Association common Romandie

13 mars 2012

Change / Release Management

Philippe MAGNE – PDG ARCAD Software

Première partie:
La valeur ajoutée business

L’objectif de l’activité Change & Release Management est de :

« Fournir un support méthodologique commun dans la gestion des changements applicatifs, quelque soient les technologies employées. Elle assure une qualité constante des livrables quel que soit le taux de changement »

- Traçabilité des évolutions
  - Qui fait quoi, quand, où, pourquoi ?
- Sécurité des évolutions
  - Synchronisation
  - Non régression
- Automatisation
  - Haute disponibilité applicative

Quelques généralités

Facteurs de croissance du marché

- **Pressions réglementaires:**
  - SOX, Bâle III, Solvency II, LSF,
- **Référentiels de bonnes pratiques:**
  - ITIL, CMM, COBIT,
- **Alignement business / IT:**
  - Agilité du SI,
  - Modernisation du SI,
- **Rationalisation des coûts:**
  - Automatisation des processus

Change & Release management: pourquoi ?

- Sensibilité: augmentation du niveau d'organisation des changements logiciels:
  - Y a-t-il des environnements définis (développement, test, production) ?
  - Les relations Etudes/production sont-elle bonnes ?
  - Y a-t-il déjà eu des cas de régressions de sources ?
  - Quelle est la fréquence des mises en production ?
  - Les déploiements sont-ils automatisés ?
  - La réception de nouvelles versions de progiciels n'engendre-t-elle pas une charge de travail conséquente en interne ?

Change & Release management: pourquoi ?

- Le réflexe de « maintenabilité » du SI,
- Un socle de base indispensable à une bonne organisation informatique,
- La colonne vertébrale pour l'ensemble des activités de changement
  - Développement,
  - Tests,
  - Production,
  - Documentation,
  - Support technique.

common Romandie

L'association romande des utilisateurs de systèmes IBM

Pour passer du stade artisanal à l’ère industrielle.

Terminologie

Norme AFNOR

"Ensemble des activités (manuelles ou automatisées) permettant d'identifier et de définir les éléments de configuration et toutes leurs relations. Elle permet de contrôler les évolutions durant le cycle de vie du logiciel, d'archiver chacun des états successifs et de vérifier que chacun de ces états est complet et cohérent".

(Norme **AFNOR : NF Z 67-102**)

Norme ISO 10007 (*)

Activités d'ordre technique et organisationnel comprenant :
l'identification de la configuration,
la maîtrise de la configuration,
l'enregistrement de l'état de la configuration,
l'audit de la configuration.

(*) Adoptée par le CEN (Comité Européen de Normalisation)

Différences

- Gestion de version vs Gestion de Configuration,
- La gestion de configuration permet en plus :
  - de gérer les demandes de modification du système à faire évoluer,
  - de mettre en correspondance les demandes de modifications avec les changements apportés au système.
- Ex: CVS = gestion de version,
- ALM vs SCM,
- L'ALM couvre un périmètre plus large…

L'ALM

- Requirements Management,
- Modelling,
- Design,
- Project Management
- **Change Management,**
- **Configuration Management,**
- Build Management,
- Testing,
- **Release Management,**
- **Deployment,**
- Monitoring and Reporting.

« Collaborer »

« Collaborer »

- De nombreux acteurs:
  - Développeurs
  - Test/recette
  - Production

- + de « nouveaux »…
  - DBA
  - Architecte

« Collaborer »

- De nombreuses technologies
  - Applications « Legacy »
  - Client/serveur
  - Web
    - Interface
    - Webservices

« Collaborer »

- Un manque de visibilité à tous les stades du changement

« Collaborer »

- Technologies différentes signifie:
  - Diversité des cultures,
  - Diversité des méthodes de travail,

- D’où un réel challenge dans l’industrialisation

« Réduire les risques »

« Réduire les risques »

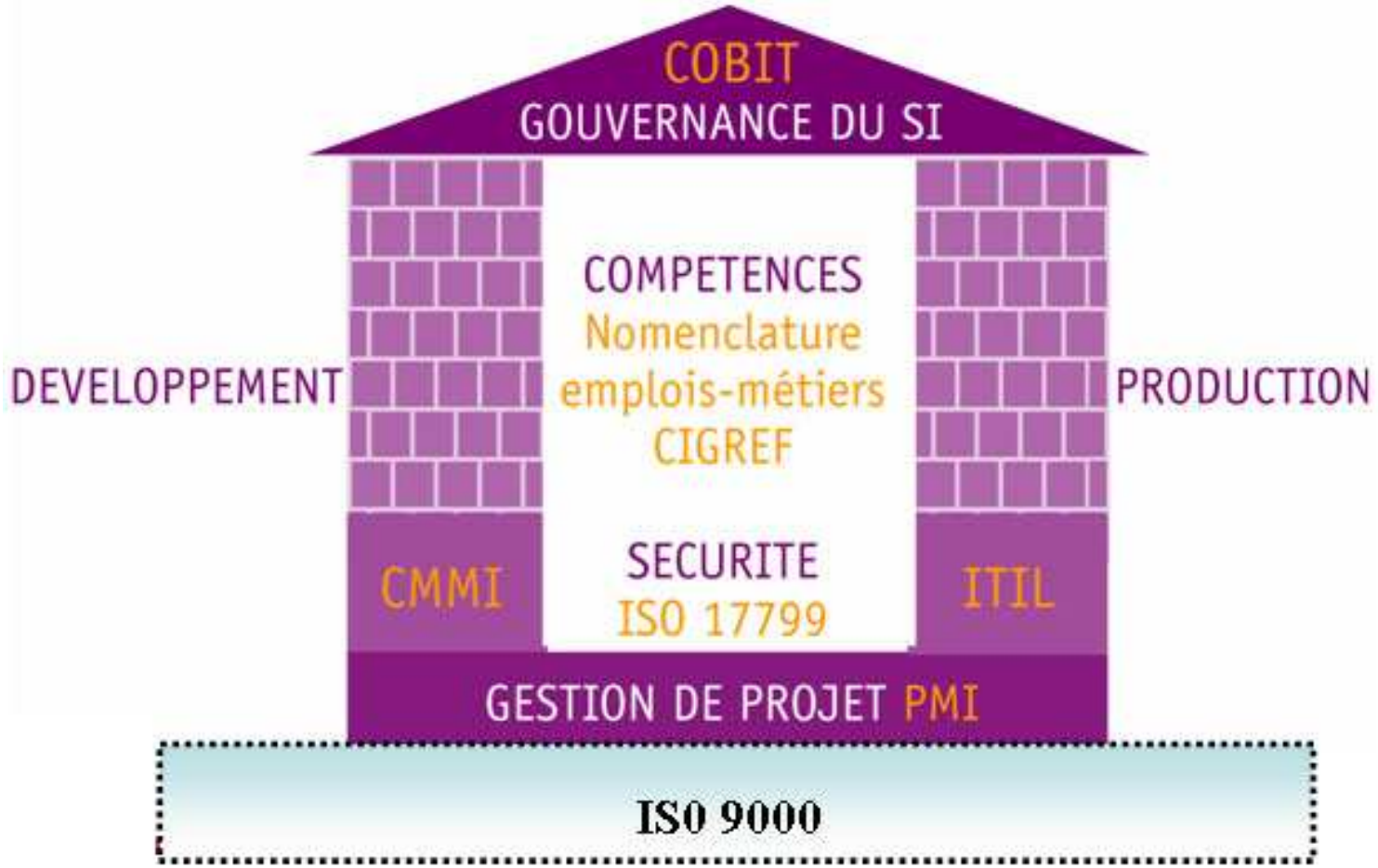
- **Indisponibilité** de certaines fonctions du SI
  - Temps de mises en production non maitrisés
  - Mises en production durant heures ouvrées

« Réduire les risques »

- Régressions suite aux nouvelles versions
  - Fonctions mal (voire pas) testées
  - Non adéquation au besoin exprimé

« Adopter les meilleurs pratiques »

Les différents référentiels



COBIT

- PO4 – Définir l'organisation et les relations de travail,
- PO10 – Gérer les projets,
- PO11 – Gérer la qualité.

- AMP2 – Acquérir des applications et en assurer la maintenance,
- AMP3 – Acquérir une infrastructure technologique, en assurer la maintenance,
- AMP5 – Installer les systèmes et les valider,
- AMP6 – Gérer les changements.

- DS9 – Gérer la configuration,
- DS10 – Gérer les problèmes et les incidents,
- DS11 – Gérer les données.

COBIT

Ce que vérifie l'auditeur :

- Existence de procédures,
- Mise en place des contrôles prévus lors de l'audit précédent,
- Leur application,
- Leur pertinence et leur efficacité,
- Qu'elles sont évaluées et révisées périodiquement.

ITIL: Le concept

- Construire et faire évoluer des bonnes habitudes de travail dans un cadre structuré pour obtenir de meilleures performances,
- Il est basé sur l'obtention d'un haut niveau de Qualité de service du SI avec une attention particulière sur la Relation Client.

Les atouts d'ITIL

- Souple: non normatif,
- Pragmatique: ancré dans l'action (du déjà vu),
- Adapté/intelligent/subtil (du bon sens),
- Modulaire: convient aux petites structures comme aux grosses dans une démarche de petits pas,
- Référentiel commun: acteurs interne DSI, DSI et métiers, DSI et fournisseurs, éditeurs de logiciels,
- Approche qualité: le bon processus, la bonne organisation, la bonne mission, la bonne formation, les bons outils, les indicateurs de mesure…

Les bénéfices d'ITIL

- **Intégration** de l'informatique dans les processus métiers de l'entreprise,
- **Alignement** de l'informatique sur les métiers de l'entreprise,
- Changement culturel vers le **service,**
- Une organisation claire, sytématique et auditable,
- Capacité à absorber un nombre important de changements avec une grande **fiabilité,**
- Un travail sur le service perçu.

Le configuration management

- L’un des processus phare du référentiel ITIL,
- Positionné dans le “Service Transition” (ITIL V3).

Service Transition

- Planifier et manager les ressources nécessaires pour déployer une release en production,
- Etablir et maintenir l'intégrité de tous les services et configurations durant leur installation (transition),
- Produire une bonne qualité d'information et de connaissance pour permettre une prise de décision pertinente entre la phase de test et la phase de mise en production,
- Produire des mécanismes reproductibles de construction et de déploiement des releases dans les environnements de test et de production,
- Assurer que le service peut être managé, exploité et supporté selon les exigences et contraintes spécifiées dans le service Design.

CMMi

Maturity Level 2:

- CM - Configuration Management,
- MA - Measurement and Analysis,
- PMC - Project Monitoring and Control,
- PP - Project Planning,
- PPQA - Process and Product Quality Assurance,
- REQM - Requirements Management,
- SAM - Supplier Agreement Management.

Maturity Level 3:

- DAR - Decision Analysis and Resolution,
- IPM - Integrated Project Management +IPPD,
- OPD - Organizational Process Definition +IPPD,
- OPF - Organizational Process Focus,
- OT - Organizational Training,
- PI - Product Integration,
- RD - Requirements Development,
- RSKM - Risk Management,
- TS - Technical Solution,
- VAL – Validation,
- VER – Verification.

Maturity Level 4:

- QPM - Quantitative Project Management.

Maturity Level 5:

- OPP - Organizational Process Performance,
- CAR - Causal Analysis and Resolution,
- OID - Organizational Innovation and Deployment.

« Automatiser »

- Auditer l’existant
  - Réduction du périmètre des composants à maintenir
  - Sécuriser et optimiser les changements

- Analyses d’impact outillées
  - Gains de productivité
  - Garantie d’exhaustivité

- En phase de développement
  - Augmentation de l’automatisation
  - Simplification des tâches quotidiennes du développeur

- => Concentration du développeur sur ses tâches métier

- Contrôler l’intégrité des lots à livrer:
  - Garantie de non régression
  - Suppression des anomalies « techniques » en amont du cycle de validation
  - Garantie de synchronisation des changements inter-technologies

- => Gains de temps très importants en phase d’intégration

- Transferts inter-environnements automatisés:
  - Suppression des procédures manuelles,
  - Traçabilité des process,
  - Sécurité des process (Rollback automatique),
  - Capitalisation des modes opératoires.

- => Augmentation de la disponibilité des applications

- Transferts inter-environnements multi-plateformes
  - Simplicité d'administration
    - les même outils gèrent tous les types de composants
  - Sécurité maximale
    - Le rollback est activable à n'importe quel stade du process
  - Synchronisation garantie

- => Garantie de compatibilité des briques applicatives

En conclusion…

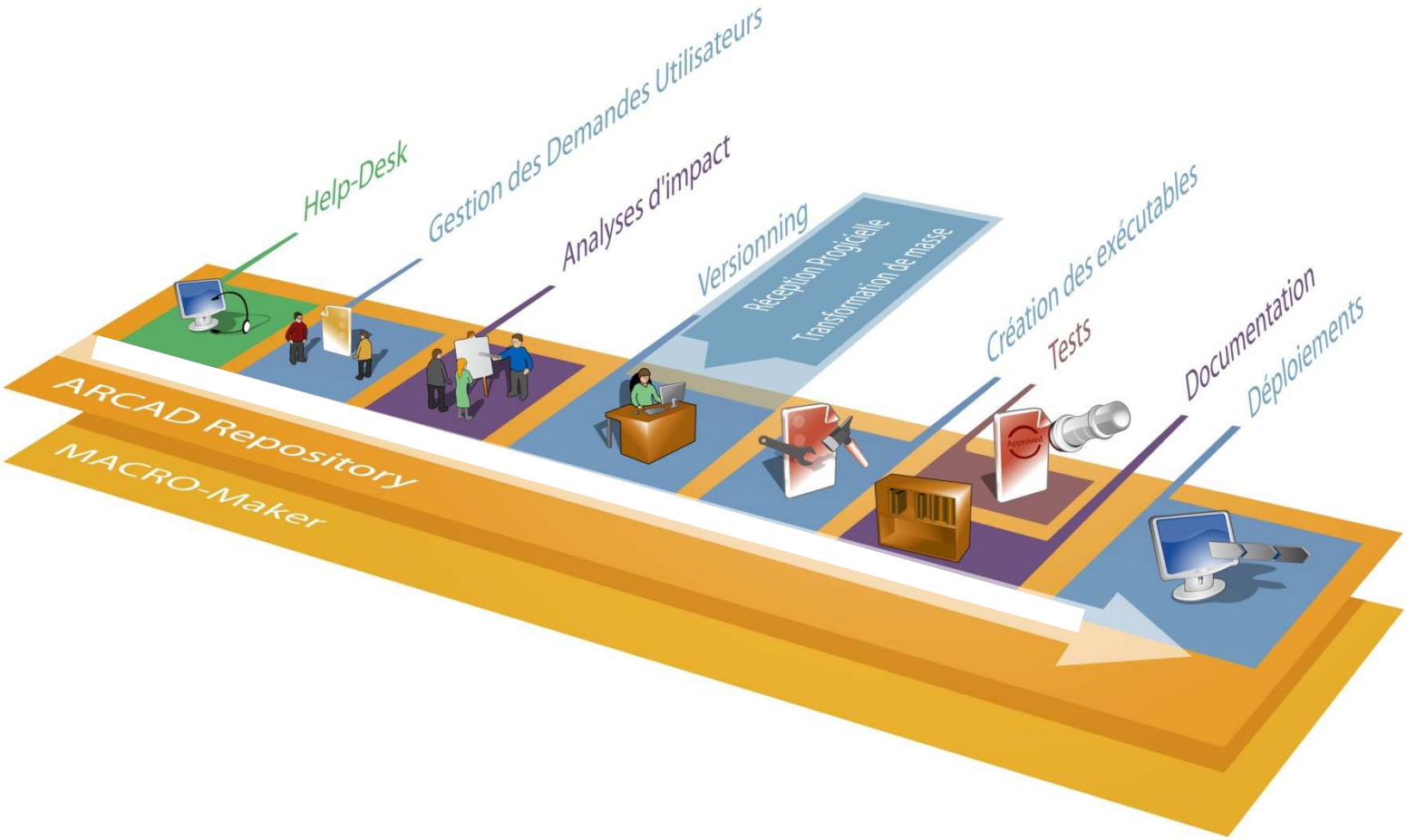
Conclusion

- Une sécurité maximale des processus de changements,
- Un ROI < un an,
- Une qualité de service accrue,
- Une facilité d'intégration de nouvelles ressources,

- Compliance / contraintes règlementaires

Deuxième partie:
Composantes techniques

La chaîne logistique des évolutions logicielles



1- Référentiel

Référentiel : rôle

- Identifier l’ensemble des composants logiciels,
- Constituer (ou reconstituer) une référence unique,
- Regrouper par domaines fonctionnels.

Référentiel : moyens

- Base centrale unique de référencement des composants logiciels,
- Assurer la sécurité et le contrôle logique d’accès,
- Référence protégée.

Différences référentiels de CM et de méta-données

- CM,
  - Dématérialisation du code source,
  - Optimisation de l’espace disque,
  - Sécurité d’accès “par défaut”,
- Méta-données,
  - Simples pointeurs sur les sources,
  - Gestion des dépendances,

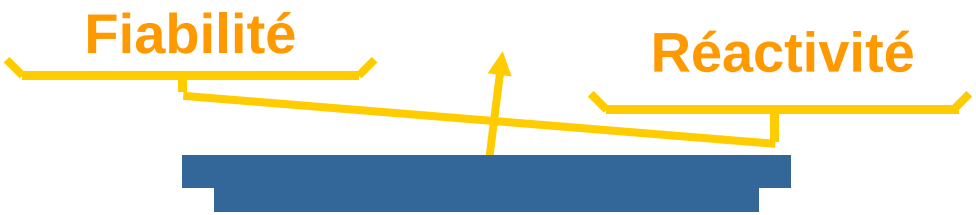
Insertion dans le référentiel

- = Application des rêgles de topologie des composants,
- = Vérification de l’automatisation de la création des exécutables.

2- Versionning

Principes

- Avoir des états stables,
- Connaissance,
- Indispensable à l’archivage,
- Coordonner les projets d’évolution,
- Compatibilité,
- Non régression,
- Maîtriser la « balance ».



Versionning de développement

- Raisonne non régression des sources,
- Indicateurs d’allocation,
- Mécanismes de comparaison/fusion.

Versionning de production

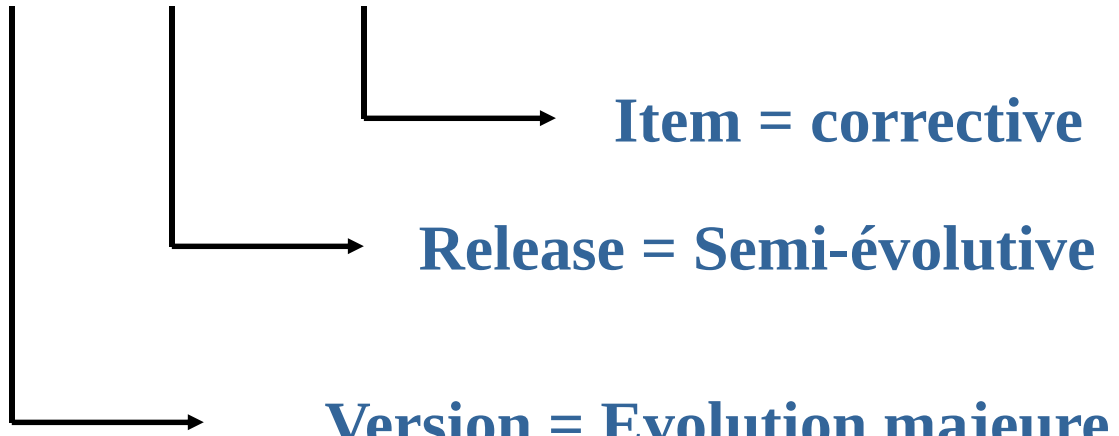
- Sert à labelliser les niveaux d’applications,
- Constitue un langage commun entre:
  - Le support technique,
  - Les études,
  - La production,
  - L’utilisateur.

Versionning de production

- Raisonne “global”,
- Développements internes +,
- Nouvelles versions de progiciels +,
- Données de paramétrage +,
- Middleware +,
- Outils d’administration (scheduler, gestion d’alertes, etc.).

Exemple de numérotation

88.99.A00



Le « branching »

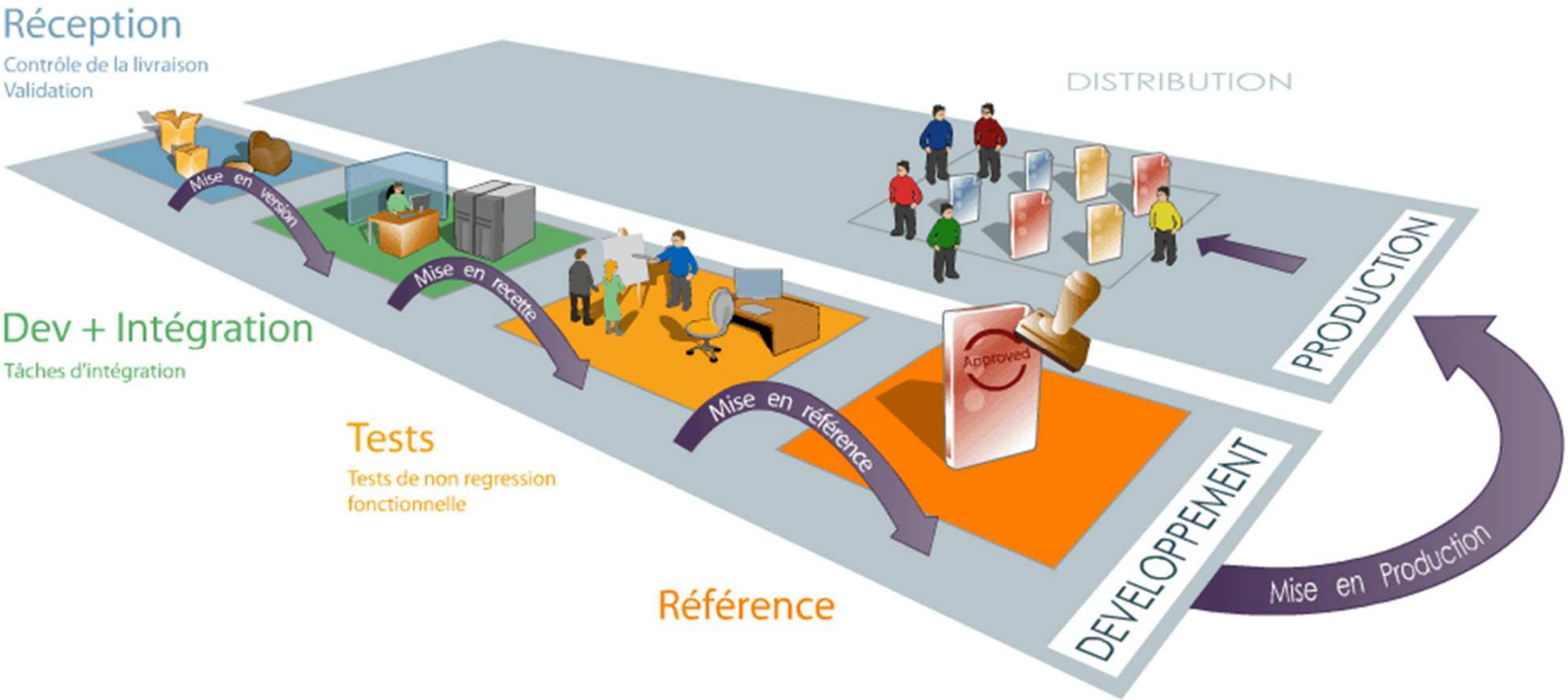
- Permet de gérer des dérivés d’un code source,
- Résoud les problèmes de corrective sur des versions antérieures.

Archivage

- Fait partie intégrante des processus,
- Durant les phases de développement,
- A la mise en production:
  - Archivage “légal”,
  - Reconstitution d’un niveau antérieur d’une application.

3- Espaces de travail

Exemple de configuration



Principes

- Définir des espaces de travail étanches,
- Définir la position physique (machine, partition),
- Définir leur contenu (Σ ou Δ),
- Définir les principes de gestion des données dans ces espaces,
- Définir les règles de sécurité/confidentialité,
- Définir les principes de mise à jour / configuration.

Catégories d’environnement

- Production:
  - Unique ou multiples,
  - Gestion de la sécurité/confidentialité,
  - Base de données utilisateur.

Catégories d’environnement

- Référence:
  - Unique par définition,
  - Protégée,
  - Base de données “semi-vide” si plusieurs environnements de production (=environnement de réplication pour la définition d’une nouvelle production).

Catégories d’environnement

- Test/qualification/pré-production:
  - Font partie intégrante du processus d’homologation,
  - Protégés,
  - Base de donnée “pleine”,

  => définir les principes de rafraichissement des données.

Catégories d’environnement

- Développement:
  - Font partie intégrante du processus de développement,
  - Non protégés,
  - Base de donnée “pleine”,

  => définir les principes de rafraichissement des données.

Catégories d’environnement

- Réception:
  - Servent à “décortiquer” le contenu d’une nouvelle version d’un progiciel,
  - Non protégés,
  - Pas de base de données (hormis des données de paramétrage).

Catégories d’environnement

- Formations:
  - Temporaires,
  - Non protégés,
  - Base de donnée permanente ou créée à la demande avec des données de production.

Conseils méthodologiques

- La démultiplication des environnements
  - démultiplie:
    L'espace disque nécessaire,
    La charge d'administration.

- Un environnement de test ne sert à rien s'il n'est pas utilisé (sic),
  - La clé = données:
    => gestion rigoureuse,

4- Gestion des rôles et responsabilités

Principes de base

- Isolation totale de la production:
  - Séparation physique (machines ou partitions logiques),
  - Elle ne doit jamais contenir de sources,
  - Elle ne doit pas être accessible aux équipes de développement.

Rôles et responsabilités

- Responsable d’application:
  - ⇔Chef produit dans l’industrie,
  - Mandaté par l’administrateur (⇔Officier de sécurité),
  - Coordinateur des grands projets d’évolution,
  - Responsable de la configuration de l’application.

Rôles et responsabilités

- Chef de projet d'une version:
  - Mandaté par le responsable d'application,
  - Gère son équipe de développement (approche ouverte ou hiérarchique),
  - Définit les recetteurs,
  - Déclenche les mises en test,
  - Coordinateur des changements dans un projet d'évolution,
  - Responsable des environnements de développement.

Rôles et responsabilités

- Développeur:
  - Mandaté par le chef de projet de la version (facultatif),
  - Reçoit et réalise des tâches de développement/maintenance,
  - Réalise les analyses d'impact au niveau technique,
  - Alloue les composants nécessaires,
  - Assure la traçabilité des évolutions,
  - Saisit les infos de suivi de projet,
  - A l'issue des tests d'intégration:
    - Récupère les composants nécessitant une adaptation dans l'environnement de développement,
    - Les re-transfère dans le ou les environnements de test.

Rôles et responsabilités

- Administrateur base de données:
  - Mandaté par le responsable d’application,
  - Seul habilité à faire évoluer des composants de type base de données (tables, vues, etc.).

Rôles et responsabilités

- Recetteur:
  - Mandaté par le chef de projet d’une version,
  - Ne fait pas partie de l’équipe de développement,
  - En charge des tests d’intégration,
  - Travaille dans un ou plusieurs environnements de test,
  - Doit traçer les anomalies rencontrées,
  - Doit valider le lot de modifications à l’issue de ses tests,
  - Responsable des environnements de test/qualification.

Rôles et responsabilités

- Logithécaire:
  - Mandaté par l’administrateur,
  - Fait partie de l’équipe de production,
  - En charge de la mise à niveau de la référence,
  - En charge des mises en production,
  - Doit valider le lot de modifications,
  - Responsable des environnements de référence et de production.

Conseils méthodologiques

- Attention: partie sensible:
  - C’est le coté négatif de la GCL perçu par les développeurs.

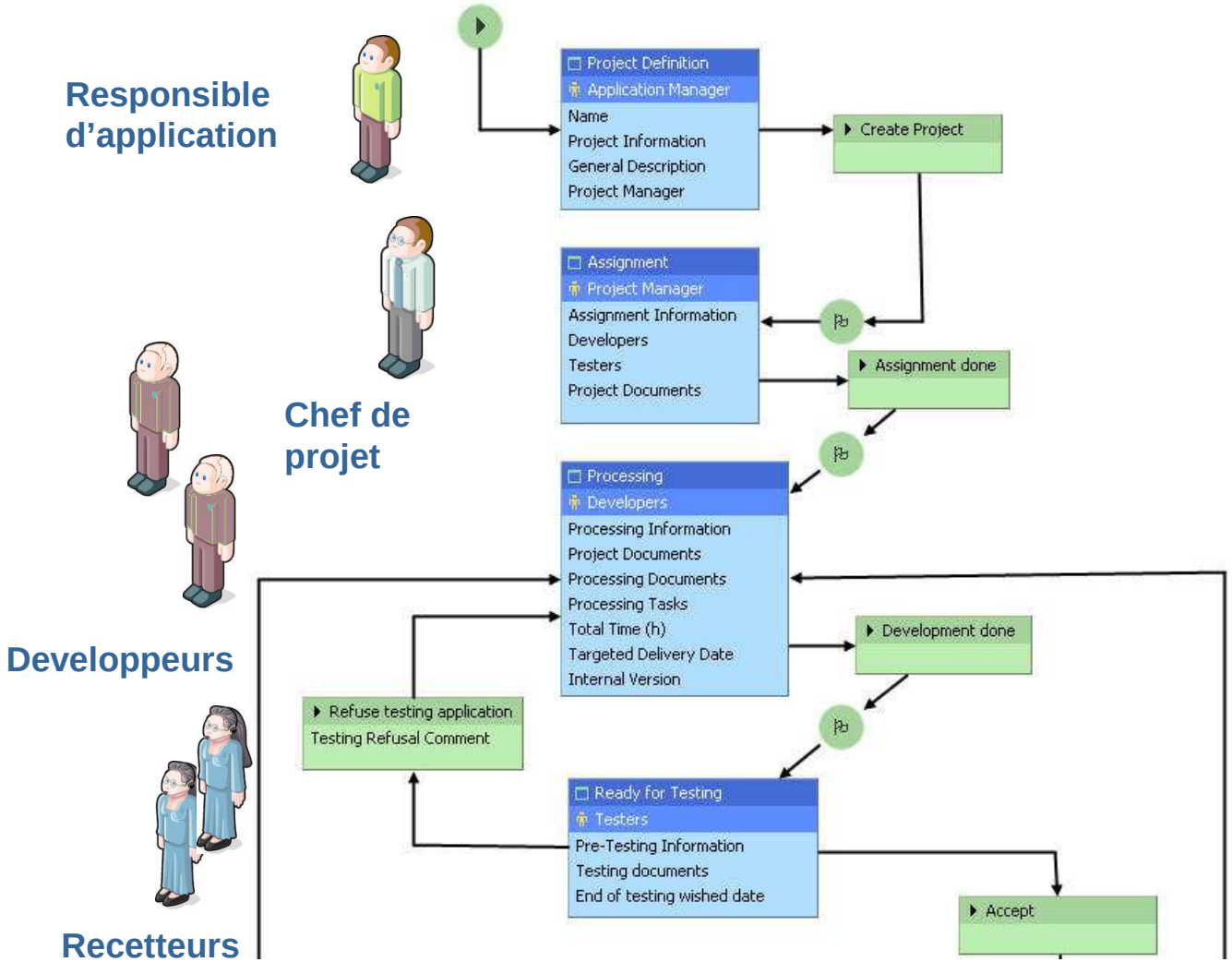
- Par contre: indispensable:
  - Fin de la guerre Production/études,
  - Partage clair des rôles.

5- Workflow

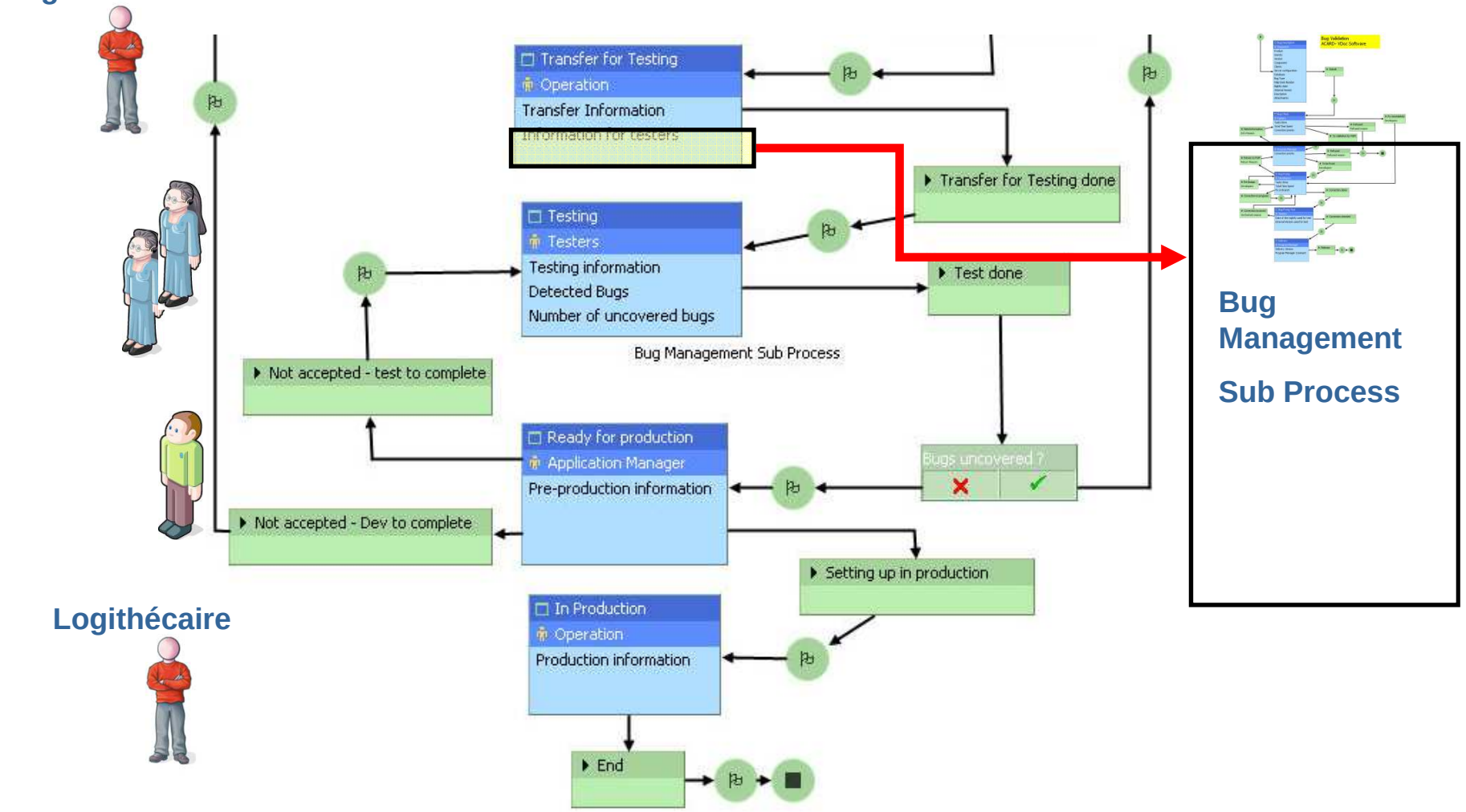
La GCL : une série de process

- Définition des acteurs, des tâches, des changements de contexte,
- Transfert d’information automatique,
- Validations,

Process de gestion de version (1ère Partie)



Process de gestion de version (2ème Partie)

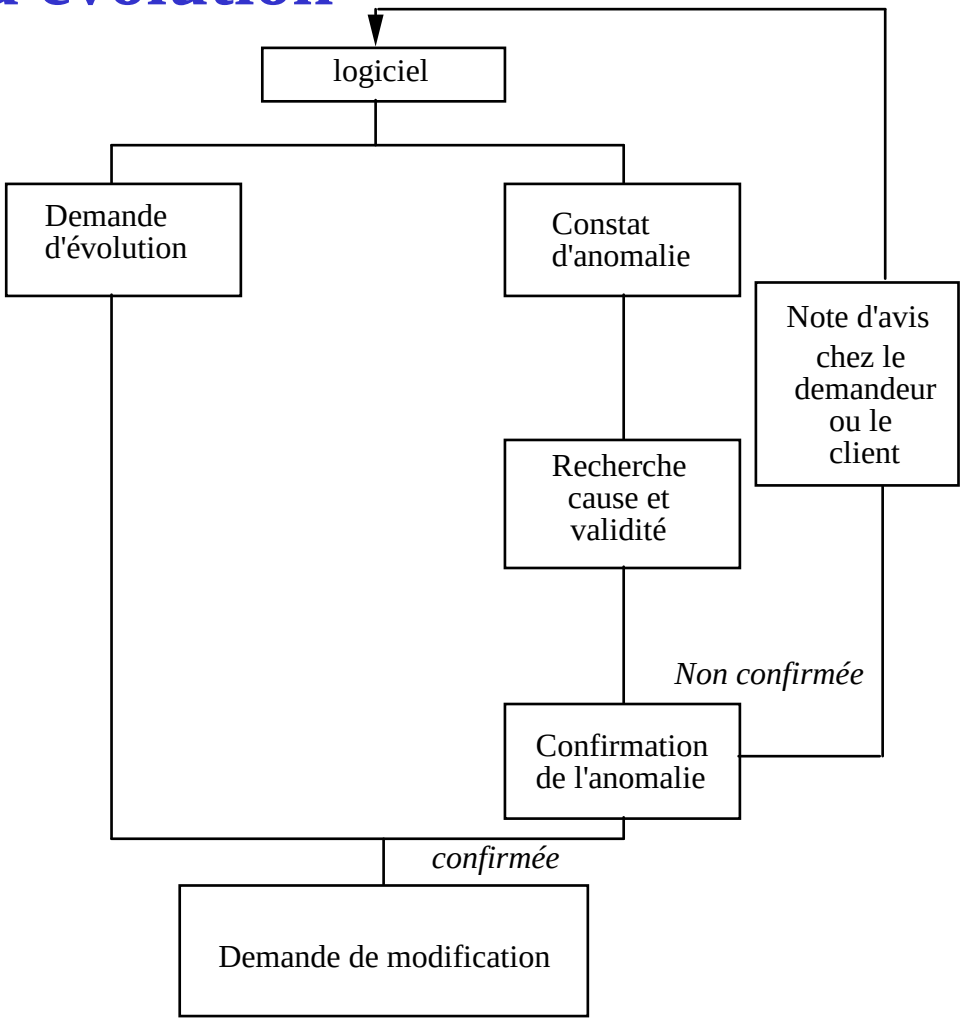


6- Gestion des demandes utilisateurs

Principes

- Formaliser l’ensemble des demandes:
  - Principe de traçabilité,
  - Eviter le “copinage”,
  - Décider des changements importants en comité,
  - Evaluer les charges,
  - Mesurer la fiabilité.

Incident ou évolution



Etablissement d'une demande modification
(extrait de la norme afnor Z 67-130)

Les fiches de tâches

- Renseigne le détail des modifications à effectuer après analyse de l’incident ou de la demande d’évolution,
- Gravité/urgence,
- Catégorie (corrective, évolutive),
- Assignation d’un responsable:
  - Du développement,
  - Du test.

- Gestion de projet:
  - Date au plus tard, au plus tôt,
  - Charge de travail.

Les fiches de tâches

- Reliée aux composants durant les phases de changement,
- Renseigne le détail des modifications apportées,
- Renseigne les infos pour le suivi de projet,
- Subdisivion de la notion de version,
- Le lien entre la technique et l'utilisateur,
- Constitue la mémoire de l'évolution de l'application,
- C'est le "pourquoi ?" de la gestion de versions (qui a fait quoi, quand, pourquoi).

7- Gestion des changements de sources

Principes

- Allocation de composants (“check-out”):
  - Jeton (exclusif ou non),
  - Visible via le référentiel,
  - Copie du composant dans l’espace de travail.

- Gestion des modifications concurrentes:
  - Comparaison/fusion,
  - Reports de modifications.

Rôle des IDE

- L’Integrated Development Environment:
  - Est le socle de base de l’acte de changement,
  - Il fournit des standards de dialogue avec les outils de GCL,
  - Le niveau d’intégration des outils de GCL dans l’IDE doit être le plus élevé possible.

- IDE les plus standards:
  - Eclipse (RDP, RAD, etc.),
  - Visual Studio.

Différences dans l’environnement PC

- La copie complète des sources sur le PC,
- La synchronisation:
  - Remontée des modifications sur le serveur,
  - Redescente des modifications des autres développeurs sur la version.
- Archivage local.

La Gestion des « locks »

- Pessimiste
- Optimiste

8- Fabrication des exécutables (build)

Principes

- Automatiser à 100% la création du code exécutable,
- => prendre en charge toutes les particularités de configuration,

Dans l’environnement IBM i

- Lancement de commandes avant ou après la phase de compilation:
  - Ex: création d’un fichier temporaire ou Override d’un fichier.

- Préservation des attributs et des droits,
- Préservation des données pour les fichiers,
- Gestion de compilations dépendantes:
  - Ex: fichiers logiques, programmes.

Dans les nouvelles technologies

- Outils d’intégration continue :
  - Maven, Buildforge, …

9 - Gestion des tests

Contenu

- Informer les “recetteurs”
- Les mettre dans un environnement dédié
- Avec des données pertinentes…
- Formaliser les anomalies rencontrées

- Validation des lots

- Automatisation progressive

9 - Gestion des mises en production

Contenu

- Allocation des composants,
- Archivage des versions n – 1,
- Transport des composants,
- Application de la politique de sécurité,
- Mise à jour du référentiel,
- Cloture du lot (version),
- Déroulement des procédures manuelles,
- Information des utilisateurs.

Périmètre d'automatisation

- Définir l'ensemble des tâches à réaliser :
  - Sur les composants,
  - Sur les données.

- Discerner les tâches automatisables,
- Pour les tâches non automatisables => écrire des procédures,
- Une tâche non automatisée est une tâche "à risque".

Sécurité des processus

- Mécanismes de reprise en cas d'anomalies :
  - Capacité d'intéraction en temps réel sur le processus.
- Mécanismes de retour à un état stable antérieur :
  - Impossibilité matérielle d'analyse de l'anomalie

  => décision de retour arrière.
- Traçabilité :
  - Logs globales,
  - Logs d'anomalies,
  - Historique des composants manipulés.

Gestion des données

- = partie critique du processus
- Sauvegarde/restauration :
  - Attention aux bases de données larges.

- Mécanismes de reprise spécifiques,

- Nouvelles techno = nouvelles contraintes :
  - Triggers,
  - Contraintes référentielles,
  - Journalisation.

Moyens

- Langages de scripts:
  - CLP,
  - ANT,
  - PERL,
  - Etc.

- Outils dédiés

Conseils méthodologiques

- Penser global (i.e: Σ plateformes),
- Raisonner “éditeur”,
- Viser le 100% d’automatisation,
- Si 100%, alors planifiable de nuit.

10- Gestion des déploiements

Multi-environnements de production

- Impossibilité de faire migrer tous les environnements de production simultanément,
- Obligation de faire de la corrective sur des plusieurs niveaux simultanément.

Notion de cumulative

- = Σ des versions entre le niveau d'un site et le niveau à envoyer:
  - Constitution automatique de lots agrégés,
  - Fusion des process de reprise.

Notion de « PTF » ou « Patch »

- = un lot de composants corrigés,
- Le lot est constitué à partir de la tâche en amont,
- Si un composant a été modifié pour plusieurs tâches, les composants des tâches sous-jacentes doivent être embarqués,
- Compatible avec un certain niveau de version:
  - => gestion rigoureuse des niveaux (particulièrement des niveaux de base de données).
- Les cumulatives relivrent les PTF déjà livrées,

- La petite porte qui peut faire des courants d’air.

Multi-sites avec maîtrise des configurations cibles

- Notion de groupes d’environnements,
- => Découper les tâches:
  - Préparation,
  - Envoi,
  - Installation.
- Permet de synchroniser les installations,
- Rollback multi-sites,
- Accusés d’installation = pilotage central.

Multi-sites sans maîtrise des configurations cibles

- => Process d’installation flexibles,
- Mise à jour manuelle des accusés d’installation,
- Multi-environnements sur site distant:
  - => mises en production “locales”.

Un peu de culture

- www.cmcrossroads.com,
- LE site de référence sur la GCL,
- www.cmcrossroads.com/cgi-bin/cmwiki/view/CM/,
- - Un wiki sur le sujet,
- www.cmresourceguide.com/,
- Un guide de référence.

common Romandie

L'association romande des utilisateurs de systèmes IBM

Le bon réflexe …

common Romandie

www.common-romandie.ch

info@common-romandie.ch

A bientôt …

Merci pour votre participation !