|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cahier de tests | | | | |
| **Projet :** | | Maquette du cahier tests pour les PFE | | |
| **Emetteur :** | | N. Ragot | | Coordonnées :  EPU-DI |
| **Date d’émission :** | | 22/03/2010 | | |
| **Validation** | | | | |
| Nom | | Date | Valide (O/N) | Commentaires |
|  | |  |  |  |
|  | |  |  |  |
|  | |  |  |  |
| **Historique des modifications** | | | | |
| Version | Date | Description de la modification | | |
| 00 | 03/2010 | Version initiale | | |
|  |  |  | | |
|  |  |  | | |
|  |  |  | | |
|  |  |  | | |
|  |  |  | | |
|  |  |  | | |



École Polytechnique de l’Université de Tours

64, Avenue Jean Portalis

37200 TOURS, FRANCE

Tél. +33 (0)2 47 36 14 14

[www.polytech.univ-tours.fr](http://www.polytech.univ-tours.fr)

**Département Informatique**

Sommaire

[Cahier de tests 5](#_Toc257036361)

[1. Introduction 5](#_Toc257036362)

[2. Tests unitaires 5](#_Toc257036363)

[2.1. Démarche 5](#_Toc257036364)

[2.2. Planification 5](#_Toc257036365)

[2.3. Description des tests 5](#_Toc257036366)

[2.4. Rapports de tests 6](#_Toc257036367)

[3. Tests d’intégration 6](#_Toc257036368)

[3.1. Démarche 6](#_Toc257036369)

[3.2. Planification 6](#_Toc257036370)

[3.3. Description des tests 6](#_Toc257036371)

[3.4. Rapports de tests 6](#_Toc257036372)

[4. Tests de validation du système 6](#_Toc257036373)

[4.1. Démarche 6](#_Toc257036374)

[4.2. Planification 6](#_Toc257036375)

[4.3. Description des tests 6](#_Toc257036376)

[4.4. Rapports de tests 6](#_Toc257036377)

[ANNEXE 1 : Fiches de tests unitaires 7](#_Toc257036378)

[ANNEXE 2 : Fiches de tests d’intégration 8](#_Toc257036379)

[ANNEXE 3 : Fiches de tests de validation du système 9](#_Toc257036380)

# Cahier de tests

1. Introduction

Précisez-ici de façon très générale les types de tests mis en œuvre ainsi que l’organisation du document.

Rappels : Les tests sont « toujours » conçus dans l’ordre inverse de leur utilisation :

* On conçoit les tests de validation fonctionnelle en phase d’analyse fonctionnelle
* En phase d’analyse technique on conçoit les tests d’intégration
* Avant la programmation on élabore les tests unitaires

Il existe différentes natures de tests (non exclusives) :

* boîte blanche (analyse « pas à pas » du déroulement de l’algorithme implémenté) ou boîte noire (vérification des sorties par rapport aux entrées)
* non régression (reprise d’un test ou de résultats précédents sur une nouvelle version de la fonctionnalité)
* performances (calcul de l’espace mémoire, du temps d’exécution…)
* robustesse (entrées ou état du système atypique)
* ergonomie (mesures qualitatives par évaluation d’utilisateurs, …)
* sécurité
* etc.

1. Tests unitaires

Objectif : mettre en évidence les problèmes pour chaque composant/fonctionnalité atomique. On est plutôt ici sur des tests types boîte blanche puis boîte noire pour valider les E/S (en supposant les préconditions remplies ou non, i.e. traitement des erreurs/exceptions).

Attention : vous devez bien faire la différence ici entre les tests effectués lors du développement (qui permettent d’élaborer le composant pour faire « en gros » ce qu’on lui demande : débogage, vérification des sorties) et les tests unitaires qui vont, à partir d’un jeu de test prédéfini, vérifier précisément le comportement du composant vis-à-vis des différents types d’entrées (entrées standards, entrées problématiques, cas d’erreurs prévisibles…).

* 1. Démarche

Description de l’environnement de test (qui fait quoi, sur quelles machines, avec quelles versions des fonctionnalités…), de la procédure mise en place (historisation, traçabilité, reporting, suivi, contrôle), des outils utilisés pour l’automatisation, etc.

* 1. Planification

Précisez pour chaque fonctionnalité ou ensemble de fonctionnalités/tâches la période pendant laquelle les tests seront effectués et le temps qui sera alloué à ces tests.

* 1. Description des tests

Sert de support à l'exécution des tests. Pour chaque fonctionnalité testée, donner la liste des tests à effectuer en précisant sa nature, les données en entrées ainsi que les sorties/modifications du système attendues et à valider.

* + 1. Fonctionnalité xxx
  1. Rapports de tests

L’objectif de cette partie est de présenter de façon synthétique le résultat de chacun des types de tests. On pourra par exemple présenter un tableau avec la liste des tests réussis, la liste de ceux qui ont échoués avec les recommandations correspondantes. Les détails sur le déroulement et les résultats des tests seront fournis en annexe avec les fiches de tests (cf. annexes).

1. Tests d’intégration

Objectif : Détecter les erreurs d’interfaçage entre les composants logiciels. On vérifie donc l’enchaînement des E/S et la validité de ces dernières en sortie de traitement, mais aussi la gestion des erreurs et la propagation des exceptions. On peut commencer à faire ici de la non régression, de l’évaluation de performances…

RQ : peut intégrer un test de recette provisoire suite au développement

* 1. Démarche
  2. Planification
  3. Description des tests

Idem qu’auparavant mais comme il faut tester ici l’interaction entre plusieurs fonctions, la description des tests consistera souvent en la définition de scénarios.

* + 1. Fonctionnalité xxx
  1. Rapports de tests

1. Tests de validation du système

Objectif : Rechercher les erreurs de haut niveau : conception, omissions et non conformités par rapport au cahier de spécifications, fonctionnement dans l’environnement final…

RQ : peut intégrer une recette finale faite par le client

* 1. Démarche
  2. Planification
  3. Description des tests

Il s’agit essentiellement ici de scénarii simples à complexes qui doivent notamment prendre en compte la communication avec des systèmes extérieurs (notamment au niveau des entrées, événements, utilisateur…).

RQ : s’il ya une recette ici, les scénarii sont en général fournis par le client. On peut aussi laisser le logiciel en test chez le client.

* + 1. Fonctionnalité xxx
  1. Rapports de tests

RQ : Dans le cas d’une recette faite par le client, celui-ci devra remplir des fiches de tests (celles fournies en annexes et adaptées (ou bien d’autres à sa guise) et signaler les erreurs ou écarts par rapport aux spécifications.

# ANNEXE 1 : Fiches de tests unitaires

On donnera ici les fiches présentant le résultat du déroulement des tests (test OK ou KO, observations en cours de tests, etc.). Le n° xx permet d’identifier le test effectué et yy permet d’identifier le nombre de fois où il a été effectué (incrémenté à chaque nouvelle vérification).

|  |  |  |
| --- | --- | --- |
| **Fiche de test unitaire** | | **Date :** |
| **Projet :** | | **Auteur :** |
|  | | |
| **Identification du test :** fonctionnalité/type de test | | **N° :** xx / yy |
|  | | |
| **Actions :** | **Résultats attendus :** | **OK / KO** |
| 1. Vérification de l’algo | 1. | 1. |
| 2. Compilation | 2. | 2. |
| 3. Test avec les entrées : … | 3. | 3. |
| 4. | 4. | 4. |
|  |  |  |
|  | | |
| **Résultats obtenus et commentaires :** | **Actions correctrices envisagées :** | |
| 1. | 1. | |
| 2. | 2. | |
| 3. | 3. | |
| 4. | 4. | |

# ANNEXE 2 : Fiches de tests d’intégration

Adapter ici les fiches de l’annexe 1.

# ANNEXE 3 : Fiches de tests de validation du système

Adapter ici les fiches de l’annexe 1.