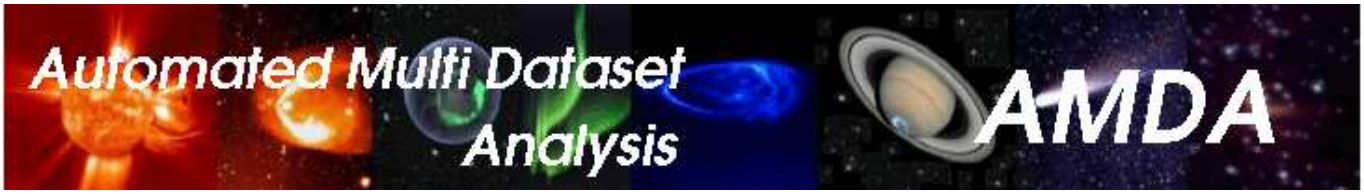
Projet : AMDA



PLAN ET DOSSIER DE TESTS

Rédigé par : *Coordinateur* : *Thomas VERBEKE*	Diffusé à : CNES et IRAP
Approuvé par : *Coordinateur* : *Thomas VERBEKE*	

LISTE DES MODIFICATIONS DU DOCUMENT

Vers.	Date	Paragraphes modifiés	Description des modifications
01.0	23/11/2010	tous	Création du document

SOMMAIRE

1 INTRODUCTION

1.1 OBJET DU DOCUMENT

L'objectif de l'activité de tests est de vérifier et démontrer la conformité du logiciel aux spécifications validées. Dans le cas d'AMDA-NG les spécifications découlent des cas d'utilisation décrits dans MU-AMDA-001

Le présent document décrit l'activité de tests mise en œuvre par l'équipe projet AKKA Technologies dans le cadre du projet AMDA-NG

Il est constitué par le plan de tests, qui décrit la stratégie de tests, le déroulement global de l'activité de tests, ainsi que son suivi.

Le document est complété au fur et à mesure de l'avancement du projet par un dossier de tests avec les descriptions des tests et les résultats obtenus.

1.2 DOMAINE D'APPLICATION

AMDA-NG est composé de quatre modules principaux

- une IHM web
- un cœur applicatif
- un fournisseur de web services
- un serveur de données (DD serveur)

Dans le cadre de ce document, seuls les trois premiers items sont concernés par les tests.

Le plan de tests est appliqué dans le cadre de tests unitaires, d'intégration et de validation.
La non régression est assurée par l'application de tous les tests définis dans les trois précédentes catégories.

2 STRATEGIE DE TESTS

La stratégie de test a pour objectifs d'assurer que tous les modules sont fonctionnellement valides individuellement (tests unitaires), qu'ils interagissent correctement entre eux selon les interfaces définies (tests d'intégration), et qu'ils fournissent les services de haut niveau demandés (tests de validation)

L'organisation mise en place pour y répondre est la suivante :

- Sont testés tous les modules ou classes écrits dans le cadre de la nouvelle IHM, du cœur applicatif et du fournisseur de web services.
- Les tests unitaires sont effectués au plus tôt de la fin du développement d'un module ou d'une classe.
- Les tests d'intégration doivent être fait en continue (selon une périodicité à définir) compte tenu du nombre de développeurs intervenant sur le projet.
- Les tests de validation sont exécutés à la fin de la réalisation de tous les éléments.
- Les tests sont automatisés au maximum pour être réalisés fréquemment, sur tous les environnements cibles. Afin de répondre à cette exigence, SELENIUM est mis en place pour les tests liés à l'IHM et HUDSON est utilisé pour assurer les tests d'intégration.
- Les tests seront réalisés sur les machines de développements situés au CESR.
- Chaque développeur est responsable des tests unitaires des sources dont il est l'auteur. Les tests d'intégration et de validation sont réalisés ou pilotés par le chef de projet CESR.

Remarque :

Dans le cadre du projet, les tests unitaires et les tests d'intégration seront automatisés (CPPUNIT, SELENIUM). Le temps de déroulement des campagnes de tests correspondantes sera donc faible. En contrepartie, l'application étant essentiellement graphique (IHM) les tests de validation seront essentiellement réalisés avec l'outil SELENIUM mais certains tests non automatisables pourront rester manuels. Le temps de validation est donc directement lié au nombre de scripts à dérouler.

2.1 PRINCIPES GENERAUX

Le modèle conceptuel de la gestion de projet est le modèle itératif de type agile (à confirmer)

L'activité de tests est un processus transversal se déroulant tout au long du cycle de développement du produit logiciel.

La stratégie de tests est étudiée dès le début de la phase de conception, puis elle est complétée et mise à jour à chaque phase du projet.

Les types de tests effectués par l'équipe projet sont décomposés en fonction de leurs niveaux :

- Tests unitaires
- Tests d'intégration
- Test de validation

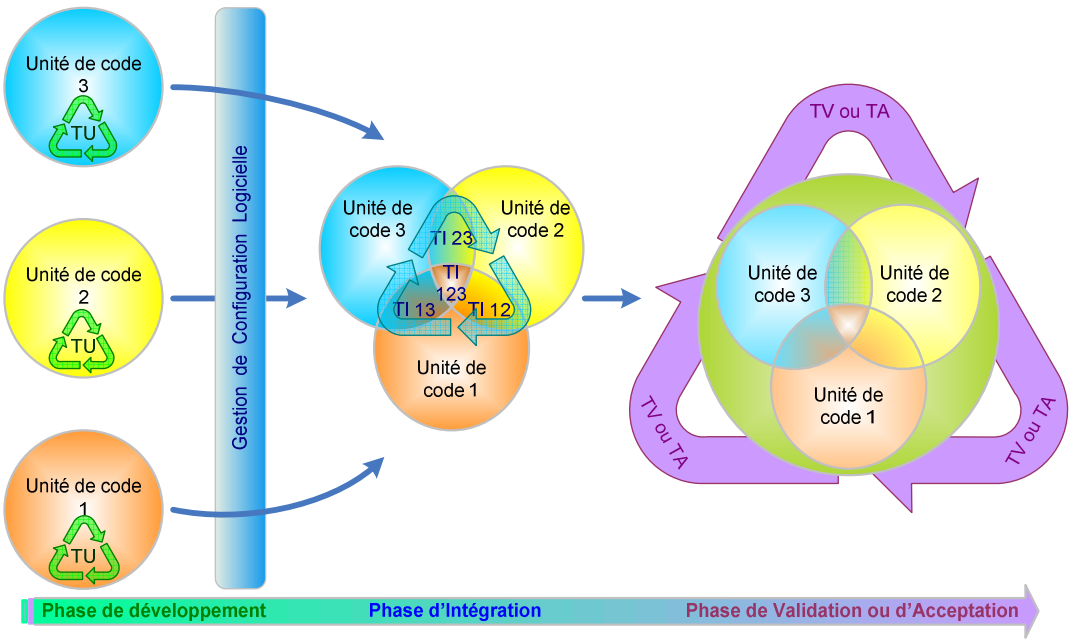
Quelque soit son niveau, un test est une procédure permettant d'examiner une « **portion** » d'un logiciel. Il vérifie une «hypothèse» basée sur le triplet :

- Données d'entrée,
- Portion logicielle à tester,
- Résultats attendus.

Les qualités premières d'un test sont :

- Qu'il soit reproductible,
- Qu'il ait des résultats interprétables aisément,
- Qu'il soit automatique (dans la mesure du possible).

Durant le cycle de vie d'un projet logiciel, les niveaux (phases) de tests cités précédemment s'enchaînent de la façon suivante :



Tous les tests réalisés dans le cadre d'AMDA-NG sont des tests fonctionnels (ou boîte noire) et non structurels (ou boîte blanche).

A chaque type de tests correspond une phase de tests, qui est présentée individuellement dans ce document.

Remarques :

- Compte tenu de l'avancement du projet, ce document n'indique pas, pour le moment, de tests de performance, de robustesse ou de vulnérabilité. Cependant ils devront être définis dans une version ultérieure de ce document.
- La stratégie pour assurer la non régression est la réalisation de l'ensemble des tests de validation en utilisant systématiquement les mêmes jeux de données en entrée. Chacun de ces tests sans exception devant avoir pour résultat un résultat positif.

2.2 ELEMENTS CONCERNES

Sont concernés par les tests toutes les nouvelles classes écrites dans le cadre du projet AMDA-NG (classes JAVASCRIPT, JAVA et C++) et au minimum toutes fonctionnalités décrites dans MU-AMDA-001 complétées par les fonctionnalités liées à la gestion des workspaces.

Ne sont pas concernés par les tests les codes source liés à DD serveur, ni les classes IDL déjà écrites dans le cadre d'AMDA (si IDL était reconduit comme outil de calcul et de tracé)

2.3 RESPONSABILITES

Compte tenu de l'équipe de développement actuelle sont définis les rôles et les activités suivantes :

Intervenant	Rôles	Activités
Caroline Darmon	Responsable de tests Concepteur de tests (unitaire) Testeur (unitaire)	Exécution correction
Myriam Bouchemit	Responsable de tests Concepteur de tests (unitaire et validation) Testeur (unitaire et validation)	préparation des plates-formes constitution des jeux de données exécution correction suivi des résultats validation

Intervenant	Rôles	Activités
Elena Budnik	Responsable de tests Concepteur de tests (unitaire et intégration) Testeur (unitaire et intégration)	constitution des jeux de données exécution correction suivi des résultats

Remarque :

Ce tableau est à compléter suivant l'évolution des ressources présentes sur le projet AMD-NG. De plus, d'autres intervenants ponctuels peuvent effectuer des tests de validation (utilisateurs scientifiques)

2.4 DOCUMENTS UTILISES

Ce plan et dossier de tests est le point d'entrée de l'activité de tests. Il est complété par de fiches de tests, dont le modèle est présenté en annexe ainsi que par des fiches de jeux de tests (ensemble de scénarios et leur ordre d'exécution)

Il n'y a pas de fiches d'anomalies matérialisées car la gestion des faits techniques et de leurs suivis se feront sous l'outil JIRA.

2.5 REGLES D'IDENTIFICATION

2.5.1 Identification des tests

Les règles d'identification des tests sont les suivantes :

<Type de test>-<Scénario>-<Numéro d'ordre> où :

- **<Type de test>** : **TU**, **TI**, **TV** respectivement pour : **T**est **U**nitaire, **T**est d'**I**ntégration, **T**est de **V**alidation.
- **<Scénario>** : fonction, classe, module, fonctionnalité du composant testé.
- **< Numéro d'ordre>** : numéro chronologique

2.5.2 Identification des résultats de tests

Les résultats des tests sont classés, après leur obtention, dans les catégories de conformité suivantes :

- **« OK »** : résultat conforme aux exigences
- **« POK »** : résultat faisant l'objet d'une limite donc Partiellement conforme
- **« NOK »** : résultat non conforme aux exigences

2.6 ENVIRONNEMENT DE TESTS

2.6.1 Configuration matérielle

Les machines utilisées pour les tests sont les suivantes :

Code	Type de machine	Système d'exploitation	Description
CDPP	XXX	FEDORA 11	Machine hébergeant le serveur http (APACHE) et le serveur d'application (TOMCAT). Cette machine héberge aussi le serveur de données DD serveur
PC Client	PC	LINUX WINDOWS XP WINDOWS 7	Client de l'application web d'AMDA.
PC Client	MAC	MAC OS X	Client de l'application web d'AMDA.

2.6.2 Outils

Les outils de génération ou de compilation qui doivent être installés sur les plates-formes de tests sont les suivants :

Outil	Version	Plate-forme
IDL	6.4	SERVEUR
ANT	XXX	DEVELOPPEMENT et SERVEUR
GCC	XXX	DEVELOPPEMENT et SERVEUR
APACHE	XXX	SERVEUR
JAVA	1.6.0_22	DEVELOPPEMENT et SERVEUR
TOMCAT	XXX	SERVEUR
EXT-JS	XXX	DEVELOPPEMENT et SERVEUR

Les outils de déroulement automatiques des tests et de gestion des faits techniques sont les suivants

Outil	Version	Plate-forme
HUDSON	1.386	SERVEUR
SELENIUM	1.0	SERVEUR
JIRA	4.2	SERVEUR
ANT	XXX	DEVELOPPEMENT

Les outils de permettant la validation des tests d'IHM sont les suivants :

	Système d'exploitation			
	WINDOWS XP	WINDOWS 7	MAC OS X	LINUX
FIREFOX	X	X	X	X
CHROME	X	X	X	X
OPERA	X	X	X	X
SAFARI	X	X	X	0
INTERNET EXPLORER	X	X	0	0

2.7 DEROULEMENT DE L'ACTIVITE DE TESTS

Ce paragraphe a pour objectif de décrire le déroulement global des tests.

La démarche générale utilisée pour chaque phase de tests peut être décomposée en plusieurs étapes :

- La préparation des tests
- L'exécution des tests
- L'évaluation des résultats de tests
- La clôture et le bilan des tests

Pour les tests unitaires, la séquence des items 2 et 3 peut être répétés autant de fois que nécessaire jusqu'à correction du code incriminé.

Pour les tests d'intégration, la même séquence n'est effectuée qu'une fois par HUDSON. Par contre, la périodicité d'exécution de cette séquence est à définir par l'équipe projet.

Pour les tests de validation, une seule itération de la séquence des quatre items est à réaliser.

Dans le cadre d'une gestion de projet en mode itératif, les tests unitaires et d'intégration doivent être effectués tout le long de la période de développement (notamment avec les outils automatiques que sont HUDSON et SELENIUM) mais les tests de validation sont à réaliser seulement en fin de période si tous les tests précédents ont été réalisés avec succès.

L'avancement des tests est réalisé automatiquement par HUDSON.

La date d'arrêt des tests de validation est décidée par l'équipe au début d'un cycle de développement

2.8 GESTION DES ANOMALIES

Les anomalies relevées suite aux tests d'intégration et de validation sont identifiées, traitées et suivies conformément au Plan de Gestion de Configuration du projet.

Elles sont gérées par le biais de déclaration de faits techniques sous JIRA, comme suit :

- description de l'anomalie suite au constat de résultat non conforme au résultat attendu
- analyse de l'anomalie pour déterminer la cause racine de l'erreur (spécification, conception, codage, gestion de configuration)
- décision de correction de l'anomalie

Le suivi des anomalies s'effectue également par le biais de l'outil JIRA

2.9 RISQUES

A ce jour pas de risques spécifiques identifiés

3 TESTS UNITAIRES

3.1 OBJECTIFS/LIMITES

Les tests unitaires consistent à tester individuellement les composants du produit logiciel.

Les objectifs des tests unitaires sont de s'assurer :

- que le maximum de combinaisons de choix, d'options pour les différents champs présents dans les écrans soit testé
- que les classes ou les paquetages du noyau fournissent les résultats attendus lorsque ceux-ci ont les mêmes données en entrée
- qu'après toute modification de code (maintenance corrective et/ou évolutive) il n'y ait pas de dysfonctionnement
- qu'il n'y ait pas de régression (la couverture des tests unitaires devant être immuable dans le temps)

Il est à noter que dans un premier temps certains tests unitaires liés à l'IHM pourront être partiellement réalisables car le retour de ces écrans fait appel à des fonctionnalités du noyau.

3.2 DOCUMENTS DE REFERENCE

Les tests unitaires sont réalisés selon certains cas d'utilisation présents dans MU-AMDA-001 ainsi que d'après les exigences indiqués dans le document DA-AMDA-001.

3.3 ELEMENTS CONCERNES

Les tests unitaires sont requis à minima pour les composants critiques.

A compléter lors de la mise en place des tests

Identifier les types de composants susceptibles d'être concernés par les tests unitaires.

Préciser les critères de criticité d'un composant.

Exemples de type de composants :

- Fonctions,

- Classes,

- Objets IHM,

-…

3.4 JEUX DE DONNEES

A compléter lors de la mise en place des tests

Identifier les sources de données utilisées (fichiers, bases de données, jeux de tests données) lors de l'exécution des tests.

3.5 METHODOLOGIE DE TESTS

Les tests unitaires seront des tests « boite noire » ; pas de test unitaire pour chaque branche de décision du code ou pour chaque chemin du graphe de décision car à priori les tests unitaires de par leur importante nature combinatoire « externe » effectueront la couverture interne des portions de code. Cependant, cette politique pourra être affinée suivant les résultats du plan de test.

Il faut donc que les tests « boite noire » couvrent :

- les cas de test pour les valeurs nominales des données en entrée de l'élément,
- les cas de test pour les valeurs limites des données en entrée de l'élément,
- les cas de test pour chaque chemin du graphe d'état des objets IHM
- les cas de test pour chaque contrôle (champ obligatoire, format de saisie, etc.)

Chaque test est décrit par une fiche de test à laquelle peut être associé un programme de test dans le cas de tests unitaires automatisés (de type SELENIUM, CPPUNIT, autre…). Les rubriques des fiches de tests sont les suivantes:

- Description du test. Elle donne l'objectif du test et rappelle la ou les exigences fonctionnelles qui sont à vérifier par ce test.
- Mise en œuvre. Elle décrit notamment :
    - Le contexte d'exécution, qui représente l'état du système nécessaire au déroulement du test : variables d'environnement, état des bases de données, etc.
    - La procédure de lancement du test.
- Résultats attendus. Cette rubrique décrit les résultats, les valeurs attendues à la fin du test, et/ou les actions à accomplir pour vérifier l'état du système à la fin de l'essai.

3.6 SCENARIOS DE TESTS

A compléter lors de la mise en place des tests

Identifier et définir les différents scénarios et les cas de tests associés.

Présenter l'ordre d'exécution des scénarios.

3.7 RESULTATS DE TESTS

Les résultats des tests d'intégration sont enregistrés dans les fiches de test présentées en « 8 ANNEXE »

4 TEST D'INTEGRATION

4.1 OBJECTIFS/LIMITES

Le but des tests d'intégration est de valider le fait que toutes les unités composant une application et développées indépendamment fonctionnent correctement ensemble de façon cohérente. Ils ciblent les interfaces (logicielles et IHM) qui permettent aux différents modules d'interagir entre eux.

Les objectifs généraux de la phase des tests d'intégration est de s'assurer :

- que les écrans d'IHM associés à une fonctionnalité particulière d'AMDA (plot, download, etc.) fournissent bien le résultat attendu
- que les différents modules devant communiquer au sein d'une même couche du noyau le fasse correctement (intégration horizontale)
- que les modules de différentes couches du noyau communiquent correctement

Il n'y a pas de test d'intégration lié à la synchronisation de plusieurs instances d'AMDA-NG (pas de tests donc liés à RSYNC ou HEARTBEAT)

4.2 DOCUMENTS DE REFERENCE

Les tests d'intégration sont réalisés selon les cas d'utilisation présents dans MU-AMDA-001 ainsi que d'après les exigences indiqués dans le document DA-AMDA-001.

4.3 ELEMENTS CONCERNES

A compléter lors de la mise en place des tests

Identifier les types d'éléments concernés par les tests d'intégration.

Exemples de type d'éléments :

- Programme principal,

- Ensemble de classes collaboratrices,

- Composants de haut niveau d'une couche de l'architecture logicielle (ex : composants de la couche Métier, de la couche Application),

- Ecran IHM …

4.4 JEUX DE DONNEES

A compléter lors de la mise en place des tests

Indiquer les fichiers, bases de données, jeux de tests données utilisés pendant les tests d'intégration.

4.5 METHODOLOGIE

La méthode de vérification associée aux tests d'intégration, peut, selon les unités testées, être :

- tests d'enchaînement des modules ou écrans,
- vérification des appels de fonction avec différents paramètres d'appel (nominaux, aux bornes, hors du domaine),
- vérification des interfaces entre unités logicielles (tests des arguments en entrée et des valeurs en retour),
- tests d'intégration entre le serveur et l'IHM web

Les résultats attendus en retour du noyau d'AMDA-NG doivent être les même que ceux d'AMDA.

Comme pour les tests unitaires, seuls seront effectués des tests de type « boite noire »

Pour les éléments de l'IHM, la méthode d'intégration sera descendante (utilisation de bouchons pour simuler les éléments de bas niveau non encore testés ou mis au point)

Pour les éléments du noyau, la méthode d'intégration sera ascendante (les éléments de bas niveau sont préalablement mis au point ou testés unitairement avant leur intégration avec un élément de plus haut niveau)

La stratégie d'intégration avec les systèmes externes sera directe (tous les éléments sont intégrés en une seule fois)

4.6 SCENARIOS DE TESTS

A compléter lors de la mise en place des tests

Identifier et définir les différents scénarios et les cas de tests associés.

Présenter l'ordre d'exécution des scénarios.

4.7 RESULTATS DE TESTS

Les résultats de tests d'intégration sont enregistrés dans des fiches de test présentées en « 8 ANNEXE »

5 TESTS DE VALIDATION

5.1 OBJECTIFS/LIMITES

Le but des tests de validation est de vérifier que le logiciel livré respecte les exigences définies dans le dossier de spécifications :

- Exigences fonctionnelles,
- Exigences de performances,
- Exigences de robustesse,
- Exigences de sécurité.

5.2 DOCUMENTS DE REFERENCE

Les tests de validation sont réalisés uniquement selon les cas d'utilisation présents dans MU-AMDA-001 ainsi que d'après les spécifications de l'IHM concernant toutes fonctionnalités liées au workspace.

5.3 ELEMENTS CONCERNES

Les éléments concernés par les tests de validation sont :

- Les écrans de la nouvelle IHM
- Les enchainements d'IHM
- Les écrans liés au concept de workspace
- Les web services offerts par AMDA-NG

5.4 JEUX DE DONNEES

A compléter lors de la mise en place des tests

Indiquer les fichiers, bases de données, jeux de tests, utilisés pendant les tests de validation.

5.5 METHODOLOGIE

La méthode de vérification associée aux tests de validation peut être :

- tests relatifs aux cas d'utilisation,
- tests relatifs à l'IHM,

- tests de fonctionnement de l'application par rapport aux spécifications fonctionnelles,
- tests de robustesse, tests de performance,
- tests des interfaces fonctionnelles avec d'autres systèmes ou applications,
- tests en mode multiutilisateurs,
- tests de bon fonctionnement sur environnement cible,
- vérification des accès aux fichiers et aux bases de données : confidentialité, intégrité,
- vérification du bon déroulement des procédures de sécurité,
- vérification de la maniabilité et de l'ergonomie du logiciel,

5.6 SCENARIOS DE TESTS

A compléter lors de la mise en place des tests

Identifier et définir les différents scénarios et les cas de tests associés.

Présenter l'ordre d'exécution des scénarios.

Si besoin, créer une fiche par cas de tests, avec les données en entrée et les résultats attendus.

5.7 RESULTATS DE TESTS

Les résultats de tests de validation sont enregistrés dans des fiches de test présentées en « 8 ANNEXE »

6 TESTS DE NON-REGRESSION

6.1 CRITERES DE TESTS DE NON-REGRESSION

Les critères qui permettent de décider de l'opportunité des tests de non-régression sont les suivants :

- modification d'un élément intervenant dans plusieurs fonctionnalités
- modification de tout élément intervenant dans la génération et la communication de résultats à des applications externes d'AMDA-NG (web services)
- A compléter

6.2 METHODOLOGIE DE TESTS DE NON-REGRESSION

Pour plus de sécurité, la méthodologie de tests de non régression proposée est la suivante :

- repasser tous les cas de test concernant l'élément modifié,
- repasser les cas de tests connexes à la modification

7 MATRICE DE COUVERTURE DES TESTS

ID exigences	Description
ARD_LOG_010	*L'administrateur doit pouvoir ajouter un utilisateur avec son mot de passe*
ARD_LOG_020	*L'administrateur doit pouvoir modifier un mot de passe*
ARD_LOG_030	*Le système doit pouvoir accepter un utilisateur connu par son login /mot de passe, on dit qu'il est identifié pour le système*
ARD_LOG_040	*Le système doit accepter dix utilisateurs identifiés en « GUEST » au maximum en simultané.*
ARD_WS_010	*Le système doit pouvoir retourner la liste des workspaces d'un utilisateur*
ARD_WS_020	*Le système doit pouvoir retourner la description (arbre affiché) d'un workspace utilisateur ou de l'espace partagé*
ARD_WS_030	*Le système doit pouvoir sauvegarder la description (arbre affiché) d'un workspace pour un utilisateur ou pour l'espace partagé*
ARD_WS_040	*Le système doit pouvoir copier une sous partie d'un workspace dans un autre workspace pour un utilisateur ou pour l'espace partagé*
ARD_WS_050	*Le système doit pouvoir supprimer une sous partie d'un workspace pour un utilisateur ou pour l'espace partagé*
ARD_WS_060	*Le système doit pouvoir sauvegarder un élément d'un workspace (par exemple la description d'un paramètre composé, la description time table, etc.) pour un utilisateur*
ARD_WS_070	*Le système doit pouvoir retourner un élément d'un workspace (par exemple la description d'un paramètre composé, la description time table, etc.) d'un utilisateur ou de l'espace partagé*
ARD_WS_080	*Le système doit pouvoir organiser son workspace ou l'espace partagé sous forme d'arbre i.e. création, renommage, suppression de répertoire ou d'élément du workspace*
ARD_PAR_010	*Le système doit pouvoir retourner l'arbre des paramètres basiques.*
ARD_PAR_020	*Le système doit pouvoir valider un paramètre composé. i.e. Valider que la formule décrivant la composition du paramètre a une syntaxe correcte correspondante*
ARD_PAR_030	*Le système doit pouvoir définir un alias sur un paramètre et le sauvegarder dans l'espace utilisateur*
ARD_PAR_040	*Le système doit gérer une description d'un paramètre composé comme un élément du workspace*
ARD_TT_010	*Le système doit pouvoir fusionner des intervalles définis d'une time table*
ARD_TT_020	*Le système doit pouvoir trier des intervalles définis d'une time table*

ID exigences	Description
ARD_TT_030	*Le système doit pouvoir faire l'intersection de deux time tables*
ARD_TT_040	*Le système doit pouvoir faire une opération de décalage dans une time table*
ARD_TT_050	*Le système doit pouvoir filtrer une time table selon une condition*
ARD_TT_060	*Le système doit gérer une description d'une time table comme un élément du workspace*
ARD_COND_010	*Le système doit pouvoir rechercher des intervalles de temps (timeTable) suivant une condition. La condition est donnée par une formule respectant la syntaxe.* *Le résultat de la recherche est :* - *une time table créée et sauvegardée dans l'espace utilisateur,* - *et deux fichiers l'un contenant les données de l'intervalle répondant à la condition, l'autre celles, ne répondant pas à la condition. Une description de ce résultat est retournée.*
ARD_COND_020	*Le système doit gérer une description d'une condition comme un élément du workspace*
ARD_PLOT_010	*Le système doit pouvoir tracer un graphique représentant les variations de paramètres sous forme de fichier PNG PDF ou POSTSCRIPT gzippé et retourner le fichier.*
ARD_PLOT_020	*Le système doit pouvoir choisir automatiquement un type de tracé (spectrogramme, série, ...) pour chaque paramètre suivant un fichier de configuration.*
ARD_PLOT_030	*Le système doit pouvoir modifier son tracé via les attributs des paramètres.*
ARD_PLOT_040	*Le système doit gérer une description de tracé comme un élément du workspace*
ARD_LOAD_010	*Le système doit pouvoir créer un fichier contenant les données d'un paramètre composé pour une time table donnée dans le but d'un téléchargement. Le fichier est en texte, la partie Web se charge de faire soit un fichier HTML, soit un fichier à télécharger.*
ARD_LOAD_020	*Le système doit gérer une description de requête de téléchargement comme un élément du workspace*
ARD_CDATA_010	*Le système doit pouvoir ajouter un centre de données*
ARD_CAT_010	*Le système doit gérer les catalogues comme un élément du workspace*
ARD_PLUG_010	*Le système doit pouvoir accepter des « plugins » via un fichier de configuration*
ARD_PLUG_020	*Le système doit pouvoir accepter une autre forme de tracé via un « plugin ».*
ARD_PLUG_030	*Le système doit pouvoir accepter d'autres fonctions de calcul via un « plugin ».*
ARD_ROB_010	*Tolérance aux pannes: un « plantage » doit impacter de manière minimale un utilisateur en termes de perte de données et de paramétrage.*

ID exigences	Description
ARD_PLA_010	*L'ensemble d'AMDA-NG doit fonctionner sous Linux mais être indépendant de la distribution de Linux.*
ARD_NDR_010	*taux de disponibilité de 100%*
ARD_NDR_020	*Temps de maintenance maximal autorisé*

Le tableau suivant permet de voir quelle fiche de test couvre quelle(s) exigence(s) (A compléter avec les identifiants des fiches)

	Test unitaire	Test d'intégration	Test de validation
ARD_LOG_010			
ARD_LOG_020			
ARD_LOG_030			
ARD_LOG_040			
ARD_WS_010			
ARD_WS_020			
ARD_WS_030			
ARD_WS_040			
ARD_WS_050			
ARD_WS_060			
ARD_WS_070			
ARD_WS_080			
ARD_PAR_010			
ARD_PAR_020			
ARD_PAR_030			
ARD_PAR_040			
ARD_TT_010			
ARD_TT_020			

	Test unitaire	Test d'intégration	Test de validation
ARD_TT_030			
ARD_TT_040			
ARD_TT_050			
ARD_TT_060			
ARD_COND_010			
ARD_COND_020			
ARD_PLOT_010			
ARD_PLOT_020			
ARD_PLOT_030			
ARD_PLOT_040			
ARD_LOAD_010			
ARD_LOAD_020			
ARD_CDATA_010			
ARD_CAT_010			
ARD_PLUG_010			
ARD_PLUG_020			
ARD_PLUG_030			
ARD_ROB_010			
ARD_PLA_010			
ARD_NDR_010			
ARD_NDR_020			

8 ANNEXE

8.1 FICHES DE TESTS

A compléter

9 DOCUMENTS APPLICABLES ET DE REFERENCE (A/R)

A/R	Référence	Titre
A	CDPP-NT-32500-379-SI	Dossier des cas d'utilisation d'AMDA V1.0 édition du 20/07/2010
A	CDPP-AR-32500-382-SI	Dossier d'architecture du noyau d'AMDA-NG V1.0 édition du 19/11/2010

10 GLOSSAIRE ET ABREVIATIONS

10.1 GLOSSAIRE

Terme	Définition
Campagne de tests	Période continue d'activités de tests, comprenant tout ou partie des scénarios de tests
Cas de test	Un cas de tests est défini par l'état des données en entrée du test, les actions à mener par le testeur et les résultats attendus.
Jeux de tests	Il s'agit de créer des cas concrets d'utilisation de l'application à partir : • de sa description formalisée dans les spécifications • des règles définies dans le plan et dossier de tests (principalement en matière d'ergonomie) La définition de jeux de tests permet de prévoir et de formaliser la manière et les critères sur lesquels l'application va être recettée. Les jeux de tests ne peuvent être définis qu'une fois les spécifications validées.
Jeux de données	Ensemble de données de tests utilisées dans le même scénario (ou jeux d'essai)
Scénario de tests	Enchaînement chronologique de cas de tests, cohérent fonctionnellement et dont le résultat est un état stable du système.
Test boîte blanche (ou test structurel)	Méthode de test qui consiste à concevoir les données d'entrée et les résultats attendus en examinant la structure interne de l'objet à tester.
Test boîte noire (ou test fonctionnel)	Méthode de test qui consiste à concevoir les données d'entrée et les résultats attendus à partir des fonctions spécifiées de l'objet à tester, sans examiner sa structure interne.
Test de bon fonctionnement	Les tests de bon fonctionnement font partie des tests de validation. Il s'agit de tests effectués sur l'environnement cible (ou un environnement de simulation) et qui consistent à s'assurer que le système fonctionne normalement dans cet environnement. Au minimum, les tests de bon fonctionnement consiste à lancer l'application et à s'assurer qu'il s'exécute normalement sans blocage. Les tests de bon fonctionnement peuvent aussi consister à repasser tout ou partie des tests de validation effectués sur l'environnement de tests, voire des tests complémentaires nécessités par des différences entre les environnements.
Test d'intégration	Activité permettant de vérifier que les interfaces communes à plusieurs composants permettent bien de réaliser le comportement attendu entre ces composants. Les tests d'intégration répondent aux exigences de conception.
Test unitaire	Activité permettant de vérifier qu'un composant, pris isolément, satisfait à ses exigences fonctionnelles et techniques. Ils répondent à la conception détaillée si elle a été réalisée.

Terme	Définition
Test de validation	Activité permettant de vérifier que le logiciel, dans son ensemble, satisfait à ses exigences fonctionnelles et techniques y compris dans son environnement cible. Les tests d'intégration répondent aux exigences de spécification.

10.2 ABREVIATIONS

Abréviation	Nom détaillé
IHM	**I**nterface **H**omme **M**achine