



Intégration de Rational/ClearCase avec les outils de développement

Auteur :

Eric Suignard
Consultant

e-mail : <mailto:eric.suignard@techap.com>

Société : TECH-AP

<http://www.techap.com/>

Résumé :

Les projets de développement logiciel utilisent aujourd'hui de plus en plus d'outils de développement dont l'intégration devient un souci majeur.

La multiplicité de ces outils (exemples : Visual Age, WebLogic, WSAD, Jbuilder, Oracle, Toad...) rend l'intégration avec ClearCase et UCM toujours plus complexe. ClearCase, avec ou sans la démarche UCM, doit collaborer avec les différents "repositories" propriétaires de ces outils.

Par ailleurs, l'intégration de ces outils doit être pensée de telle sorte que le processus de développement soit fluide, tout en tenant compte des contraintes nouvelles (fichiers .jar, .war...), et permette une mise au point correcte des bases de données.

Enfin, l'émergence du langage XML et des techniques de fabrication basées sur ce langage (Exemple : Ant) modifie la thématique de la fabrication. On adaptera l'utilisation de ClearCase, en fonction de cette évolution.

ClearCase, dans ses concepts et son architecture, est capable de fédérer tous les outils de développement. C'est d'autant plus vrai qu'il existe aujourd'hui des interfaces standard entre ClearCase et les outils de développement : SCC et WebDav. Toutefois, l'intégration nécessite des connaissances certaines sur des produits comme VisualAge, Jbuilder, Weblogic, Visual Interdev ou même Oracle.

Nous proposons une réflexion sur l'intégration de ClearCase non seulement avec les produits cités, mais également par type d'utilisation (architecture web, architecture applicative, architecture de base de données...) et un recensement des démarches à suivre.

Mots clés :

Gestion de configuration, gestion des changements, méthodologie, ClearCase, UCM, processus, XML, outils de développement, repositories, Visual Age, WebLogic, WSAD, Oracle.

Pour toute information complémentaire, e-mail : info.techap@techap.com

© 2002. Copyright TECH-AP. Reproduction et diffusion de ce document strictement interdites sans autorisation préalable de la société TECH-AP.



SOMMAIRE

1. INTRODUCTION.....	2
2. CONTEXTE.....	2
2.1. NECESSITE D'UNE INTEGRATION	2
2.2. AUGMENTATION DE LA COMPLEXITE DES ATELIERS DE DEVELOPPEMENT	3
2.3. ADAPTATION DU PROCESSUS DE DEVELOPPEMENT.....	4
2.4. TRANSFORMATION DES METHODES DE FABRICATION	5
3. ELEMENTS DE REPONSES	5
3.1. VISUAL AGE	5
3.2. ORACLE.....	8
3.3. VISUAL INTERDEV.....	9
3.4. WSAD	10
3.5. FABRICATIONS.....	10
4. GENERALISATION.....	13

1. INTRODUCTION

Les projets de développement logiciel utilisent aujourd'hui de plus en plus d'outils de développement. L'intégration de ces outils avec les outils de gestion de configuration doit être souple et s'adapter aux méthodologies modernes de développement des projets. A partir d'un retour d'expérience, l'objectif de ce document est de :

- réfléchir sur l'intégration entre ClearCase et ces outils de développement,
- souligner les difficultés que l'on peut rencontrer,
- proposer des solutions ou des façons d'aborder ces problèmes.

2. CONTEXTE

2.1. Nécessité d'une intégration

Le nombre d'outils de développement que l'on rencontre actuellement est dû à plusieurs raisons : volonté de réduire les temps de développement, augmentation du volume de code généré, augmentation du niveau d'abstraction...

Ces outils s'appuient souvent sur une gestion de fichiers. S'ils n'ont pas de système de gestion de configuration, il est alors souhaitable de les faire collaborer avec un logiciel de gestion de configuration ; s'ils ont un système de gestion de configuration intégré, celui-ci doit être complet, robuste, scalable et performant.

Dans tous les cas, on peut être amené à vouloir utiliser ces outils avec ClearCase, ne serait-ce que pour mettre en œuvre une solution globale et transverse au projet. Une telle intégration consiste à :

- faire en sorte que ClearCase et les outils de développement fonctionnent ensemble



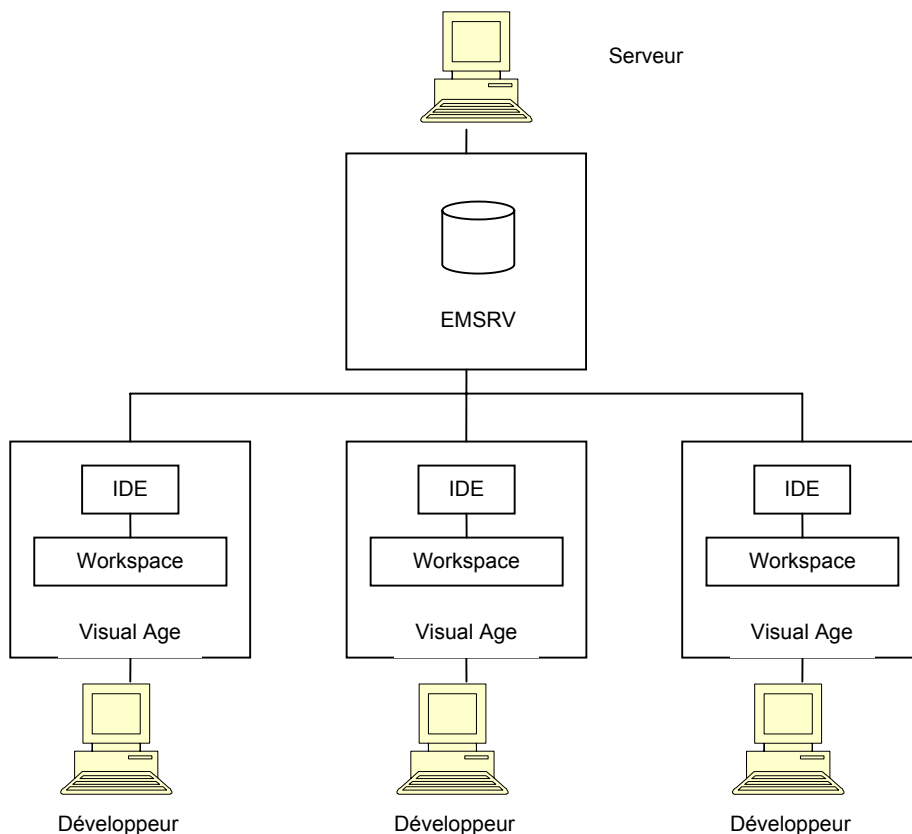
correctement sur un plan technique,

- adapter le processus de développement, de telle sorte que tous les intervenants du projet puissent accéder aux artefacts conformément aux rôles et aux responsabilités qui leur sont conférés et fassent évoluer le code applicatif tout au long de son cycle de vie.

2.2. Augmentation de la complexité des ateliers de développement

Les outils de développement rendent aujourd'hui nombreux. De plus en plus puissants, ils sont de plus en plus complexes (exemple : Visual Age, WSAD...) et nécessitent de faire collaborer ClearCase avec des repositories propriétaires. Ce point doit être résolu pour ClearCase de base, ainsi que pour UCM.

Dans le cas de Visual Age, des mécanismes propriétaires de traçabilité sont intégrés : notion d'éditations ouvertes, de fork, de fusion... Par ailleurs, un système d'accès concurrents s'appuyant sur EMSRV permet de travailler à plusieurs développeurs en parallèle.



Les mécanismes de gestion de configuration proposés de base sont efficaces et sont bien adaptés au développement de code Java. Toutefois, ils présentent rapidement des limitations sur les fonctionnalités classiques fournies par ClearCase : enregistrement très succinct de l'historique des modifications, pas d'assistance dans la fusion entre branches pour un ensemble de classes, pas de vision sur des ensembles de classes en dehors de



la notion de package, gestion de trop bas niveau pour les retours en arrière, scalabilité sommaire dans le volume des repositories, pas de trigger permettant de contraindre les développeurs à une politique de développement dûment identifiée et implémentée, ...

L'objectif consiste ici à bénéficier de la multiplicité et de la puissance des ateliers de développement, tout en utilisant un logiciel de gestion de configuration éprouvé.

2.3. Adaptation du processus de développement

L'intégration de ces outils doit être pensée de telle sorte que le processus de développement soit suffisamment souple, afin de faciliter le travail des développeurs, tout en étant rigoureux et fiable.

Besoin d'assouplissement

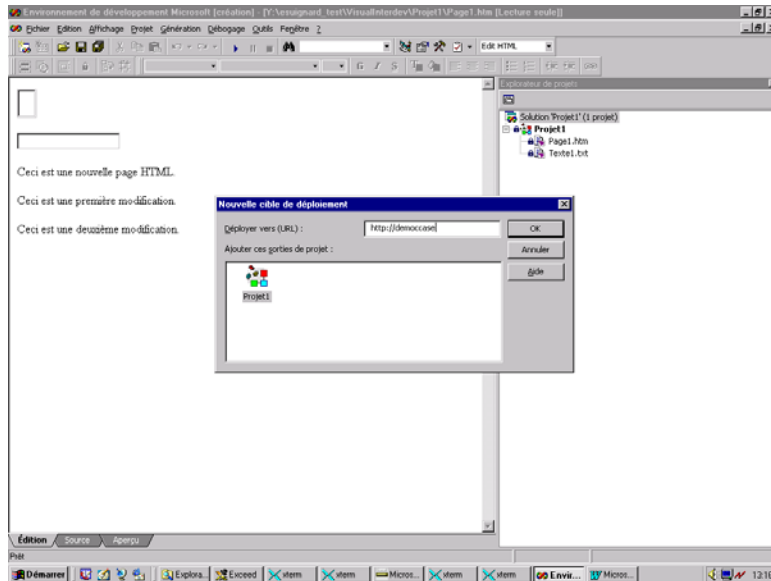
Aujourd'hui, dans le cadre des développements d'applications constituées de bases de données (exemple : Oracle), l'on observe certaines difficultés : pas de mécanisme de réservation/libération d'objets stockés dans les bases, pas de traçabilité des modifications apportées, mécanisme très sommaire pour les livraisons dans les différents environnements (développement, validation fonctionnelle, production, maintenance...). Une coopération entre ClearCase et Oracle doit être établie, de telle sorte que les opérations réalisées dans un environnement soient tracées et que le passage entre deux environnements soit cohérent et complet.

Besoin de plus de contraintes

Par ailleurs, les mécanismes de déploiement récemment apparus (exemple : fichiers .jar, .war...) peuvent facilement court-circuiter les objectifs de traçabilité. En effet, il est très facile aujourd'hui de déployer et publier sur un serveur d'applications une archive Java sans respecter les principes de :

- traçabilité (pas de checkout/checkin et impossibilité de savoir qui a publié une archive et pourquoi),
- gestion de la cohérence (publications asynchrones et pouvant être incohérentes entre elles),
- rôles (tout le monde peut publier n'importe quel composant à n'importe quel moment),
- identification (pas de label sur les composants livrés, ni sur l'assemblage de ces composants, pas de signature dans les sources ou les binaires).

Il s'agit ici d'introduire davantage de contraintes dans le processus de développement.



Selon le cas, la gestion de configuration avec ClearCase permet de faciliter le travail des développeurs, ou bien d'introduire davantage de contraintes.

2.4. Transformation des méthodes de fabrication

Enfin, l'émergence d'XML se poursuit et transforme les outils. Aujourd'hui, de plus en plus de projets ont des méthodes de fabrication construites sur XML et Ant qui modifient la thématique de la fabrication. On adaptera l'utilisation de ClearCase, en fonction de cette évolution.

3. ELEMENTS DE REPONSES

ClearCase est suffisamment mûr pour être capable d'apporter des solutions globales à toutes les problématiques de développement logiciel et particulières à tous les cas de figure. C'est d'autant plus vrai qu'il existe aujourd'hui des interfaces standard entre ClearCase et les outils de développement : SCC et WebDav. Toutefois, l'intégration nécessite des connaissances certaines sur les technologies de développement.

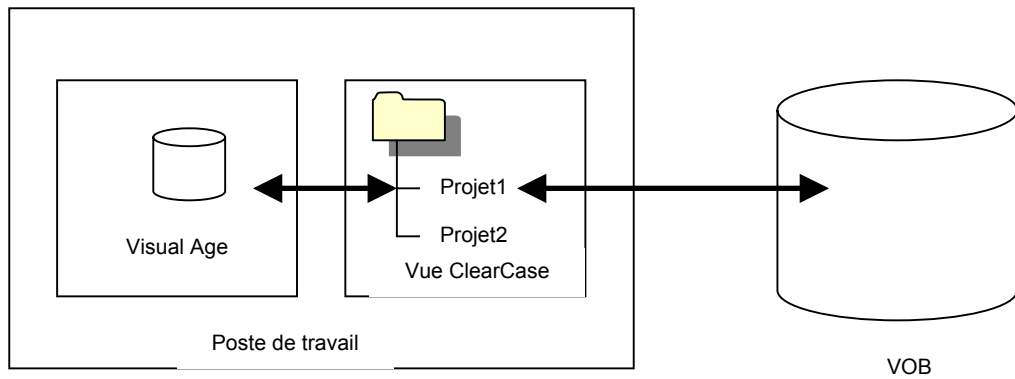
3.1. Visual Age

Par rapport aux mécanismes de gestion de configuration proposés, ClearCase assure une meilleure traçabilité, une meilleure identification et une meilleure sécurité des développements. Par ailleurs, ClearCase s'intègre facilement et permet d'ajuster au mieux les processus de développement.

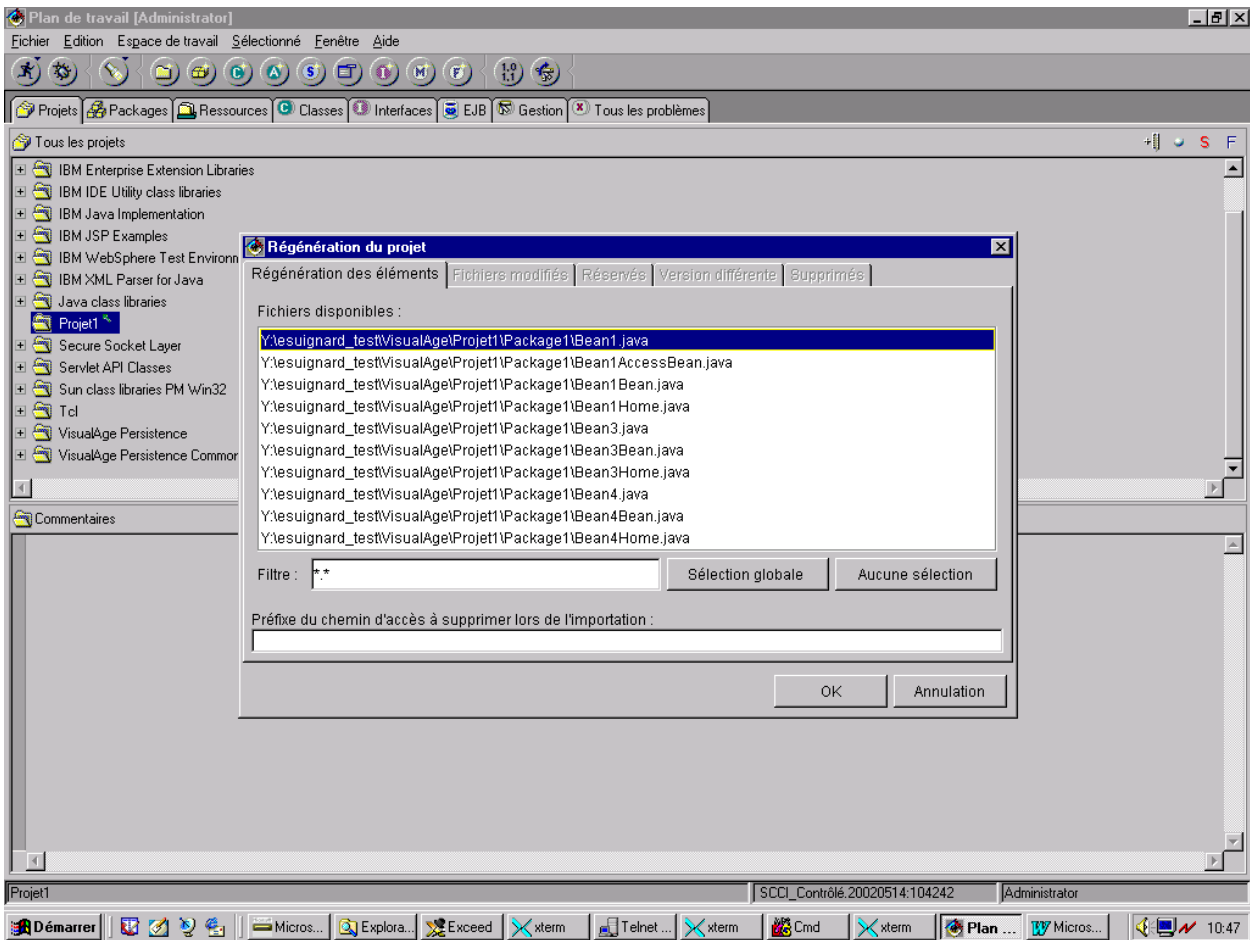
L'intégration entre Visual Age et ClearCase consiste à coupler l'espace de travail et le repository Visual Age avec une vue ClearCase. De même que l'on synchronise une vue statique avec les VOB ClearCase, on synchronise l'espace de travail Visual Age avec la



vue ClearCase, par l'appel à une fonction Visual Age. Afin que les modifications apportées par un développeur ne perturbent pas celles des autres développeurs, un espace de travail Visual Age doit être local à un poste de travail. La synchronisation se fait alors poste par poste.



Une synchronisation projet par projet permet d'intégrer dans l'espace de travail local les modifications apportées par d'autres développeurs et suit le processus de développement (modifications dans les environnements de développement, fabrication dans l'environnement d'intégration, fusions entre branche/stream pour livraison...).



Nous attirons l'attention sur le fait que ce couplage ne peut être réalisé qu'à partir de Visual Age V3.x.

Les applets sont des composants logiciels comportant une représentation graphique. Visual Age offre la possibilité de développer des applets et de définir leur "composition visuelle" (boîte de dialogue, menus, boutons...). Cette représentation graphique est conservée dans le repository local et, par défaut, ne correspond à aucun fichier à gérer en configuration sous ClearCase. Toutefois, il existe un moyen pour encoder, dans une méthode Java `getBuilderData()`, la composition visuelle en une sorte de "uencode"; cette méthode est contenue dans une classe Java qui est gérée en configuration d'une façon classique. Pour cela, on sélectionnera l'option "Génération de la méthode de métadonnées" accessible à partir du menu "Options"/"Génération de code".

Exemple :

```
private static void getBuilderData() {
  /*V1.1
  **start of data**
    D0CB838494G88G88G10D40FACGGGGGGG...
    86111988322A6A84D31950F02E886AA9D0DF..
  **end of data**/
}
```



S'agissant de la politique de branche, la présence de cette méthode `getBuilderData()` contenant un code semi-binaire interdit de faire des fusions non triviales ou non automatiques entre branches. Le développement d'applets avec Visual Age impose de travailler sur une seule branche et **ne supporte pas complètement UCM**.

Par ailleurs, il existe une autre information conservée dans le repository local et contenue dans aucun fichier géré en configuration : les définitions des groupes d'EJB. On limite alors le nombre de groupes d'EJB car ils doivent être créés manuellement, poste par poste.

Ceci a un impact si l'on souhaite accéder simultanément à plusieurs configurations (par exemple, une configuration en développement et une configuration de maintenance d'une version mise en production), au travers de vues ClearCase dédiées. L'accès aux sources via Visual Age se fait dans un espace de travail connecté à un repository et synchronisé avec une vue ClearCase. C'est dans l'espace de travail que sont définis les projets Java et leurs emplacements dans une vue ClearCase. Pour passer d'une configuration en développement à une configuration en maintenance, la définition de ces emplacements doit être modifiée. Afin d'éviter de devoir supprimer les projets Java, les recréer et redéfinir les groupes d'EJB, on sera amené à créer sur le poste de travail deux instances Visual Age, l'une pour la configuration de développement, l'autre pour la configuration en maintenance. Une instance est constituée d'un espace de travail et d'un repository.

Afin d'éviter les limitations mentionnées ci-dessus (utilisation des fonctionnalités UCM pour le développement d'applets et gestion des groupes d'EJB), on pourra passer à des ateliers du type WSAD.

3.2. Oracle

Périmètre de la configuration et gestion des espaces

Une configuration est constituée de fichiers accessibles au travers d'une vue. Mais, dans le cas du développement d'une base de données, elle est également constituée des structures des tables Oracle, ainsi que du code contenu dans la base (procédures stockées, fonctions...). Ces informations sont conservées dans des fichiers SQL, afin de tracer les modifications apportées et construire les environnements (développement, intégration, livraison...).

Le cloisonnement entre environnements est réalisé pour les fichiers à l'aide de vues ClearCase; s'agissant des données, le cloisonnement peut être réalisé avec des serveurs Oracle différents, ou bien des instances Oracle différentes (i.e. des bases différentes stockées sur le même serveur), ou bien encore des schémas (ou users) Oracle différents. Pour des raisons de souplesse, de performance et de facilité d'administration, on essaiera de cloisonner les environnements Oracle au niveau des schémas. On sera alors amené à mettre en place un ou plusieurs schémas de développement, un schéma d'intégration, un schéma de livraison, un schéma de maintenance, ...

Un transfert entre espaces consiste à se positionner dans une nouvelle vue ClearCase, effectuer les opérations classiques (fusion entre branches, fabrication, pose d'étiquette, ...) et à exécuter dans un nouveau schéma des scripts SQL. Toutefois, il faut savoir



distinguer les transferts *ex-nihilo* des transferts additifs : un ordre SQL "create or drop table XXX" est tout à fait correct lors de la première mise en production mais serait catastrophique s'il était lancé lors des mises en production suivantes.

Une solution consiste à conserver tous les ordres SQL affectant la structure d'une table dans des fichiers séparés : un fichier pour la création de la table, puis autant de fichiers qu'il y a de modifications apportées à la structure de la table. La mise en production et un retour en arrière sur une version précédente peuvent alors être réalisés en toute sécurité de la façon suivante :

- Exporter les données,
- Rejouer les scripts SQL,
- Importer les données.

Accès à la configuration

Les modifications apportées aux bases de données peuvent être réalisées par les développeurs ou déclenchées lors des transferts entre espaces. Dans tous les cas, elles s'appuieront nécessairement sur des vues ClearCase. Un outil de gestion de configuration doit en effet être la base commune à partir de laquelle tous les environnements sont mis à jour.

Il existe de nombreux outils de développement de base de données et certains offrent des fonctionnalités de gestion de configuration. La version 7 de Toad offre une interface SCC qui facilite ces opérations. Par ailleurs, le produit Oracle Repository offre des fonctionnalités de gestion de configuration accessibles au travers des produits Oracle JDeveloper (V3.2 ou supérieure), Oracle Reports and Forms et Oracle Designer. Cette gestion de configuration est encapsulée dans la base Oracle, ne s'appuie pas sur un système de fichiers et **n'est pas compatible avec ClearCase**.

Toutefois, les nouveaux ateliers de développement du type WSAD proposent des modules de modélisation de données (XDE) qui invoquent des fonctions de gestion de configuration ClearCase et qui résolvent bon nombre de limitations mentionnées ci-dessus.

3.3. Visual InterDev

Les mécanismes de déploiement proposés par les ateliers de développement doivent être gérés avec ClearCase. Ces mécanismes opèrent à la fois sur des fichiers (.jar, .war) et sur des arborescences complètes (répertoire WEB-INF).

Comme on l'a vu, il est très facile de déployer et publier un composant sur un serveur d'applications à partir d'un atelier de type Visual InterDev. Dans la mesure du possible, on proscriera ces mécanismes et l'on cherchera à faire en sorte que la publication soit effectuée à partir d'une vue ClearCase. Il faut donc s'assurer que le répertoire de publication est contenu dans une vue ClearCase et que cette vue a des protections telles que seules les personnes habilitées à la publication peuvent modifier les fichiers présents sous ce répertoire.

Là encore, si l'on souhaite bénéficier de mécanismes de déploiement robustes, on



s'orientera vers des ateliers du type WSAD qui proposent des assistants de déploiement très élaborés.

3.4. WSAD

WSAD est un framework de développement logiciel construit sur deux composants logiciels : WSW et Eclipse (Open Source). L'intégration avec ClearCase a été implémentée au sein même de WSAD à l'aide de plug-in et ne passe pas par une interface externe (du type SCC ou WebDav). La collaboration entre les deux logiciels en est d'autant plus étroite : on accède directement au travers de WSAD à un grand nombre de fonctions ClearCase :

- Création d'une vue,
- Mise à jour d'une vue statique,
- Accès aux fonctions UCM,
- Accès aux propriétés d'un élément ou d'une version d'un élément,
- Accès à l'arbre de version et à l'historique d'un élément,
- ...

Il n'y a guère que la pose d'une étiquette qui reste encore en dehors de l'atelier.

Si l'on souhaite accéder simultanément à plusieurs configurations (par exemple, une configuration en développement et une configuration de maintenance d'une version mise en production), il suffit de créer autant de vues ClearCase et autant d'espaces de travail WSAD.

3.5. Fabrications

Avec certains ateliers de développement (Visual Age, par exemple), la fabrication est entièrement prise en charge par l'atelier. Il n'est alors pas nécessaire de mettre en œuvre clearmake qui permet de partager les objets dérivés, rédiger les audits de fabrication, compiler sur plusieurs machines en parallèle...

Par ailleurs, les fabrications qui s'appuient aujourd'hui sur Ant peuvent être intégrées aux mécanismes ClearCase. Elles s'appuient sur des règles formalisées en XML et définissent les cibles, les dépendances et les actions à réaliser. Elles permettent de fabriquer du code Java pour intégration et déployer les packages sur un serveur d'application (du type WebLogic).

Nous avons aujourd'hui les moyens d'intégrer, dans un processus de fabrication sous Ant, des opérations ClearCase (checkout, checkin, pose de label...). Certaines de ces opérations sont fournies dans les dernières distributions Ant (voir ci-dessous l'exemple 1). Toutes les autres opérations peuvent être codées en XML ou en Java (voir ci-dessous l'exemple 2).

Exemple 1 :

```
<project name="MyProject" default="dist" basedir=". ">
  <property name="src" value="src"/>
  <property name="build" value="build"/>
```



```
<property name="dist" value="dist"/>
<property name="inc" value="inc"/>
<property name="exe" value="\${build}\total.exe"/>
<property name="dep" value="\${build}\total.obj \${build}\init.obj"/>

<taskdef name="ccmklbtype"
classname="org.apache.tools.ant.taskdefs.optional.clearcase.CCMklbtype" />
<taskdef name="ccmklabel"
classname="org.apache.tools.ant.taskdefs.optional.clearcase.CCMklabel" />

<target name="init">
  <tstamp/>
  <mkdir dir="\${build}"/>
</target>

<target name="compile" depends="init">
  <javac srcdir="\${src}" destdir="\${build}"/>
</target>

<target name="dist" depends="compile">
  <mkdir dir="\${dist}/lib"/>
  <jar jarfile="\${dist}/lib/MyProject.jar" basedir="\${build}"/>
</target>

<target name="clean">
  <delete dir="\${build}"/>
  <delete dir="\${dist}"/>
</target>

<target name="c-obj" depends="init">
  <apply executable="gcc" dest="\${build}" parallel="false">
    <arg value="-c"/>
    <arg value="-I\${inc}"/>
    <arg value="-o"/>
    <targetfile/>
    <srcfile/>
    <fileset dir="\${src}" includes="*.c"/>
    <mapper type="glob" from="*.c" to="*.obj"/>
  </apply>
</target>

<target name="c-exe" depends="c-obj">
  <exec executable="gcc">
    <arg line="-o \${exe}"/>
    <arg line="\${dep}"/>
  </exec>
</target>

<target name="cc-co">
  <cccheckout viewpath="C:\view\test_view\ant_test\build.xml"
reserved="true"
nowarn="true"
comment="Some comment text"/>
  <ccuncheckout viewpath="C:\view\test_view\ant_test\build.xml"
keepcopy="true"/>
</target>

<target name="cc-lb">
  <ccmklbtype replace="true" label="ANT_TEST_V1" comment="My comment"/>
  <ccmklabel recurse="true" label="ANT_TEST_V1" comment="My comment"
viewpath="C:\view\test_view\ant_test" />
</target>
```



```
</project>
```

Exemple 2 :

```
<project name="MyApp" default="dist" basedir=". ">  
  <property name="livraison.dir"    value="../Build" />  
  <property name="livraison.zip"    value="MyApp.zip" />  
  
  <target name="sauvegarde.war" depends="web, ressources">  
    <echo message=" - Check Out du fichier ZIP." />  
    <execon executable="cleartool">  
      <arg value="co"/>  
      <arg value="-comment"/>  
      <arg value="&#171;Modification sous Ant&#171;"/>  
      <arg value="\${livraison.zip}"/>  
      <fileset dir="\${livraison.dir}">  
        <include name="\${livraison.zip}"/>  
      </fileset>  
    </execon>  
  </target>  
</project>
```

On peut donc continuer à appeler des fonctions ClearCase dans une fabrication avec Ant. Toutefois, comme avec les ateliers de développement, certains mécanismes de fabrication clearmake (tels que le winkin) présentent aujourd'hui moins d'intérêt qu'avant : les packages sont déployés et n'ont plus besoin d'être réutilisés dans d'autres fabrications. En revanche, la fabrication avec ClearCase conserve toute son actualité avec des langages de programmation tels que C#.



4. GENERALISATION

ClearCase, dans ces concepts et son architecture, doit être capable de fédérer tous les outils de développement. Il est possible que, ponctuellement, pour un problème donné, ClearCase soit moins adapté que certaines offres. En revanche, aujourd'hui, seuls les outils de gestion de configuration évolués tels que ClearCase offrent une solution globale multi-plateformes (Unix, Windows), robuste, évolutive, supportant un grand nombre de formats de fichier (texte, binaire, Word, XML, ...) et proposant une intégration avec la plupart des logiciels du marché (Word, Visual C++, Visual Age, Visual InterDev, WSAD, .Net, ClearQuest, Rose...).

Par ailleurs, les solutions directement apportées par les outils de développement sont souvent très basiques voire approximatives. La gestion de configuration est un processus intégré à part entière et il est inutile de réinventer la roue sans cesse. Il vaut mieux s'appuyer sur des outils qui sont éprouvés depuis longtemps et fournissant des fonctionnalités puissantes (findmerge, UCM, clearmake, metadata, ...).

La mise en œuvre de la gestion de configuration sur un projet peut alors davantage se focaliser sur la définition du processus de développement à implémenter :

- Gestion et assemblage des composants applicatifs,
- Gestion des versions,
- Définition des rôles des intervenants,
- Définition des activités (développement, livraison, mise en production, maintenance...) et des modes opératoires associés,
- ...

Maintenant, il faut penser l'intégration de ClearCase non seulement produit par produit, mais également par type d'utilisation (architecture web, architecture applicative, architecture de base de données...) et recenser les ateliers logiciels compatibles, les méthodes à suivre et les pièges à éviter, de façon à faciliter et fluidifier le processus de développement. C'est l'orientation qu'ont suivie les ateliers de développement actuels WSAD et .Net.