

Cycles de vie : Management

Deux processus distincts

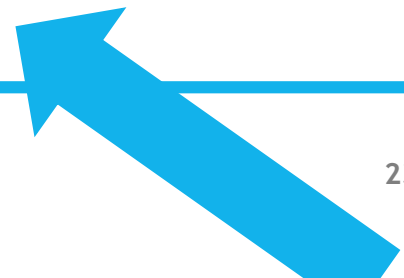
Développement logiciel

Processus Management

- Planifier le travail
- Planifier les livraisons
- Allouer les ressources
- Gérer le budget, les coûts
- Surveiller l'avancement des travaux
- Gérer les risques

Processus développement

- Créer le logiciel selon les demandes du client (requirements)
- Gestion SDLC et évolutions
 - Environnements
 - Configuration management
 - Change control



Environnements de développement

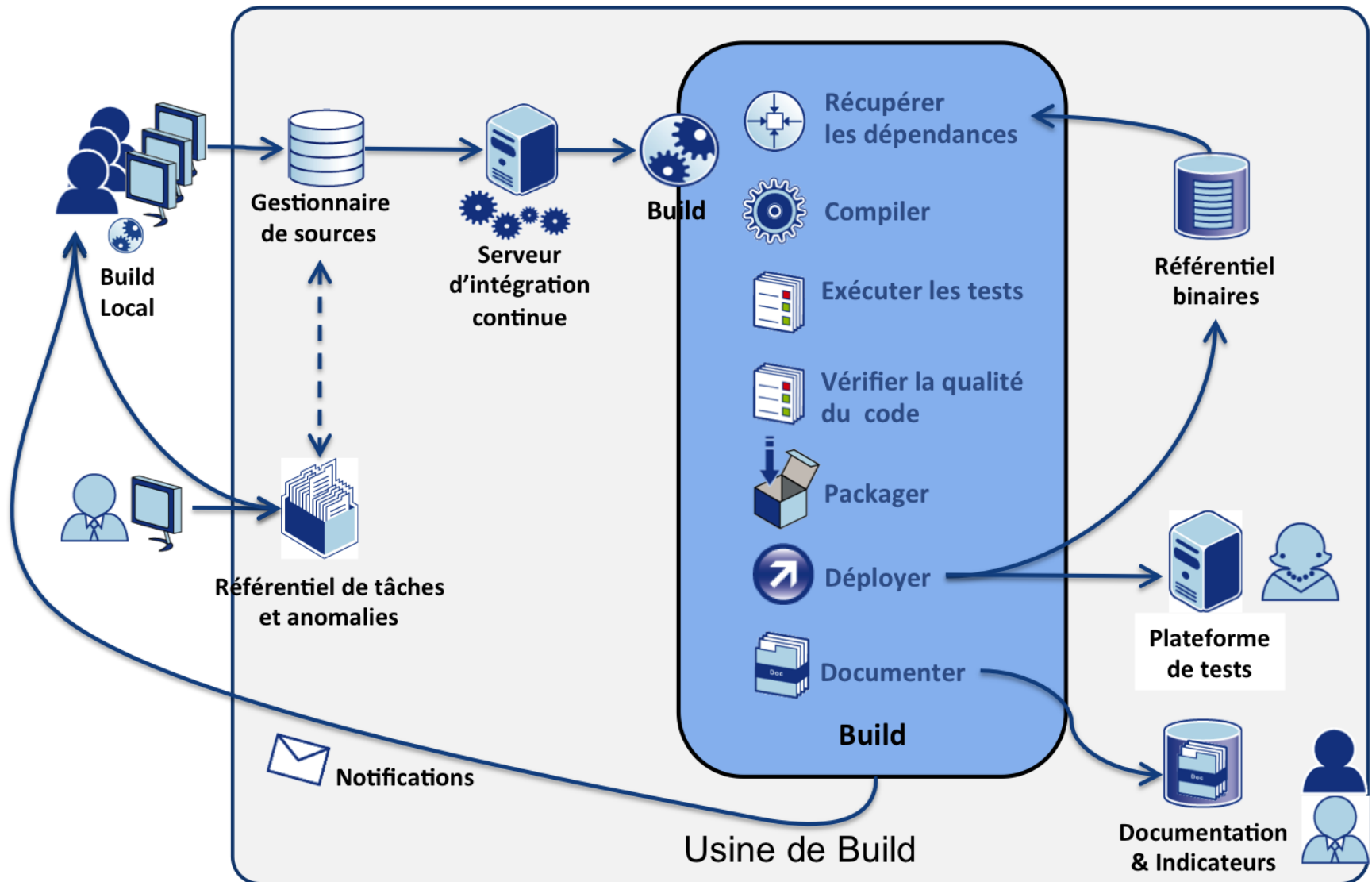
- Outils de développement
 - Eclipse
 - pgadmin
- Ne pas oublier
 - Serveur de fichiers
 - Authentification...
- MAIS AUSSI et surtout:
 - Jenkins
 - SVN
 - studentittools
 - Jacoco
 - Browser
 - Check styles
 - Trello...

Usine logicielle

- Un ensemble structuré de logiciels qui fonctionnent ensemble pour permettre le développement
 - Non seulement outils de programmation, compilation, tests, déploiement et sauvegarde (gestion de sources...) mais aussi
 - de gestion de documents
 - de gestion du cycle de vie d'un projet, de suivi des activités
 - de gestion des licences
 - Tableaux de bord.
- ➔ Outils connectés entre eux

Usine logicielle IRIS sa

Autre exemple



Qualité du code



- Software qui vérifie la qualité du code
- Exécute au moins quatre outils :
 - Checkstyle
 - PMD
 - Findbugs
 - Corbertura

Outils de développement

Jira Software



- Créer des user stories et des tickets
- Planifier des sprints
- Affecter des tâches
- Définir des priorités et états pour les tâches
- Visibilité totale

ARHS
Developments
Belgium – client
Proximus

Stage observation
Ronsmans Thomas
2016-2017

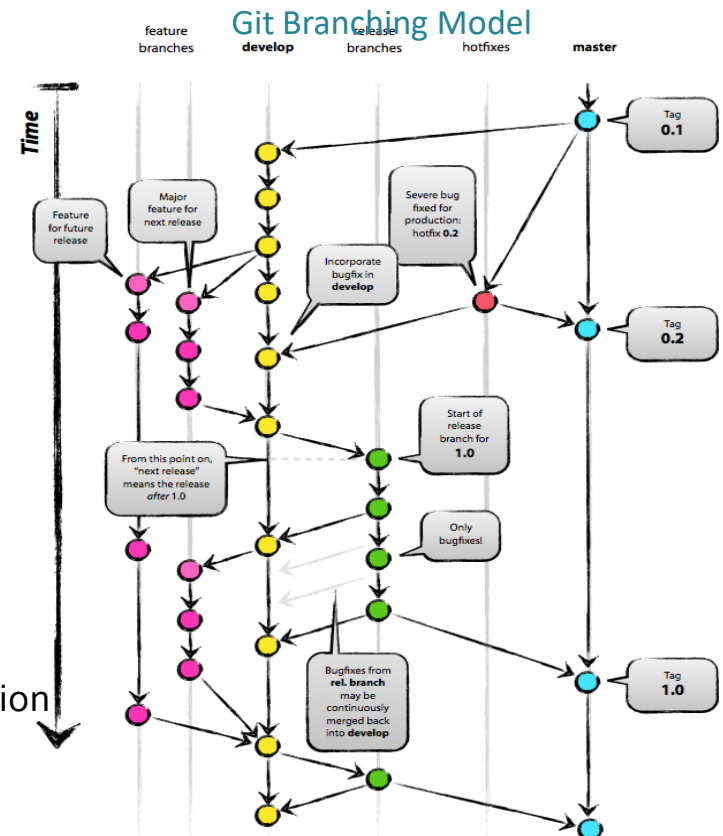
Outils de développement

Git

- Branching model :
 - Branches
 - Commits
 - Updates
- Séparation du travail
- Chaque membre d'équipe peut travailler indépendamment sur sa version
- Pas de problèmes générés lors de la mise en commun

GitLab

- Repository central : contient le code source de l'application
- Composé de plusieurs directory
 - Directory principal (Projet)
 - Plusieurs autres directory => Modifications des programmeurs



ARHS Developments Belgium – client Proximus
Stage observation Ronsmans Thomas 2016-2017



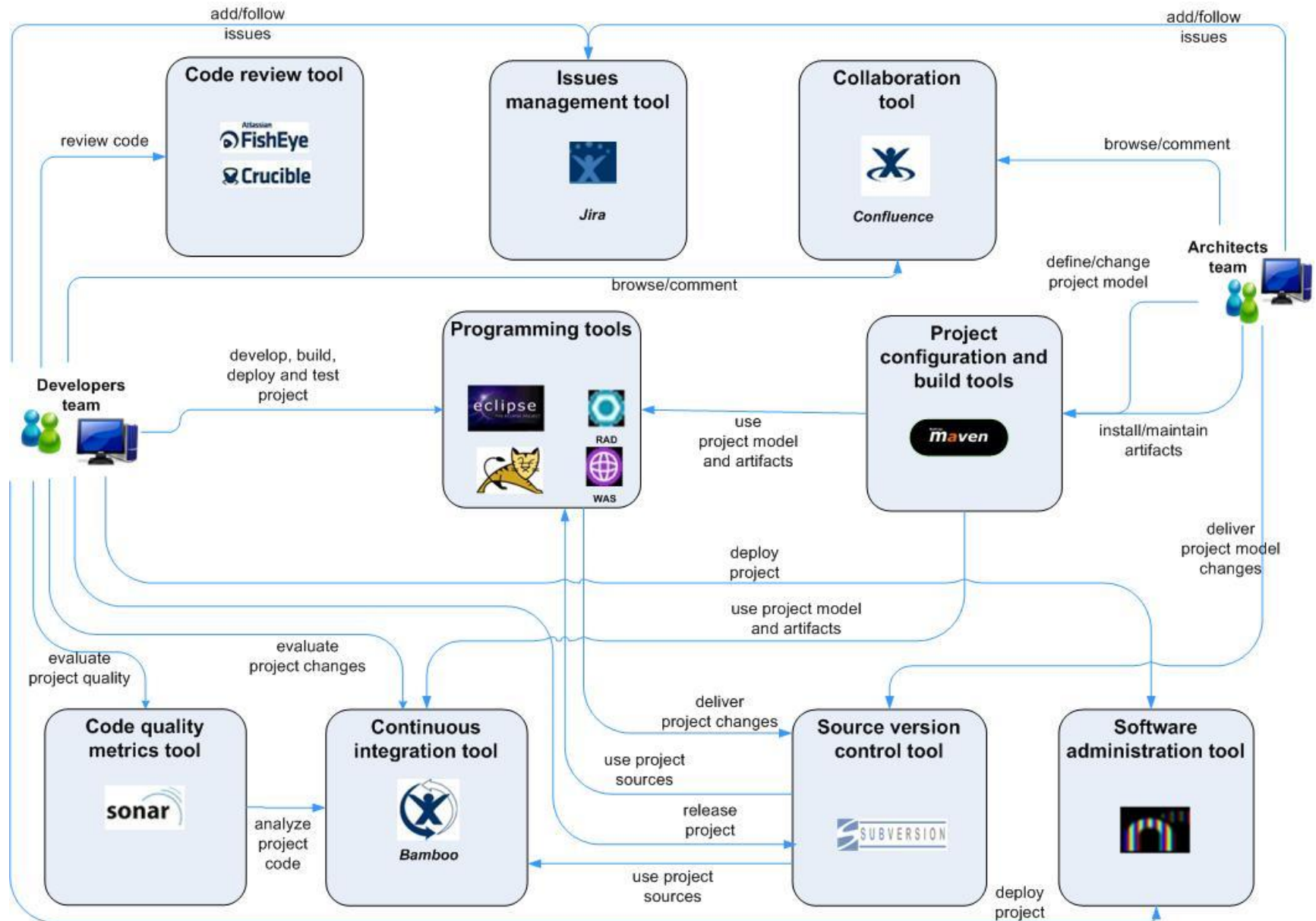
Jenkins

Jenkins :

- Outil open source
- Intégration continue
- Rebuild le package lors d'un évènement
 - Commit
 - Update
 - ...
- Test d'intégration et unitaires automatiques
- Personnalisable

ARHS Developments Belgium – client Proximus

Stage observation Ronsmans Thomas 2016-2017



Gestion de configuration

"Software Configuration Management is a disciplined approach to managing and controlling the evolution of software development and maintenance practices and their software products as they change over time" [© Ovum Ltd 1998](#)

Gestion de configuration

- Gérer la description technique d'un système et de ses divers composants ; gérer **l'inventaire de tous les éléments de la configuration** (configuration items)
- Gérer **l'ensemble des modifications** apportées au cours de l'évolution du système (→ change control ; contrôle des changements apportés au système)
- Gérer **les versions** (releases)

Pourquoi ?

Gestion de la complexité

- Nombreux développeurs
- Sous-systèmes interdépendants
- Différents langages
- Évolution des systèmes

Pour avoir un logiciel

- Fiable, robuste,
- Maintenable ...

Voir qualité du logiciel - chapitre 4

Exemples d'activités :

- Contrôle de versions
- Définition du style de codage
- Tests de non-régression
- Contrôle des changements...

Objectifs /1

Quels sont les objectifs poursuivis ?

- Que les demandes de changement et les rapports de problèmes (bugs) soient gérés dans le système
 - Utilisation d'un programme de gestion de bugs
 - Vérification que chaque demande de client soit introduite dans le logiciel.

Objectifs /2

- Que des bugs fixés (résolus) précédemment ne réapparaissent pas mystérieusement
 - Si cela arrive, le bug est rouvert et est marqué dans le programme de gestion de bugs
 - Mesure : comptage du nombre de bugs rouverts dans une version : il devrait être égal à 0

Objectifs /3

- Que les activités de développement soient sous contrôle
 - Mesure : il est possible de produire un état des développements en moins d'une demi-journée.
- Que la documentation soit à jour et correcte
 - Ceci est difficile à atteindre.
 - Une solution est de maintenir un numéro de version et la date de dernière mise à jour du document dans chaque document produit.
 - Mesure : chaque document doit avoir un numéro de version et la date lui correspondant.

Objectifs /4

- Que tous les « configuration items » soient identifiés
- Que chaque release puisse être construite précisément
- Que les différences entre deux releases soient facilement identifiées
- **Qu'une release envoyée à un client soit complète** (pas de fichier manquant).

1. Inventaire des éléments et leur gestion

Inventaire

- Informations
 - Documentation
 - Méthodes de travail
 - Spécifications techniques
 - Définition des bases de données
 - Scénarios de tests
 - Enregistrement des tests
 - Dossiers de validation
 - Composants externes
 - Dossiers sécurité
- Code source
- Environnement
- Exécutables
- Ressources nécessaires au développement
- Fichiers INI
- Scripts pour créer la base de données
- ...

Inventaire (2)

- Organisation des informations
- Conventions de nommage
- Nomenclature de versions de fichiers
- Documents relatifs à l'assurance et au contrôle qualité
- Standard
 - Un standard est une **spécification technique établie par l'équipe** et qui **repose sur les acquis** de la science, des technologies et de l'expérience.
- ...

1.1. Organisation des informations

Organisation

- Définir l'arborescence pour stocker les différents items de configuration et le lien avec le numéro de version du logiciel:
 - les fichiers de documentation
 - les exécutables
 - les jeux de tests
 - les enregistrements de l'exécution des tests...
- Définir la structure des numéros de version du logiciel
- Définir la structure des documents

Documents relatifs à l'assurance et au contrôle qualité










- Voir leur place dans l'organisation des informations
- Faire le lien avec le plan qualité (voir chapitre 4)

Nomenclature

Exemple : n.m.ooooo.p

- n.m : release majeur
- ooooo : numéro de build
- p :
 - par défaut, valeur 0
 - si branchement, représente le numéro séquentiel de branchement sur le numéro de build

Exemple Microsoft

 bfsvc.exe	16/07/2016 13:42	Application	60 KB	10.0.14393.0
 hh.exe	16/07/2016 13:42	Application	18 KB	10.0.14393.0
 notepad.exe	16/07/2016 13:43	Application	238 KB	10.0.14393.0
 winhlp32.exe	16/07/2016 13:42	Application	10 KB	10.0.14393.0
 write.exe	16/07/2016 13:42	Application	11 KB	10.0.14393.0
 splwow64.exe	22/11/2016 23:55	Application	128 KB	10.0.14393.351
 explorer.exe	04/03/2017 08:03	Application	4,565 KB	10.0.14393.953
 regedit.exe	04/03/2017 07:18	Application	313 KB	10.0.14393.953
 HelpPane.exe	28/03/2017 07:14	Application	953 KB	10.0.14393.1066

Tous les exemples sont extraits des bonnes pratiques développées dans la société IRIS sa de 1997 à 2009.

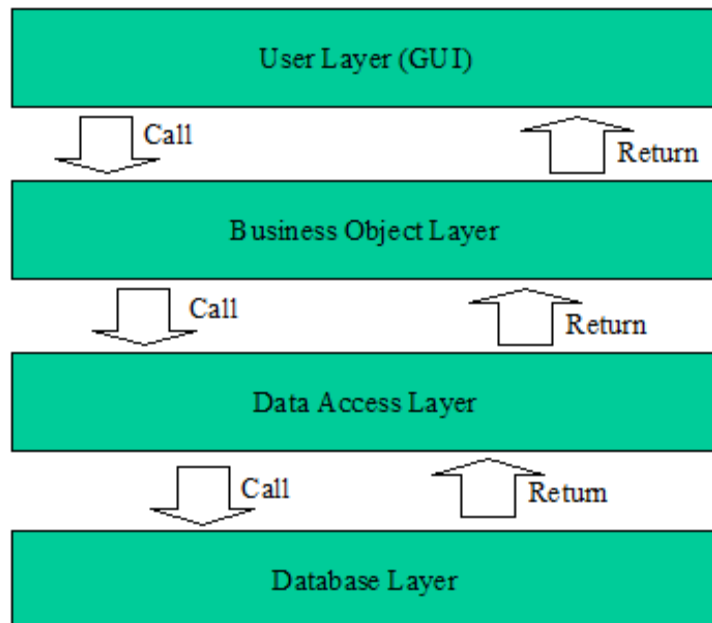
1.2. Standards

Standards d'architecture

1 Program Structure Standards (physical & logical)

1.1 Logical

In keeping with standard software engineering principles, all new systems must be developed using a layered architecture.



The advantages of this approach are well documented in most software engineering texts.

Objects and functions within one layer are only allowed to access objects and functions in the layer immediately below.

This greatly improves portability and re-use of objects - even more than that, though, is the fact that it makes the code much easier to understand.

1.1 *Physical*

1.1.1 Code Modules

Systems should be divided into individual self-contained modules and, ideally, most modules should map directly onto one business or data access object. There will also be a number of utility modules containing functions, base classes etc. that do not map onto an object.

1.1.2 Installation-specific information

When required, a system may need to retrieve installation-specific information from an additional file. This information should be stored in the application INI file.

Standards de programmation

1 Coding Standards

1.1 General

1.1.1 Language

The language to use for comments and identifiers is English.

1.1.2 Labels

To allow easy portability between languages, all text should be stored away from the source code.

See the documentation for usage/implementation details.

1.1.3 Naming convention for all development tools

All names (functions, objects...) have a name beginning with an uppercase letter and other are in lowercase. If the name is a concatenation of several words each one has its first letter in uppercase (e.g.: ThisIsAnExample).
Name must be significant.

1.1.4 Generalities

Avoid the use of hardcoded constant. Use const names, eg: const int DayPerWeek = 7;
Never use duplicated or redundant code with small variations. Design and factorize.

1.2 *Coding Standards for each language used*

...

Standards de documentation

5 Documentation Standards

We are managing the documentation, version by version.

5.1 *Code Documentation*

A common problem with code level documentation is that it is too detailed. Consequently, ensuring that the documentation is up-to-date and accurate is a laborious and time-consuming problem. To combat this, documentation for ICP projects should be at a slightly higher level. The following table outlines what needs to be included. Low level documentation about individual code segments should be included in the code where it is most useful.

The technical documentation must cover:

- the functions and classes that are exported
- dependencies
- files, database tables, registry keys.

Code documentation should be compatible with Doxygen syntax.

5.2 Database Documentation

The documentation of the database should not be limited to the documentation of the data structure as more and more database systems include the notion of triggers and stored procedures. However, as this notion is not a component of the ICP databases yet, the documentation will include the following items :

- An entity-association schema (which refers to the entity-association model). This powerful model allows to represent the structure of data within tables, the relations between tables, the integrity and coherence constraints.
- An additional document will give the datatype of each field, its length and a short description of its content.

One thing that should never appear in code documentation is large reproductions of the code itself. Developers work hard to avoid code duplication within their systems so it should be avoided in the documentation also – it is very difficult to maintain.

Templates

Les standards couvrent aussi le template de document.
Tous les documents doivent avoir le même modèle.

Exemple : [CHANGECT ProductName Template.pdf](#)

1.3. Éléments des méthodes de travail

Exemples :

- Traitement des requirements au long du cycle de vie
[SOFTWARE-development.docx](#)
- Niveaux de tests
[TESTING-Levels.docx](#)

2. Gestion des changements

3. Gestion des versions

Versions (1)

- Démarrer le développement d'une nouvelle version :
Quand ? Comment ? Pourquoi ?
 - Où sauver l'information?
 - Liste des demandes prises en compte
- Préparer une version :
 - Changement de / nouvelle technologie?
 - Spécifications (correspondant aux nouvelles demandes)
 - Préparation des environnements
 - Préparation du document de contrôle des changements

Versions (2)

- Pendant le développement de la version, **contrôle des changements** :
 - Suivi
 - Nouveau code
 - Nouveaux tests
 - Revue du code
- Préparation du déploiement
- En fin de changement, décommissionning
 - Mettre à jour les documents
 - Enregistrer la version dans le document de configuration.

[SCM Releases.doc](#)

Questions - réponses