;
- **Production d'objets**: cette relation décrit pour chaque objet, l'outil qui l'a produit ;
- **Type de message**: cette relation décrit le type de chaque message ;
- **Emission de messages**: cette relation décrit pour chaque message l'outil qui l'a émis ;
- **Réception de messages**: cette relation décrit pour chaque message de requête, le message susceptible d'être la réponse à ce message ;
- **Interaction entre outils**: cette relation associe à chaque couple d'outils la valeur logique *vrai* ou *faux* s'il peut avoir ou non une communication possible entre les deux outils.

4.2.4 Les relations de vérification des propriétés de service

Les relations de vérification de propriétés de service utilisent les relations décrites précédemment pour analyser la sémantique des messages et mettre en évidence la propriété d'un service.

- **Serv_de_don** : cette relation décrit pour chaque message de service de données, la donnée ou l'objet qui y est référencé ;
- **Vérif_sans_don** : cette relation associe à chaque message une valeur logique *vrai* ou *faux* selon que le message est ou non un message de service de données ;
- **Vérif_don_emi** : cette relation associe à chaque message une valeur logique *vrai* ou *faux* selon que le message est ou non un message de service de données contenant dans ses paramètres une donnée émise et non compatible avec le format du récepteur ;
- **Verif_obj_mod**: cette relation associe à chaque message une valeur logique *vrai* ou *faux* selon que le message est ou non un message de service de données contenant dans ses paramètres la référence d'un objet persistant à modifier et non compatible avec le format du récepteur.

4.2.5 Opérateurs de conversion de données

Ces deux opérateurs correspondent aux propriétés donnée émise et référence émise et permettent de gérer les données contenues dans les messages ou les objets persistants.

- **Convert_par_mod** : l'opérateur convert_par_mod permet de convertir une donnée contenue dans un paramètre de message en émission, en une autre donnée respectant le format du destinataire.
- **Convert_obj_mod** : l'opérateur convert_obj_mod permet de convertir un objet persistant et produit par un autre outil au format de l'outil demandeur de service.

Les différentes étapes de conversion de données sont basées sur l'unification des définitions des interfaces des données, la résolution des problèmes d'homonymie et de synonymie des attributs contenus dans les interfaces et ensuite la transformation des données. Ces opérateurs peuvent être enrichis par composition d'un ensemble minimal de primitives proposées dans [BoPe94].

La technique d'intégration des outils est basée sur trois concepts [Perr94] : 1) le concept de classes d'outils, qui englobe toutes les informations relatives au contexte dans lequel les outils vont évoluer ; 2) le concept de méta-représentation des données, qui englobe la sémantique liée aux objets et 3) le concept de vérification des contraintes, qui donne la possibilité de protéger les objets contre les transformations impossibles.

4.2.6 Relation de vérification des services de données

La relation **Controle_data** utilise les relations et opérateurs précédemment définies pour vérifier les échanges dynamiques de données entre outil. Cette relation associe à chaque 3-uplets (message_demande, outil, message_réponse) une valeur logique *vrai* ou *faux* selon que le message_réponse est oui ou non conforme à la demande de service de données message_demande émis par l'outil.

4.3 Les avantages du mécanisme

L'avantage fondamental de ce mécanisme est qu'il permet de combiner le contrôle et les données et de palier à certains inconvénients de propositions existantes : par rapport à l'approche FIELD, il intègre un module de vérification de la sémantique de messages ; par rapport à l'approche basée sur les IDLs, il permet d'intégrer et de maintenir de nouveaux outils ; par rapport à l'approche d'interopérabilité orientée objet, il permet l'échange dynamique de données.

Le mécanisme permet d'une part la vérification des propriétés de messages de services, ce qui facilite l'interopérabilité de données entre outils et d'autre part le respect de la synchronisation, ce qui permet de vérifier les commandes d'invocation avant d'autoriser un outil à gérer les données d'un autre outil, sans mettre en cause la cohérence et l'intégrité de celles-ci.

5 Conclusion

Dans ce travail, nous avons proposé une approche du contrôle adapté aux environnements d'intégration d'outils logiciels basé sur la vérification des messages de services échangées entre outils en interactions. La démarche s'appuie sur deux concepts fondamentaux : (a) la sémantique des messages échangées, et, (b) la généralisation et l'abstraction des services d'interconnection dans le serveur de messages.

Ces deux concepts offrent un support flexible qui permet de contrôler la communication entre outils et de réaliser des invocations implicites d'outils. Nos propositions ont été illustrées dans le cas d'échanges de données. Cette qualité pourtant nécessaire est soit absente dans les propositions d'échange dynamique de données [BoPe94,Perr94], soit ne gère pas la sémantique des messages échangés dans les propositions d'échanges statiques de données entre outils [Kons93,Reis95c].

Dans l'état actuel, les mécanismes proposés n'offrent qu'un cadre générique de vérification pour la dimension données. Il apparaît donc nécessaire d'approfondir l'étude de ces concepts, afin de les adapter aux particularités des autres dimensions de l'intégration. En particulier celui de l'interopérabilité de procédés logiciels.

Nous expérimentons certaines des idées exposées dans cet article en réalisant un prototype. Ce prototype conçu autour de l'outil d'aide à la décision, du convertisseur générique de données et d'un serveur de messages permettra de disposer d'une base de validation de nos propositions. Actuellement, le convertisseur générique est implémenté en Tcl/Tk et nous envisageons de le coupler

avec notre outil d'aide à la décision inspiré de FOREST et d'un serveur de messages Softbench.

Références

[ArMe92]J.E. Arnold and G. Memmi. Control integration and its role in software integration. In *Toulouse '92*, pages 107–116, Toulouse, December 1992. Fifth Int. Conf.: Software Engineering & its Applications, Proceedings.

[BoPe94] N. Boudjlida and O. Perrin. A formal framework and a procedural approach for data interchange. In IEEE Computer Society Press, editor, *ICSI'94*, pages 476–485, São Paulo, Brazil, August 1994.

[Brow92] A.W Brown. Control integration through message passing in a software development. Technical Report CMU/SEI-92-TR-35, Carnegie Mellon University, Dec. 1992.

[Caga90] M. Cagan. The hp softbench: An architecture for a new generation of software tools. *Hewlett–Packard Journal*, pages 36–47, Juin 1990.

[Call93] J.R. Callahan. Software packaging. *PhD thesis, CS Department-University of Maryland*, 1993.

[DEC93a]DEC. Dec fuse encase manual. *Digital Equipement Corporation*, March 1993.

[DeGr94] J.-C. Derniame and V. Gruhn. Development of process-centered ipses in the alf project. *Journal of Systems Integration*, 4:127–150, 1994.

[Fran91a]R. Frankel. Introduction to the tooltalk service. *Technical report, SunSoft, Inc.*, September 1991.

[Gall90] D. Garlan and E. Ilias. Low-cost, adaptable tool integration policies for integrated environments. *ACM Software Engineering Notes, 15(6)*, 15(6):1–10, December 1990.

[IBM93] IBM. Somobjects developer toolkit: Technical overview. Technical Report Version 2.0, Internationnal Busi ness Machines Corporation, November 1993.

[Kons93] D. Konstantas. Object oriented interoperability. In *ECOOP '93*, pages 80–102, Kaisersland, Germany, July 1993. 7th European Conference on Object Oriented Programming, LNCS 707.

[OMG92]OMG. The object model architecture guide. *Technical Report OMG Document Number 91.11.1*, Sept. 1992.

[OSF92] OSF. Open Software Foundation: DCE application development guide. *Tech. Report Cambridge, USA*, 1992.

[Perr94] O. Perrin. Un modèle d'intégration d'outils dans les environnements de développement de logiciels. *PhD Thesis, CRIN-Université Henri Poincaré Nancy 1*, 1994.

[Reis90b] S.P. Reiss. Connecting tools using message passing in the FIELD environment. *IEEE*, p. 57–66, July 1990.

[Reis90a] S.P. Reiss. Interacting with the FIELD environment. *Software-Practice and Experience*, 20(S1):S1/89–S1/115, June 1990.

[Reis95c] S.P. Reiss. Fragments: A mechanism for low cost data integration. Technical Report CS-95-13, Departement of computer Science, Brown University, May 1995.

[ThNe92]I. Thomas and B.A. Nejmeh. Definitions of tool integration for environment. *IEEE Software*, pages 29–35, March 1992.

[Tiak95] P.F Tiako. Communication et contrôle d'outils dans les environnements de développement de logiciels. *Rapport de DEA Informatique, University of Nancy I*, Septembre 1995.

[Wass90] A.I. Wasserman. Tool integration in software engineering environments. In *Software Engineering Environments*, pages 137–149, Berlin, 1990. International Workshop on Environments, LNCS *No.* 467.

[ZaBr94] P. Zarrella and A.W Brown. Remplacing the message service component in an integration framework. Technical Report CMU/SEI-94-TR-17, Carnegie Mellon University, Sept. 1994.