Environnements de logiciels basés sur la connaissance et Mécanismes d'intégration[1]

N. Boudjlida et H. Basson[2]
C.R.I.N. - Université de Nancy 1 - B.P. 239
54506 - Vandoeuvre Lès Nancy Cedex (France)

Résumé : Les mécanismes d'intégration d'outils dans un environnement de développement assisté de logiciel (CASE) visent la production d'environnements dont les outils couvrent tout le cycle de développement. La notion de procédé de développement de logiciel concerne, quant à elle, la modélisation et la réalisation des différentes activités qui contribuent à un développement coopératif de logiciel. Dans un environnement de logiciel, la coopération est le plus souvent abordée à un bas niveau en vue de fournir des mécanismes sûrs d'intégration et de contrôle du déroulement des activités. Dans cet article, en nous appuyant sur les solutions adoptées dans ALF, un projet ESPRIT, nous montrons que l'intégration d'outils dans un environnement n'est pas uniquement une affaire de mécanismes, mais qu'elle recouvre également des aspects langage et modèles de représentation. Ces solutions sont, par ailleurs, comparées au modèle de référence d'environnements CASE défini par l'ECMA.

Mots Clés : Environnement Intégré de développement de logiciel, Procédé de logiciel, Modélisation et Interprétation de Procédé, Système de Gestion d'Objets, Intégration d'outils et de Rôles, modèle de référence ECMA.

Abstract : Tool integration in Software Engineering Environments aims at producing environments whose toolset covers the whole life-cycle. Software Process addresses supports for modelling and/or executing the various activities which contribute to a cooperative software development. In a Software Environment, cooperation is most often tackled at basic levels to provide safe mechanisms to integrate coordination and control of the activities. In this paper, on the basis of the solutions we adopted in ALF, an ESPRIT Project, we show that integrating tools in an environment is not only a matter of mechanisms, but it covers linguistic features, representation models and basic mechanisms. Afterward, we contrast ALF solutions to the ECMA reference model for frameworks of CASE Environments.

Keywords : Integrated Software Project Support Environment, Software Process, Software Process Modelling and Enaction, Object Management System, Tool and Role integration, ECMA reference model.

[1] In Actes des Journées Internationales des Sciences Informatiques, Tunis-Tunisia, May 1992.

[2] e-mail address : nacer@loria.crin.fr ou basson@loria.crin.fr; Tel: (33) 83 91 21 11

Environnements de logiciels basés sur la connaissance et Mécanismes d'intégration [3]

N. Boudjlida et H. Basson

1 Introduction

Un des problèmes clés dans la construction des environnements de développement assisté de logiciels (CASE ou Computer-Aided Software Engineering) est de rendre disponibles des outils pouvant communiquer, partager des données, et coopérer de façon cohérente pour l'accomplissement d'une ou de plusieurs tâches du développement. L'ensemble des outils d'un environnement CASE doit pouvoir évoluer sans perturber le ou les développements en cours. Pour satisfaire ces besoins, la variété des mécanismes d'intégration dont disposent les structures d'accueil sur lesquelles sont construits les environnements, est un critère déterminant de la qualité de ces environnements.

La catégorie d'environnements CASE que nous considérons ici est celle des Environnements de Développement de Logiciel (EDL) basés sur la modélisation de procédés logiciels (PL) (Software Process Model) [2]. Cette modélisation, bien que potentiellement bénéfique pour la flexibilité et l'évolutivité des environnements CASE, a pour conséquence l'intégration d'un grand nombre d'activités hétérogènes. En effet, dans un projet donné, les activités de gestion, de dévelopement, de l'assurance qualité, et de gestion de versions et de configurations sont vues comme un ensemble des procédés logiciels, i.e. "un ensemble d'activités dépendantes qui contribuent à la production du système logiciel en tant que processus cohérent sujet au raisonnement" [3]. Une fois de plus, l'aspect intégration apparaît comme un critère vital pour la construction de ce type d'environnements.

Sur un autre plan, la relation entre les EDL et les PLs peut être considérée sous deux aspects. Le premier concerne le bien-fondé de les lier étroitement, et le second porte sur la problématique technique de leur intégration. Durant la dernière décennie, nombre de projets (ALF [4], ASPIS [5], DAIDA [6],MARVEL [7], ARCADIA [8], IPSE 2.5 [9], KBSA [10]) ont clairement adopté une réponse positive au premier aspect. Ils proposent des moyens pour décrire des procédés logiciels et offrent des mécanismes pour exploiter cette description. L'exploitation de la description (nous parlerons d'activation ou d'interprétation d'un modèle de procédé), vise à contrôler et à offrir de l'assistance au cours du développement du produit logiciel. Elle concerne aussi la compréhension, l'amélioration, l'analyse et le monitoring des procédés logiciels. Dans ce contexte, la disponibilité d'un environnement

[3] Ce travail est partiellement financé par la Commission de la Communauté Européenne dans le cadre du programme ESPRIT (Projet N∘ 1520).

support pour un modèle de procédé logiciel constitue un grand pas vers l'exécution, la gestion et l'amélioration de ces procédés.

L'objectif de cet article est double. Nous présentons, dans un premier temps, les solutions qui ont été adoptées dans ALF [4], un projet ESPRIT ayant pour objectif la réalisation d'un environnement intégré basé sur la modélisation explicite de procédés logiciels pour supporter le développement de projets. Dans un second temps, ces solutions sont situées par rapport au modèle de référence d'environnements CASE défini par l'ECMA [11].

Dans la section 2, nous introduisons plus précisément la notion d'intégration. Dans la section 3, nous exposons brièvement quelques mécanismes d'intégration disponibles sur PCTE [12], la plateforme de développement du système ALF. Ensuite, nous présentons les concepts et les mécanismes d'intégration comme ils ont été abordés dans le système ALF, depuis le niveau externe (i.e. le niveau de la modélisation de procédés logiciels) jusqu'au niveau interne (i.e. le niveau architecture du système) (section 4). La section 5 situe les solutions proposées dans ALF par rapport au modèle de référence ECMA. Enfin, la dernière section (section 6) est consacrée à des points de discussion et à des remarques finales.

2 L'intégration d'outils dans les environnements de logiciels

L'intégration d'outils dans un environnement de développement de logiciel vise la production d'environnements dont les outils couvrent tout le cycle de développement. Ces outils s'appliquent à divers aspects du développement, mais, ils ont également besoin de partager des données, d'opérer sur un même ensemble de données, de se communiquer des données, des événements, etc.

Wasserman [13] considère le problème selon cinq dimensions :

- La plateforme :
  c'est un ensemble de services système permettant de décharger les applications des services du système d'exploitation, de l'architecture du réseau, si réseau, etc. Dans le cadre du projet ALF, PCTE (voir la section 3) constitue la plateforme de développement.

- La présentation :
  du point de vue de l'utilisateur, les outils doivent avoir la même apparence (même *"look and feel"*). Dans cette dimension, l'exemple du Mac Intosh est peut-être le plus significatif.

- Les données :
  cette dimension concerne le partage de données inter-outils et la gestion des relations entre les objets produits par des outils différents. Ce volet de l'intégration peut être réalisé par l'intermédiaire de fichiers ou de bases de données communes, selon un format commun de représentation des données, ... A titre d'exemple, les environnements de développement de bases de données supportant des langages de quatrième génération illustrent assez bien cette dimension, dans la mesure où ils se basent sur une représentation unique des données fournies par le dictionnaire de la base.
  Cependant, des transformations de structures peuvent parfois être nécessaires. Actuellement, des formes de représentation candidates à une standardisation semblent émerger (CDIF [14], IRDS [15, 16]) pour faciliter l'intégration par les données.

- Le contrôle :
  il s'agit de permettre aux outils de se communiquer des événements et, éventuellement, d'y réagir. Cette dimension de l'intégration peut être concrétisée par des mécanismes de notification directe (l'outil contient un segment de code de notification), par des triggers, ou par un service de messages ...

- Le procédé de développement :
  c'est la possibilité de définir un modèle de développement et de contrôler son exécution. Cette dimension de l'intégration est la "branche la moins mûre" à l'heure actuelle. Entrent dans cette dimension, dans le domaine des systèmes d'informations, des travaux comme ceux entrepris dans Merise [17] ou REMORA [18], appliqués à la "dynamique" du développement d'un système d'information. Entreraient également dans cette dimension, les environnements L4G qui offriraient un pilotage automatique ou semi-automatique des activités qui se déroulent en leur sein. En effet, à l'heure actuelle, nous disposons d'environnements dotés d'outils dont les règles et les conditions d'enchaînement ne sont ni explicites ni contrôlées. Le but de l'intégration de procédés de développement est, notamment, de pallier cette faiblesse.

3 La Gestion des Objets et les Mécanismes d'Intégration dans PCTE

Un environnement intégré de développement doit offrir différents types de supports, dont des supports pour les interfaces utilisateurs, la gestion des objets, la distribution, la communication et l'assistance. En outre, il doit être doté d'outils et de mécanismes adéquats pour se comporter comme un interprète d'un modèle de procédé de fabrication de logiciel. Le projet PCTE [19] (a basis for a Portable Common Tool Environment), visait la définition d'un ensemble d'interfaces publiques pour décharger l'utilisateur de préoccupations de "bas niveau", telles la gestion des objets de l'environnement ou les problèmes de répartition. Ces définitions ont donné lieu à diverses implémentations, dont le système EMERAUDE [20].

PCTE offre trois catégories de mécanismes et de primitives :

1. *ceux dits de base*, qui incluent des mécanismes d'exécution, de contrôle et de communication (via des tubes, des signaux et des transmissions de messages).

2. *ceux dédiés aux interfaces utilisateurs* qui permettent le développement d'interfaces intégrées en utilisant deux types d'agents, communicants par files de messages : *l'agent utilisateur* gère les fenêtres, les menus, les commandes, ..., et *l'agent application* lie l'agent utilisateur et les applications.

3. *et enfin les mécanismes et les primitives relatifs à la distribution* des objets et des environnements.

Par ailleurs, le *système de gestion d'objets* de PCTE permet de représenter et de manipuler de façon uniforme les objets de l'environnement. Il asssure également leur intégrité et contrôle les accès.

Le *mécanisme d'intégration* d'outils sous PCTE est fondé sur la notion de schéma de travail (Working Schema ou WS), un WS étant un schéma de type Entité-Relation représentant la vue qu'a un outil sur la base d'objets. De ce fait, l'intégration de tout outil dans un environnement basé sur PCTE débutera par le définition de son schéma de travail. Par suite, si l'outil utilise des primitives de bas niveau du système hôte, il faudra lui associer une enveloppe pour l'adapter à la représentation de la base d'objets. Ainsi, le *changement d'un outil* intégré en un autre peut être transparent si les deux outils ont le même WS et les mêmes effets sur la base. Si tel n'est pas le cas, le changement s'opère selon la même stratégie que celle utilisée lors de l'ajout d'un nouvel outil. La *composition d'outils*, quant à elle, se fait soit par intégration en un WS unique des WSs des outils à composer, soit en définissant un WS propre à l'outil composé. Le contrôle de flot entre les fragments d'outils pourra être exprimé comme un shell-script.

Ainsi, PCTE offre la plupart des types de services (gestion d'objets, distribution, interface utilisateur) attendus d'un environnement intégré. Il pêche, néanmoins, par l'absence de support pour l'assistance et le contrôle de procédé. Le projet ALF [4] a pour but de pallier ces insuffisances. Ses objectifs, son architecture et ses principaux mécanismes font l'objet du paragraphe qui suit.

4 Le projet ALF

Le projet ALF vise à produire une structure d'accueil d'environnements basée sur la connaissance. Plus précisément, il a pour objectif d'étendre PCTE avec des capacités de modélisation et d'interprétation de procédés, avec un large spectre de capacités d'assistance. Dans ce contexte, un système ALF [21] peut être vu comme un interprète "intelligent" d'un Modèle de Procédé de Fabrication de Logiciel.

4.1 Le modèle de description de procédé et son langage de définition

Le modèle de description de procédés, appelé méta-modèle de procédés assistés (Meta-Model for Assisted Software Process) et le langage de définition associé permettent de spécifier la variété de connaissances relatives à la structure des objets, leurs relations, leur dynamique et leurs contraintes. Une description utilisant ce méta-modèle est appelée modèle de procédé assisté de fabrication de logiciel (Model for Assisted Software Process ou MASP). Cinq concepts servent à définir le méta-modèle. Ils sont rapidement passés en revue ci-dessous, une présentation détaillée se trouvant dans [4, 22]. En outre, nous mettrons en évidence les principales caractéristiques du langage de définition associé.

1. **Le modèle d'objets** permet de construire une description abstraite des structures des objets, de leurs propriétés et de leur relations. Il se base sur un modèle de données de type Entité-Relation étendu par la relation de sous-type, les attributs multivalués et la possibilité de décrire des objets composites.

   Le langage de définition de MASPs permet de décrire ces structures. Il permet aussi la ré-utilisation de descriptions existantes grâce à des mécanismes d'importation et d'extension de types de données.

   Exemples

```
- Definition de types d'attributs
        flag  : BOOLEAN
        teste : BOOLEAN := FALSE
- Definition de types d'objets sans attributs ni liens
        c_program_dir : SUBTYPE OF sys-dir
        c_program : SUBTYPE OF program
- Extension d'un type d'objet
        EXTEND c_program WITH
          ATTRIBUTE  teste,
                     NO_de_version ;
```

```
  LINK       source : COMPOSITION LINK TO c_program_source ;
             object : COMPOSITION LINK TO object_module ;
END c_program
```

2. **Le modèle d'opérateurs** est une abstraction des activités liées au développement de logiciel. Les opérateurs sont typés dans le sens où un type (comme édition_de_textes) définit une classe d'opérateurs. Un opérateur (comme Emacs ou VI) est alors un membre d'une classe. Ce mécanisme de typage devrait faciliter l'intégration de nouveaux outils au sein de l'environnement. Par ailleurs, le fait qu'un opérateur nécessite ou non une inter-action avec l'utilisateur, est une information qui peut être associée au type de l'opérateur. Elle sert essentiellement à faire jouer un rôle "actif" à l'environnement, dans la mesure où celui-ci pourra, dans certaines situations, prendre l'initiative d'activer un opérateur non-interactif sans stimulus de l'utilisateur.

   Ainsi, la spécification d'un type d'opérateur comporte une signature, une pré et une post-condition, et une nature (interactif ou non) optionnelle. Les pré et les post-conditions sont formulées comme des expressions logiques typées [23] construites à l'aide de prédicats de comparaison, de quantificateurs, de connecteurs logiques, de symboles de prédicats correspondants à une vue prédicative du modèle d'objets, etc. Enfin, le langage de spécification offre des clauses d'importation qui permettent de ré-utiliser des spécifications de types d'opérateurs.

   Exemple

   ```
   audit: (IN _p: project; OUT _r: report)
         PRECOND : audit_autorise(_p) = TRUE
         POSTCOND: audit(_p)=TRUE AND proj_to_rep(_p,_r,NO_KEY)
         KIND    : INTERACTIVE
   IMPORT audit AS imported_audit
   ```

3. **Les caractéristiques** d'un MASP sont un ensemble de contraintes qui qualifient des états de cohérence durant le processus de développement. Elles peuvent également aider le système à mettre en oeuvre des mécanismes de raisonnement lorsque l'une d'elles est violée. Syntaxiquement, une caractéristique se présente comme une expression pouvant comporter une partie événement qui précise la situation dans laquelle l'expression doit être évaluée.

   Exemple

   ```
   EXPRESSION_MODEL_IS
      compile : ON UPDATE OBJECT body_compiled OF body_unit EVALUATE
   ```

```
                body_unit(_b) AND body_compiled(_b,TRUE)
END_EXPRESSION_MODEL
```

4. **Le modèle de règles** permet d'exprimer des disciplines à suivre et les réactions attendues du système face à certaines situations. Ces règles sont à la base des initiatives que peut prendre le système. Chaque règle associe une expression (partie condition) à un type d'opérateur (action).

   Exemples

   ```
   IF number_errors(_mod) > 0 THEN edit(_mod)

   IF ON INVOKE OPERATOR report_ph(_module, _Pb)
   THEN send_warning_to_project_manager(_module,_Pb)

   IF ON EXIT OPERATOR compiler(_x)
         EVALUATE ok_compilation(_x) EVALUATED TRUE
   THEN link(_x,/lib/maths.l)
   ```

5. **Le modèle d'ordre** permet de spécifier des contraintes de type contrôle de flots sur les opérateurs. Le langage de spécification permet d'exprimer, entre autres, la séquence (e.g. edit(x) ; compile(x)), la simultanéité (e.g. {print(x)}), la concurrence (e.g. *read(x)* || *print(x)*), le groupement (e.g. *(lex(x); parse(x))*, l'alternative (e.g. *print(x)* | *pretty_print(x)*) et l'itération (e.g. [*test(x)*] *(1 TO 5)*).

Ainsi, les concepts du méta-modèle d'ALF couvrent des aspects structurels (modèle d'objets pour décrire les différents types d'objets de l'environnement), des aspects dynamiques (modèle d'opérateurs spécifiant les outils à utiliser pour réaliser les activités), et des aspects de contrôle (pré et post-conditions, modèle d'ordre et caractéristiques). En outre, le modèle de règles permet d'exprimer la connaissance nécessaire aux fonctions d'assistance. Tous ces concepts coopèrent à un niveau dynamique [24] pour faire jouer un rôle actif à un système ALF durant le développement effectif.

4.2 L'intégration des rôles et des outils au niveau conceptuel

L'intégration des diverses facettes et activités du cycle de vie d'un logiciel est faite, d'abord, lors de la définition du procédé. En effet, les composants d'un MASP peuvent comporter des composants spécifiques à différents rôles ou à différentes phases du cycle de vie (comme, par exemple, des types d'objets ou d'opérateurs propres à un gestionnaire de projets). Ces divers composants peuvent être intégrés au sein d'un MASP unique ou au sein d'un MASP structuré. Le mécanisme de *structuration de MASPs* permet de spécifier un type d'opérateur comme un MASP à part entière. Ainsi, on peut imaginer un "MASP racine" ne comportant que les seules spécifications de "types d'opérateurs racines" pour les différents rôles et phases, chacun de ces types pouvant être ensuite raffiné en un MASP simple ou structuré.

Par ailleurs, vu que les activités et les rôles peuvent utiliser des objets et des outils communs, des mécanismes *d'importation et d'extension* de définitions de types ont été introduits tant au niveau du méta-modèle qu'au niveau de son langage, permettant ainsi l'expression du partage d'information et de vues différentes sur le même "univers".

La *composition d'outils* est également possible grâce à la mise en oeuvre conjointe des mécanismes de structuration et d'importation : la spécification de l'outil composé peut être faite comme un MASP contenant la spécification des outils composants. Le *changement* d'outils existants ou l'*intégration* de nouveaux outils au sein d'un environnement basé sur ALF est aussi initialement abordée au niveau du MASP. Ce type d'opération sur l'environnement peut n'avoir aucune incidence sur les MASPs existants. Si le nouvel outil est d'un type conforme au type d'un opérateur existant, l'intégration est opérée lors de la phase d'instanciation (voir le paragraphe 4.3). Mais, si l'on ne dispose pas d'une spécification appropriée au nouvel outil, l'approche envisagée consiste à étendre un MASP avec la spécification d'un nouveau type d'opérateur. Evidemment, cette opération ne doit pas aller à l'encontre de la consistence du MASP sur lequel elle est opérée. Elle peut, d'autre part, induire la modification de certains composants du MASP, comme son modèle d'objets, au cas où celui-ci ne comporterait pas des types d'objets nécessaires au nouvel outil, ou son modèle de règles pour exprimer le nouveau contrôle de flot.

4.3 L'intégration au niveau instance

Comme nous venons de le voir, un MASP intègre différents types et sources de connaissance dans une description statique. Avant d'en faire une interprétation, c'est-à-dire de mettre effectivement en oeuvre un procédé guidé par son modèle, nous devons passer par une phase d'instanciation où des objets sont créés, et des outils sont liés à leurs types. Cette phase donne naissance à un modèle instancié (Instantiated MASP ou IMASP). Elle concerne essentiellement l'outillage effectif de l'environnement d'interprétation d'un MASP par l'établissement de liens entre des types d'opérateurs et des outils du système hôte.

En premier chef, une base PCTE ad-hoc est générée en compilant la description du MASP. L'instanciation effective est alors réalisée au travers d'un dialogue contrôlé par l'utilisateur. Les informations relatives à cette phase font aussi partie de la base d'objets. Il est à noter que le système ALF autorise une instanciation partielle grâce au mécanisme d'instanciation par nécessité.

En outre, lors de l'intégration d'outils au sein de l'environnement, un mécanisme d'enveloppage peut être associé à celui d'instanciation. Il permet d'encapsuler un outil dans une enveloppe qui comprend :

- dans une *préface*, ce qui doit être réalisé avant d'activer l'outil,
- dans un *corps*, la correspondance entre la spécification formelle du type d'opérateur et la structure du message d'appel de l'outil. Le corps de l'enveloppe peut aussi contenir des informations supplémentaires, comme l'indication des options de l'outil.
- dans une *post-face*, la spécification de ce qui doit être entrepris après l'activation de l'outil, comme son impact sur la base d'objets.

Ainsi, outre les capacités d'outillage d'un environnemnt et d'intégration d'outils, ces mécanismes améliorent les mécanismes d'intégration de PCTE et donnent à un système ALF un niveau "confortable" de flexibilité et d'extensibilité.

4.4 L'intégration au niveau de l'architecture du système

L'architecture du système ALF (Figures 1 et 2) est définie par quatre couches fonctionnelles dont nous donnons une brève présentation (voir [21] pour plus de détails).

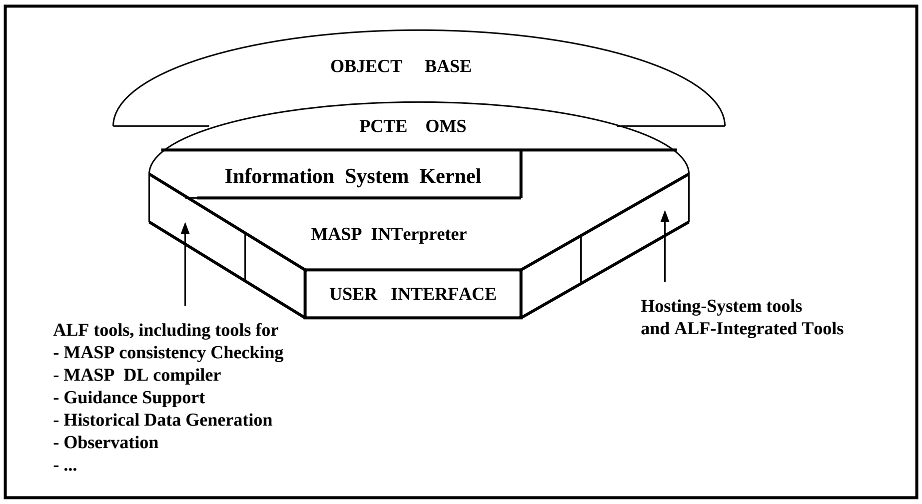

Figure 1 : ALF : Architecture du Système

4.4.1 Sous-système d'informations : intégration des données

Le Sous-Système d'Informations (SSI) a pour charge de gérer les informations relatives aussi bien aux modèles de procédé, qu'aux projets dont le développement doit se conformer à un modèle. Ce sous-système est implanté sur PCTE et il utilise les services du système

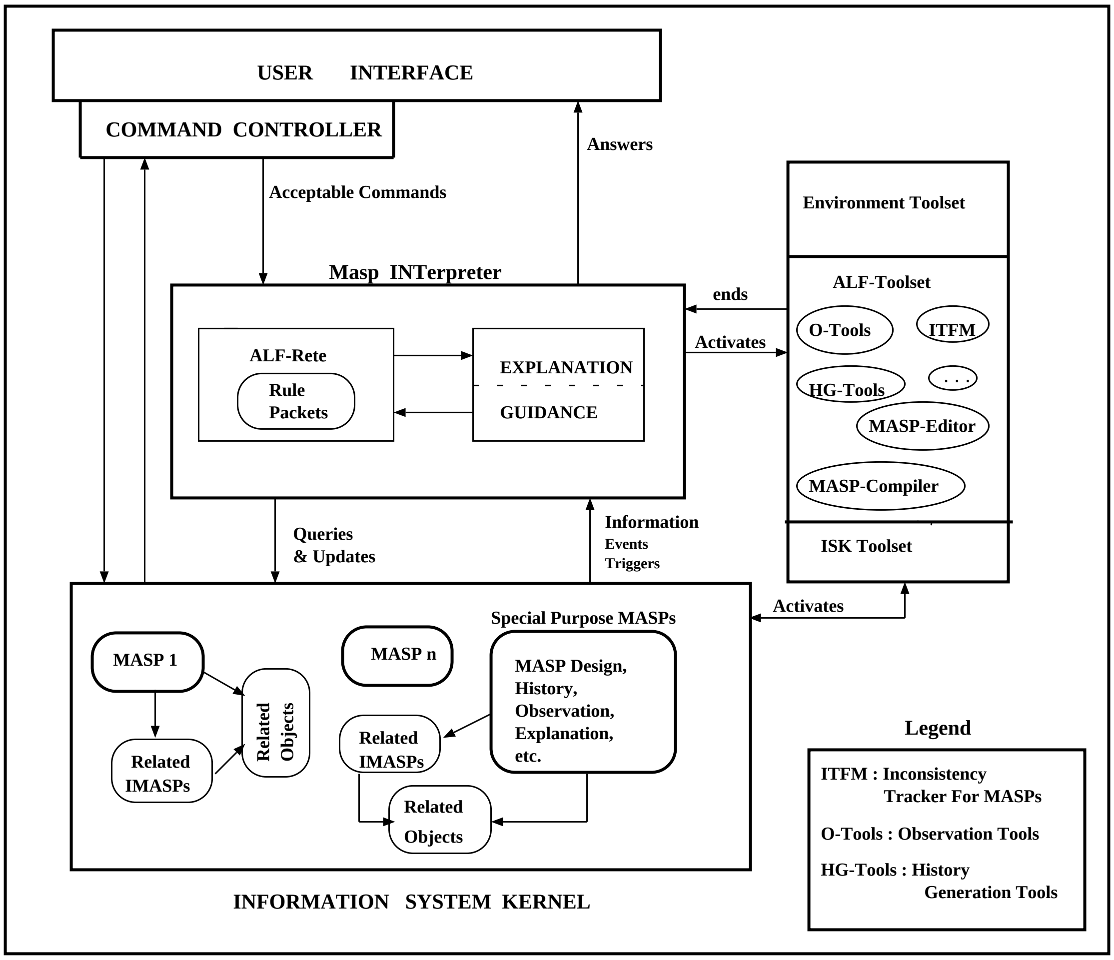

Figure 2 : ALF : Structure du Système

de gestion d'objets de celui-ci. Il est à la base de l'intégration des modules du système ALF et de l'interopérabilité de ses outils.

Pour cela, le SSI fournit un schéma PCTE qui modélise le méta-modèle de procédé. Cette représentation constitue la structure de référence pour les modules du système ALF. Des agents de présentation auront pour charge, partant de la structure commune, de la transformer en une représentation spécifique à un module ALF, quand nécessaire. Par exemple, un agent présentera de l'information extraite du système d'informations en une représentation à la PROLOG pour les besoins du module chargé de vérifier la consistence d'un MASP, un autre agent construira une représentation appropriée pour le moteur d'inférence, etc.

D'autre part, le SSI fournit un ensemble de primitives de manipulation de données à l'intention des composants d'un système ALF. L'instanciation initiale de la base d'information, résultant de la compilation de la description syntaxique d'un MASP, évoluera alors au fur et à mesure des diverses phases d'instanciation (Instanciation de MASPs, création d'objets par les utilisateurs ou par les outils, etc).

4.4.2 Sous-système de pilotage : intégration des supports d'exécution, de contrôle, de coordination et d'assistance

La couche Système de Pilotage (SP) est chargée de coordonner l'activité des composants du système ALF. L'interprète de MASPs (MINT), comme son nom l'indique, interprète un modèle de procédé. Il réagit à une initiative de l'utilisateur (une commande) en contrôlant, tout d'abord, si elle est autorisée dans l'état courant du développement. Cette fonction de contrôle s'appuie, évidemment, sur des informations du MASP présentes dans la base d'informations (spécifications des types d'opérateurs, identification des outils instanciant des types d'opérateurs, ordres). L'exécution effective de la commande n'est faite que si la phase de contrôle est passée avec succés.

Un flot de déclencheurs et d'événements peut être transmis au MINT par le SSI pour l'informer de toute modification apportée à la base d'objets. Ce flot de communication peut provoquer le déclenchement de certaines règles du modèle de procédé, le MINT étant chargé du déclenchement.

Par ailleurs, si le MINT constate une violation de caractéristiques, il poura mettre en oeuvre des mécanismes de génération et d'exécution de plan d'opérateurs en exploitant les spécifications des pré et des postconditions. Si aucun plan rétablissant la caractéristique violée ne peut être généré, le MINT défait les actions réalisées par la commande de l'utilisateur.

Ces diverses fonctions sont implantées par le moteur d'inférence ALFRete et les mécanismes de déclencheurs, de communication et de contrôle de la concurrence de PCTE. ALFRete est une extension du système à base de règles XRete [25] incluant des fonctions de communication avec PCTE, de raisonnement en arrière, de structures composites et d'activation dynamique de paquets de règles, ces derniers étant construits à partir de la connaissance décrite dans le MASP (paquet planning, paquet ordre, paquet caractéristiques, paquet règles, ...).

4.4.3 L'interface Utilisateurs

La couche interface utilisateurs est constituée d'interfaces dédiées aux différents types d'utilisateurs (conception et test de consistence de MASPs, gestion de projets, développement, ...). Elle comprend une couche Présentation, une couche Contrôle du dialogue et une couche Application. La couche présentation comporte un gestionnaire de la présentation externe, un gestionnaire de fenêtres et le système X-window. La couche de contrôle du dialogue "comprend" et gère le dialogue utilisateur-application. Les dialogues sont définis à l'aide d'objets de dialogue tels que des boutons, des fenêtres et des menus. Enfin, la couche application est un système guidé par des événements et inter-agissant avec le contrôleur de dialogues.

4.4.4 Le sous-système d'exécution (ou opérant)

Le sous-système opérant gère et contrôle des processus, appelés Processus Logiciels Assistés (Assisted Software Processes ou ASP) qui sont créés et lancés par le MINT "sous" un IMASP, c'est-à-dire que l'exécution d'un ASP est guidée par la connaissance fournie par son IMASP, celui-ci constituant une partie du contexte d'exécution du processus. Par ailleurs, plusieurs ASPs peuvent être issus d'un même IMASP, comme ils peuvent agir concurremment sur certains objets (par exemple, un ASP de développement et un ASP de gestion de ce développement peuvent agir sur des objets communs). La gestion des ASPs est réalisée grâce aux mécanismes de PCTE pour la gestion de la concurrence et du partage de données.

5 ALF vs ECMA

De manière similaire au dimensionnement de l'intégration présenté ci-avant (Section 2), le modèle de référence de l'ECMA (European Computer Manufacturers Association) pour les plateformes d'environnements CASE [11] définit une liste des services requis. Ce modèle constitue un cadre conceptuel et fonctionnel dans lequel on peut décrire et comparer les structures d'accueil d'environnements. Il aborde l'intégration selon huit dimensions : la base de données, l'intégration des données, les outils, la gestion de tâches (correspondant à l'intégration de procédés logiciels), les messages, l'interface utilisateur, la sécurité, l'administration et la configuration. Chaque dimension est à son tour examinée selon différentes perspectives : visions Conceptuelle/Externe/Interne de la dimension, Règle/Opération/Données requises, Types/Instances/Métadonnées nécessaires, et Relations inter-services/Justification/Compréhension des services. En outre, un ensemble de services est identifié pour chaque dimension.

Comparer le système ALF au modèle de référence de l'ECMA dans sa totalité n'est pas dans nos intentions, dans cet article. Dans ce qui suit, nous ne considérons que la dimension gestion des tâches. Elle concerne la définition, l'exécution et l'historique des tâches. Elle traite également de la notion de transaction et de la gestion des rôles et des événements.

- *La gestion des rôles* : Le modèle de référence reconnaît le besoin de disposer d'un ensemble standard de types de rôles, mais il n'en donne pas une définition précise.

Il en est de même dans ALF. Il existe divers moyens pour définir et instancier des rôles [26], mais il n'existe pas d'ensemble prédéfini de types de rôles.

- *La définition des tâches* : est-elle prescriptive ou descriptive ? Comment affecter des tâches à des rôles ? Comment prendre en compte le contexte d'exécution des tâches, leur granularité variable, leurs relations ? Dans ALF, la spécification des tâches est de nature descriptive, c'est-à-dire que le langage de description de modèles de procédé est de type déclaratif. Les mécanismes de structuration de MASPs permettent d'exprimer les relations inter-tâches, comme ils permettent de prendre en compte leurs diverses granularités. En effet, nous avons montré qu'une tâche pouvait être spécifiée comme un type d'opérateur (cas des tâches de fine granularité), comme elle pouvait être spécifiée comme un MASP à part entière (cas des tâches de grosse granularité).

  En outre, les tâches sont réalisées dans le cadre d'un rôle, dans un contexte. Ceci impose que l'on puisse affecter des tâches à des rôles et des ressources à des tâches. Comme nous l'avons déjà dit, dans ALF, il n'y a pas de rôle générique. Néanmoins, la définition d'un type de rôle peut se faire par le biais d'un MASP contenant la spécification des tâches associées au rôle. Par suite, la relation entre une tâche et son contexte d'exécution peut être établie lors de la phase d'instanciation par le biais d'un IMASP.

- *L'exécution des tâches* : D'un point de vue interne, il s'agit de contrôler et de coordonner l'exécution des tâches, les moyens techniques pouvant aller d'un simple shell à un langage sophistiqué de programmation de procédé. Dans notre cas, la nature descriptive du langage de description a conduit au développement d'un "moteur" d'exécution à base de règles utilisant les mécanismes de base de PCTE.

  D'un point de vue externe, la dimension exécution de tâches concerne la capacité qu'a un environnement d'informer sur l'état des diverses tâches. Dans ce cadre, ALF offre une large palette de fonctions d'assistance, dont *WHAT CAN BE DONE, WHAT TO DO, EXPLAIN* [2, 24].

- *Les transactions* : ALF gère les transactions courtes en utilisant les mécanismes de base de PCTE, les transactions longues n'étant, pour le moment, pas prises en compte.

- *La gestion des événements*, c'est-à-dire la possibilité de définir des événements ou des types d'événements et de leur associer des actions pour permettre la communication inter-outils. Du point de vue externe, il existe divers types d'événements utilisables lors de la spécification de MASPs, essentiellement dans les modèles de règles et de caractéristiques. Du point de vue interne, PCTE a été étendu par un mécanisme de déclencheurs (triggers) utilisé par le MINT.

- *L'historique* : ALF propose des MASP prédéfinis pour la gestion de l'historique des objets et pour l'observation des activités. Ces modèles peuvent être adaptés à des besoins spécifiques en utilisant les mécanismes de définition de MASPs (importation de types, extension ...).

6 Conclusion

Dans cet article, nous avons essayé de mettre en évidence le fait que la notion d'intégration dans un environnement de logiciels nécessite, à la fois, des mécanismes de base et des modèles de représentation et de description, pour uniformément offrir différents types de supports. Dans ce contexte, MARVEL [7] peut être considéré comme un système qui intègre les supports d'assistance et de contrôle de procédé de fabrication de logiciel dans la phase de développement. ARCADIA [8] et KBSA [10] sont des projets très ambitieux dont l'achèvement devrait aboutir à des noyaux d'environnements incorporant la majeure partie des types d'intégration attendus d'un environnement. Enfin, la famille de projets PCTE (PCTE, PCTE+, PACT ...) a le mérite d'offrir les mécanismes de base pour le développement de logiciels et d'environnements, même si l'on peut déplorer l'absence de support pour les fonctions d'assistance.

Le projet ALF a pour principal objectif de pallier ce manque. Il propose, à travers son méta-modèle de description de procédé de fabrication de logiciels, une approche multi-paradigme pour la modélisation et la spécification de procédés. La notion d'intégration y est considérée tout d'abord, de façon conceptuelle, à un niveau descriptif où des mécanismes et des constructions syntaxiques sont fournies pour exprimer, de manière "plate" ou structurée, les différents types de connaissance. Cette connaissance peut concerner des phases particulières du cycle de vie, comme des rôles spécifiques.

Par suite, une structure logique de référence intègre les éléments constituants la description d'un modèle de procédé. ALF propose d'autres mécanismes, comme l'instanciation de MASPs et la définition d'enveloppes pour des outils, pour "instrumenter" et instancier un environnement. La plupart de ces mécanismes servent également à faire évoluer un environnement existant.

Enfin, au niveau le plus bas, le système ALF est doté de mécanismes de base permettant l'interprétation, la coordination et le contrôle d'un modèle de procédé de fabrication de logiciel.

Références

[1] CASE88. Proceedings of the Second International Workshop on Computer-Aided Software Engineering, Cambridge, MA, July 1988.

[2] N. Boudjlida, O. Gervaise, C. Godart, J.C. Derniame, and K. Benali. Vers des ateliers de logiciel supportant des méthodes. "toward software environments that support methods". In *Actes des journées internationales des sciences informatiques*, Tunis - Tunisia, April 1988.

[3] J.C. Wileden and M. Dowson, editors. *Proceedings of the International Workshop on the Software Process in Software Environments*, Trabuco Canyon, CA, March 1985. Also as Software Engineering Notes, 11,4, August 1986.

[4] K. Benali, N. Boudjlida, F. Charoy, J-C. Derniame, C.Godart, Ph.Giffiths, V.Gruhn, Ph.Jamart, A. Legait, D.E. Oldfield, and F. Oquendo. Presentation of the **ALF** project. In *Proceedings of the International Conference on System Environment Development and Factories*, Berlin, Germany, may 1989.

[5] R. Burlon and M. Tagliati. ASPIS : Knowledge-based Assistance for System Development. *TSI*, 9(2):175–182, 1990.

[6] M. Jarke. DAIDA : Conceptual Modeling and Knowledge-based Support of Information Systems Development Processes. *TSI*, 9(2):121–133, 1990.

[7] G.E. Kaiser, P.H. Feller, and S.S. Popovitch. Intelligent Assistance for Software Development and Maintenance. In *IEEE - Software*, 1988.

[8] R.N. Taylor, F.C. Belz, L. Clarke, L.J. Osterweil, R.W. Selby, J.C. Wileden, A. Wolf, and M. Young. Foundations of the ARCADIA Environment Architecture. In *Proceedings of the ACM SIGSOFT-SIGPLAN Software Engineering Symposium on Practical Software Development Environments*, Boston, 1988. Appeared also in Software Engineering Notes, 13(5), February 1989.

[9] B. Warboys. The IPSE 2.5 Project : Process Modelling as the basis for a Support Environment. In *Proceedings of the International Conference on System Environment Development and Factories*, Berlin - Western Germany, May 1989.

[10] D.A. Cerino and F.S. LaMonica. An overview of Rome Air Development Center's Software Life Cycle Support Environment. In *Proceedings 5th Annual Knowledge-Based Software Assistant Conference*, pages 1–12, Syracuse, N.Y., September 1990. Rome Air Development Center.

[11] ECMA. A Reference Model for Frameworks of CASE Environments. Technical Report ECMA TR/55, European Computer Manufacturers Association, December 1990.

[12] I. Thomas, G. Boudier, F. Gallo, and R. Minot. An overview of PCTE and PCTE+. In *Proceedings of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments*, November 1988.

[13] A.I. Wasserman. Tool Integration in Software Engineering Environments. In *Software Engineering Environments: International Workshop on Environments*, Berlin, 1990. Springer-Verlag.

[14] D. Ornstein. Developping a CASE Interchange Standard : EDIF/CASE. In *[1]*, 1988.

[15] F. Gey, W. Benson, and J.L. Mc Carthy. IRDS Standards : A Superior Mechanism for CASE Information Repository and Exchange. In *[1]*, 1988.

[16] M.H. Law. Significance of IRDS Standard for CASE. In *[1]*, 1988.

[17] H. Tardieu, A. Rochfeld, and R. Coletti. *La methode MERISE.* Editions Organisation, Paris, 1983.

[18] C. Rolland, O. Foucaut, and G. Benci. *Conception de systèmes d'information : la mèthode REMORA*. Eyrolles, Paris, 1988.

[19] PCTE. A Basis for a Portable Common Tool Environment - Functional Specification. Technical report, PIMB, November 1988. Version 1.5, For C.

[20] GIE EMERAUDE. The Emeraude Environment, August 1987.

[21] I. Campbell, J.C. Derniame, J. Garbajosa, V. Gruhn, S. Psonis, and J.D. Zucker. The ALF System. Techical Report ALF/UDO-VG/WP-4-4-4-D1, ALF-Consortium, March 1991. Issue 1.0.

[22] N. Boudjlida and J.C. Derniame. Toward Process-Model-Based Software Environment Kernels. In *10th International Conference of the Chilean Computer Science Society*, Santiago de Chile, July 1990.

[23] N. Boudjlida. Formal Foundations for Value type quantification and Some Others Semantic Features. Technical Report ALF/NCY-NB/CT-4, ALF-Consortium, June 1990. Issue 0.1.

[24] N. Boudjlida, F. Charoy, J.C. Derniame, and C. Godart. Modeling of Software Process Assistance. In *CASE'89 - 3rd Workshop on Computer-Aided Software Engineering*, London, July 1989.

[25] THOMSON. Specification du langage XRete (XRete Specification Language). Technical report, THOMSON CSF, 1989. In French.

[26] J.C. Derniame, K. Benali, N. Boudjlida, C. Godart, and J. Lonchamp. Roles Cooperation through Software Process Instantiation. In *[27]*, pages 77–82, 1990.

[27] ISPW6. "Support for the Software Process". Proceedings of the 6th International Software Process Workshop, October 1990. Hakodate, Japan, Edited by IEEE Computer Society Press.