PROCESSUS METIERS DE LA PRODUCTION

SOMMAIRE

1 Les processus métiers de la Production

1.1 Le contrat de service

Une application a pour objet de délivrer un ensemble de fonctionnalités à ses utilisateurs. Ce service délivré est lui-même encadré dans un contrat de service fixant les engagements de chacun des acteurs. Ce contrat implicite ou non, définit les plages d'ouverture du service, les performances attendues, la durée maximale d'indisponibilité tolérée et la perte de données acceptée en cas de sinistre majeur. La production informatique a la responsabilité et s'engage de son côté à mettre en œuvre les dispositions nécessaires pour assumer ses obligations en échange de quoi la maîtrise d'ouvrage utilisatrice du service s'engage à respecter un certain nombre de consignes explicitées dans le SLA (Service Level Agreement - ou Contrat de Service).

Un contrat de service pragmatique et tangible doit permettre d'être en mesure de vérifier que les engagements pris puissent être tenus.

1.2 Les éléments du contrat de service

Les différents critères du contrat de service sont :

Disponibilité du service

- Définir les heures d'ouverture du service pour le TP, pour les traitements batch, pour les travaux de maintenance,
- la durée maximale d'indisponibilité programmée,
- la durée maximale d'indisponibilité non programmée,
- Assurer la production des traitements et la disponibilité des résultats.

Performances

- La disponibilité des résultats des traitements dans les délais impartis,
- Les performances TP,
- La mise à disposition des données aux clients dans les délais impartis (Interfaces, fichiers…).

Confidentialité

- Garantir la confidentialité des informations,
- Gérer les droits.

Sécurité

- Garantir l'intégrité des données,
- Garantir la persistance des données d'entreprise,
- Garantir la reprise sur incident.

Fiabilité

- Maîtriser l'impact des évolutions,
- Etablir rapidement un diagnostic sur incident,
- Capitaliser les problèmes rencontrés,
- Mesurer la fiabilité.

1.3 Les différents acteurs

Les différents acteurs sont :

Les exploitants

Les exploitants assurent la production des environnements techniques et applicatifs. Leurs principales tâches sont :

- Assurer le suivi des traitements de production des environnements techniques (sauvegardes, purges, contrôles, capacity planning…),
- Assurer le suivi des traitements des applications (Ordonnancement des traitements, sauvegardes applicatives, sauvegardes des données, reprises,…),
- Assurer le suivi des flux de données et la mise à disposition (émission, réception, sauvegardes des résultats),
- Participer aux recettes de production.

Les supports utilisateurs

- Assurer le support téléphonique premier niveau,
- Assurer le suivi des incidents,
- Gérer l'escalade en fonction de la criticité,
- Invoquer le support second niveau.

Les supports seconds niveau

- Maintenir un niveau d'expertise pour accomplir leur mission,
- Surveiller et améliorer les performances du système,
- Surveiller les ressources disponibles,
- Résoudre les problèmes techniques « pointus »,
- Proposer des solutions palliatives,

- Proposer des solutions d'éradication des problèmes,
- Apporter un soutien technique aux différents métiers et processus (notamment la recette).

Les différents processus métier de la production partent de la mise à disposition de moyens techniques (matériels et logiciels), méthodologiques et humains pour la mise en oeuvre des environnements des applications.

Il convient ensuite d'assurer la livraison des éléments à mettre en production suite à la recette fonctionnelle qui elle, fait partie des processus du développement, de qualifier la complétude de cette livraison et d'en assurer l'intégration dans l'environnement de production.

La recette d'intégration (ou VABF) permettra alors de définir et de disposer des éléments indispensables au bon fonctionnement en production (Sauvegardes, ordonnancement, procédures de maintenance du bon fonctionnement des environnements, disponibilité d'indicateurs de surveillance des ressources, traçabilité des opérations…).

La pré-production (ou VSR) aura quant à elle pour objet de valider la qualité de service régulier. Lorsque la qualité est suffisante, la production peut alors démarrer, l'ensemble des éléments conçus dans la phase d'intégration étant opérationnels.

2 Les processus de la mise en production

2.1 Introduction

Pour garantir la qualité du service telle qu'exigée par le SLA, les différents acteurs de l'informatique doivent mettre en œuvre des processus métiers transverses qui alimentent les processus éléments indispensables aux processus métiers de suivi de la production.

Ces processus métier sont :

La mise en production, allant de l'installation des environnements techniques à la recette de production fait intervenir plusieurs processus métier et peut se décliner comme suit :

- La mise à disposition des environnements système (machines et systèmes d'exploitation),
- La mise en œuvre des environnements techniques que sont les logiciels de base comme un SGBD, un serveur d'applications…,
- La mise en œuvre de l'exploitation (Arrêt/Démarrage, Purges, sauvegardes, Suivi des ressources, surveillance, supervision, tableaux de bord,…),
- La mise en œuvre de l'environnement applicatif (logiciels, données, droits…),
- La mise en oeuvre de l'exploitation de l'application (Ordonnancement, sauvegardes des résultats, contrôles et transmission des résultats d'exploitation, tableaux de bord, surveillance et supervision).

On voit clairement apparaître différents processus comme :

- La mise à disposition des environnements matériels, qui est un processus à part entière qui nécessite la livraison et l'installation des machines en salle machine,
- L'installation du système d'exploitation conformément aux besoins de l'application,

- L'étude de la sécurisation des composants en fonction du SLA et du plan de backup éventuel si la machine est susceptible d'héberger des applications dont le SLA est élevé (haute disponibilité…) ,
- Une recette pour valider l'environnement.

La mise à disposition des environnements techniques est un autre processus métier qui peut nécessiter l'intervention de spécialistes du domaine. Ce processus nécessite l'installation du composant technique conformément aux règles et aux normes en vigueur, la création et le dimensionnement de l'enveloppe de gestion (Base, Circuits virtuels…en fonction du type de composants).

2.2 Les ressources gérées par les services techniques

2.2.1 Les ressources gérées

Il existe différents types de ressources :

- Les ressources de stockage,
- Les ressources d'exécution,
- Les ressources de communication.

Pour assurer le contrat de service proposé il va falloir :

- Assurer la disponibilité des ressources,
- Garantir la sécurité d'accès aux ressources,
- Garantir les performances des ressources.

Assurer la disponibilité de la ressource c'est :

- La mettre à disposition conformément aux exigences du service demandé,
- Surveiller la consommation qui en est faite pour adapter la ressource à la consommation qui en est faite et à son évolution (capacité planning),

- Gérer le taux d'occupation, et la ressource disponible.

Garantir la sécurité d'accès aux ressources, c'est :

- Protéger l'accès en implantant les mécanismes de confidentialité d'accès,
- Protéger les chemins d'accès par des mécanismes de redondance si nécessaires,
- Vérifier l'intégrité de la ressource,
- Prévoir une ressource alternative pour le plan de secours si nécessaire,
- Inclure les éléments nécessaires à la mise à disposition de la ressource sur le site de secours (procédures de configuration, dimensionnement, installation,
- Sauvegarder la ressource, la restaurer.

Garantir les performances de la ressource, c'est :

- Vérifier que l'usage qui en est faite est conforme à l'usage attendu,
- Surveiller le temps de service de la ressource aux différents types de sollicitation,
- Gérer la répartition de la consommation sur l'ensemble des ressources disponibles,
- Arbitrer les contentions et incompatibilité d'utilisation,
- Optimiser son usage.

2.2.2 Les procédures associées aux ressources gérées

Assurer la disponibilité de la ressource, c'est :

Install <Ressource> = Mettre à disposition la quantité de ressource spécifiée
Quantité de ressource allouée

ChgState <Service> = Mettre la ressource online ou offline.
OK ou KO en fonction de l'état demandé (ON ou OFF)

GetState <Service> = Donne l'état de la ressource.
OK ou KO en fonction de l'état demandé (ON ou OFF)

Check <Ressource> = Vérifier l 'existence de la ressource
Code retour OK,KO

Check <Ressource> <Etat> = Vérifier que la ressource est dans l'état <Etat>
Code retour OK,KO

Use <Ressource> = fournit l'utilisation de la ressource
La quantité allouée

La quantité utilisée

La quantité disponible

Le pourcentage d'utilisation

Le pourcentage libre

Stat <Ressource> = Qui écrit dans un fichier table le résultat de la méthode Use pour fournir un fichier afin de générer un graphique représentant l'évolution de l'utilisation de la ressource.

Fréquence ✳ Cyclique selon les cycles définis.

Garantir la sécurité d'accès aux ressources, c'est :

Check_Protect <Ressource> = Vérifie les droits
Liste des différences par rapport au référentiel de droits

Check_AltPath <Ressource> = Vérifie que la sécurisation est active, si nécessaire (Par exemple vérifie que le miroir est OK).
OK,KO

Check_integrity <Ressource> = Vérifie les règles d'intégrité
OK,KO + erreurs sur stdout

Save_FullOffLine <Storage> = Sauvegarde la totalité de la ressource persistante alors qu'elle n'est pas utilisée
OK,KO

Save_FullOnLine <Storage> = Sauvegarde la totalité du contenu (s'il y a lieu) de la ressource en cours d'utilisation.

OK,KO

Save_IncOffLine <Stotage> = Sauvegarde incrémentale du contenu (s'il y a lieu) de la ressource en dehors de son utilisation

OK,KO

Save_IncOnline <Storage> = Sauvegarde incrémentale du contenu (s'il y a lieu) de la ressource en dehors de son utilisation

OK,KO

Verif_Save <Storage> = Vérifie que la sauvegarde est conforme aux attentes

% de conformité

Cette procédure peut être basée sur les caractéristiques des sauvegardes précédentes (pour du stockage : volume sauvegardé la veille ou sur un nombre de fichier ou les deux), l'objectif étant d'adjoindre à un contrôle technique de type code retour un contrôle « métier ».
La sauvegarde d'un fichier vide, alors qu'il ne devrait pas l'être, donnera un contrôle technique OK alors que le contrôle métier ne sera pas OK.

Save_Definition <Ressource> = Sauvegarde les caractéristiques de la ressource

OK,KO

Cette procédure doit permettre de sauvegarder les éléments permettant de définir les ressources sur un système de back up d'architecture différente (comme c'est souvent le cas dans un plan de secours).
Garantir les performances de la ressource, c'est :

Get_Load <Ressource> = Permet d'obtenir la charge générée par les accès à la (ou aux) ressource(s), permet aussi de voir l 'équilibrage.

Nb de requêtes utilisant la ressource

Temps moyen d'obtention

Temps moyen d'attente

List_Queue <Ressource> = Voir le contenu de la file d'attente d'accès à la ressource.
Liste des tâches utilisant la ressource

Check_Accelerator <Ressource> = Permet de vérifier que les mécanismes permettant d'accélérer les accès aux ressources sont opérationnels.

Cette méthode est liée fortement au type de ressource considérée, il peut s'agir de caches, de mécanismes d'indexation…

ViewMgrLog <Log> = Voir le log de suivi du fonctionnement du gestionnaire de ressource

ViewClientLog <Log> = Voir le log de suivi de l'utilisation client de la ressource.

HistoMgrLog <Log> = Historier et purger les logs du gestionnaire

HistoClientLog <Log> = Historier et purger les logs des clients utilisateurs de la ressource.

3 La production des traitements

La production informatique doit assurer le suivi de la production des applications et notamment de leurs traitements automatisés ou non. L'ordonnanceur et automate de production fournit un certain nombre de mécanismes permettant d'assurer la planification et le déclenchement des traitements selon un planning de production prédéfini et d'assurer l'enchaînement des traitements en fonction de leur cinématique fonctionnelle.

Pour cela la production doit :

- Programmer les traitements et enchaînements dans l'ordonnanceur, définir leur planification et calendrier d'exécution,
- Identifier les éléments de suivis et contrôles,
- Valider les consignes de reprise sur incidents,
- Assurer la sécurité des données en cas de problème applicatif,
- Vérifier le bon fonctionnement des traitements et enchaînements et notamment la validité des éléments déclenchant,
- Vérifier la cohérence d'ensemble.

Pour pouvoir assurer l'ensemble de ces tâches la production va devoir disposer des éléments nécessaires et pour cela va s'adresser au développeur ou au fournisseur des traitements.

Les éléments dont il va avoir besoin seront les suivants :

La description des traitements :

- Objets en entrée des traitements et leur utilisation,
- Actions exécutées par le traitement,
- Objets en sortie des traitements et utilisation,
- Objets temporaires créés,
- Les contraintes horaires d'exécution,

- Les ressources nécessaires,
- La granularité transactionnelle,
- La cadence de traitement,
- Les compatibilités du traitement avec les activités externes,
- Les éléments de suivi du traitement, les éléments de contrôle des résultats du traitement.

Pourquoi avons-nous besoin de cet ensemble d 'éléments, de la part des développeurs ?

Si l'on compare un traitement informatique au fonctionnement d'une usine, les matières premières seraient les objets en entrée, le produit fabriqué serait les objets en sorties, le traitement serait le poste de travail les copeaux seraient les objets temporaires à éliminer, le tapis roulant serait alors assuré par l'ordonnanceur. Un poste de travail effectue une série d'actions dans un environnement particulier. Il doit être adapté aux conditions opérationnelles dans lesquelles il effectue sa tâche et respecter une certaine cadence de fabrication avec une bonne fiabilité afin de ne pas fragiliser l'ensemble de la chaîne de fabrication. Il en va de même pour les traitements applicatifs, il est simplement plus difficile de définir ce que sont réellement la cadence, les réglages et les contraintes opérationnelles. Pourtant, ils existent, quels sont-ils ?

Imaginons qu'un traitement soit susceptible d'être exécuté pendant la journée, ce qui est une contrainte, il en résulte que la granularité transactionnelle devra être fine afin d'éviter la pose de verrous trop nombreux rendant impossible tout accès transactionnel en mise à jour. Cette granularité pourrait être un réglage, si ce traitement valide des liasses le nombre de liasses validées par minute pourrait être la cadence. De même qu'une pièce ne peut se trouver sur deux postes de travail en même temps, ce qui au-delà de l'évidence est une contrainte, une table de base de données ne peut généralement pas être modifiée par deux traitements simultanément ce qui doit être pris en compte dans l'ordonnancement du processus. Il va sans dire que le bon fonctionnement global nécessite des points de contrôles tant du fonctionnement de la chaîne de fabrication que de la qualité de la production. Les travaux de maintenance permettront de maintenir l'outil de production en état.

En poursuivant l'analogie on mettrait en évidence que le temps d'exécution est une forme de suivi qualité, que les erreurs détectées peuvent conduire à des maintenances etc.…

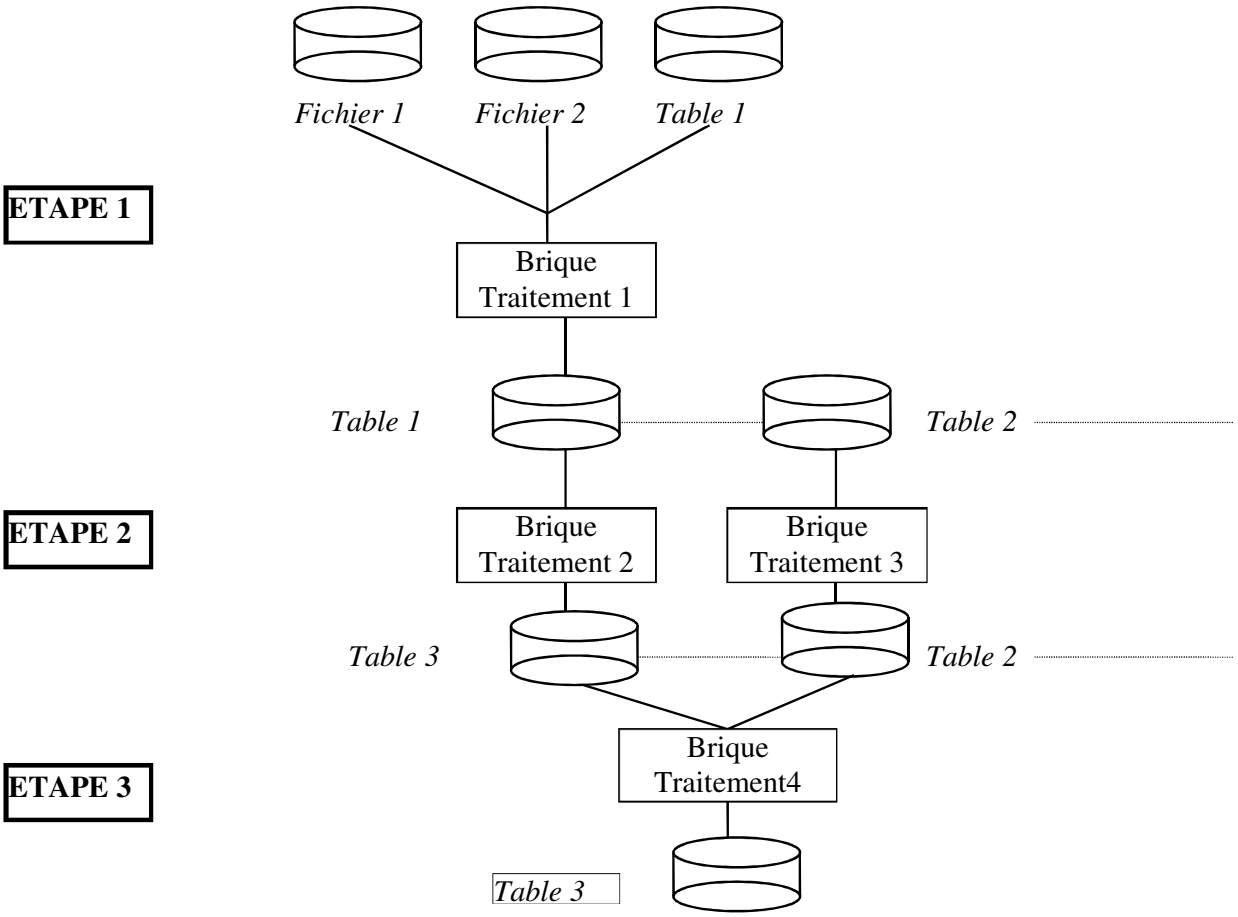
Revenons aux différents éléments dont nous avons besoin pour garantir la production des traitements.

4 Le référentiel d'exploitation

4.1 La description des traitements

4.1.1 Introduction

La description des traitements est fournie par la documentation du fournisseur de logiciel ou du développeur sous forme de dossier de production ou organigramme détaillé des traitements. Dès lors la description des traitements peut se représenter de la façon suivante :



Chaque page d'organigramme représente alors une phase fonctionnelle.

Afin de décrire ce type d'enchaînement nous avons introduit les notions de phases, étapes et briques. Ces différentes notions sont utilisées pour décrire les enchaînements fonctionnels dans un référentiel qui sera utilisé ultérieurement par la production.

4.1.2 Les phases

Une phase est définie de la façon suivante :

C'est un ensemble cohérent de traitements fonctionnels.
Exemple : Traitement des liasses (PH_TRT_LIASSES)
Une phase est composée de n étapes en séquence.
Une phase a des contraintes d'ordonnancement identiques pour toute la durée de la phase idéalement une phase à une unité de traitement identique pour l'ensemble de ses briques ce qui permet à partir d'un comptage initial d'évaluer le temps d'exécution approximatif de la phase et des ressources nécessaires à son exécution.

L'identification des phases est avant toute fonctionnelle puis il faut vérifier que ses contraintes fonctionnelles et techniques d'ordonnancement sont homogènes.

Exemple : Si la phase doit s'effectuer hors TP, toutes les briques de la phase sont exécutées hors TP.

La notion de phase a été introduite pour faciliter l'ordonnancement des briques en décrivant un premier niveau d'enchaînement des briques.
Cela permet de n'avoir à se préoccuper que de l'ordonnancement des phases l'enchaînement définitif des briques en découlant directement.
Le planning d'exécution est relié à la phase.
Un comptage initial des éléments à traiter est souhaitable en début de phase.
Un traitement de contrôle de fin de phase est également souhaité.

4.1.3 Les étapes

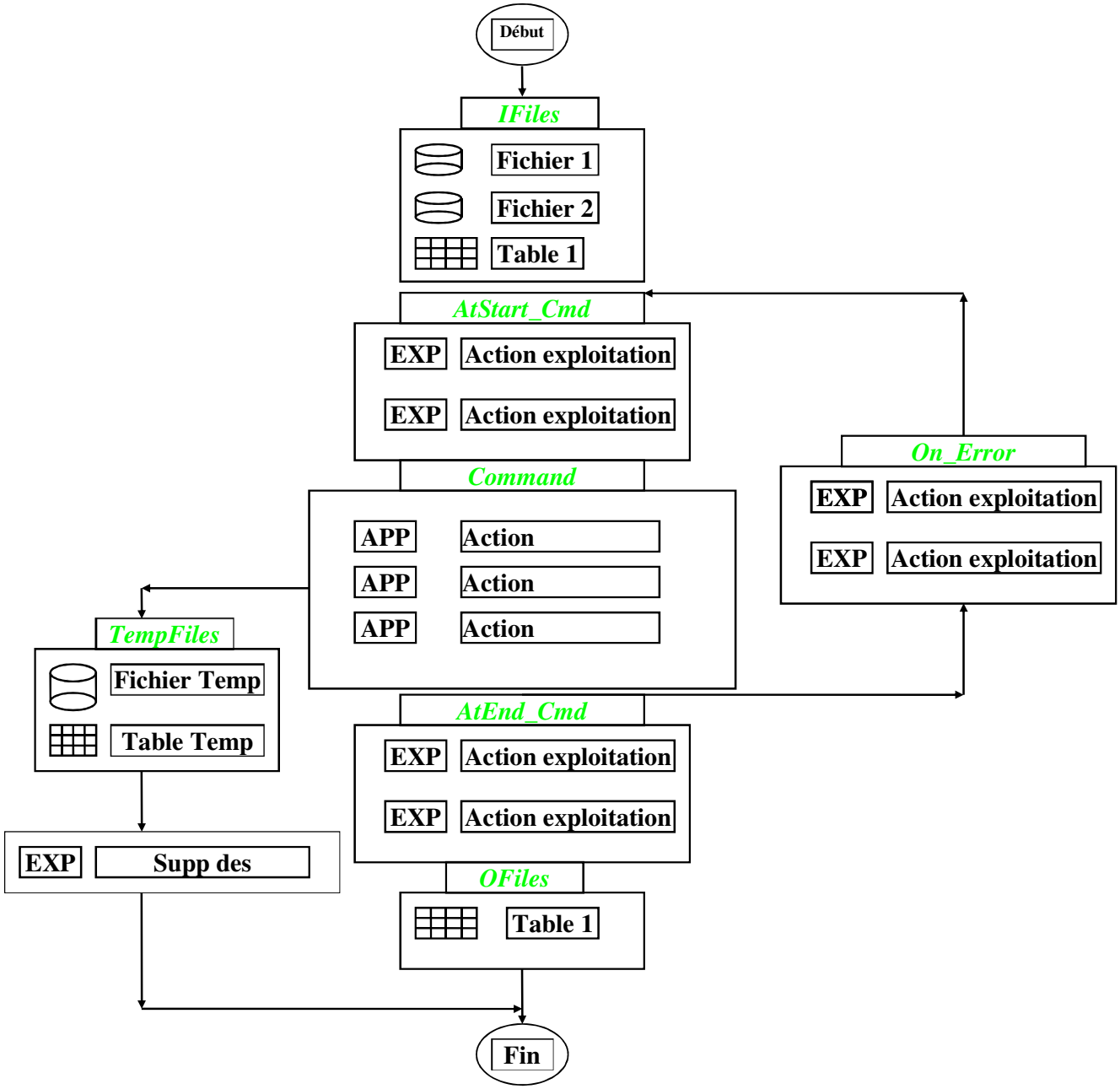
Les étapes n'ont pas de sens à priori elles ne servent qu'à décrire la parallélisassions de certains traitements à l'intérieur d'une phase. Les étapes n'existent que par nécessité dans le modèle de description des traitements. Il s'agit en réalité d'une notion technique permettant de décrire les phases telles que représentées dans le paragraphe I.1.1 du présent chapitre.

4.1.4 Les briques

Les briques représentent la notion fondamentale de la description des traitements. Leurs caractéristiques sont les suivantes :

- Elles constituent une unité fonctionnelle,
- Elles sont des unités de reprise (En cas d'échec d'une brique, toute la brique est ré exécutée),
- Ce sont les atomes de l'ordonnancement des traitements,
- Chaque brique correspond à un script à ordonnancer .

On peut représenter une brique comme suit :



Une brique de traitement a des objets en entrée qui sont les données lues (*IFiles*), des objets en sortie, ce sont les données mises à jour (*OFiles*), le corps du traitement lui-même (*Command*) qui est une suite d'actions applicatives telles que l'exécution d'un programme java, d'un tri..., des actions d'exploitation à effectuer après traitement.(*AtEnd_Cmd*) telles que la mise à disposition des éditions ou fichiers résultats par envoi via FTP. Une brique peut également créer des objets temporaires qui doivent être supprimés en fin de brique. La brique constituant une unité de reprise, elle doit également comporter des actions de remise en état afin que la brique puisse être ré exécutée lorsque la cause de l'erreur aura été inhibée. Il faut donc lui adjoindre des actions d'exploitation avant traitement (*AtStart_Cmd*), tels que des sauvegardes ou le changement du nom d'un fichier...ainsi que les actions de remises dans l'état initial en cas d'erreur (*On_Error*).

La brique possède une unité de traitement et une cadence de traitement. : par exemple un nombre de lignes de modification d'entreprise. La cadence étant le nombre de lignes traitées par unité de temps (Minutes ou secondes).

Des ressources sont associées à la brique certaines ressources sont quantitatives (Les roll back segments Oracle par exemple) et devraient être fournies en fonction du nombre d'unités traitées.

Un contrôle peut être également associé.

A l'aide des phases, étapes et briques il devient relativement aisé de décrire les traitements applicatifs et de les enrichir des actions d'exploitation indispensables pour la reprise des traitements en cas de dysfonctionnement.

4.2 Les méthodes et paramètres

4.2.1 Les méthodes

Les méthodes sont les scripts standard permettant d'effectuer des actions particulières. Bien que regroupées, on peut distinguer deux types de méthodes :

- les méthodes applicatives qui permettent de décrire les traitements applicatifs ce sont les tris, les exécutions de programmes JAVA, les chargements et déchargements de données, ces méthodes sont fournies par les études... ,
- et les méthodes d'exploitation qui permettent d'effectuer les sauvegardes des données avant traitement, vérifier les conditions d'exécution du traitement, les mises à disposition des éditions, les transferts de fichier ces méthodes sont généralement définies par la production.

Les méthodes s'appliquent à du matériel de production fichiers Unix, tables Oracle, programmes.

Une méthode peut n'avoir de sens que vis à vis d'un type matériel de production particulier (table Oracle) ou plusieurs types de matériel (Tables et fichiers).

Un traitement individuel pourra ainsi être décrit comme une suite d'actions (méthodes) sur des objets particuliers (programmes, fichiers, tables...). Chaque action fait référence à une méthode prédéfinie.

4.2.2 Les paramètres

Pour des raisons d'automatisation les briques une fois ordonnancées ne devraient pas comporter de paramètres sinon ils s'agiraient de traitement « à la demande ». Les méthodes elles, comportent très souvent des paramètres et les paramètres sont alors décrits dans les actions de la brique. Par exemple un tri doit savoir sur quelles colonnes du fichier il opère. L'action décrite dira **« Trier fichier toto col3 »** par exemple. De même une action applicative peut être paramétrée **« Exécuter monprog paramètre1 paramètre2…. »**. Par contre les paramètres fournis ne bougeront plus une fois le script de production réalisé et ordonnancé. Si tel n'est pas le cas et que les paramètres varient d'un jour sur l'autre le traitement sera considéré comme traitement « à la demande ». De même certains paramètres peuvent être liés à la configuration de l'environnement de production, ces paramètres se traduisent

généralement par des variables d'environnement (sous Unix et Linux) c'est le cas de l'identifiant d'une base de données, du chemin d'un répertoire de travail etc.…
Ces paramètres sont résolus au moment de l'exécution en appelant une fonction standard d'initialisation (profile Unix…), ces paramètres ne sont pas des paramètres de traitements mais des paramètres d'implantation des composants techniques, d'ailleurs ils sont communs à bon nombre de traitements.

4.3 Les objets : fichiers, tables et codes

4.3.1 La virtualisation des objets : les variables d'environnement

Afin de faciliter l'accès aux objets de production, les caractéristiques de stockage des objets sont généralement fournies par des variables d'environnement. Par exemple le nom de propriétaire des tables sera fourni par la variable PROPRIO, de sorte que si le nom du propriétaire change seule la valeur de la variable PROPRIO sera changée et l'ensemble des traitements en bénéficieront. Il en sera de même pour les répertoires de stockage des codes exécutés. Plus généralement, les variables d'environnement sont utilisées pour s'affranchir de l'implantation physique des objets manipulés.

L'ensemble des variables d'environnement décrivant l'implantation des objets de production est regroupé dans un fichier « profil de production ». Chaque traitement utilise le profil de production.

4.3.2 Les méthodes de production d'application

Les méthodes de production d'application représentent l'ensemble des actions qui pourront être exécutées au cours des traitements, il peut s'agir de l'exécution de code application (code Java) ou l'exécution de code spécifique au service comme par exemple le déchargement de données, le transfert d'un fichier…

Le transfert de fichier utilisera le code spécifique du service technique de transfert (FTP, CFT…). De même, le déchargement de données utilisera le code spécifique du service de données (Oracle, DB2…).

4.4 La fin d'un traitement

L'élément déclenchant un traitement au cours de l'exécution d'une chaîne planifiée de traitements est, en premier lieu, le statut de fin du (des) traitement(s) précédent(s). Ce statut est généralement le code retour du traitement (ou « exit code, return code…. »), Sous Unix toute commande exécutée a un code retour qui est égal à 0 si la commande s'est exécutée correctement ou différents de zéro dans le cas contraire. Lorsque ce code est différent de zéro sa valeur correspond à une erreur comme indiqué dans le fichier système */usr/include/sys/errno.h.*

Un script shell Unix renvoie le code retour de la dernière commande exécutée. Il est primordial que la dernière commande exécutée par un script de production soit toujours exit $CODEXIT où $CODEXIT est la valeur du code retour valorisée à la suite des différents tests effectués avant la sortie du script. L'intérêt est de gérer soit même la gravité de l'erreur rencontrée, toutes les erreurs n'ayant pas le même sens. Nous préconisons que le code renvoyé soit un code de sévérité et non pas d'identification de l'erreur. L'erreur étant identifiée par un message d'erreur dans le fichier log de suivi des traitements.

Nous utilisons les codes :

- 0 pour OK,
- 201 pour un warning,
- 202 pour une erreur bloquante,
- 203 pour une erreur bloquante dont l'impact est plus vaste que le script courant.

Ces codes ont le mérite de ne jamais être renvoyé par une commande standard et donc de s'assurer que le code retour est bien géré.

4.5 Les fonctions de logs de production

Les scripts de production font systématiquement appel à des fonctions d'écriture dans des fichiers logs. Ces fonctions permettent d'écrire des logs parfaitement formatés disposant de l'ensemble des informations requises. Ces informations sont les suivantes :

Date	Tampon horaire indiquant quand le message a été écrit
domaine	Domaine fonctionnel ayant émis le message comme par exemple BACKUP pour les procédures de sauvegardes
père	Numéro de process du père du script ayant émis le message
fils	Numéro de process du script ayant émis le message
proc	Nom complet du script ayant émis le message les appels successifs sont séparés par : Exemple : P_sauve_db :F_check_db Signifie que c'est la fonction F_check_db appelée par la procédure P_sauve_db qui a émis le message
typemsg	Indique si le message est un message de : début de script : DEBUT fin ou résultat du script : RESULTAT informatif : INFO erreur : ERREUR
Numerror	Indique Le numéro éventuel de l'erreur si le type du message est ERREUR.
Séverité	Indique la sévérité du message : 0=OK 1=Warning 2=Erreur bloquante 3=Erreur bloquante grave qui peut avoir une incidence sur les autres traitements.
Message	Le contenu du message

4.5.1 Code retour et logs des briques et des traitements d'application

Il faut distinguer le traitement application en tant que tel, de la brique de traitement ; le traitement application peut-être un exécutable du code de l'application ou une méthode de production d'application telle que mentionnée ci-dessus.

La brique de traitement est l'unité ordonnancée, c'est une unité de reprise, elle doit donc vérifier ses conditions d'exécution (La base de donnée est-elle accessible, l'espace disque est-il suffisant, la sauvegarde des données est-elle bonne ?). C'est seulement lorsque les conditions d'exécution auront été vérifiées que le traitement d'application proprement dit sera déclenché, ensuite les résultats du traitement seront exploités (Transfert, sauvegarde…).

Une brique de traitement comme nous l'avons décrit précédemment comporte des actions avant traitement, les actions du traitement et des actions après traitement.

Une brique de traitement peut donc être interrompue soit parce que son environnement d'exécution n'est pas conforme à ce qu'il devrait être, soit parce que le traitement d'application s'est mal terminé ou encore parce que l'exploitation des résultats n'a pas pu se faire correctement. Le code retour du traitement d'application indique la bonne fin de celui-ci. Le code retour de la brique indique la bonne fin ou non de la brique c'est à dire que les conditions d'exécution étaient bonnes, que l'exécution du traitement s'est bien passé et que les résultats du traitement ont pu être exploités (Contrôles, Sauvegardes, transmission….

4.6 Les différentes méthodes à développer

Les méthodes d'acquisition de données

- Réception,
- Contrôle,
- Historisation,
- Purges.

Les méthodes de manipulations des objets de production

- Copie,
- Renommage,
- Chargement,
- Déchargement,

- Suppression,
- Création, dimensionnement.

Les méthodes d'exécution des traitements de données

- Formatage,
- Filtrage,
- Tri,
- Exécution de codes ,
- Vérification d'intégrité,
- Recyclage des rejets,
- Les méthodes de vérification des environnements de production,
- Les méthodes de contrôle des résultats.

Les méthodes de mise à disposition des résultats de production

- Mise en forme,
- Emission,
- Archivage, historisation,
- Externalisation, sécurisation,
- Extraction,
- Interprétation (données de synthèse…).

5 Stratégies de sauvegardes et externalisation

5.1 Les stratégies de sauvegardes

Les stratégies de sauvegardes se composent de :

Sauvegardes totales, incrémentales et différentielles.

Nous appelons ici sauvegardes totales d'un environnement les sauvegardes qui prennent la totalité de l'environnement sans aucune considération de filtrage sur date de modification, création ou autre.

Nous appelons sauvegarde incrémentale d'un environnement toute sauvegarde ne prenant en compte que ce qui a été modifié par rapport à la sauvegarde précédente et ce quelle que soit la nature de la sauvegarde précédente.

Nous appelons sauvegarde différentielle d'un environnement toute sauvegarde ne prenant en compte que ce qui a été modifié par rapport à la sauvegarde totale précédente.

Par exemple si on fait une sauvegarde totale à J0, une sauvegarde incrémentale à J1, une sauvegarde incrémentale à J2, Une sauvegarde différentielle à J3.

La sauvegarde J1 ne contient que le delta en le jour J1 et J0

La sauvegarde J2 ne contient que le delta entre J2 et J1

La sauvegarde J3 contient le delta entre J3 et J0 et donc le delta entre J3 et J2, J2 et J1 et J1 et J0.

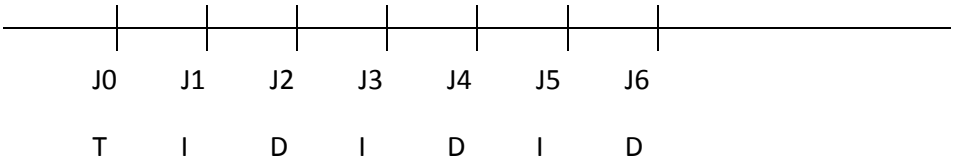
Une sauvegarde différentielle est donc le cumul des sauvegardes incrémentales précédentes jusqu'au point initial constitué par la sauvegarde totale.

Nota : La sauvegarde J1 est à la fois incrémentale et différentielle.

L'intérêt des sauvegardes différentielles est de réduire le nombre de sauvegarde à restaurer. L'intérêt de la sauvegarde incrémentale est de réduire le volume sauvegardé au strict minimum.

Une stratégie de sauvegarde se construit à parti de ces différents types de sauvegardes en fonction des contraintes de temps et de volume.

Sur un environnement peu volumineux, rapide à sauvegarder, on pourra se permettre d'effectuer des sauvegardes totales quotidiennes. Pour un environnement volumineux et lourd à sauvegarder, on choisira des sauvegardes totales hebdomadaires et des sauvegardes incrémentales quotidiennes. De plus si le délai de restitution en cas d'incident est critique on remplacera une sauvegarde incrémentale sur deux par des sauvegardes différentielles pour réduire le nombre de lot de sauvegarde à restaurer.



T: Totale

I: Incrémentale

D : Différentielle

On voit que cette stratégie permet de n'avoir à restaurer que 3 sauvegardes au plus quelque soit le cas de figure : la sauvegarde totale, une sauvegarde différentielle et une sauvegarde incrémentale.

Si on remplace les sauvegardes différentielles par des sauvegardes incrémentales, le nombre de sauvegardes à restaurer en cas d'incident dépend du jour de l'incident. Pour restaurer à J6 il faudra J0, J1, J2, J3, J4, J5, J6 soit 7 sauvegardes alors qu'avec les sauvegardes différentielles précédentes il aurait fallu J0+J6 soit 2 sauvegardes au lieu de 7.

5.2 Les types de sauvegardes

Il existe deux types de sauvegardes les sauvegardes logiques et les sauvegardes physiques.

Les sauvegardes physiques, consistent à sauvegarder des enveloppes sans se soucier du contenu interne et de l'organisation des objets stockés dans l'enveloppe. Pour l'exemple une sauvegarde base fermée Oracle est une sauvegarde physique, on ne voit pas le contenu lorsqu'on sauvegarde.

Les sauvegardes logiques permettent de sauvegarder le contenu organisé de la structure en parcourant les liens spécifiques de la structure, un export Oracle est une sauvegarde logique car on parcours les différentes structures dictionnaires, users, tables…

5.2.1 Les sauvegardes logiques

Avantages :

Ces sauvegardes ont l'avantage de parcourir la structure et donc de valider que les liens internes sont corrects. Par exemple lorsque l'on sauvegarde un système de fichier on parcourt l'arborescence dans sa logique (répertoire, sous-répertoires, fichiers etc…).

Inconvénients :

La sauvegarde logique exige que la structure soit « en-ligne » et donc elle peut être en cours d'utilisation, il n'y a pas de garantie d'intégrité des données entre elles sauf si des mécanismes spéciaux les garantissent (cas de l'option CONSISTENT du export Oracle). La sauvegarde logique nécessite l'existence **préalable** de la structure pour être restaurée. Il faut donc disposer des scripts de création de la structure.

Les sauvegardes logiques sont souvent utilisées pour des environnements dont les composants sont peu liés, c'est le cas des systèmes de fichiers ou la cohérence inter-fichiers est peu importante.

5.2.2 Les sauvegardes physiques

Avantages :

La sauvegarde physique peut se faire alors que la structure n'est pas « en-ligne » et de ce fait garantie une consistance des données internes à la structure. Aucun accès n'est effectué pendant la sauvegarde. C'est le cas d'un système de fichier non monté sous Unix et que l'on sauvegarde physiquement par sauvegarde du disque logique. C'est également le cas d'une base Oracle arrêtée et dont on sauvegarde les fichiers servant d'implantation physique à la base de données (Datafiles, control files, Redologs…). La garantie d'intégrité est beaucoup plus forte.

Inconvénients :

Les structures ne sont pas validées et il n'est pas impossible qu'une anomalie de structure soit sauvegardée. Pour une base Oracle il serait possible qu'un bloc Oracle corrompu fasse partie de la sauvegarde physique sans que l'on s'en rende compte.

On choisira les sauvegardes physiques pour des environnements nécessitant une forte cohérence, c'est à dire lorsque les objets logiques sont fortement liés, comme c'est le cas des tables de base de données. Les bases de données sont sauvegardées régulièrement par des sauvegardes physiques.

5.3 Les sauvegardes pour l'externalisation

Les sauvegardes pour l'externalisation sont souvent des copies des sauvegardes de la stratégie définie : copie des sauvegardes totales, incrémentales et différentielles mise en œuvre, qu'elles soient physiques ou logiques.

En plus de ces copies, on adjoindra les éléments de construction de l'environnement et des bandes boot.

Les bandes boot

Elles ont pour rôle de redescendre un environnement système de démarrage conforme aux caractéristiques principales de l'environnement initial. C'est à partir de cet environnement que seront redescendues les autres sauvegardes.

Les éléments de construction de l 'environnement

Ces éléments sont descendus prioritairement après la bande boot afin de construire l'environnement complet de réception des autres sauvegardes.

Le processus de restauration sur site externe est le suivant :

- Mise à disposition de la ressource,
- Boot bande et restauration de l'environnement système,
- Restauration des sauvegardes incrémentales du système,
- Restauration des données de configuration et création de l'environnement complet,
- Mise en œuvre de l'environnement complet,
- Restauration des environnements selon l'ordre de priorité défini au préalable (Sauvegardes totales + Incrémentales + différentielles).

6 Déroulement et organisation

6.1 La formation, la sous-traitance de certaines tâches

Le nombre de composants techniques nouveaux que les personnels de production ont à gérer sont souvent importants et complexes.

La formation des équipes est à ce titre primordiale, mais il semble également essentiel de focaliser les acteurs sur les savoirs réellement nécessaires à l'exercice de leurs fonctions. Les tâches trop spécialisées pourront être sous-traitées.

6.2 La répartition des tâches

Suivant la quantité des tâches à effectuer, il est essentiel d'identifier les responsabilités de chacun. Ces responsabilités ne sont pas toujours clairement définies et les frontières sont parfois floues. Qui est responsable de l'ordonnanceur ? Qui est responsable de l'outil de sauvegarde ? Qui développe les scripts de gestion des environnements ? Qui développe les scripts de la production des traitements ? Qui développent les scripts de mise en production ? Qui définit les règles et normes d'écriture ? Qui fournit le support technique sur les différents composants de l'architecture logicielle de la production ?

Toutes ces questions doivent trouver une réponse claire pour que chacun sache ce qu'il a à faire. Il est également essentiel de focaliser les différentes équipes sur les outils essentiels à leur fonction et donc de les spécialiser sur ces outils.

6.3 Conception et réalisation de l'environnement de production

Une fois les rôles de chacun définis, il faut distribuer les tâches aux différents acteurs. Parmi ces tâches certaines nécessitent une ressource dédiée et ne sont pas compatibles avec une activité opérationnelle de production.

Les tâches particulièrement prenantes sont :

- La définition des normes et règles d'écriture des scripts,
- La définition des normes et règles d'implantation d'un composant technique,
- La réalisation des scripts de gestion des composants techniques,
- La réalisation des scripts de production des traitements.

Il est donc nécessaire de prévoir de dédier certaines tâches à certains acteurs.