Réalisation d'une recette fonctionnelle outillée avec Quality Center et Quick Test Professional

Cathy Sevre

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

Centre d'enseignement de PARIS

MEMOIRE

présenté en vue d'obtenir

LE DIPLOME D'INGENIEUR CNAM

SPÉCIALITÉ : Informatique

OPTION : Architecture et ingénierie des systèmes et des logiciels

par Cathy SEVRE

Réalisation d’une recette fonctionnelle outillée avec Quality Center et Quick Test Professional

Soutenu le 21/06/2011

JURY :

Président : Yann POLLET
Membres : Jean-Michel DOUIN
Yves LALOUM
Jean POMPEE
Emmanuelle TEXIER

REMERCIEMENTS

Je souhaite remercier

Laurent Moyeux, mon tuteur au sein de RTE, pour m'avoir aidé à mettre en place ce projet.

Pierre Brunel, Patrick Dufaure, Jean Pompée pour m'avoir accueillie dans le département DPCM et l'équipe PARC.

Emmanuelle Texier et toute l'équipe du projet EMMA pour le temps qu'ils m'ont consacré.

Le Professeur Yann Pollet pour ses conseils tout au long de ma mission.

Ainsi que l'ensemble des personnes de chez RTE pour leur accueil.

Résumé

La phase de test consiste à vérifier que le logiciel répond bien aux exigences des utilisateurs. Cette phase a souvent été le parent pauvre du processus de production. Mais, avec l'arrivée des systèmes de plus en plus complexes, les tests sont devenus indispensables. La plupart des sociétés connaissent désormais l'importance des tests et de nombreux outils ont été conçus pour les aider dans cette démarche.

J'ai intégré l'équipe PSI/DPCM/PARC qui développe des applications spécifiques aux besoins métiers de RTE (Réseau de Transport d'Electricité). RTE est l'entreprise qui gère le réseau électrique français de transport d'électricité à haute et très haute tension.

L'utilisation d'applications sur des périmètres critiques nécessite de s'assurer du bon fonctionnement des applications avant qu'elles soient mises en production. Pour cela des périodes de recette sont prévues dans les plannings d'évolution des applications. Afin d'optimiser ces processus de recette l'équipe PSI/DPCM/PARC à décidé de s'outiller des logiciels Quality Center pour le suivi de la recette et Quick Test Professional pour l'automatisation des tests. Dans ce cadre, j'ai contribué à la définition d'une méthodologie de tests basée sur ces outils et mis en œuvre cette méthode sur plusieurs applications.

Mots clés

Automatisation, tests fonctionnels, tests de non-régression, qualité logiciel, Quality Center, Quick Test Professional.

Summary

The goal of test phase is to ensure that the software meets user requirements. It has often been the lazy phase of the production process. But with the rising of systems more complex, testing has become critical. Most companies now recognize the importance of testing and many tools have been designed to assist in this process.

I joined the team PSI/DPCM/PARC which develops specific applications business needs of RTE. RTE is the company that manages the French electricity transport electricity at high and very high voltage.

The use of critical applications requires to ensure to get the desired response from the system before they are put into production. To achieve this, test periods are scheduled during the development applications process. To optimize the test process the team PSI/DPCM/PARC decide to equip them up with Quality Center for monitoring the test phase and Quick Test Professional for test automation. I defined a testing methodology based on these tools and implemented this method on several applications.

Keywords

Sofware test automation, functional testing, regression testing, software quality assurance, Quality Center, Quick Test Professional.

TABLE DES MATIERES

INTRODUCTION

La recette applicative consiste à valider le produit par rapport aux attentes et exigences de l'utilisateur, à maitriser le risque de mise en place et à garantir le niveau de qualité requis pour le logiciel. La recette permet de vérifier et valider l'adéquation du produit livré avec le contrat de service et les besoins effectifs du client.

Cette tâche est encore trop souvent mise de côté. Pourtant les logiciels prennent une place de plus en plus importante dans notre vie. Une erreur peut avoir de très lourdes conséquences que ce soit sur le plan financier ou, plus grave, sur le plan humain.

Selon une étude américaine, c'est près de soixante milliards de dollars qui sont perdus chaque année aux Etats-Unis suite à des problèmes de qualité dans les applications logicielles [NIST2002].

Certaines sociétés commencent à prendre conscience de l'importance des tests. Mais lorsque l'idée du test est acceptée, le projet n'en est pas réussi pour autant. Les tests nécessitent être menés selon une démarche et être organisés. Ils doivent être gérés comme un projet à part entière. Comment réaliser correctement une recette fonctionnelle outillée ?

De nombreux outils existent pour faciliter la réalisation des recettes applicatives. Nous allons voir plus particulièrement les outils Quality Center et Quick Test Professional d'HP. Il s'agit des outils choisis par RTE où j'ai effectué un stage de neuf mois pour réaliser ce mémoire.

Dans un premier temps, nous allons présenter le contexte de ce mémoire et ma mission au sein de RTE.

Ensuite, nous décrirons quelques notions relatives aux tests logiciels, montrant l'importance des tests, les principes généraux des tests et la comparaison entre les tests manuels et les tests automatisés.

Puis nous verrons comment réaliser une recette fonctionnelle outillée avec les outils HP en détaillant différentes fonctionnalités de ceux-ci.

Pour finir, nous parlerons de la mise en œuvre concrète de ces outils avec les applications EMMA (Java) et DECOMPTES (C++).

1. PRESENTATION DU CONTEXTE

1.1 RTE

1.1.1 Présentation générale

RTE (Réseau de Transport d'Electricité) est une société anonyme filiale du groupe EDF. Il s'agit d'une entreprise de service public qui a pour mission l'exploitation, la maintenance et le développement du réseau haute et très haute tension. Elle est garante du bon fonctionnement et de la sûreté du système électrique.

RTE achemine l'électricité entre les fournisseurs d'électricité (français et européens) et les consommateurs, qu'ils soient distributeurs d'électricité ou industriels directement raccordés au réseau de transport.

Elle est dotée d'un statut d'entreprise indépendante qui garantit sa neutralité : elle assure un accès libre et équitable à tous les utilisateurs du réseau, acteurs du marché de l'électricité.

Avec 100 000 km de lignes comprises entre 63kV et 400kV et quarante-cinq lignes transfrontalières, le réseau géré par RTE est le plus important d'Europe.

1.1.2 Ses missions

1.1.2.1 Assurer un service public de haut niveau

RTE remplit d'importantes missions de service public définies juridiquement. Elles sont précisées dans le contrat de service public conclu avec l'Etat et exercées sous le contrôle de la Commission de Régulation de l'Energie (CRE), qui fixe notamment les tarifs d'utilisation des réseaux publics de transport et de distribution (TURPE).

Le contrat de service public, signé en 2005, apporte des garanties sur le maintien d'un haut niveau de service public de l'électricité en France.

Il précise les engagements de RTE, en vue d'assurer la pérennité des missions de service public que le législateur lui a confiées, en particulier dans deux domaines : la gestion du réseau public de transport et la sûreté du système électrique.

1.1.2.2 Gérer les infrastructures de réseau

RTE doit, au coût le plus juste pour la collectivité, entretenir le réseau, renforcer sa robustesse et le développer en fonction de la demande, en veillant à réduire son impact environnemental. Il doit assurer la continuité et la qualité de la fourniture de courant.

1.1.2.3 Gérer les flux d'électricité sur le réseau

RTE doit veiller à la sécurité d'approvisionnement et alerter les pouvoirs publics en cas de risque de rupture.

La sûreté de fonctionnement du système électrique est au cœur des responsabilités confiées par la loi à RTE, en tant que gestionnaire du réseau de transport public français.

1.1.2.4 Contribuer au bon fonctionnement du marché de l'électricité

RTE garantit à tous les utilisateurs du réseau de transport d'électricité un traitement sans discrimination, sur la base de tarifs d'accès publics, indépendants de la distance entre fournisseur et consommateur.

RTE favorise la fluidité des échanges. Ses solutions de gestion des flux préservent au maximum la liberté des acteurs du marché et font appel à leurs initiatives. Il travaille à développer les capacités d'interconnexion, en coopération avec les autres GRT. Un effort indispensable, vu la position géographique centrale du réseau français.

1.1.2.5 Contribuer au développement durable

Grâce à ses prévisions de consommation, RTE fournit une information utile pour l'ensemble des acteurs de l'énergie et le grand public, afin d'encourager les comportements responsables de tous, y compris des consommateurs.

De plus, il se mobilise pour intégrer au mieux l'électricité produite à partir d'énergies renouvelables ou pour limiter le recours à une production fortement émettrice de gaz à effet de serre lorsque cela est possible.

1.1.3 Organisation

RTE regroupe 8 500 collaborateurs répartis entre des Unités Régionales et des Fonctions Centrales.

RTE est structuré en sept régions électriques. Ce découpage de la France est un héritage ancien, antérieur au découpage régional administratif de 1972, qui trouve son origine dans le maillage électrique français.

L'activité de RTE s'organise autour de deux grands métiers indissociables : Système Electrique et Transport Electrique.

1.1.3.1 Les unités Système Electrique

Les unités Système Electrique de RTE gèrent l'équilibre offre-demande, les flux sur les réseaux 400kV, 225kV, 90kV et 63kV, l'accès au réseau pour les sites des clients implantés en France et le développement du réseau régional. Il comprend le Centre national d'exploitation du système (CNES) et sept unités régionales.

1.1.3.2 Les unités Transport Electrique

Les unités Transport Electrique de RTE assurent la maintenance et l'exploitation des ouvrages ainsi que l'ingénierie des projets touchant au développement du réseau. Il dispose du Centre national d'expertise réseaux (CNER) et de sept unités régionales.

1.1.3.3 Les centres de conduite ou « dispatching »

Le système électrique français est exploité par des centres de conduite réseau également appelés dispatching national (CNES) et régionaux.

Le Centre National d'Exploitation du Système de RTE (CNES) est responsable de l'équilibre entre la production et la consommation, la maîtrise du plan de tension et des transits sur le réseau de 400kV et de la gestion des échanges d'électricité entre la France et ses pays voisins aux frontières par les quarante-cinq lignes d'interconnexion électrique.

Les régions électriques, par le biais de leur dispatching régional, ont la responsabilité de la surveillance du réseau de 400kV en appui du CNES, de la maîtrise du plan de tension et des transits sur les réseaux inférieurs à 400kV (225kV, 90kV et 63kV) et de la télécommande des postes haute tension.

Chacune des sept régions est dotée de son dispatching régional : des équipes se relaient vingt-quatre heures sur vingt-quatre et sept jours sur sept pour veiller à l'équilibre production-consommation au niveau de la région concernée, en supervisant en temps réel l'état du réseau haute et très haute tension.

Les équipes des dispatching ordonnent en temps réel les manœuvres nécessaires pour aiguiller l'électricité de manière optimale, quelle que soit la situation à laquelle elles peuvent être confrontées (incidents sur le réseau, aléas climatiques…). Les ordres sont soit télécommandés, soit transmis par téléphone (lignes sécurisées) vers les postes de transformation électriques sur tout le territoire du réseau de RTE.

En cas d'apparition de perturbations sur le réseau, les dispatching des sept régions électriques apportent leur soutien au CNES pour mettre en œuvre les dispositions conservatoires visant à maintenir l'intégrité du réseau et son fonctionnement habituel, ou à le reconstituer au plus vite après un écroulement partiel ou total.

En cas d'incident de grande ampleur, des actions, tant automatiques que manuelles, sont mises en œuvre pour éviter un écroulement total du réseau et faciliter la reconstitution du système. Il s'agit alors d'actions de conduite exceptionnelles, comme le recours au délestage qui consiste à arrêter volontairement l'approvisionnement d'un ou de plusieurs consommateurs pour rétablir rapidement l'équilibre entre la production et la consommation du réseau.

La figure ci-dessous nous montre l'organisation générale de RTE.

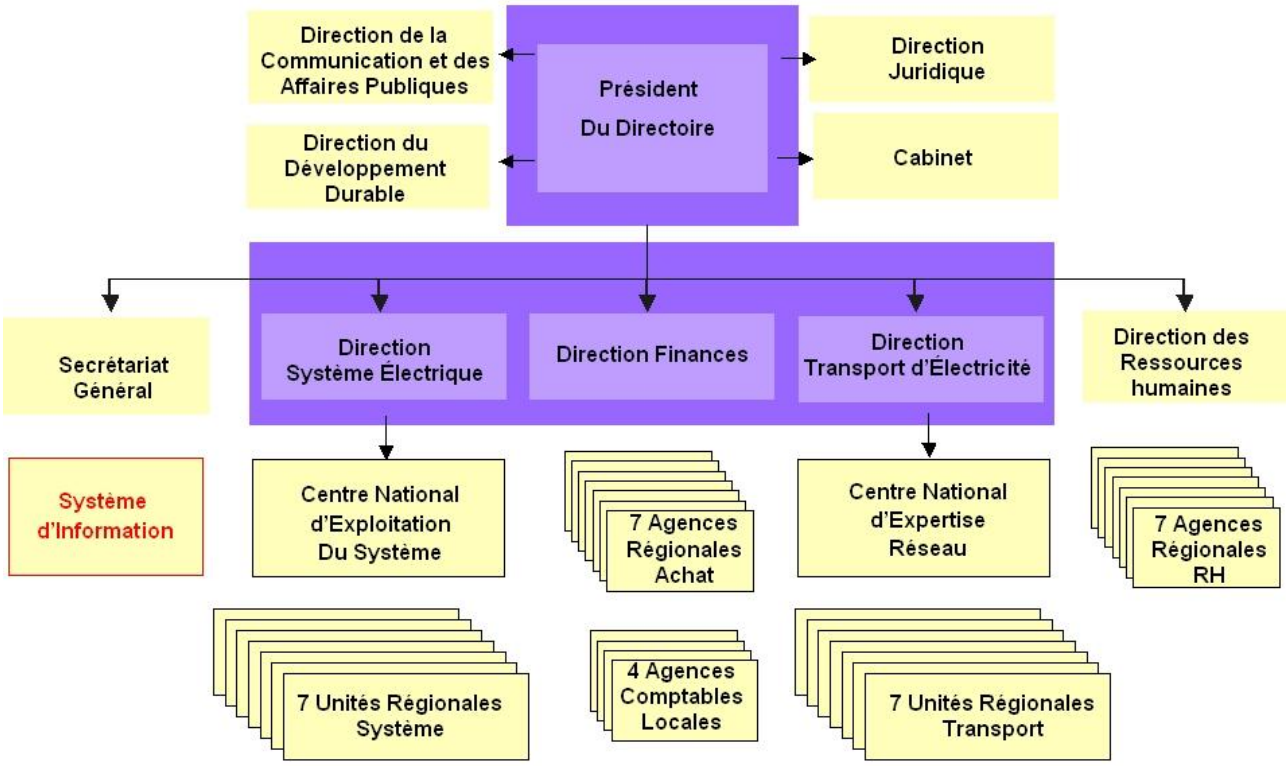


Figure 1 : Organisation de RTE

1.2 Système d'Information

1.2.1 Présentation générale

Le Système d'Information (SI) de RTE assure le pilotage national des services d'ingénierie et d'exploitation de l'entreprise.

1.2.2 Organisation

L'organisation de la fonction SI est alignée avec les Directions métiers de RTE.

Les Pilotes Stratégiques de Programme (PSP) pilotent les Programmes SI. Chaque PSP est responsable de l'ensemble des projets et applications de son programme.

Le Système d'Information de RTE est constitué de deux entités.

1.2.2.1 Le Centre d'Expertise et d'Exploitation du Système d'Information (CEESI)

Sa mission est de garantir la mutualisation de l'expertise et la fiabilité du SI dans une logique industrielle. Elle est garante de l'adéquation, de la cohérence, de l'optimisation du SI et travaille en partenariat avec les métiers à l'amélioration de leur performance.

1.2.2.2 Le Centre d'Ingénierie et de Maintenance des Programmes du Système d'Information (CIMPSI)

Cette entité a pour mission d'assurer, pour les métiers de l'entreprise, au travers des programmes du SI, la performance des projets et du maintien en condition opérationnelle des applications.

La figure ci-dessous permet de visualiser l'organisation du SI de RTE.

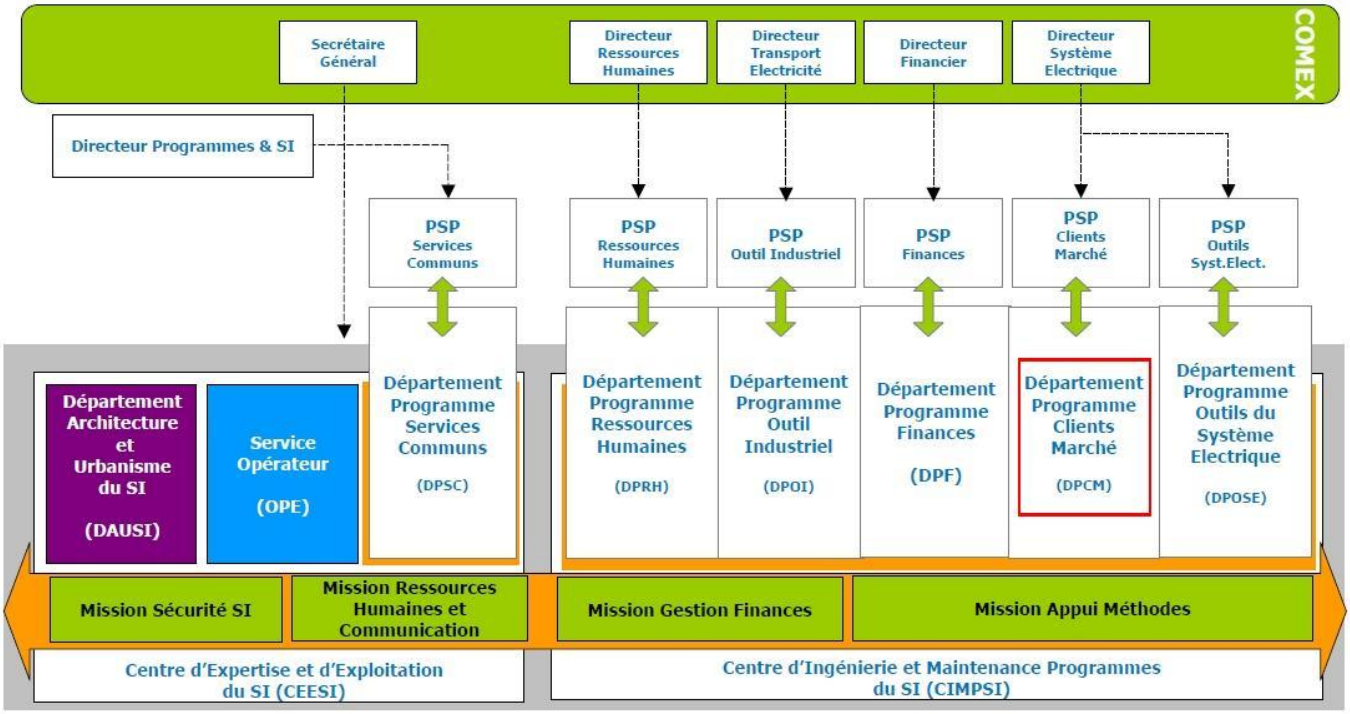


Figure 2 : Organisation du SI de RTE

1.3 Département Programme Clients Marché (DPCM)

1.3.1 Ses missions

L'objectif du département est de concevoir, adapter et maintenir les applications du Système d'Information de l'entreprise liées au marché de l'électricité français et européen, à la fois pour les clients de RTE et pour les acteurs internes.

En pratique, il s'agit de :

- Assurer la facturation de l'accès au réseau (TURPE) et des services de RTE ;
- Mettre à disposition des chargés de relations clientèles des outils de partage de la connaissance des clients et de gestion notamment du contrat d'accès au réseau ;
- Gérer les offres d'ajustement et les programmes de production pour assurer l'équilibre entre l'offre et la demande ;
- Publier des informations attendues par les acteurs du marché ;
- Gérer les échanges avec les acteurs du marché pour qu'ils accèdent aux interconnexions tout en respectant les règles de sûreté.

1.3.2 Organisation

Le département est organisé en trois équipes selon un découpage des activités par domaine fonctionnel, soit une trentaine d'agents.

- Une équipe SAP-CM (SAP Clients Marché) pour les applications « SAP » (référentiel des données de comptage, facturation, référentiel clients) ;
- Une équipe PARC (Programmation Ajustement Référentiels et Clients) pour les applications relatives à l'équilibre offre/demande, aux Responsables d'Equilibres et aux référentiels associés ;
- Une équipe IMGH (Interconnexions, Marché, Gestion des Hypothèses) pour les outils de gestion des interconnexions internationales.

1.4 Ma mission

1.4.1 Mes objectifs

RTE cherchant à mettre en place une solution industrielle de tests de non-régression sur des applications utilisées dans le domaine du marché de l'électricité, j'ai intégré pendant neuf mois l'équipe PARC du Département Programme Clients Marchés.

Tout d'abord j'ai participé à la définition d'une méthodologie de test basée sur l'outil Quality Center (QC) pour l'application EMMA.

Ensuite j'ai implémenté des tests automatisés de non-régression sur cette application en mettant en œuvre l'outil Quick Test Professional (QTP).

Puis j'ai rédigé un document de bonnes pratiques des outils QC et QTP dans le cadre de l'application EMMA.

Pour finir, j'ai formé d'autres équipes projets à l'utilisation de ces outils et je les ai conseillées dans la mise en place de ceux-ci.

1.4.2 Planning de mes actions

Périodes	Actions
Mai 2010	J'ai fait des recherches concernant l'organisation générale des tests et j'ai commencé à prendre en main l'outil Quality Center (QC).
Début juin 2010	J'ai présenté QC lors de la réunion des chefs d'équipes de DPCM et j'ai assisté à une formation de trois jours sur l'outil Quick Test Professional (QTP).
Mi-juin à fin juin 2010	J'ai saisi les tests de non-régression du client URSE de l'application EMMA dans QC (exigences et fiches de test)
Juillet à mi-juillet 2010	J'ai automatisé les tests de non-régression du client URSE à l'aide de l'outil QTP. J'ai également mis en place les scripts réutilisables (chargement d'un dump,…).
Mi-juillet à mi-août 2010	J'ai scripté quelques tests de non-régression du client Tracnes d'EMMA. Ces tests étaient déjà rentrés sous QC.
Septembre à mi-octobre 2010	Une nouvelle version de mise en production d'EMMA a été livrée et est à tester (V2.3). J'ai relancé les scripts automatiques crées avec la version précédente (V2.2). J'ai participé à l'analyse des erreurs et j'ai apporté quelques corrections aux scripts.
Mi-septembre 2010	J'ai présenté QTP à une autre personne de l'équipe PARC et nous avons testé l'outil sur l'application TEMPETE.
Fin septembre 2010	J'ai essayé QTP sur l'application DECOMPTES.
Début octobre 2010	J'ai fait une présentation des outils QC et QTP à l'équipe SAP-CM.
Octobre à novembre 2010	J'ai saisi des tests de non-régression des clients Tracnes et Validation d'EMMA dans l'outil QC. Puis j'ai lancé l'automatisation de ces tests dans QTP.
Décembre 2010	J'ai aidé à la mise en place des outils QC et QTP pour quelques tests de non-régression de l'application TEMPETE.
Fin décembre 2010 à mi-janvier 2011	J'ai automatisé dans QTP la création des dumps de référence nécessaire aux tests d'EMMA.
Janvier à février 2011	J'ai ajouté de nouveaux tests de non-régression dans les outils QC et QTP pour l'application EMMA.
Mi-février 2011	La version 2.4 d'EMMA a été livrée et doit être testée.
Mi-février à fin février 2011	J'ai relancé les tests automatisés sur la V2.4 et j'ai analysé les résultats.

1.5 Choix des outils de gestion des tests

Contrairement à la mission de Benoît Courty [COURTY2006], auditeur CNAM ayant effectué un mémoire au sein de RTE sur un thème proche, lorsque je suis arrivée chez RTE, je n'ai pas eu à comparer les différents outils de gestion des tests présents sur le marché et à choisir l'un d'entre eux.

Ce choix a été effectué par les collaborateurs de la Mission Appui Méthodes (MAM). Leurs missions sont de concrétiser, partager et industrialiser les processus des métiers de l'ingénierie et de la maintenance SI de RTE. Ils accompagnent les équipes projet et maintenance SI pour la mise en œuvre des méthodes préconisées.

Leur action de proximité contribue au développement d'une culture partagée au sein de RTE.

1.5.1 Outils du marché

De nombreux éditeurs ont mis en place des outils permettant de gérer les tests à l'aide d'un référentiel et de les rendre automatisés.

Voici les principaux éditeurs présents sur le marché :

1.5.1.1 HP (Mercury)

Racheté par HP en 2006, Mercury reste sans conteste le leader des outils de gestion des tests avec Quality Center. L'outil Quick Test Professional assure l'automatisation des tests fonctionnels et des tests de non-régression.

1.5.1.1 IBM

IBM, après le rachat de Rational en 2003, dispose d'un catalogue d'outils de test particulièrement exhaustif avec des produits tels que Rational Quality Manager qui est un environnement de gestion de test centralisé et Rational Functional Tester pour l'automatisation des tests.

1.5.1.2 Micro Focus (Borland et Compuware)

En 2009, la société Micro Focus a racheté Borland et les activités de tests et d'assurance qualité logiciels de Compuware.

- Borland (Segue)

Borland a construit son offre gestion de la qualité avec le rachat de Segue en 2006. Il propose une gamme de produits pour améliorer les performances et la qualité des applications.

Avec SilkCentral Test Manager, l'éditeur propose un outil permettant de gérer l'intégralité du processus de test des applications. Cet outil propose une intégration transparente avec les autres outils et technologies Borland.

SilkTest est un système d'automation des tests. SilkPerformer est un outil de test de performance et de charge.

- Compuware

Compuware propose avec QACenter une gamme d'outils de test automatisés : gestion des tests avec QADirector, tests de performance avec QALoad, portail de tests avec QAPortal, gestion des spécifications avec Reconcile, test fonctionnel avec TestPartner…

1.5.2 Outils choisis par RTE

C'est en 2007 que MAM a choisi pour outiller ses recettes d'utiliser Rational Functional Tester d'IBM.

Ce choix paraît assez surprenant après avoir lu le mémoire de Benoît Courty [COURTY2006] car celui-ci ne conseille absolument pas ce logiciel. Il explique que cet outil utilise régulièrement les coordonnées de l'écran lors de l'enregistrement du script. Le script est donc impossible à rejouer s'il y a un changement de résolution du navigateur ou une modification dans l'affichage de l'écran de l'application.

Mais ce choix peut s'expliquer par le fait que RTE utilise déjà un outil d'IBM : ClearQuest qui permet de gérer les anomalies rencontrées lors des recettes.

Fin 2007, l'équipe projet de l'application EMMA a voulu automatiser ses tests de non-régression. Ce premier essai a été un échec. Après avoir passé plusieurs mois à se former sur ce nouvel outil et à automatiser quelques tests, il s'est avéré que le re-jeu des tests était impossible, même sur une même version de l'application. L'outil ne reconnaissait pas certains composants des écrans complexes de l'application. De plus une gestion de ces composants par coordonnées n'était pas possible du fait d'une gestion dynamique de ces écrans.

Le second choix effectué par MAM en 2009 a été de référencer les outils QC et QTP d'HP.

L'équipe projet de l'application EMMA a commencé à utiliser QC au premier trimestre 2010, puis QTP à mon arrivée en mai 2010.

2. NOTIONS DE TEST LOGICIEL

2.1 L'importance des tests

2.1.1 Quelques exemples

Il existe de nombreux exemples d'erreurs informatiques (que ce soit de spécification, de conception ou de codage) qui ont empêché le fonctionnement correct d'un programme.

Les conséquences peuvent être bénignes (comme un défaut d'affichage) à gravissimes (mort d'homme).

- Mariner I :

En juillet 1962, la première sonde spatiale interplanétaire du programme Mariner de la NASA, Mariner I, est lancée pour une mission d'analyse de Vénus. Quatre minutes après son lancement, le lanceur dévie de sa trajectoire et doit être détruit.

Cause du problème : une instruction du programme de guidage écrit en Fortran, contenait une virgule à la place d'un point.

- Therac 25 :

La machine Therac 25 chargée de traiter des patients atteints du cancer leur a administré une dose mortelle de radiations.

Cause du problème : un logiciel de contrôle déficient.

- Vol inaugural d'Ariane 5 :

Le 4 juin 1996, le lanceur Ariane 5 explose en Guyane lors de son premier vol de qualification.

Cause du problème : la réutilisation du logiciel de guidage d'Ariane 4 (éprouvé, n'ayant jamais failli) dans Ariane 5 sans analyse des conditions de son fonctionnement ni tests sur simulateur. Or le logiciel ne faisait pas certains tests de débordement car il avait été prouvé que le débordement ne pouvait physiquement pas arriver compte tenu des caractéristiques d'Ariane 4, notamment de sa vitesse. Malheureusement, Ariane 5 était beaucoup plus rapide…

- Mars Climate Orbiter :

La sonde Mars Climate Orbiter devait se mettre en orbite autour de Mars pour prendre des mesures climatiques de la planète. Le 23 septembre 1999, elle s'est désintégrée pendant la mise en orbite, causant un échec irrémédiable de la mission.

Cause du problème : l'une des composantes du système d'altimétrie transmettait les données dans un système d'unités, tandis que la composante logicielle rattachée fonctionnait dans un autre système.

L'enquête a mis en évidence que certains paramètres avaient été calculés en unités de mesure anglo-saxonnes et transmises telles quelles à l'équipe de navigation, qui attendait ces données en unités du système métrique.

Dans tous ces cas, une erreur de réalisation ou de conception d'un logiciel conduit un système automatique à faire autre chose que ce pourquoi il est fait. Différents termes sont employés pour relater ces problèmes, nous allons voir leurs définitions.

2.1.2 Chaine de l'erreur

Selon l'IEEE (Institute of Electrical and Electronics Engineers) : « Le test est l'exécution ou l'évaluation d'un système ou d'un composant, par des moyens automatiques ou manuels, pour vérifier qu'il répond à ses spécifications ou identifier les différences entre les résultats attendus et les résultats obtenus. »

Lorsque les tests sont correctement réalisés et utilisés, ils permettent de mettre en avant des défaillances du logiciel, c'est-à-dire des fonctionnements anormaux aux vues de ses spécifications. [PRADAT2009]

Un défaut dans un logiciel provient d'une erreur humaine qui peut être apparu dans n'importe quel cycle de vie du logiciel (mauvaise compréhension des spécifications, erreur de réalisation lors du codage,…).

Avec l'introduction du défaut dans le logiciel, une défaillance peut apparaître.

Une défaillance est la manifestation du défaut dans le logiciel. Elle se caractérise par des résultats inattendus ou un service non rendu comme une erreur de calcul, un problème de traitement de données,…

Il faut noter que tout défaut ne conduit pas systématiquement à une défaillance [PRADAT2009]. Un logiciel peut comporter un grand nombre de défauts qui, s'ils ne sont jamais exercés en fonctionnement, peut se comporter correctement.

Une suite de défaillance peut entraîner une panne du logiciel ou du système. Une panne se révèle par un arrêt total ou partiel du logiciel.

La figure ci-dessous illustre ce mode de propagation d'une erreur.

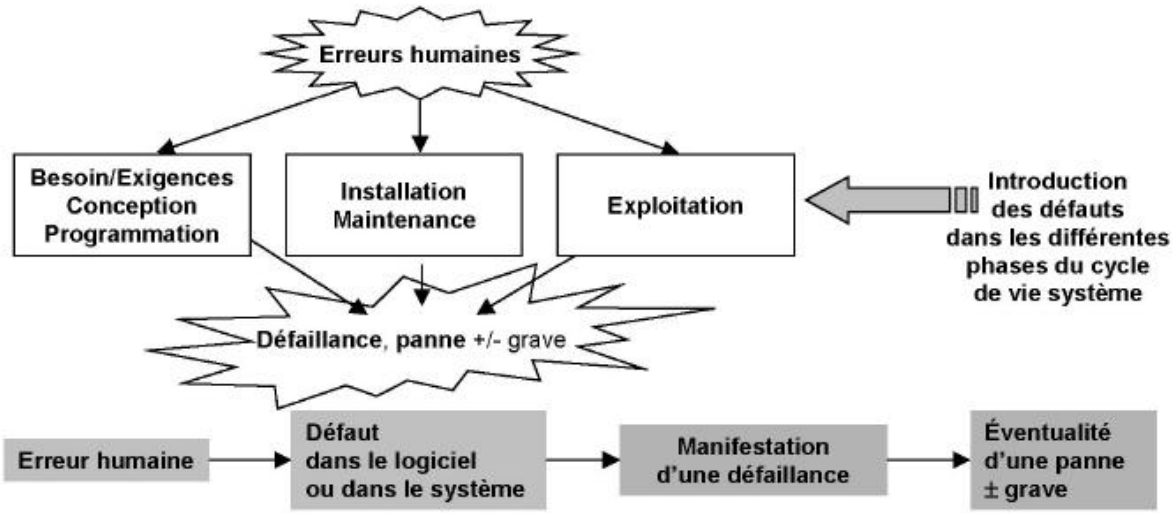


Figure 3 : Chaîne de l'erreur

2.2 Cycles de développement

Les cycles de développement des logiciels prennent en compte toutes les étapes de la conception d'un logiciel.

Il existe différents modèles de cycles de développement : les cycles linéaires (cycle en cascade, cycle en V,…) et les cycles itératifs. Nous allons voir plus en détail le cycle en V car il s'agit de celui qui est utilisé par RTE.

2.2.1 Cycle en V

Le cycle de développement en V est le plus couramment utilisé. [LORTHIOIR2005]

Il est représenté par un V dont la branche descendante contient toutes les étapes de la réalisation du logiciel, et la branche montante toutes les étapes de tests du logiciel. La pointe du V, quant à elle, représente la réalisation concrète du logiciel, le codage.

A chaque étape de conception correspond une étape de test.

Le modèle de cycle en V montre que les tests peuvent intervenir tout au long du processus de développement du logiciel.

Nous pouvons voir ci-dessous la figure du cycle en V.

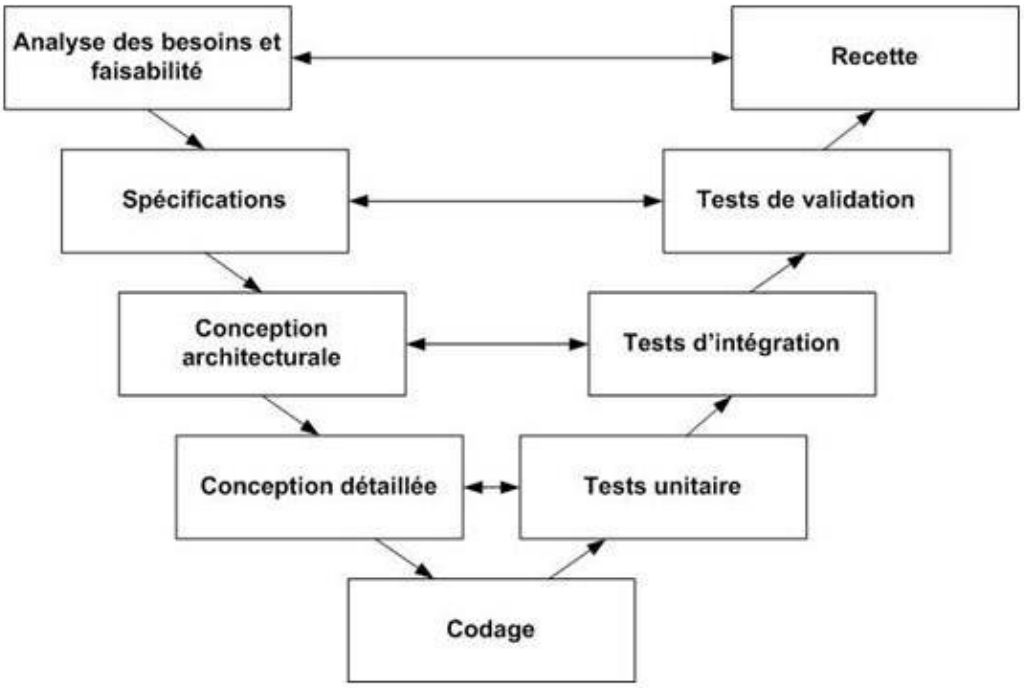


Figure 4 : Cycle en V

Le problème avec le cycle en V est qu'il est parfois difficile à appliquer rigoureusement.

Par exemple, il est quelquefois nécessaire de prendre en compte des changements importants dans les spécifications dans une phase avancée du projet.

Il n'est pas adapté à tous les projets informatiques car certains réclament un peu plus de souplesse.

C'est pour cette raison qu'est apparu la méthode du cycle itératif qui tente de formaliser une approche plus pragmatique et maniable.

2.2.2 Cycle itératif

Ce modèle de cycle de vie prend en compte le fait qu'un logiciel peut être construit étape par étape. Le logiciel est spécifié et conçu dans son ensemble. La réalisation se fait par incréments de fonctionnalités. Chaque incrément est intégré à l'ensemble des précédents et à chaque étape le produit est testé exploité et maintenu dans son ensemble.

Il existe différentes méthodes de cycle itératif : Processus unifié, Développement rapide d'applications, Extreme Programming (XP),…

La figure suivante montre une représentation du cycle de vie itératif.

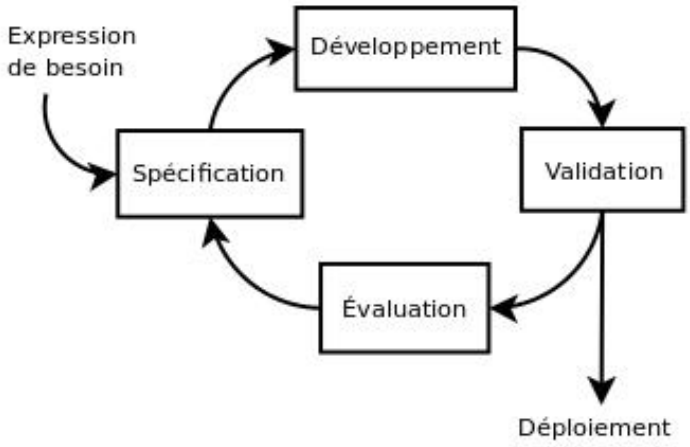


Figure 5 : Cycle itératif

Ce cycle est plus en phase avec la réalité et permet la prise en compte de l'évolution.

Avec une méthode itérative, les tests sont en général réitérés plus souvent, à chaque incrément. Du fait des livraisons plus fréquentes, le test de non-régression est d'autant plus indispensable. [LEGEARD2009]

2.3 Les tests

2.3.1 Les types de test

Nous allons décrire les différents types de tests liés au cycle de développement logiciel.

2.3.1.1 Tests unitaires

Les tests unitaires permettent de s'assurer du fonctionnement correct d'une partie déterminée d'un logiciel ou d'une portion d'un programme. Il s'agit pour le programmeur de tester un module, indépendamment du reste du programme, ceci afin de s'assurer qu'il répond aux spécifications fonctionnelles et qu'il fonctionne correctement en toutes circonstances.

En test unitaire, l'approche de conception des tests sera plutôt de type « boîte blanche » c'est à dire que les tests seront conçus dans un objectif de couverture du code du module testé. [LEGEARD2009]

2.3.1.2 Tests d'intégration

Les tests d'intégration viennent une fois que les tests unitaires ont été validés. Ils ont pour but de vérifier le fait que toutes les parties développées indépendamment fonctionnent bien ensemble de façon cohérente.

2.3.1.3 Tests de validation

Les tests de validation ont pour objectif de s'assurer que le logiciel réalise effectivement tout ce que le client est en droit d'en attendre. Ils permettent de vérifier si toutes les exigences décrites dans le document de spécification du logiciel sont respectées.

2.3.1.4 Tests fonctionnels

Les tests fonctionnels englobent tous les types de tests qui visent, au moins pour une partie, la conformité de l'application testée avec ses exigences fonctionnelles.

Le test fonctionnel est essentiellement de nature « boîte noire », c'est-à-dire que l'application testée est vue au travers de ces interfaces. [LEGEARD2009]

2.3.2 Les tests de non-régression

Les tests de non-régression permettent de vérifier, pour une nouvelle version, que les modifications apportées n'ont pas entraîné d'effets de bord non prévus qui pourraient dégrader le comportement du logiciel antérieurement validé.

Ces tests sont fastidieux, car ils doivent être les plus exhaustifs possibles, afin d'assurer que le logiciel fonctionne de la même manière. Or la nature du phénomène de régression impose de tester à nouveau un grand nombre de fonctionnalités, précédemment testées et validées, alors que les fonctionnalités récemment validées sont peu nombreuses. En particulier, quand il s'agit de corrections d'erreurs, le nombre de tests tend à rester constant, alors même que le nombre de modifications apportées diminue. Le rendement de la phase de test décroît donc rapidement.

2.3.3 Les principes généraux des tests

Tester est une activité complexe qui demande expérience et méthode. [PRADAT2009]

Il existe de nombreux principes indispensables à connaître pour réaliser au mieux cette activité. En voici quelques-uns.

2.3.3.1 Mise en évidence de défauts

Les tests permettent de mettre en évidence la présence de défauts mais ils ne peuvent en aucun cas prouver qu'un logiciel n'a pas du tout d'erreurs. Si aucun défaut n'est découvert, ce n'est pas une preuve qu'il n'en reste pas. [PRADAT2009]

2.3.3.2 Tester le plus tôt possible

Il est important d'effectuer les tests le plus tôt possible dans le cycle de développement du logiciel. La correction d'un bug au moment des exigences est moins coûteuse que cette même correction au moment du déploiement de l'application puisque celles-ci sont de plus en plus difficiles à localiser et impactent une partie du logiciel de plus en plus importante.

C'est ce que nous montre la courbe d'évolution du coût de correction d'une anomalie en fonction de la phase projet dans laquelle elle est détectée.

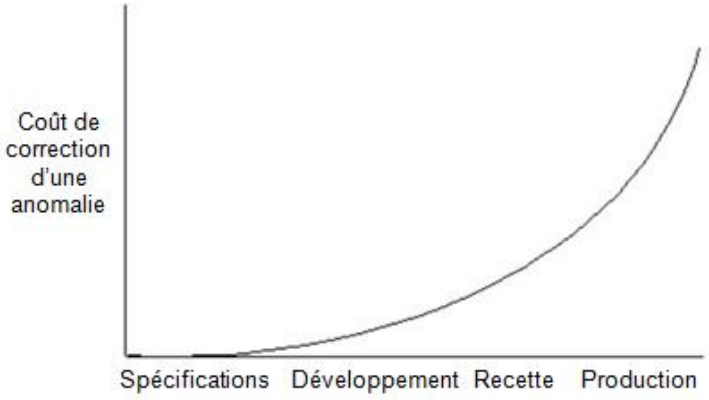


Figure 6 : Evolution du coût de correction

Les études menées sur le coût associé à la détection d'une erreur montre que si une erreur décelée lors de la phase d'élaboration du cahier des charges coûte 1 alors que la même erreur décelée en phase de conception coûte 10 et une erreur décelée en phase d'exploitation coûte 100.

2.3.3.3 Impossible de tout tester

Il existe un nombre inimaginable de combinaisons pour tester les programmes, de ce fait il est impossible de tout tester.

Comme le précise [LORTHIOIR2005], il a été démontré scientifiquement, que la preuve du zéro défaut est un problème indécidable. Il est donc impossible de garantir l'absence de défaut dans les logiciels.

Pour [LEGEARD2009], il ne faut pas tendre vers du logiciel zéro anomalie mais vers un niveau de sureté de fonctionnement et pouvoir maitriser les conséquences d'un dysfonctionnement.

Il est important de définir une stratégie de test. Pour cela il est faut équilibrer les tests boîte noire et les tests boîte blanche et effectuer différentes techniques d'assurance qualité, comme les revues ou les inspections. Les tests doivent être définis dans un périmètre. Pour définir ce périmètre, l'analyse des risques est indispensable.

2.3.4 Les tests dirigés par les risques

Comme il est impossible de tout tester, il faut appliquer une stratégie de couverture fonctionnelle pour maîtriser le triplet qualité/délai/coût. [LEGEARD2009]

Pour ce faire, les tests dirigés par les risques peuvent être utiles car il n'y a pas beaucoup d'intérêt de passer du temps à tester une fonctionnalité pour laquelle l'impact d'une quelconque défaillance serait nul.

Les tests dirigés par les risques consistent à prendre chaque fonction principale de l'application et à définir si celle-ci est souvent utilisée et quel sera l'impact d'un éventuel dysfonctionnement.

Ensuite, le résultat de cette analyse du risque conduit à attribuer une priorité à chaque fonctionnalité.

2.3.4.1 Les facteurs de risque

- Fréquence d'utilisation :

La première étape de l'analyse du risque est de parcourir l'ensemble des fonctionnalités et de définir le degré d'utilisation de chacune d'elles.

Il existe différents degrés :

- Inévitable : Partie de l'application que tous les utilisateurs rencontreront nécessairement.
- Fréquent : Partie de l'application utilisée par les utilisateurs, mais pas dans toutes les sessions.
- Occasionnel : Partie de l'application qu'un utilisateur moyen n'utilisera pas, mais qui peut être utilisée par un utilisateur expérimenté en cas de besoin.
- Rare : Une partie de l'application qui n'apparaît que dans certains cas, dans le cadre d'opérations complexes. La majorité des utilisateurs ne l'utilisent pas, mais elle joue un rôle dans certaines situations.

- Impact :

Ensuite, il faut analyser l'impact d'un dysfonctionnement sur l'utilisateur ou sur l'environnement externe à l'application.

Comme pour la fréquence d'utilisation, il existe plusieurs niveaux :

- Catastrophique : Un problème qui peut causer un blocage de l'ordinateur, l'arrêt complet d'une application ou l'obligation de redémarrer l'ordinateur.
- Grave : Si cette fonction ne marche pas, l'application peut continuer, mais le risque de perdre des données ou d'utiliser des données corrompues est important.
- Modéré : Le problème rencontré perturbe l'utilisateur, mais ce dernier peut le contourner en effectuant des actions complémentaires.
- Ennuyeux : Si cette fonction ne marche pas, on peut continuer à utiliser l'application, mais elle peut poser des problèmes ultérieurement.

Le fait de combiner les fréquences d'utilisation avec les impacts sur les utilisateurs va permettre de classer les fonctionnalités par priorités.

2.3.4.2 Priorité du risque

Pour finir il faut assigner des priorités du risque à chaque fonctionnalité, cela servira à valoriser l'importance que l'on doit leur accorder. Une priorité élevée définira les problèmes qui doivent impérativement être résolus, une priorité moyenne les problèmes qui pourront être résolus ultérieurement, mais qu'il ne faut pas négliger, et enfin une priorité basse les problèmes dont la solution peut attendre et qui seront traités en dernier.

Pour affecter des priorités, il est possible de se baser sur le tableau ci-dessous :

Fréquence / Impact	Inévitable	Fréquent	Occasionnel	Rare
Catastrophique	**Elevée**	**Elevée**	**Elevée**	**Moyenne**
Grave	**Elevée**	**Elevée**	**Moyenne**	**Basse**
Modéré	**Moyenne**	**Moyenne**	**Basse**	**Basse**
Ennuyeux	**Moyenne**	**Basse**	**Basse**	**Basse**

2.3.5 Les tests manuels et les tests automatisés

Il existe deux catégories de tests : ceux effectués manuellement par le testeur et ceux rendus automatisés.

2.3.5.1 Les tests manuels

Les tests manuels sont exécutés manuellement par un testeur d'après un plan de test donné. En cas d'erreurs, le testeur réalise une analyse qui fera l'objet d'une fiche d'anomalie si nécessaire.

L'avantage pour les hommes est de pouvoir interpréter les tests et de les extrapoler. [MACIAK2001]

Mais ces tests ont aussi de nombreux inconvénients, surtout en ce qui concerne les tests de non-régression.

Le temps consacré aux tests de non-régression devient de plus en plus significatif au fur et à mesure des versions livrées de l'application. Cela demande une charge de travail non négligeable dans un délai limité. [LORTHIOIR2005]

De plus, réaliser des tests manuels est une activité laborieuse, longue, coûteuse et peu reproductible.

Au final, les tests manuels conduisent souvent à négliger le test de non-régression pour des raisons de coûts, mais aussi parce que répéter les mêmes tests devient très pénible pour le testeur. [LEGEARD2009]

2.3.5.2 Les tests automatisés

Les tests automatisés permettent de rejouer, autant de fois que l'on souhaite, les manipulations qu'un testeur ferait.

La mise en place de ces tests demande un travail plus conséquent qu'avec des tests manuels mais une fois programmés ces tests sont stables et ils se substituent aux tâches les plus fastidieuses et répétitives. [MACIAK2001].

Par contre, il s'avère que, dans certaines situations, l'automatisation n'est pas adaptée. Pour Tamila Lorthioir [LORTHIOIR2005], il faut rendre les tests automatisés pour les cas suivants :

- Les tests de non-régression dans un cycle itératif :

Comme nous l'avons vu dans la partie précédente, un des inconvénients des tests manuels est le re-jeu des tests de non-régression.

S'il y a de nombreux livrables pour une application, l'automatisation est un atout indéniable.

- Les tests de non-régression suite à une migration technique
- Les tests sur des plateformes différentes
- Les tests de logiciels dans des secteurs critiques

Pour Benoit Courty [COURTY2006], les tests automatisés ont également l'avantage d'être fiables puisque ces tests peuvent vérifier toujours aussi consciencieusement les résultats, même après une énième exécution, ce qui est loin d'être le cas avec un humain.

Avant de se lancer dans une recette fonctionnelle automatisée, il y a de nombreux points à prendre en compte.

Comme l'explique Tamila Lorthioir [LORTHIOIR2005], il n'est pas si évident de faire fonctionner les outils d'automatisation des tests avec les applications de l'entreprise qui contiennent très

souvent des éléments incompatibles avec ces outils. L'entreprise doit donc prévoir un projet « pilote » afin de se rendre compte si ces outils sont en adéquation avec ses attentes et ses logiciels.

Comme nous l'avons vu précédemment, tous les tests ne sont pas automatisables. Il faut définir une stratégie de test et regrouper tous les scénarios et cas de tests manuels et automatisés. Les cas de tests les plus pertinents seront automatisés progressivement en fonction de leur priorité.

De plus l'automatisation est coûteuse par rapport aux tests manuels et le retour sur investissement n'est pas immédiat.

La figure ci-dessous compare les courbes de coût des tests manuels par rapport aux tests automatisés :

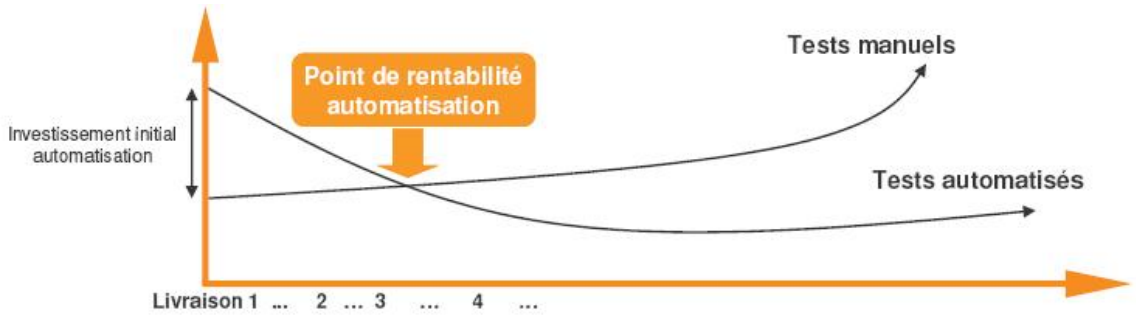


Figure 7 : Rentabilité de l'automatisation

A priori, le retour sur investissement se fait à partir de la troisième livraison, mais tout dépend du coût de l'investissement initial.

L'investissement initial de l'automatisation dépend surtout de la couverture des vérifications que la société souhaite traiter pas les tests automatisés. [VALTECH2006]

Plus la société cherche à étendre la couverture par les tests automatisés, plus la croissance des coûts devient élevée.

Le ticket d'entrée représente le coût initial : coût des licences, installation, formation des testeurs.

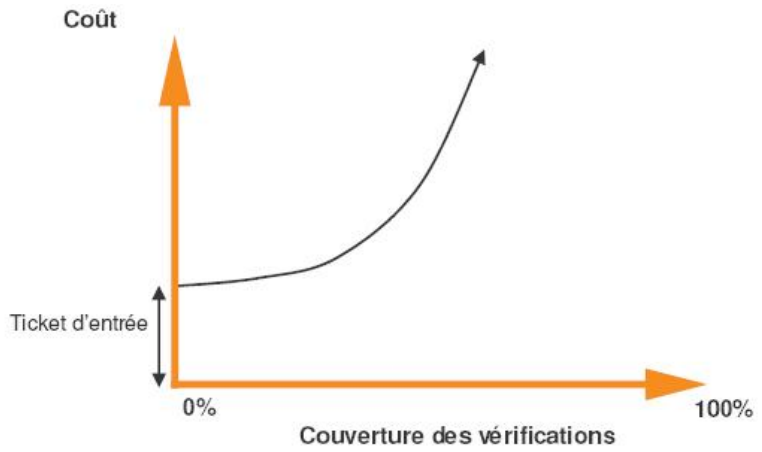


Figure 8 : Courbe coût investissement initial

Un autre point relevé par Tamila Lorthioir [LORTHIOIR2005] est qu'il ne faut pas faire passer l'automatisation au détriment d'autres tests. Si on se concentre uniquement sur l'automatisation des tests de non-régression, on risque de négliger les tests sur les nouvelles fonctionnalités.

L'un des problèmes des tests automatisés est la maintenance. Il s'agit de la raison majeure de l'abandon des outils de tests. Durant la vie du logiciel, des fonctionnalités seront ajoutées, d'autres modifiées. Tous les cas de tests automatisés devront alors être maintenus en fonction des

spécifications. Ceci engendre forcément une charge et un coût de test supplémentaire mais c'est indispensable si l'on souhaite poursuivre le travail d'automatisation.

Pour finir, il ne faut pas croire que les tests automatisés vont éliminer les tests manuels et supprimer totalement les testeurs. Ce sont les testeurs qui définissent et écrivent les cas de test qui seront ou non automatisés. De plus, l'analyse des rapports de tests produits par l'automate nécessite une charge de travail non négligeable afin d'évaluer le résultat. L'interprétation des résultats n'est jamais totalement automatisable, et les outils ne fournissent généralement qu'un niveau d'interprétation trivial.

3. REALISATION D'UNE RECETTE FONCTIONNELLE OUTILLEE

Nous allons voir dans cette partie, comment réaliser une recette fonctionnelle outillée avec les outils choisis par RTE : Quality Center et Quick Test Professional.

Avant de mettre en place ces outils sur les applications, il m'a fallu apprendre à les utiliser et découvrir toutes les fonctionnalités disponibles pour apporter un meilleur savoir-faire aux équipes projet de RTE. Nous allons voir dans ce chapitre les différentes fonctions apportés par ces outils.

Avant toute chose, il faut bien évidemment mettre en place l'infrastructure pour la phase de préparation de l'exécution des tests. Cette infrastructure est à mettre en place dès le début de la recette, elle concerne l'environnement de test (support des versions applicatives) et le choix des données. Nous allons voir un peu plus en détail les données car la qualité des jeux de données influence directement la qualité des tests.

3.1 Les données

3.1.1 Les données en entrée

Les données en entrée doivent être spécifiées pour chaque test. Certaines données peuvent être réutilisées pour plusieurs tests.

Elles se décomposent en deux catégories : les jeux de données et les bases de données de tests.

3.1.1.1 Les jeux de données en entrée

Les jeux de données sont à rentrés dans l'application pour effectuer les tests.

Il peut s'agir de valeurs à saisir ou de fichiers à injecter dans l'application.

La spécification des données est à réaliser au moment de la production des tests c'est-à-dire pendant leur préparation et non pendant leur exécution [LEGEARD2009].

3.1.1.2 Les bases de données en entrée

Les tests fonctionnels doivent s'appuyer sur une base de données spécialement conçue pour le test, et donc indépendante des bases utilisées en production et en développement. Cette pratique permet de garantir la stabilité des données.

Pour [LEGEARD2009], Il n'est pas nécessaire d'avoir la volumétrie d'une base de production ni même une base de grande taille pour les tests. Il suffit que les occurrences de données répondent aux besoins de l'exécution des différents cas de test.

Les données de test peuvent être construites de façon spécifique ou bien extraites d'une base de production, ou combiner des deux approches.

Comme le fait remarquer Tamila Lorthioir [LORTHIOIR2005], la problématique des tests automatisés implique la restauration d'une base de référence toujours identique avant chaque re-jeu.

De plus, une bonne gestion des données ne s'arrête pas à leur simple utilisation.

Il est également nécessaire d'enrichir cette base, de la faire évoluer et de la sauvegarder à la demande. Pour ce faire, il est indispensable de mettre en place des procédures d'import/export de cette base de référence.

3.1.2 Les données en sortie

Les données en sortie correspondent aux résultats attendus. Elles doivent être renseignées pour chaque étape des tests.

Elles peuvent prendre plusieurs formes :

- Une description :

Les données en sortie peuvent être une simple description du résultat attendu dans la fiche de test.

- Une image :

Elles peuvent être représentées par une image de l'application qu'il faudra comparer avec le résultat observé.

- Un fichier :

Il peut également s'agir d'un fichier généré par l'application.

Contrairement aux données en entrée, les données en sortie sont des données spécifiques à chaque test, c'est-à-dire qu'elles ne peuvent pas être réutilisées.

3.2 Utilisation de Quality Center

Le logiciel Quality Center (QC) est un outil de gestion du processus qualité de l'éditeur HP Mercury. Il gère de la collecte des besoins au suivi des anomalies, en passant par la planification, l'organisation et l'exécution des tests.

Pour ce faire, il dispose de plusieurs modules ayant un rôle bien distinct.

3.2.1 Module Management

Le module « Management » permet la gestion des versions de l'application et des cycles de test.

3.2.1.1 Releases et cycles

Sous QC les versions de mise en production sont appelées des releases. Elles sont caractérisées par un ensemble d'exigences qui définit le périmètre complet de la version.

Les cycles correspondent à des étapes de développement et de test d'une version applicative. Dans QC, un cycle est rattaché à une release.

Ces deux éléments sont caractérisés d'une date de début et d'une date de fin de recette. Il est très important de renseigner ces informations car cela permet de suivre la couverture des tests sur la version.

3.2.1.2 Suivi de la recette

L'onglet « Management » dispose de nombreux graphiques pour suivre la progression de la recette. Pour afficher des données correctes, il est impératif de rentrer les dates de début et de fin de la recette.

Pour chaque release et cycle, il y a dans la partie « Progress » un graphique représentant la progression de la couverture des tests. Nous pouvons voir un exemple de ce graphique à la page suivante.

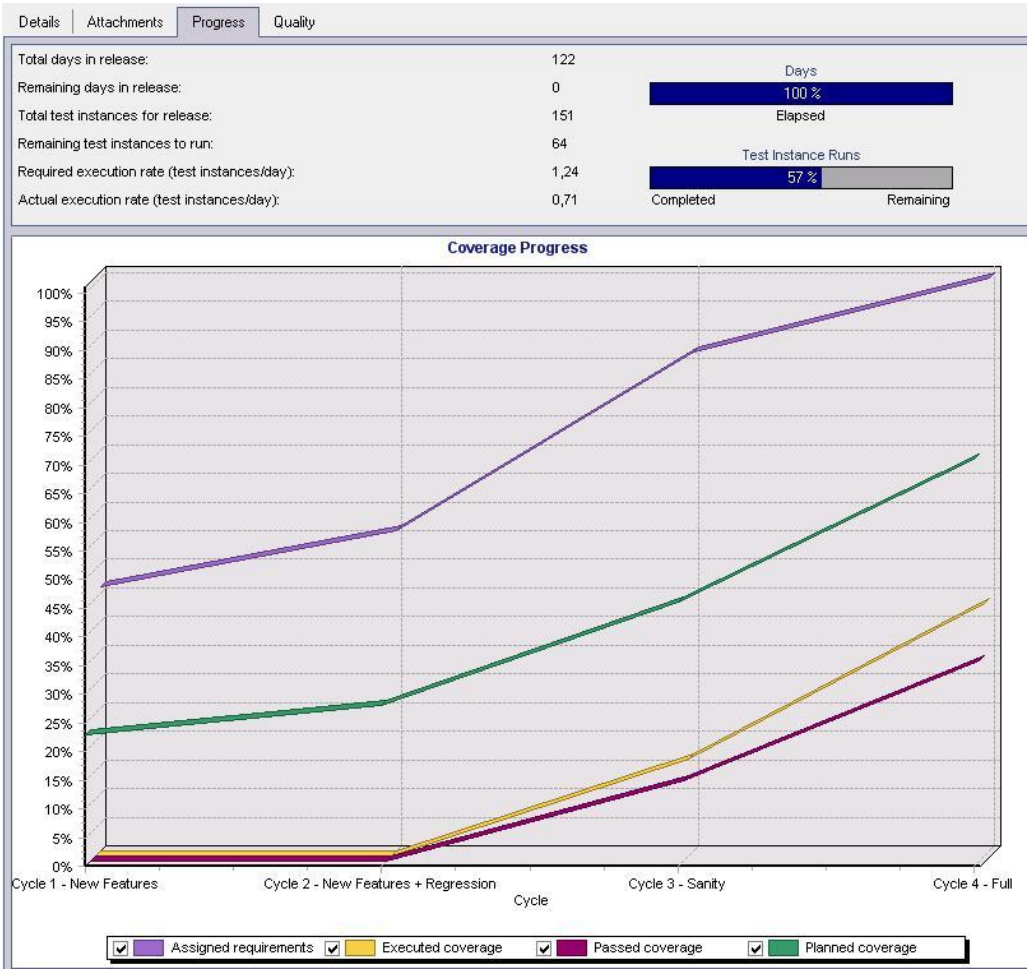


Figure 8 : Progression de la recette dans l'onglet « Management »

3.2.2 Module Requirements

Le module « Requirements » permet de gérer les exigences auxquelles doit répondre l'application.

Il est possible de :

- Gérer l'arborescence des exigences ;
- Créer, modifier et visualiser les exigences ;
- Lier les exigences à un ou plusieurs cycles ;
- Joindre les fiches de tests aux exigences ;
- Evaluer les risques liés à la non satisfaction de l'exigence.

3.2.2.1 Exigences

Pour qu'un test soit valable et efficace, il est nécessaire qu'il réponde aux exigences décrites dans les spécifications du logiciel. Une exigence est l'expression d'un besoin.

Une exigence doit toujours être quantifiable, mesurable, vérifiable et non ambiguë. [RTE]

Dans le référentiel rédigé par l'équipe MAM de RTE, il est précisé que chacune de ses modifications doit être validée par le référent métier.

Les exigences doivent être assez détaillées mais pas trop non plus. Si l'on rentre trop dans le détail cela rendra difficile la navigation et on peu très vite avoir une liste très importante d'exigences. Si l'on ne détaille pas assez les exigences cela génèrera un très grand nombre de fiche de test par exigence, ce qui n'est pas souhaitable car peu modulable pour la suite.

Il existe plusieurs types d'exigences prédéfinis dans QC :

- Les exigences « racines » : « Business » et « Folder » qui correspondent à des répertoires et l'exigence « Group » qui permet de regrouper des exigences entre-elles.
- Les exigences « nœuds » : « Functional » (exigences fonctionnelles) et Testing (exigences non-fonctionnelles).

3.2.2.2 Lien entre les exigences

Une exigence peut être liée à une autre exigence.

Dans l'onglet « Requirements Traceability », nous pouvons visualiser et modifier les relations entre les exigences. Cet onglet permet de voir rapidement les impacts dus à un changement sur une exigence.

3.2.2.3 Lien entre les exigences et les cycles

Il est très important d'associer les cycles de tests aux exigences pour le suivi de la recette.

Pour ce faire, il faut se placer sur un répertoire ou une exigence puis cliquer droit et sélectionner « Assign To Cycle ». Si l'on se place sur une exigence, celle-ci sera la seule à être modifiée. Si l'on se place sur un répertoire toutes les exigences qui se trouvent dans celui-ci seront impactées.

Il est possible de cette manière d'associer ou de dissocier un ou plusieurs cycles à des exigences.

Les cycles étant liés à des releases, celles-ci sont automatiquement associées aux exigences.

3.2.2.4 Lien entre les exigences et les fiches de tests

Dans la partie « Test coverage », nous pouvons visualiser pour chaque exigence les fiches de test associées à celle-ci, ainsi que le taux de couverture des tests. Il est également possible dans cette étape de lier une fiche de tests à une exigence ou de supprimer un lien.

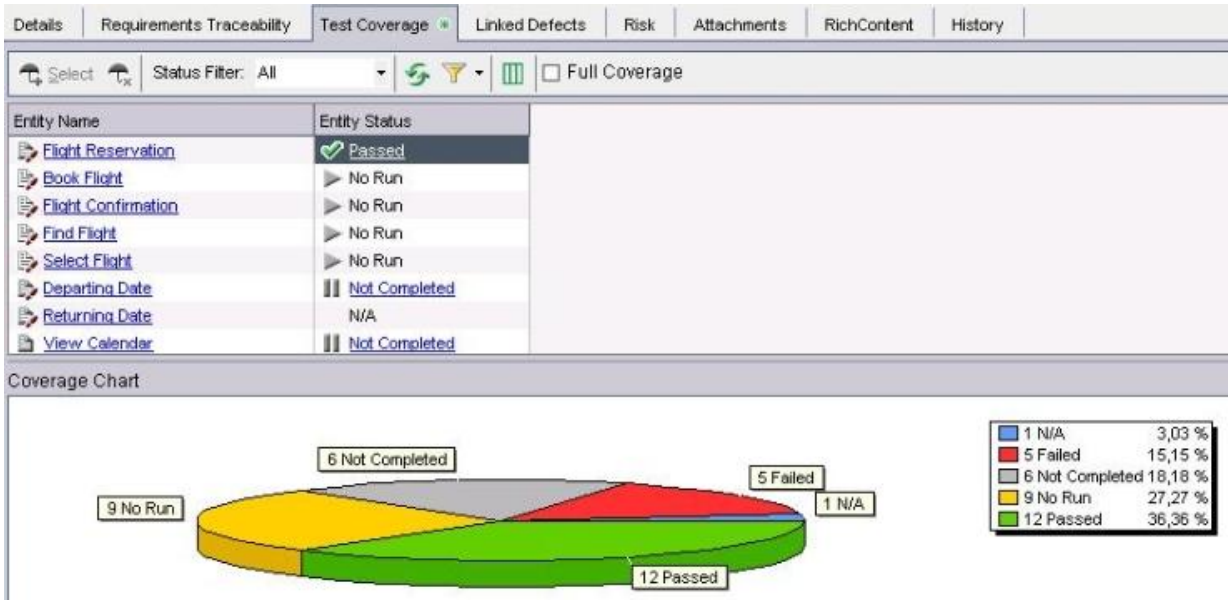


Figure 10 : Taux de couverture des tests dans l'onglet « Requirements »

3.2.2.5 Analyse du risque

Dans l'onglet « Requirements » se trouve une partie « Risk ». Cette fonctionnalité permet d'analyser les risques pour un ensemble d'exigences. Comme nous avons vu dans la partie « 2.3.3 Les tests dirigés par les risques » il n'est pas toujours possible de tout tester lors d'une recette. Dans ce cas il faut tester en priorité les exigences qui poseront le plus de problème en cas de bug.

Pour chaque exigence, il faut renseigner les risques encourus dans l'onglet « Risk Assessment » et la complexité pour cette fonctionnalité dans « Functionnal Complexity ».

Pour les risques encourus, il y a huit valeurs à renseigner qui sont liés à l'exigence :

- Le type de processus : il s'agit d'un calcul complexe, d'une mise à jour de la base de données ou d'une modification simple de l'application ;
- Les conséquences d'une erreur : il y a des conséquences légales, il y a des conséquences graves mais elles ne sont pas légales ou il n'y a pas de conséquences ;
- La fréquence d'utilisation dans l'application : très souvent, souvent ou rarement ;
- Le nombre ou l'importance des utilisateurs : élevé, moyen ou faible ;
- Le type de changement : l'exigence est nouvelle, a changé ou est inchangée ;
- La maturité de l'application : immature, intermédiaire ou mature ;
- Le taux de défauts acceptables : élevé, moyen ou faible ;
- Le nombre d'écrans : supérieur à quatre, entre deux et quatre ou inférieur à deux.

Lorsque tous ces champs sont renseignés, une valeur de A à C est indiquée. La valeur A signifie que le risque est élevé.

Il est possible de saisir directement une valeur sans passer par les différentes listes déroulantes, il suffit de cocher la case « Use custom Risk » et de rentrer la valeur.

Dans l'onglet « Functionnal Complexity », il y a quatre champs à saisir :

- Le nombre d'étapes à accomplir : supérieur à trois, entre deux et trois ou inférieur à trois.
- La quantité de données impactées : élevé, moyen ou faible.
- La dépendance avec d'autres systèmes : dépendances externes, dépendances internes ou pas de dépendance.
- Le niveau de calcul du processus : complexe, moyen ou faible.

Dès que les champs sont saisis, une valeur de 1 à 3 est renseignée. La valeur 1 signifie que la fonctionnalité est complexe.

Il faut effectuer ces étapes pour l'ensemble des exigences.

Puis, en se plaçant sur un répertoire, nous retrouvons l'analyse du risque calculé par QC pour les exigences. Nous pouvons voir un exemple d'analyse du risque à la page suivante.

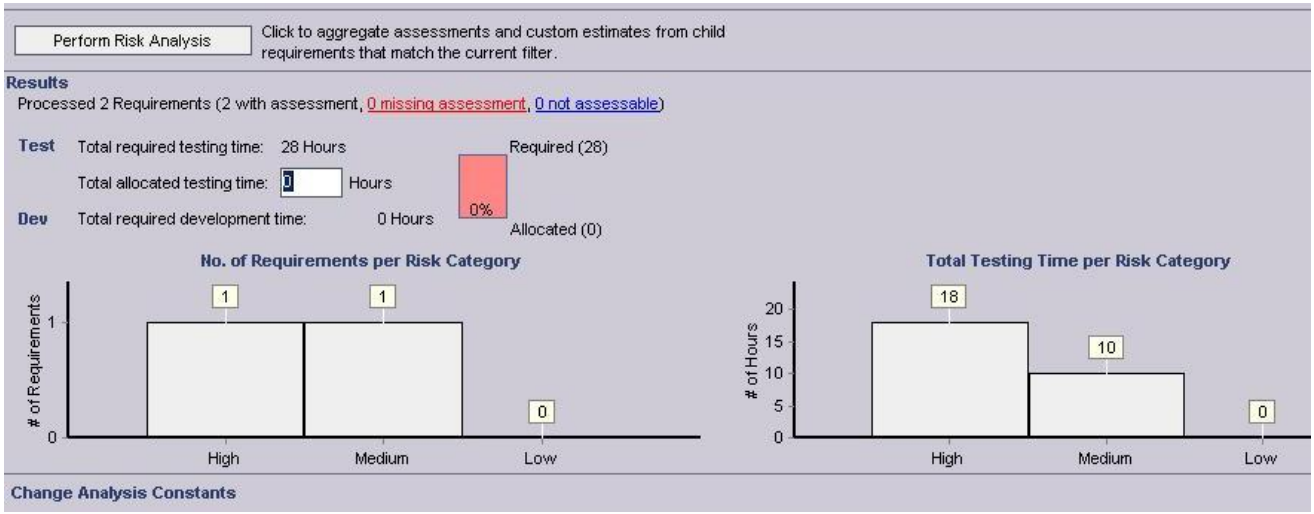


Figure 11 : Analyse du risque par QC

3.2.3 Module Business components

Ce module doit être utilisé pour décrire de façon modulaire des tests de flux métiers.

Il permet de :

- Gérer l'arborescence des composants de tests
- Créer, modifier et visualiser des composants de tests

3.2.3.1 Composant élémentaire

Un composant est une brique élémentaire de test, composée d'un petit nombre d'étapes, de données entrantes et de données sortantes (au niveau de l'objet). La combinaison de ces briques permet de décrire de façon modulaire des cas de test complexes tels que des flux métier.

Les composants ne sont pas exécutables indépendamment, ils doivent être instanciés dans des tests dans les module « Test Plan ».

3.2.3.2 Création d'un composant élémentaire

Il est possible de créer un composant manuel ou automatisé (crée à partir de l'outil Quick Test Professional).

Il faut ensuite renseigner une description, ainsi qu'une pré et une post condition à son déroulement. Certaines informations sont déjà renseignées, comme la date de création, le statut (qui est par défaut « En développement »),…

Des commentaires éventuels peuvent être indiqués dans la partie « Discussion Area ».

3.2.3.3 Paramètres d'entrée et de sortie

Dans l'onglet « Parameters », il est possible de rentrer des paramètres d'entrée et de sortie. Les paramètres peuvent être du type : « String », « Number », « Date », « Boolean » ou « Password ». On peut indiquer une valeur par défaut à ce paramètre.

3.2.3.4 Scénario du composant

Dans l'onglet « Design Steps », le composant doit être décrit plus précisément, pas à pas, avec le nom de l'étape, une description et le résultat attendu.

3.2.4 Module Test Plan

Dans cet onglet seront décrits les tests permettant de valider la conformité de l'application aux exigences.

Ce module permet de :

- Gérer l'arborescence des fiches de test ;
- Créer, modifier et visualiser les fiches de test ;
- Lier les fiches de test à une ou plusieurs exigences ;

3.2.4.1 Fiches de test

Plusieurs types de fiches de test existent :

- « Manual » : fiche de test à exécuter manuellement ;
- « Flow » ou « Business Process » : fiches de test composées d'appel à des composants élémentaires (« Business Components »). Un « Test Flow » est un séquençage de « Business Components », tandis qu'un « Business Process » peut faire appel à des « Business Components » ainsi qu'à des « Test Flows ».
- Automated : fiche de test en lien avec des outils annexes (comme QTP).

Pour chaque fiche de test il faut décrire brièvement le test dans la partie « Détails » en indiquant l'objectif du test, les pré-conditions pour l'exécuter et les post-conditions.

Il faut renseigner les étapes du test pas à pas dans l'onglet « Design Steps », comme pour les composants décrits dans le chapitre précédent.

3.2.4.2 Lien entre les fiches de test et les exigences

Il est primordial de rattacher une fiche de test à une exigence pour pouvoir suivre la recette.

Dans QC il est possible de lier une fiche de tests à plusieurs exigences. Le problème est que si le test est en erreur, il est impossible de déterminer quelle est l'exigence qui est en erreur.

3.2.4.3 Longueur des fiches de test

Les fiches de test doivent être réutilisables et facilement maintenables c'est pour ces raisons qu'il faut que celles-ci soient relativement courtes. Plutôt que de prévoir une fiche de test avec le cas nominal et tous les cas d'erreur, il est préférable de prévoir une fiche de test pour le cas nominal et des fiches de test supplémentaires pour les cas d'erreur.

De plus, dans le cas des tests manuels, où le coût en temps est très élevé, cette pratique est très intéressante. En cas de re-livraison il n'est pas nécessaire de tout re-tester si seulement une partie est à faire, cela permet de faire gagner du temps.

3.2.5 Module Test Ressources

Ce module permet de gérer des ressources externes pouvant être nécessaires pour le test.

Cet onglet est utile si on utilise l'automatisation des tests. Les ressources disponibles peuvent être les objets de l'application, des fonctions externes, la liste des variables d'environnement,…

3.2.6 Module Test Lab

Le module « Test Lab » permet de :

- gérer la planification des tests au sein des Tests Sets (création, ajout d'instances de test, ordre d'exécution) ;
- exécuter des tests (exécution, modification du statut de chaque étape, création de Defect si nécessaire).

3.2.6.1 Tests Sets

Les Tests Sets sont des ensembles de fiches de test que l'on souhaite exécuter. Après avoir crée un Test Set il faut donc sélectionner les fiches de test à passer.

3.2.6.2 Exécution des fiches de tests

Après avoir sélectionné les fiches de test, il est possible de modifier l'ordre d'exécution de celles-ci dans l'onglet « Execution Flow ».

Certains tests peuvent être lancés en parallèle. Par défaut, un test s'exécute lorsque celui qui le précède, quelque soit son statut, est terminé. Il est possible de déclarer qu'un test peut être exécuté uniquement si le test précédent a le statut « passed » (flèches vertes), c'est-à-dire si ce dernier n'a pas d'erreur.

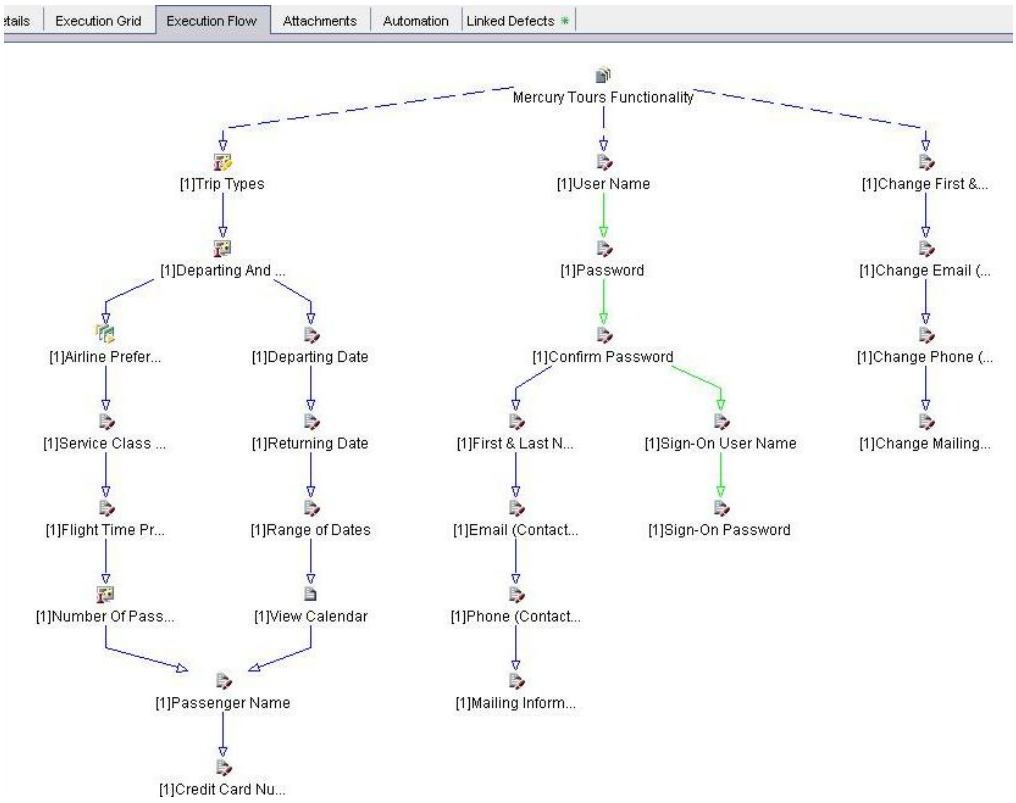


Figure 11 : Flux d'exécution des fiches de tests

Un test peut également être déclenché par un « Time Dependency » en lui spécifiant une date et une heure d'exécution.

3.2.6.3 Exécution des tests

Il y a deux manières d'exécuter des Tests Sets, soit avec le bouton « Run », soit avec « Run Test Set ».

« Run Test Set » permet d'exécuter toutes les fiches de test du Test Set.

Avec le bouton « Run », seulement la ou les fiches de test sélectionnées sont exécutées.

L'exécution des tests se passe différemment si les fiches de test sont manuelles ou automatisés (crées dans QTP).

- De façon manuelle

Pour l'exécution des Tests Sets de façon manuelle, une page apparaît avec le détail du test à lancer (pré-condition,…). Il faut cliquer sur « Begin Run ».

Le détail du test étape par étape s'affiche.

Ensuite il faut modifier le statut en fonction du résultat observé et celui qui est attendu. Si les deux résultats sont différents, le statut doit passer en échec et la description du résultat observé doit être renseignée dans la partie « Actual ».

- De façon automatisée

Pour l'exécution des Tests Sets de façon automatisée, il faut renseigner, pour chaque fiche de tests, la colonne « Planned Host Name » qui contient l'adresse IP de la machine où se trouve QTP puis cliquer sur « Run ».

Une fois que le test est terminé, le statut est renseigné automatiquement et la fenêtre d'exécution des tests peut être fermée.

QTP fournit un rapport de test à QC. Pour l'afficher, il faut double-cliquer sur un des tests. Une fenêtre s'affiche alors avec l'historique de l'exécution de ce test. Il suffit de sélectionner une des exécutions et de cliquer sur « Launch Report » pour que le rapport se lance. Le rapport permet de trouver plus facilement à quel moment le test a échoué.

Lorsqu'un test automatisé est en échec, il faut analyser en détail les erreurs afin de trouver l'origine des écarts obtenus lors du re-jeu.

Plusieurs causes peuvent être découvertes :

- Nouveau script :

S'il s'agit d'un nouveau script, l'erreur peut venir du script lui-même. Dans ce cas, après avoir modifié et recompilé le script, on peut le relancer unitairement afin de valider la correction.

- Evolution non prise en compte :

Si le script a déjà été testé dans une itération précédente, l'erreur est souvent due à une évolution de l'application. Il faut se rapprocher des chefs de projet pour savoir si la régression est apparue suite à la modification d'une règle de gestion. Si c'est le cas, il faut mettre à jour le script. Comme le dit Tamila Lorthioir [LORTHIOIR2005], toute évolution de règle de gestion doit être signalé au plus vite au responsable des tests afin d'anticiper la modification des scripts concernés. En effet il n'est pas souhaitable de découvrir les évolutions tardivement pendant l'exécution d'une campagne de tests.

- Régression fonctionnelle :

Après analyse, si l'erreur dans le script est du à une véritable régression fonctionnelle il faut créer une anomalie.

3.2.6.4 Création d'une anomalie

Suite à l'exécution des tests manuels, si une erreur est découverte dans l'application, il est possible via l'onglet « Test Lab » de créer une nouvelle anomalie.

Cette anomalie sera présente dans le module « Defects ». Dans ce module, il est possible de créer des anomalies pour les tests manuels et automatisés.

3.2.6.5 Automatisation

Dans l'onglet « Automation », plusieurs fonctionnalités sont mises à disposition pour les tests automatisés.

- Envoyer un email

Il est possible d'envoyer un email aux utilisateurs pour les informer du déroulement de l'exécution des tests. Un mail peut être envoyé si un des tests est en échec, s'il y a eu un problème au niveau de l'ordinateur ou dès que le Test Set est terminé.

- Exécution en échec

Si un test automatisé a un statut « Echec », il est possible de paramétrer une ou plusieurs exécutions supplémentaires pour ce test et de préciser l'action à effectuer lors du dernier passage du test en erreur.

Le test en erreur peut être automatiquement ré-exécuté une ou plusieurs fois. Le Test Set peut également être arrêté dès qu'un de ses tests est en erreur.

3.2.7 Module Defects

Ce module permet de tracer les erreurs d'exécution de test.

3.2.7.1 Gestion d'une anomalie

Nous avons vu avant qu'il est possible de créer une anomalie lors de l'exécution manuelle du test. Il est également possible de créer une nouvelle anomalie directement à partir du module Defects. Ce qui est indispensable en cas d'exécution automatique des fiches de test. En effet QTP ne créé pas automatiquement un defect, il faut donc effectuer une petite analyse du rapport QTP afin de créer un defect si nécessaire.

Une anomalie peut avoir une catégorie, une priorité, un statut,…

Une fois que l'anomalie est corrigée, il faut la clôturer dans QC en indiquant une date de fin.

3.2.7.2 Causes d'une anomalie

Une anomalie peut avoir trois causes :

- Un environnement défaillant ;
- Une erreur dans la fiche de test ;
- Un fait technique.

Il est possible de créer une anomalie même lors d'une exécution réussie d'un test, par exemple pour indiquer qu'une fiche de test est trop peu explicite et est à améliorer (le test ne s'en trouve pas pour autant en échec).

3.2.8 Module Dashboard

L'un des intérêts de QC est de permettre le suivi de la recette grâce à toutes sortes de graphiques et de rapports.

Ce module permet de gérer un ensemble d'indicateurs personnalisables (tableau de bord, graphiques) générés à la volée et d'éditer des rapports d'avancement.

3.2.8.1 Utilisation du Graph Wizard

La création de graphique est facilitée grâce à « Graph Wizard ». Cet outil aide à créer pas à pas le graphique que l'on a besoin.

Lors de la première étape, il faut sélectionner le thème (les anomalies, les exigences, l'exécution des tests et les briques élémentaires) et le type de graphique (diagramme circulaire, diagramme en bâtons,…).

Au moment de la deuxième étape, nous pouvons sélectionner ou non un filtre pour réaliser le graphique. Le filtre correspond aux différentes catégories du thème choisi. Si nous choisissons le thème des anomalies, les filtres qui lui sont associés sont la date de création, le statut de l'anomalie,…

Lors de la troisième étape, nous devons choisir les différentes catégories à afficher sur le graphique. Par exemple, une catégorie peut être l'axe des abscisses.

La dernière étape permet de saisir le nom du graphique et de sélectionner un répertoire dans l'onglet Dashboard où le graphique devra être classé.

3.2.8.2 Création d'un graphe manuellement

Il n'est pas obligatoire de passer par l'outil Graph Wizard, nous pouvons créer un graphique tout simplement en sélectionnant « New Graph ».

Nous retrouvons la première et la dernière étape du Graph Wizard (thème, type de graphique et son nom).

Les deuxième et troisième étapes du Graph Wizard sont ensuite regroupées dans l'onglet « Configuration ».

3.2.8.3 Création d'un rapport

- Rapport standard

Pour créer un rapport standard, nous retrouvons quasiment les même étapes que pour la création d'un graphique manuel.

- Rapport Excel

La création du rapport Excel semble assez complexe. Il faut écrire soit même la requête SQL permettant de créer le rapport que l'on souhaite afficher.

Nous avons, quand même, pour nous aider l'outil « Query Builder » qui renseigne toutes les entités (SQL ou utilisateurs) disponibles.

Dans l'onglet « Post-processing », il est possible d'écrire un script en Visual Basic Script permettant d'afficher comme nous le souhaitons le résultat de la requête dans le fichier Excel (graphique, …).

3.2.9 Bibliothèque de tests

QC permet de créer une bibliothèque de tests (Library) et une photographie instantanée de la bibliothèque (Baseline) dans le module « Management ».

Library et Baseline permettent de suivre l'évolution du référentiel de test entre deux jalons, ou entre un jalon et la version courante.

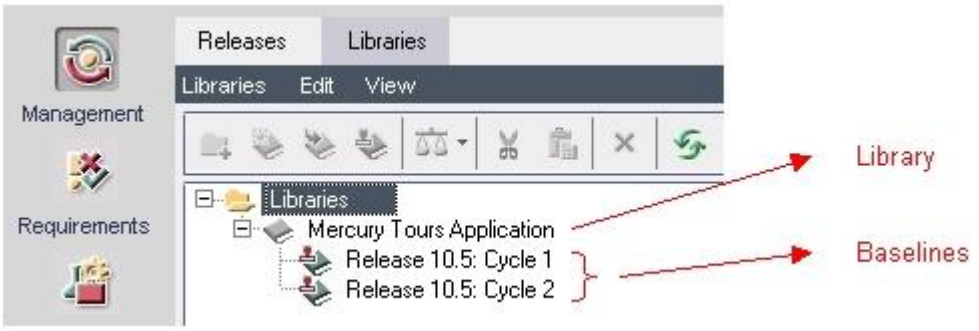


Figure 12 : Bibliothèque de tests

3.2.9.1 Library

La Library définit le cadre de comparaison, il est possible de créer une Library contenant tout le référentiel et d'autres ne couvrant qu'un thème, c'est à dire une sous-partie particulière de l'arborescence, si arborescence d'exigences et arborescence de test sont similaires. (RTE)

Les informations qui peuvent être sélectionné sont les exigences, les composants, les fiches de tests et les ressources.

3.2.9.2 Baseline

La Baseline est la photographie instantanée de la Library.

Il est conseillé, si l'utilisation de cette fonctionnalité est retenue, de créer une Baseline à chaque date clé ou jalon du projet. [RTE]

3.2.9.3 Comparaison de baselines

Il existe un outil permettant de comparer une baseline à une autre ou une baseline aux entités courantes.

Après avoir sélectionné l'outil « Compare to », nous pouvons visualiser les différences (ajouts, modifications, suppressions et déplacements) entre deux versions de l'application.

Les onglets « Requirements », « Components », … sont affichés en fonction de ce qui a été sélectionné dans Library.

Pour visualiser le détail des modifications d'une entité il faut se positionner dessus et sélectionner « Compare Entities ».

3.2.10 Les rôles utilisateurs

Des rôles utilisateurs ont été crées par le service MAM. Il existe des rôles utilisateurs par défaut sur QC mais ceux-ci posaient problème avec les licences achetées par RTE.

Nous allons voir les différents rôles utilisateurs paramétrés par MAM.

Ils donnent à chaque acteur du processus des droits sur chaque objet et son cycle de vie et délimitent son action.

Module QC \ Rôle	Release_Manager	Requirement_writer	Test_Archi	Test_Planner	Tester	Defect_Analyst
Management	Oui	Aucun	Aucun	Aucun	Aucun	Aucun
Requirement	Aucun	Partiel	Partiel	Aucun	Aucun	Aucun
Business Component	Aucun	Aucun	Oui	Aucun	Aucun	Aucun
Test Plan	Aucun	Aucun	Oui	Aucun	Aucun	Aucun
Test Ressources	Aucun	Aucun	Oui	Aucun	Aucun	Aucun
Test Lab	Aucun	Aucun	Aucun	Partiel	Partiel	Aucun
Defect	Aucun	Aucun	Aucun	Aucun	Partiel	Partiel
Dashboard	Oui	Oui	Oui	Oui	Oui	Oui

Oui	Tous les droits (sur les objets d'un module)
Partiel	Droits partiels (une partie des actions sur les objets)
Aucun	Aucun droit (lecture seule)

La visibilité des modules et des objets est directement liée aux rôles.

Module QC \ Rôle	Release_Manager	Requirement_writer	Test_Archi	Test_Planner	Tester	Defect_Analyst
Management	Oui	Oui	Non	Oui	Non	Non
Requirement	Oui	Oui	Oui	Oui	Non	Non
Business Component	Non	Non	Oui	Non	Non	Non
Test Plan	Non	Non	Oui	Non	Non	Non
Test Ressources	Non	Non	Oui	Non	Non	Non
Test Lab	Non	Non	Non	Oui	Oui	Non
Defect	Non	Non	Non	Non	Oui	Oui
Dashboard	Oui	Oui	Oui	Oui	Oui	Oui

3.2.10.1 Rôle Release_Manager

Le rôle « Release Manager » est attribué au responsable du projet de test. C'est lui qui aura la possibilité de créer et d'organiser les objets Release, Cycle, Library et Baseline, qui permettent de suivre l'avancée des travaux de recette.

3.2.10.2 Rôle Requirement_Writer

Le rôle « Requirement Writer » est attribué à un référent fonctionnel (de l'équipe projet). Il saisit les exigences dans l'outil Quality Center (dans l'équipe projet, c'est peut-être lui qui décline les exigences métier en exigences fonctionnelles et système).

Il organise l'arborescence des exigences en prêtant une attention particulière à la typologie des exigences (cela influe sur les indicateurs standards QC).

Il a pour responsabilité de faire évoluer les exigences à chaque version de l'application, de vérifier les impacts sur les objets liés (essentiellement les tests) et de prévenir les rédacteurs de test (rôle Test_Architect) afin qu'ils répercutent les modifications si nécessaire.

3.2.10.3 Rôle Test_Architect

Le rôle « Test Architect » permet :

- La conception des fiches de tests ;
- La conception des composants de test ;
- L'organisation de l'arborescence des composants et des fiches de tests ;
- La création de liens entre exigences et tests.

C'est un rôle pivot au sein du projet de test qui nécessite un acteur ayant des compétences à la fois fonctionnelles et techniques.

3.2.10.4 Rôle Test_Planner

Le rôle « Test Planner » permet de planifier la réalisation des campagnes de tests à exécuter lors des livraisons.

Il crée les Tests Sets, y affecte les instances de test, les valeurs des jeux de données et les testeurs. Il affecte les Tests Sets aux cycles de test pour la remontée des indicateurs en direction du Release Manager.

3.2.10.5 Rôle Tester

Le rôle « Tester » permet d'enregistrer le déroulement des tests définis par le « Test Planner » dans les Tests Sets et de saisir à tout moment une anomalie (Defect QC) en relation avec une fiche de test pour analyse par le « Defect Analyst ».

3.2.10.6 Rôle Defect_Analyst

Le rôle « Defect Analyst » permet d'effectuer l'analyse des anomalies de test détectées par les testeurs.

Il qualifie la cause de ces anomalies et le cas échéant fait modifier les fiches de test, les jeux de données de test, ou l'environnement de test. Si le constituant est la cause de l'anomalie de test, il crée la fiche de fait technique correspondante dans l'outil de Gestion des Faits Techniques (ClearQuest) et répercute sur le Defect le numéro de cette fiche.

3.2.10.7 L'attribution des rôles

Le demandeur, en lien direct avec MAM, décide de l'attribution des rôles aux membres de l'équipe projet. Les rôles peuvent être portés par un ou plusieurs membres de l'équipe, mais une bonne pratique consiste à avoir autant d'acteurs différents que possible (selon les contraintes du projet).

Les utilisateurs et les rôles sont définis à la création du projet QC, ou lors de l'arrivée d'un nouvel acteur dans l'équipe projet de test.

3.3 Utilisation de Quick Test Professional

Quick Test Professional (QTP) est un automate de test. Il enregistre et répète toutes les actions nécessaires à la réalisation d'un test :

- Les manipulations de clavier et de souris
- Les opérations de contrôle des résultats obtenus par rapport aux résultats attendus

Le langage utilisé pour les scripts QTP est du Visual Basic Script.

3.3.1 Ouverture de QTP

3.3.1.1 Sélection des add-ins

A l'ouverture de QTP il faut sélectionner les modules complémentaires (Add-ins). Ces modules correspondent au langage de programmation de l'application à tester (.NET, Java, Oracle, Web,…). Il est possible de sélectionner plusieurs add-ins.

3.3.1.2 Connexion avec Quality Center

La connexion entre QC et QTP se fait très simplement à partir de QTP. Il faut cliquer sur l'icône « Quality Center Connexion » et saisir les identifiants utilisés dans QC.

3.3.2 Création d'un test automatisé

Il existe plusieurs manières de créer un test automatisé.

- Soit à partir de QTP en créant un nouveau test. Pour que ce test se trouve dans QC, au moment de l'enregistrement, il faut sélectionner l'onglet « Quality Center Test Plan ».
- Soit à partir de QC :
  - En créant un nouveau test de type « quicktest_test » dans l'onglet « Test Plan »
  - En sélectionnant « generate_script » dans la partie « Design Steps » d'un test manuel. Les différents Steps seront alors des commentaires dans le test QTP.

3.3.3 Enregistrement d'un script

Pour enregistrer un script, il faut tout d'abord cliquer sur « Record » de l'outil QTP.

Une fenêtre apparait avec différents onglets. Ceux-ci sont les types d'applications disponibles pour le test. Ils se mettent à jour en fonction des modules complémentaires (add-ins) renseignés à l'ouverture de l'outil.

Il faut donc choisir le type d'application à tester, indiquer si l'application est déjà ouverte ou sélectionner un fichier exécutable,…

Ensuite il faut exécuter manuellement le test, pendant ce temps QTP enregistre les différentes actions effectuées.

3.3.3.1 L'enregistrement en mode normal

L'enregistrement en mode normal est l'enregistrement par défaut.

QTP reconnaît les objets d'une application prise en charge, détermine le type d'objet et leur attribut une classe (Bouton radio = WinRadioButton, liste déroulante = WinComboBox).

Ensuite, il enregistre les actions effectuées sur les objets.

La liste des objets et leurs propriétés est sauvegardée dans un « Object Repository » (référentiel d'objets).

Les propriétés d'un objet sont un ensemble de caractéristiques définissant l'aspect, les valeurs, l'état et l'identité d'un objet au sein d'une application. Les propriétés d'objet sont directement extraites de l'API de l'application.

Le référentiel d'objets peut être utilisé pour :

- Modifier les noms des objets ;
- Ajouter un nouvel objet ;
- Configurer les propriétés utilisées pour identifier un objet (ajout/suppression de propriétés).

La figure ci-dessous nous montre comment ajouter des propriétés sur un objet.

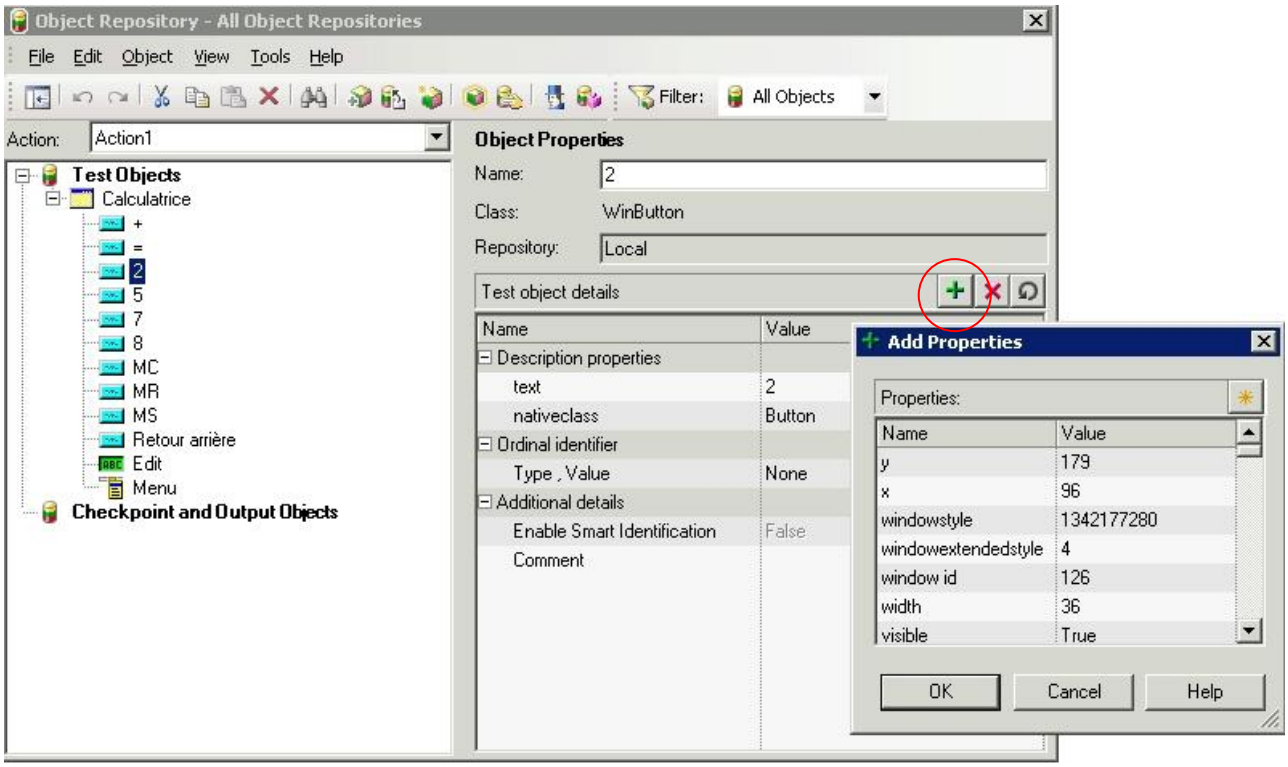


Figure 13 : le référentiel d'objets de QTP

3.3.3.2 Les vues de l'enregistrement

L'enregistrement du script peut être visualisé de différentes manières.

- Vue par objets

L'onglet « Keyword View » permet de voir l'enregistrement en fonction des objets de l'application.

Cette vue contient un tableau disposant de plusieurs colonnes :

- Item = nom de l'objet ;
- Operation = action effectuée sur l'objet ;
- Value = valeur de l'objet ;
- Documentation : détail de l'action.

Nous pouvons distinguer facilement les différents types d'objet (liste déroulante, bouton, …), comme nous le montre la figure 14.

Il est également possible de rajouter, à partir de cet écran, des actions sans passer par l'application en cliquant sur « Select an item ». Il faut choisir un objet et lui associer une action et une valeur. Si tous les objets sont enregistrés dans QTP, il n'est plus nécessaire de passer par l'application pour scripter un test.

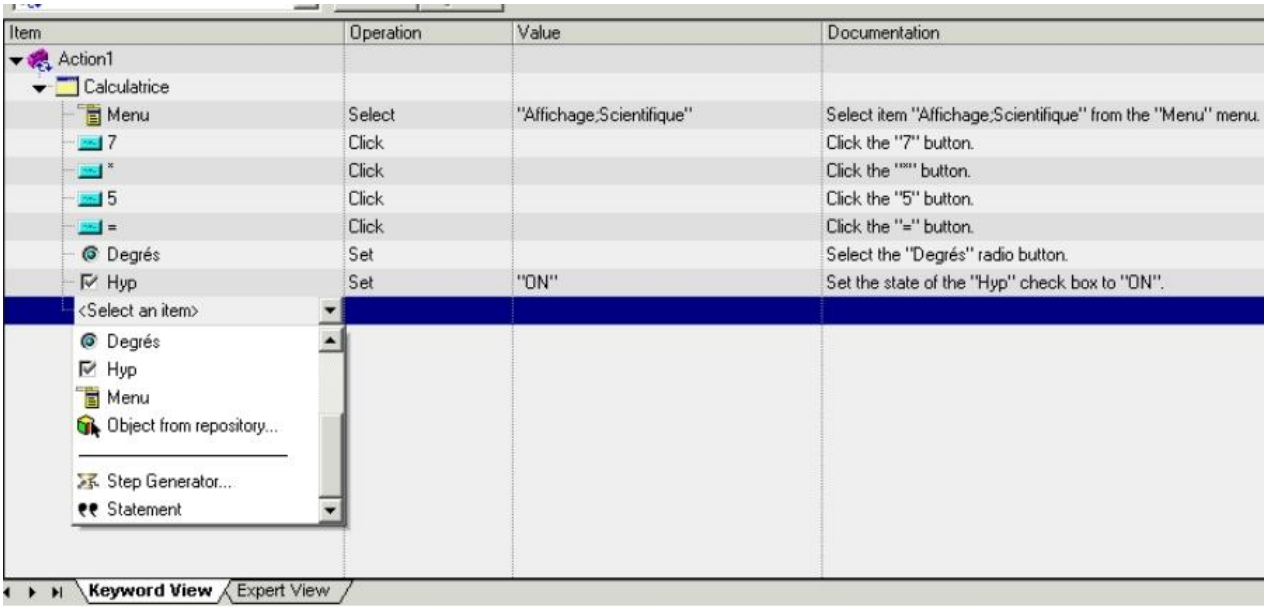


Figure 14 : Vue par objets de QTP

- Vue experte

Pour l'onglet « Expert View », il s'agit d'afficher l'enregistrement avec le langage Visual Basic Script.

Figure 15 : Vue experte de QTP

- Ecran actif (Active screen)

Dans cette partie, nous pouvons visualiser une copie de l'application pour chaque étape de l'enregistrement.

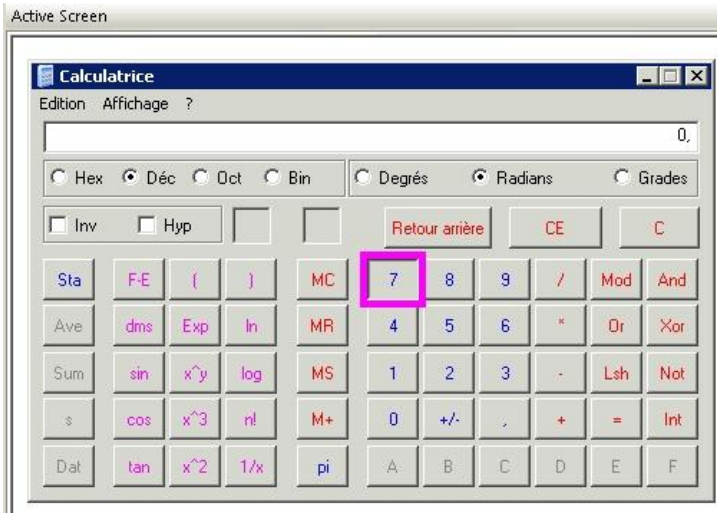


Figure 16 : Ecran actif de QTP

3.3.3.3 Les enregistrements « Low Level » et « Analog »

Si l'enregistrement en mode normal de QTP ne permet pas d'enregistrer correctement l'action, il est possible d'utiliser un des deux modes suivants :

- Enregistrement « Low level »

Ce mode permet d'enregistrer tous les objets et opérations à l'aide des coordonnées de l'application.

Voici un exemple de script généré par cet enregistrement.

```
Window("Calculatrice").WinObject("5").Click 13,11
Window("Calculatrice").WinObject("+").Click 20,13
Window("Calculatrice").WinObject("6").Click 18,15
Window("Calculatrice").WinObject("=").Click 15,12
```

- Enregistrement « Analog »

Avec l'enregistrement « Analog », QTP enregistre avec exactitude tous les déplacements de souris et entrées au clavier liés à un écran ou une fenêtre.

Il faut sélectionner l'application à tester puis effectuer les différentes actions.

Dans ce cas, le script est vu comme une seule action et n'est pas modifiable.

```
Window("Calculatrice").RunAnalog "Track1"
```

Ces deux modes sont à éviter car ils sont facilement obsolètes, soit suite à des modifications d'IHM, soit si ils tournent sur un poste avec une résolution différente. Toutes ces raisons peuvent poser des problèmes lors du re-jeu.

3.3.4 Les points de contrôle

Les points de contrôle (checkpoints) permettent de garder en mémoire les résultats attendus par l'application. Ils peuvent être insérés soit à l'enregistrement du script soit à partir de l'écran « Active Screen ».

Il en existe plusieurs sortes sous QTP.

3.3.4.1 Standard Checkpoint

Le « standard checkpoint » est utilisé pour vérifier les propriétés d'un objet. Il permet également d'enregistrer les valeurs d'un tableau se trouvant sur l'application.

- Sur les propriétés d'un objet

Après avoir sélectionné ce type de checkpoint, il faut cliquer sur l'objet que l'on souhaite tester.

QTP reconnaît le type de l'objet et par conséquent les propriétés qui lui sont attribuées. Il suffit de cocher les propriétés à vérifier.

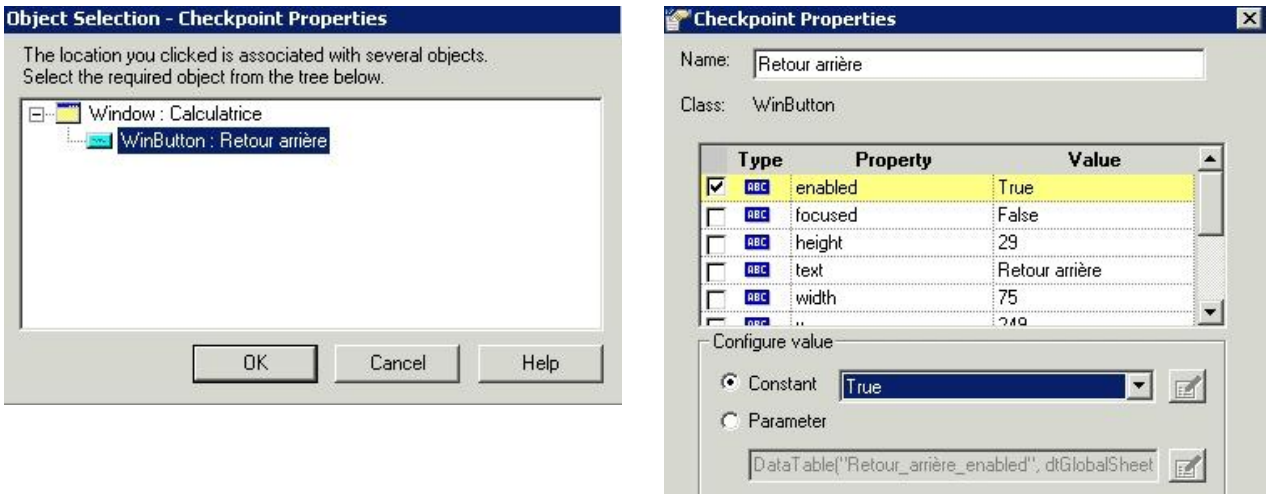


Figure 17 : Point de contrôle sur un objet

- Sur un tableau

Lorsque l'on souhaite vérifier les valeurs d'un tableau, il faut également choisir le « checkpoint standard ». Après avoir sélectionné le tableau à tester, l'écran suivant s'ouvre.

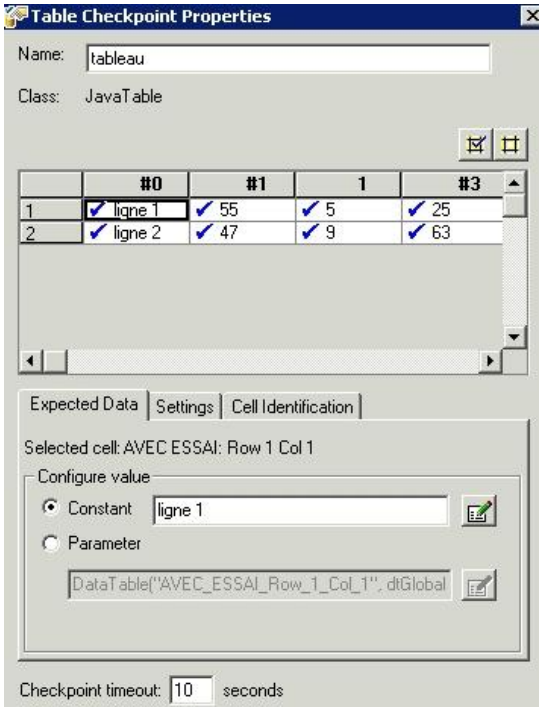


Figure 18 : Point de contrôle sur un tableau

Il est possible de sélectionner seulement quelques valeurs du tableau en cochant ou décochant certaines cellules.

3.3.4.2 Text Checkpoint

Les checkpoints « text » et « text area » permettent de contrôler un texte ou un groupe de texte.

En plus de vérifier le texte sélectionné, ces checkpoints peuvent aussi tester les mots qui entourent celui-ci.

3.3.4.3 Bitmap Checkpoint

Le « Bitmap Checkpoint » peut être utile pour vérifier des graphiques. Il suffit de sélectionner « Bitmap Checkpoint » et de cliquer sur une partie de l'application. Avec l'option « Check only selected area », nous pouvons choisir uniquement une partie de l'image.

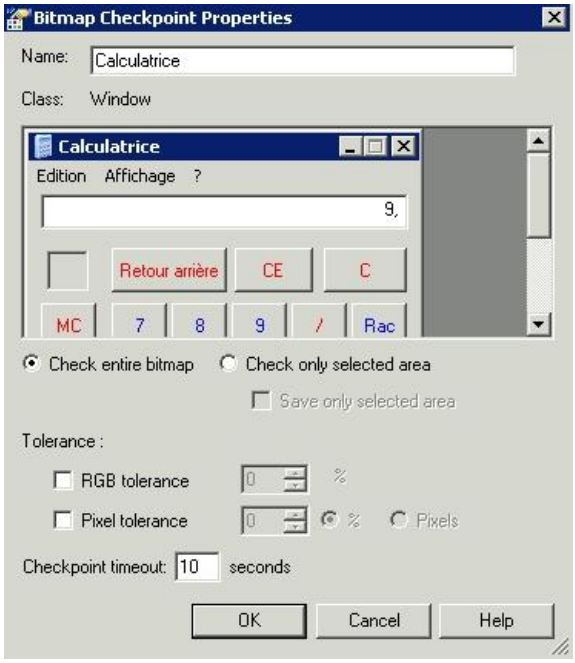


Figure 19 : Point de contrôle sur une image

3.3.4.4 Database Checkpoint

Pour effectuer une vérification sur la base de données, il faut sélectionner « Database Checkpoint ». Pour réaliser une requête, il y a le choix entre l'utilisation manuelle ou à l'aide de l'outil graphique Microsoft Query. Cet outil permet de sélectionner les tables et les champs que l'on souhaite ajouter au checkpoint. Avec la méthode manuelle, il suffit de saisir une requête SQL de sélection.

Dans les deux cas, il faudra sélectionner une source de données ODBC vers la base de données de l'application.

Les propriétés de ce checkpoint ressemblent à celles d'un tableau. Il est possible de cocher ou décocher des cellules du tableau.

3.3.4.5 Accessibility Checkpoint

Ce checkpoint permet de vérifier qu'une page Web est conforme aux règles du W3C (World Wide Web Consortium). Ce consortium international a pour but de promouvoir l'évolutivité du Web et de garantir son Interopérabilité.

Pour vérifier que la page Web est bien conforme, il faut sélectionner ce checkpoint, cliquer sur une page Web. Ensuite QTP coche les éléments qui peuvent être vérifié sur la page.

3.3.4.6 XML Checkpoint

Il y a deux manières de vérifier un fichier XML, soit en cliquant sur un fichier XML déjà ouvert, soit en envoyant le chemin du fichier à QTP.

Ensuite, QTP affiche les données du fichier. Il est possible de sélectionner les valeurs à tester.

3.3.5 La synchronisation d'objets

La synchronisation consiste à ajouter une étape dans un script de test qui indique à QTP d'attendre l'apparition d'un objet particulier avant de passer à l'étape suivante.

La synchronisation garantit la réussite de la ré exécution car le script diffère toute action jusqu'à ce que l'application soit dans l'état adéquat pour poursuivre.

Certains objets nécessitent un temps de traitement plus long :

- Une barre de progression doit atteindre 100% ;
- Un message de statut doit apparaître ;
- Un bouton doit être activé ;
- Une fenêtre ou un message contextuel doit s'ouvrir.

Pour ajouter un point de synchronisation, il faut attendre que l'objet apparaisse puis il faut cliquer dessus.

Cette fenêtre apparaît, elle permet de sélectionner la propriété de l'objet à attendre, de saisir une valeur et un temps d'attente.

3.3.6 Gestion des données

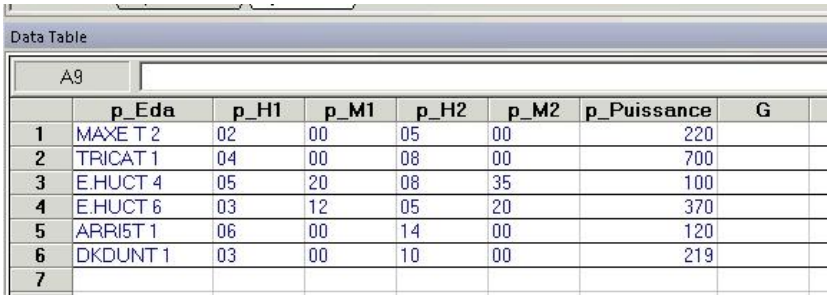
QTP propose plusieurs manières de gérer les données dans les tests.

Un test dirigé par des jeux de données désigne un test qui exécute un ensemble d'actions utilisateur avec plusieurs valeurs d'entrée. Grâce à l'utilisation de jeux de données, un seul script peut tester les fonctionnalités d'une application à l'aide d'un grand nombre de données différentes.

3.3.6.1 Table de données

QTP intègre directement une table de données interne (« DataTable ») matérialisée sous la forme d'une feuille de calcul type « Excel ». C'est sur cette feuille qu'il est possible de stocker les données à utiliser dans les tests (données d'entrée et données de sortie). L'importation et l'exportation du fichier des données est possible via l'outil. Il est également possible de renommer les colonnes pour associer une suite de valeurs à un champ à saisir plusieurs fois.

L'utilisation de paramètres d'entrée permet de se servir d'une plage de valeurs plutôt que d'une valeur codée en dur dans le script. Il est possible de ré-exécuter un même script pour tester plusieurs jeux de données d'entrée.



	p_Eda	p_H1	p_M1	p_H2	p_M2	p_Puissance	G
1	MAXE T 2	02	00	05	00	220	
2	TRICAT 1	04	00	08	00	700	
3	E.HUCT 4	05	20	08	35	100	
4	E.HUCT 6	03	12	05	20	370	
5	ARRI5T 1	06	00	14	00	120	
6	DKDUNT 1	03	00	10	00	219	
7							

Figure 20 : Table de données

Chaque ligne de données génère une itération du script. Pour paramétrer les itérations, il faut aller dans la partie « Run » du « Test Settings ». Il est possible de sélectionner le nombre de lignes à exécuter.

Pour ajouter un paramètre d'entrée dans la « DataTable », il faut enregistrer le script de test puis sélectionner une des valeurs et cliquer sur ses options de configuration. Il suffit de sélectionner le paramètre « DataTable » et donner un nom à la variable.

Voici le résultat de l'ajout de ces paramètres dans le script QTP :

```
JavaWindow("Saisie d'une consigne").JavaList("EDA :").Select DataTable("p_Eda",
dtLocalSheet)
JavaWindow("Saisie d'une consigne").JavaList("j").Select DataTable("p_H1",
dtLocalSheet)
JavaWindow("Saisie d'une consigne").JavaList("h").Select DataTable("p_M1",
dtLocalSheet)
JavaWindow("Saisie d'une consigne").JavaList("j_2").Select DataTable("p_H2",
dtLocalSheet)
JavaWindow("Saisie d'une consigne").JavaList("h_2").Select DataTable("p_M2",
dtLocalSheet)
JavaWindow("Saisie d'une consigne").JavaEdit("Puissance (MW)").Set
DataTable("p_Puissance", dtLocalSheet)
```

QTP permet aussi de créer un paramètre de sortie pour sauvegarder la valeur d'une propriété d'un objet. Lorsque le test s'exécute, QTP extrait la valeur actuelle de cette propriété et l'intègre à la table de données en mode Exécution, en tant que valeur de sortie. Cette valeur de sortie peut ensuite être utilisée comme variable d'entrée dans le test. Cela permet de réutiliser les données récupérées dans d'autres parties d'un test.

3.3.6.2 Variables d'environnement

QTP peut aussi insérer dans le script une valeur provenant de la liste de variables d'environnement. Cette liste contient des variables avec un nom et une valeur. Elle est accessible à partir du test en interne ou exportée à partir d'un fichier XML.

3.3.6.3 Variables aléatoires

QTP peut générer des chiffres aléatoires et les insérer en tant que valeurs d'un paramètre. Il est possible de choisir l'intervalle de ce chiffre et de produire un nouveau chiffre à chaque appel du paramètre, à chaque itération ou à chaque exécution du test.

3.3.7 Scénario de reprise

Des évènements et des erreurs inattendus, survenant au cours de l'exécution d'un test, peuvent fausser la procédure de test.

QTP dispose d'un gestionnaire de scénarios de reprises qui permet de :

- Détecter et gérer l'apparition d'une boite de dialogue d'erreur particulière ;
- Mettre en œuvre un scénario de reprise, si une erreur survient, permettant de poursuivre l'exécution du test ;
- Créer un scénario de reprise à l'aide de l'assistant « Recovery Scenario Wizard ».

La première étape est de sélectionner l'évènement déclenchant l'interruption du test : ouverture d'une fenêtre, propriétés et valeurs d'un objet, affichage d'un message d'erreur,…

Suivant l'évènement déclencheur sélectionné, il faut identifier la pop-up ou le type d'erreur,…

Ensuite, il faut indiquer à QTP comment se comporter après avoir découvert l'évènement déclencheur. Il est possible de cliquer sur un bouton, de fermer un processus, d'appeler une fonction,…

La dernière étape est de préciser à QTP ce qu'il doit faire après avoir surmonter l'évènement déclencheur, en indiquant le mode de reprise souhaité : redémarrer le test de puis le début, passer directement à l'étape suivante,…

Le scénario sera associé au test en cours et pourra être intégré aux autres tests.

3.3.8 Débogage et exécution du test

3.3.8.1 Déboguer un script

QTP dispose d'un module de débogage de scripts.

Il est possible d'ajouter ou de supprimer des points d'arrêt sur une ligne du script. Un point d'arrêt suspend l'exécution du test lorsque QTP l'atteint. Il est possible d'exécuter le script ligne par ligne et d'afficher la valeur d'une variable.

3.3.8.2 Exécuter un script

Il y a plusieurs paramètres à prendre en compte avant d'exécuter pour la première fois un script QTP.

Il est possible de capturer les écrans présentant des erreurs ou avertissements lors de l'exécution ou d'enregistrer la vidéo des écrans précédant les erreurs ou avertissements survenus lors de l'exécution.

En fin d'exécution du script, le fichier des résultats d'affiche. Ce rapport détaille l'exécution du test et montre le résultat des points de contrôles. Les erreurs sont facilement détectables. S'il y a une erreur dans un tableau de valeurs, les cellules erronées disposent d'une croix rouge. De plus le résultat attendu est renseigné.

Ce rapport peut être imprimé ou exporté en fichier HTML ou PDF.

Nous pouvons voir un exemple de rapport en « Annexe A ».

4. AUTOMATISATION DES TESTS DE L'APPLICATION EMMA

4.1 Présentation d'EMMA

4.1.1 Contexte

La conduite du réseau de transport d'électricité consiste à assurer en temps réel l'équilibre entre consommation et production d'électricité.

Plusieurs types d'incidents peuvent perturber cet équilibre (unité de production défaillante, ligne de transport endommagée, écart de consommation). RTE doit alors faire appel aux producteurs et aux consommateurs directement connectés au réseau de transport d'électricité pour qu'ils modifient très rapidement leur programme de fonctionnement.

RTE a donc besoin de connaître en permanence les différentes solutions disponibles ainsi que les conditions techniques et économiques de leur mise en œuvre pour retrouver le niveau d'équilibre du réseau de transport afin de déterminer équitablement les meilleures solutions.

C'est l'objet du "Mécanisme d'ajustement" que RTE a mis en place en avril 2003. L'application EMMA fournit de nombreuses fonctionnalités nécessaires à sa mise en œuvre.

4.1.2 Mécanisme d'ajustement

Les propositions d'ajustement (Offres) sont remises par les participants ayant signé un contrat d'adhésion au Mécanisme d'ajustement.

On distingue deux types d'offres :

- Offre à la hausse : augmentation de production / diminution de consommation ;
- Offre à la baisse : diminution de production / augmentation de consommation

Pour une Entité d'Ajustement (EDA : entité qui injecte ou soutire de l'énergie sur le réseau électrique), une offre est composée systématiquement :

- d'un sens d'ajustement ;
- d'une période, qui peut-être soit une journée complète, soit une plage prédéfinie ;
- d'un prix unique ;
- de conditions d'utilisation.

Les offres sont prises en compte par RTE à des heures précises, appelées guichets. A chaque période d'ajustement J correspondent n guichets (nombre et heure paramétrables). Les guichets sont paramétrés comme suit :

- trois guichets la veille, en J-1 : 16h00, 22h00 et 23h00 ;
- vingt-deux guichets temps réel en J : 0h00, 1h00, 2h00, 3h00, 4h00, 5h00, 6h00, 7h00, 8h00, 9h00, 10h00, 11h00, 12h00, 13h00, 14h00, 15h00, 16h00, 17h00, 18h00, 19h00, 20h00 et 21h00.

En fonction des besoins d'ajustement, du résultat de l'interclassement par préséance économique, et des contraintes techniques, RTE sélectionne des offres et envoie des ordres à l'acteur d'ajustement (un ordre est l'activation d'une offre pendant une période donnée à une puissance donnée).

Les offreurs finalement appelés ont alors à mettre en œuvre les moyens correspondants.

Les offres sélectionnées sont rémunérées au prix d'offre pour la durée pendant laquelle il y est fait appel.

En parallèle, en fonction des prix des offres retenues pour l'ajustement, le coût de règlement des écarts pour les responsables d'équilibre est défini.

Enfin, le contrôle du réalisé compare l'énergie attendue à l'énergie livrée.

La figure ci-dessous nous montre le déroulement du mécanisme d'ajustement.

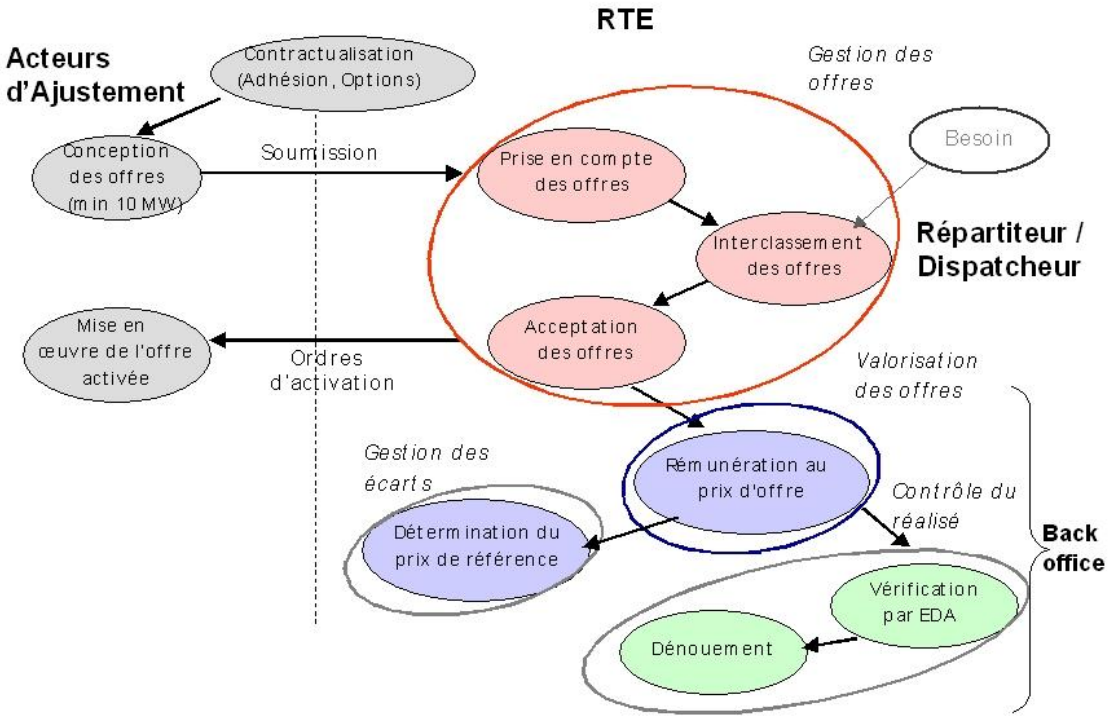


Figure 21 : Mécanisme d'ajustement

4.1.3 Modules

EMMA est composée des modules suivants :

- EMMA Traçabilité permet essentiellement :
  - D'élaborer les programmes de marche des moyens de production et de consommation à partir des déclarations de programmes et des offres d'ajustement transmises par les différents acteurs externes à RTE ;
  - D'enregistrer les événements qui permettent à RTE de justifier les ajustements demandés.
- EMMA Validation et EMMA Historique permettent de corriger a posteriori les données saisies dans EMMA Traçabilité.
- EMMA Bouclage complète EMMA Traçabilité en permettant aux utilisateurs du CNES :
  - D'acquérir les programmes d'appel (PA) des responsables de programmation (RP)
  - De réaliser le « Bouclage » c'est-à-dire :

    - En J-1 de prendre en compte les ajustements nécessaires pour lever les contraintes réseaux et garantir les marges suffisantes.

    - En infra-journalier de faire en sorte que Production = Consommation tout en évitant les contraintes réseaux et en garantissant des marges suffisantes.

  - De diffuser les programmes de marche (PM) aux applications avales.

- EMMA Régions permet aux dispatcheurs en région de voir les modifications de programmes (PA ou PM) sur les entités de leur région.
- EMMA Administration permet de modéliser des moyens de production, de consommation et d'échange et de configurer l'application.

4.1.4 Architecture

4.1.4.1 Architecture applicative

Trois environnements sont nécessaires en production :

- TR (Temps Réel) : Environnement réalisant les processus J et J-1 ;
- VALID : Environnement réalisant le processus de validation a posteriori (dix jours de données modifiables) ;
- BO : Environnement de consultation qui contient un historique de dix-huit mois de données.

Le « basculement » est, le moment du changement de jour qui provoque les traitements suivants dans la BDD de l'environnement TR :

- J est déplacé dans l'environnement VALID ;
- J-1 devient J ;
- Création J-1 vierge.

Le basculement se produit chaque jour quelques minutes après minuit. Il modifie la plupart des tables de la base de données.

4.1.4.2 Architecture technique

L'application EMMA est basée sur une architecture 3-tiers J2EE et contient :

- Une partie serveurs :
  - Des serveurs d'application Weblogic 9.1. pour les traitements applicatifs ;
  - Un serveur Oracle RAC 10.2 pour la gestion des données.
- Quatre parties clientes :
  - Tracnes pour les modules EMMA Traçabilité, Validation et Historique ;
  - Bouclage pour le module EMMA Bouclage ;
  - URSE pour le module EMMA Régions ;
  - Tracadmin pour le module EMMA Administration.

L'environnement J-1/J est déployé sur deux serveurs d'application dédiés (un normal et un secours passif).

Les environnements Validation et Historique sont déployés sur le même serveur d'application dans deux domaines Weblogic distincts.

Chaque environnement utilise un schéma de base de données dédié. Tous ces schémas sont hébergés par un Oracle RAC mutualisé.

4.1.4.3 Authentification

Les mécanismes d'identification et d'authentification des utilisateurs sont pris en charge par une connexion LDAPS à l'annuaire AAA.

Le client EMMA se connecte sur le serveur EMMA dans l'environnement pour lequel il est configuré (fichier parametre.properties). Il transmet (protocole RMI) au serveur d'application les données d'authentification (login, mot de passe). C'est ensuite le serveur qui se connecte à AAA.

Les droits d'accès sont administrés dans et par l'application. Les droits dépendent du profil de l'utilisateur.

Un mode secours a été mis en place pour pallier les indisponibilités potentielles de l'annuaire AAA. Seuls les utilisateurs déclarés directement dans EMMA peuvent se connecter en mode secours.

4.1.4.4 Flux

L'application est alimentée quotidiennement par :

- des données provenant des différents acteurs du marché d'ajustement ;
- des données produites par d'autres applications de RTE.

Elle alimente à son tour les services internes du CNES et les URSE pour le suivi des programmes et des ajustements. Les programmes d'appel, les programmes de marche simulés et les programmes de marche officiels sont fournis aux applications avales via la création de fichiers de sortie.

Les échanges (internes ou externes) de données sont assurés :

- par transfert FTP de fichiers ;
- par partage de fichiers stockés sur un espace disque dédié qui communique avec le serveur EMMA via le logiciel SAMBA (partage des données au travers du réseau entre Windows et Linux). Ces fichiers peuvent ensuite faire l'objet d'un échange vers d'autres applications ;
- par file JMS (l'interface Java Message Service permet d'envoyer et de recevoir des messages entre applications Java) ;
- par flux OPALIS, SGW ou EAI.

4.1.4.5 Ordonnancement

Un scheduler (ordonnanceur) spécifique à l'application EMMA a été développé en java pour déclencher les traitements applicatifs récurrents (import ou génération de fichiers de données par exemple).

4.1.5 Livraisons

EMMA étant une application critique, il est indispensable de s'assurer du bon fonctionnement de celle-ci avant qu'elle soit mise en production.

On peut schématiser le processus de recette comme suit :

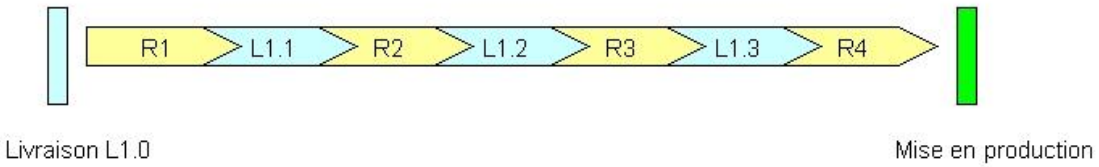


Figure 22 : Processus de recettes d'EMMA

Ou L X.X sont les livraisons de l'application et RX les périodes de recette.

Entre la livraison et la mise en production il se passe environ quatre à cinq mois. C'est le temps qu'il faut pour trois à quatre personnes pour tester les évolutions et la non-régression de l'application.

Voici le déroulement type des étapes pour la mise en production d'EMMA :

- Définition du périmètre prévisionnel ;
- Expression de besoins ;
- Spécifications : rédaction des FFT ;
- Analyse et chiffrage détaillé ;
- Définition du périmètre définitif ;
- Préparation recette : plans de tests ;
- Réalisation des évolutions ;
- Recette usine (tests réalisés avant la livraison) ;
- Correctifs ;
- Recette métier ;
- Mise en production.

Le détail du planning des versions 2.2, 2.3 et 2.4 d'EMMA sur lesquelles j'ai travaillé se trouve en « Annexe C ».

4.1.6 Les environnements de test

Il existe plusieurs environnements de tests pour EMMA car les recettes se font en parallèles. L'équipe d'EMMA dispose de deux serveurs (Lame 1 et Lame 2) ayant chacun trois instances (VALID, BO et TR). Chaque personne de l'équipe utilise un schéma avec un serveur Weblogic et une base de données qui lui sont propres.

La version d'EMMA qui se trouve sur ces schémas est celle à tester, sauf pour « EMMA_VALID_2.0 » qui dispose de la version en production. Ce schéma est utilisé en cas de bugs dans cette version.

Serveur	Schéma	Version EMMA	Utilisateur
Lame 1	EMMA_TR	Vx	Cécile
	EMMA_VALID	Vx	Aurélien
	EMMA_BO	Vx	Jean-Philippe
Lame 2	EMMA_TR_2.0	Vx-1	Gestion des incidents
	EMMA_VALID_2.0	Vx	Cathy
	EMMA_BO_2.0	Vx	Emmanuelle

4.2 Mise en œuvre de Quality Center

Lorsque j'ai commencé mon stage au sein de RTE, l'équipe d'EMMA venait de commencer à utiliser l'outil Quality Center. Après avoir passé du temps sur la prise en main de l'outil et à répertorier les différentes fonctionnalités de l'outil, je les ai conseillé dans la mise en place de l'outil.

Pour la mise en œuvre de Quality Center sur le projet EMMA, les onglets que l'équipe projet et moi-même avons principalement utilisé sont « Management », « Requirements » et « Test Lab ».

Nous n'avons pas utilisé les « Business Components » car cela ne m'a pas semblé être le plus adapté à l'application EMMA. Les « Business Components » correspondent à des briques élémentaires de l'application. Leur principal intérêt est qu'un composant peut transmettre des données à un autre composant. Cette solution n'a pas d'intérêt pour l'application EMMA puisque ses tests sont petits et spécifiques et donc très difficile à réutiliser.

En ce qui concerne le module « Test Ressources », j'ai utilisé des ressources externes pour l'automatisation des tests. Elles seront détaillées dans la partie « 4.4 Mise en œuvre de Quick Test Professional ».

Le module « Defects » ne sert pas car RTE utilise l'outil Clearquest pour le suivi des anomalies. Les anomalies faisant partie de la catégorie « Faits technique » sont obligatoirement saisies dans cet outil. J'ai quand même conseillé à l'équipe projet d'EMMA de saisir ce défaut dans QC car cela permet d'obtenir une traçabilité des anomalies rencontrées et d'avoir un suivi correct.

Pour l'onglet « Dashboard », il est pour le moment peu utilisé car la mise en place de QC se fait progressivement.

4.2.1 Module Management

4.2.1.1 Organisation

Une bonne organisation est primordiale pour le suivi de la recette. J'ai réfléchi à une meilleure organisation possible pour l'application EMMA. Cette organisation pourra également être appliquée pour d'autres applications.

Pour le module « Management », j'ai commencé par créer un répertoire avec le nom de l'application. Dans celui-ci, j'ai ajouté des répertoires pour chaque version majeure de l'application (V2.2, V2.3…).

Pour chaque version majeure il existe des versions intermédiaires (V2.2.1, V2.2.6,...). J'ai décidé d'indiquer une release dans QC uniquement pour les versions intermédiaires faisant l'objet d'une mise en production.

Par exemple, dans le schéma au début du chapitre « 4.1.5 Livraisons », nous aurions une seule release L1.0 alors que la version mise en production serait la L1.3 et toutes les campagnes de test R1 à R4 porteraient sur cette release mais dans des cycles différents. En fait nous ajoutons des cycles à la release en cas de re-livraison.

Les releases contiennent les différents cycles de test. Les cycles de test correspondent à la non-régression, à des nouvelles fonctionnalités ou à des corrections d'anomalies.

Nous pouvons voir ci-dessous, une copie d'écran de l'onglet « Management » d'EMMA.

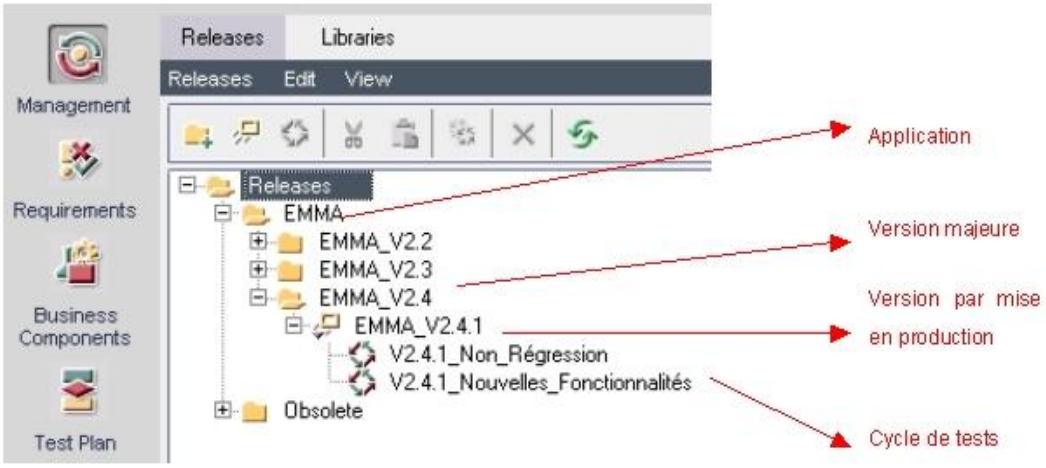


Figure 23 : Module « Management » d'EMMA

4.2.1.2 Nommage

Les releases et les cycles seront utilisés dans les autres onglets, il est très important de les repérer facilement.

C'est pour cette raison qu'il faut définir une règle de nommage.

Pour EMMA, j'ai décidé, avec l'accord de l'équipe projet, de faire contenir le numéro de version dans le nom des releases et des cycles. Pour le nom des cycles, j'ai préféré faire commencer celui-ci par le numéro de version à cause de sa longueur.

4.2.2 Module Requirements

4.2.2.1 Organisation

Pour lister les exigences, j'ai commencé par créer un répertoire avec le nom de l'application, comme pour l'onglet « Management ». Puis il a été décidé de séparer les exigences de non-régression et celles des évolutions. Nous verrons plus en détail la raison de cette décision dans la partie suivante.

Le répertoire de non-régression contient l'ensemble des exigences de l'application. Dans la partie évolution il y a les FFT (Fiches de faits techniques) qui correspondent aux nouvelles fonctionnalités, correction d'anomalies,… pour la version de l'application à tester. Ces FFT seront redirigées vers la partie de non-régression à la fin de la recette soit en modifiant une exigence déjà existante soit en créant une nouvelle exigence.

Dans le répertoire de non-régression se trouvent les différents clients d'EMMA. Ceux-ci disposent de macro exigences qui contiennent les exigences.

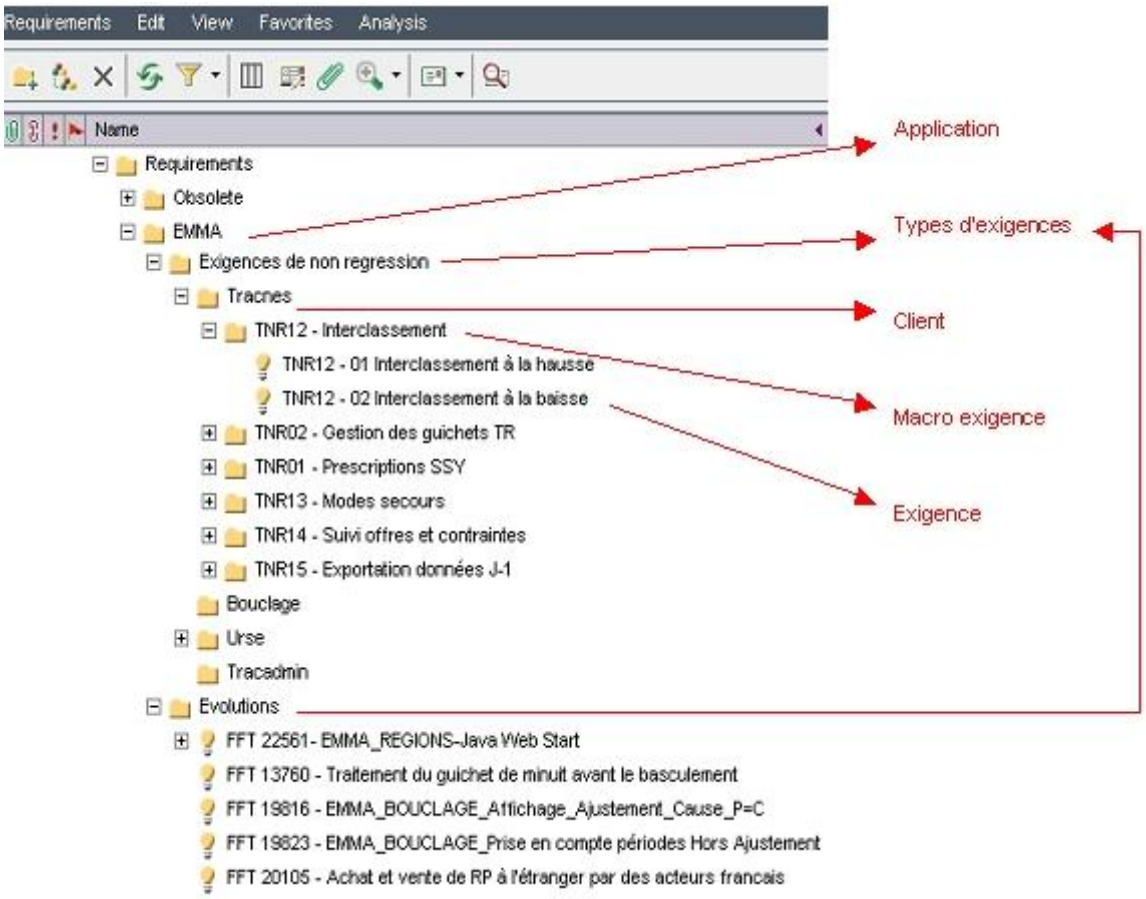


Figure 24 : Module « Requirements » d'EMMA

4.2.2.2 Problèmes rencontrés

L'organisation des exigences m'a posé problème car celle-ci doit prendre en compte la complexité des évolutions d'EMMA et la facilité à maintenir les exigences.

Lors de la mise en œuvre de QC pour EMMA, l'ensemble des exigences (évolution et non-régression) se trouvait dans un même répertoire. Avec l'équipe projet, nous avons découvert lors de la recette de la V2.3 que cette méthode ne pouvait pas convenir à l'application EMMA.

En effet la V2.3 a été une recette longue et lors de celle-ci l'équipe projet a reçu un patch à apporter sur une exigence de la version V2.2 qui se trouvait en production pour une correction d'anomalie.

Cette exigence a été changée dans la V2.3 et n'est plus du tout conforme à celle de la V2.2. Cette situation n'est pas rare pour le projet EMMA et est très difficile à intégrer dans QC car il faut garder les deux exigences jusqu'à la mise en production de la V2.3.

Le service MAM m'a conseillé d'utiliser les notions de Library et Baseline de QC (« 3.2.9 Bibliothèque de tests »). Ce fonctionnement m'a paru compliqué pour les non initiés à QC ce qui sera fréquemment le cas de l'équipe projet car celle-ci change régulièrement.

Avec l'accord de l'équipe projet, j'ai préféré séparer les exigences de la V2.2 (Exigences de non-régression) et celles de la V2.3 (Evolutions). Le problème avec cette solution c'est qu'à chaque fin de recette il faudra passer les exigences les plus critiques du répertoire « Evolutions » dans celui de « Non-régression ».

4.2.2.3 Nommage

Comme pour les releases et les cycles, il faut réfléchir à une règle de nommage pour retrouver facilement les exigences dans les autres onglets de QC.

Pour les exigences de non-régression, j'ai choisi d'utiliser la première lettre du client (« T » pour Traçabilité, « B » pour Bouclage,…) puis « NR » pour indiquer que l'exigence est dans la partie non-régression et enfin une série de numéro. On ajoute un numéro à deux chiffres a chaque fois que l'on ajoute un niveau dans l'arborescence de l'exigence. Pour les évolutions, le nom de l'exigence commence par le numéro de FFT.

4.2.3 Module Test Plan

4.2.3.1 Organisation

Le choix de l'organisation des fiches de test a été simple car j'ai choisi d'utiliser la même arborescence que pour les exigences car chaque exigence dispose d'une ou plusieurs fiches de test.

Le premier répertoire est le nom de l'application. Ensuite les évolutions et les tests de non-régression ont été séparés. Dans ces répertoires se trouvent les macro exigences qui contiennent les exigences. Les exigences contiennent une ou plusieurs fiches de test.

Dans QC il est possible de lier une fiche de test à plusieurs exigences. Cette solution m'a semblé compliquée pour le suivi de la recette. En effet si une fiche de test se retrouve en erreur, il est impossible de savoir laquelle des exigences est en erreur. J'ai donc informé l'équipe projet qu'une fiche de test devait correspondre à une seule exigence.

La figure suivante représente l'organisation des fiches de test d'EMMA dans QC.

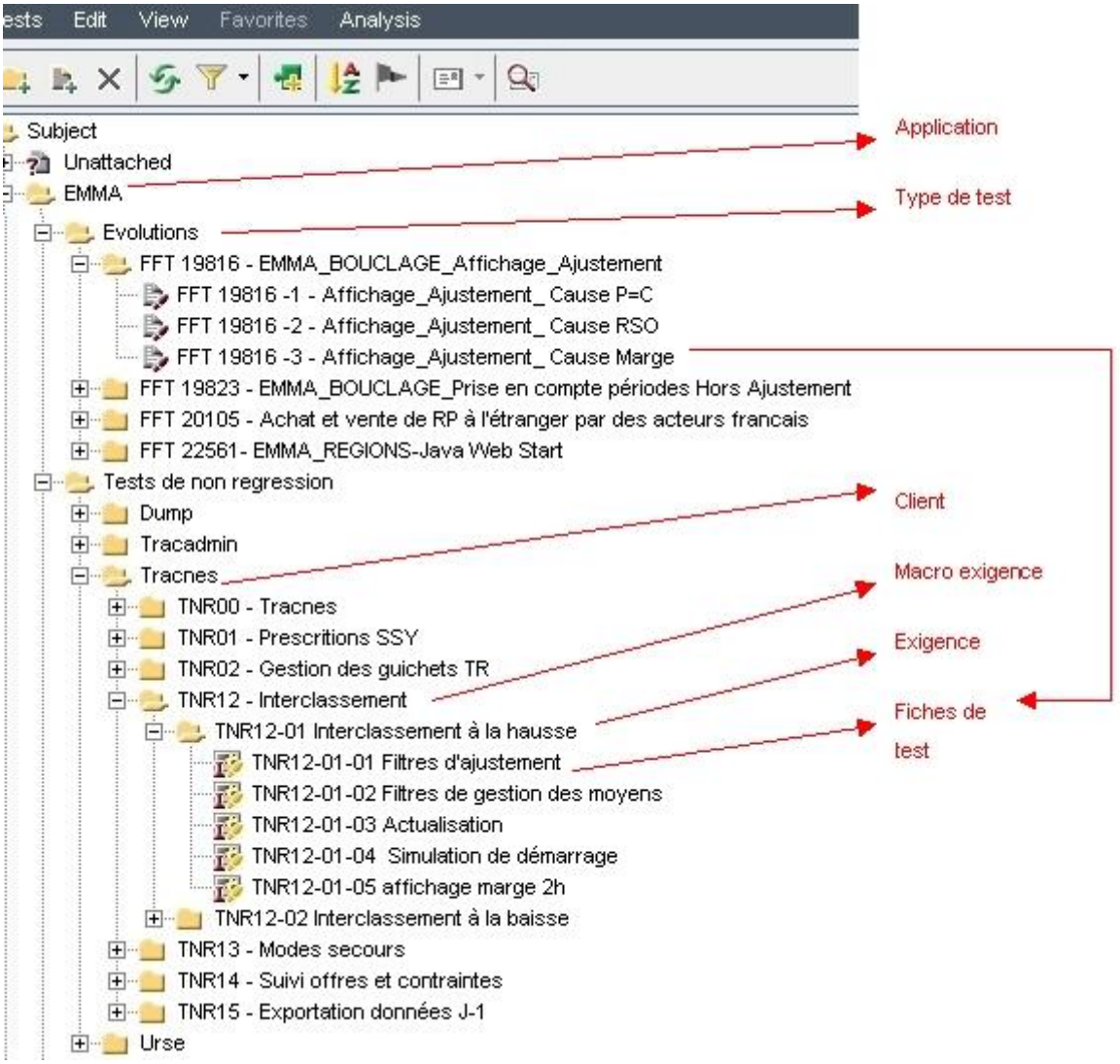


Figure 25 : Module « Test Plan » d'EMMA

4.2.3.2 Création des fiches de test à partir des exigences

QTP permet de créer l'arborescence des fiches de test à partir de celle des exigences. J'ai conseillé à l'équipe d'EMMA d'utiliser cette fonctionnalité.

Nous avons le choix entre plusieurs méthodes de conversion.

La première méthode convertit la macro exigence en fiche de test et l'exigence en étape de test. Je n'ai jamais utilisé cette méthode car elle ne correspond pas à l'organisation des exigences.

La seconde méthode permet de transformer l'exigence en fiche de test. L'avantage de cette méthode c'est une fiche de test est automatiquement liée à une exigence. Le problème c'est qu'une exigence a souvent plusieurs fiches de test.

La troisième méthode est celle que j'ai conseillé à l'équipe projet. Elle permet de convertir l'exigence en un répertoire. A l'intérieur du répertoire il suffit de créer les fiches de test qu'il faut par la suite lier à une exigence car ce n'est pas pris en compte automatiquement.

4.2.4 Module Test Lab

4.2.4.1 Organisation

Les Tests Sets étant liés à un cycle de recette, j'ai repris logiquement l'arborescence du module « Management ».

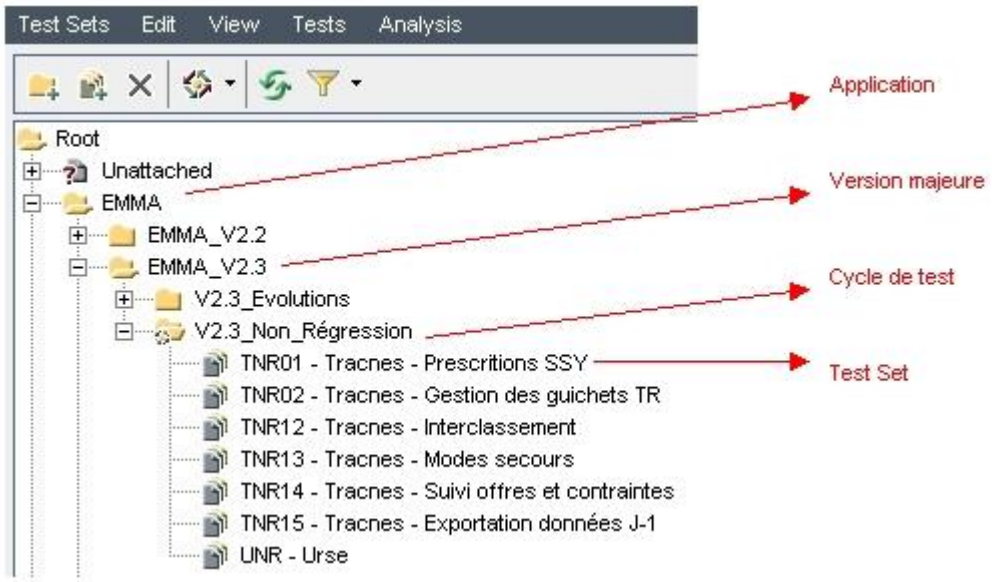


Figure 26 : Module « Test Lab » d'EMMA

4.2.4.2 Découpage des Tests Sets

Pour les tests manuels, les Tests Sets sont découpés par évolutions. J'ai fait ce choix pour plus de souplesse et de clarté. Le problème que j'ai rencontré pour les tests automatisés est que les Tests Sets ne peuvent pas être programmés pour être lancés les uns à la suite des autres. Vu le nombre de tests, il n'était pas envisageable de les regrouper dans un seul Test Set je les ai donc regroupés par macro-exigences.

4.3 Les données en entrée et en sortie

Les données en entrée et en sortie se trouvent sur un serveur à part, le serveur « referentiel_test ».

Celui-ci contient un répertoire pour les bases de données : « DumpRef_01 » et deux dossiers pour les jeux de données : « FichiersDeTest » et « Tests-Evolutions ».

Dans QC il est possible d'attacher des fichiers à chaque étape de la recette, j'ai déconseillé d'utiliser cette fonctionnalité pour plusieurs raisons.

Tout d'abord, les dumps sont assez volumineux. Ensuite ce n'est pas pratique pour l'exécution des tests puisque de nombreux fichiers doivent être transférés sur le Poste de travail puis envoyer sur un compte FTP (File Transfer Protocol). De plus pour les tests automatisés, il est impossible pour QTP d'aller chercher les fichiers se trouvant dans QC. Par contre il est très simple d'ajouter dans le script le chemin du fichier déposé sur le serveur.

4.3.1 Jeux de données

Dans le répertoire « FichiersDeTest » se trouve les données en entrée et en sortie des tests de non-régression. Le répertoire « Tests-Evolutions » contient celles des évolutions.

Dans ces deux répertoires nous retrouvons la même arborescence que l'onglet « Test Plan » de QC. Nous avons le client EMMA, la macro-exigence, l'exigence et les fiches de test. Pour chaque fiche de test, il y a le répertoire « resultat » qui correspond aux données en sortie et le répertoire « src » qui contient les données en entrée.

Les fichiers se trouvent dans un répertoire dont le nom est l'étape du test. Ces étapes sont détaillées dans l'onglet « Test Plan ». Elles permettent de savoir à quel moment les fichiers doivent être injectés dans l'application EMMA (pour les données en entrée) ou doivent être comparées avec les fichiers générés par EMMA (pour les données en sortie).

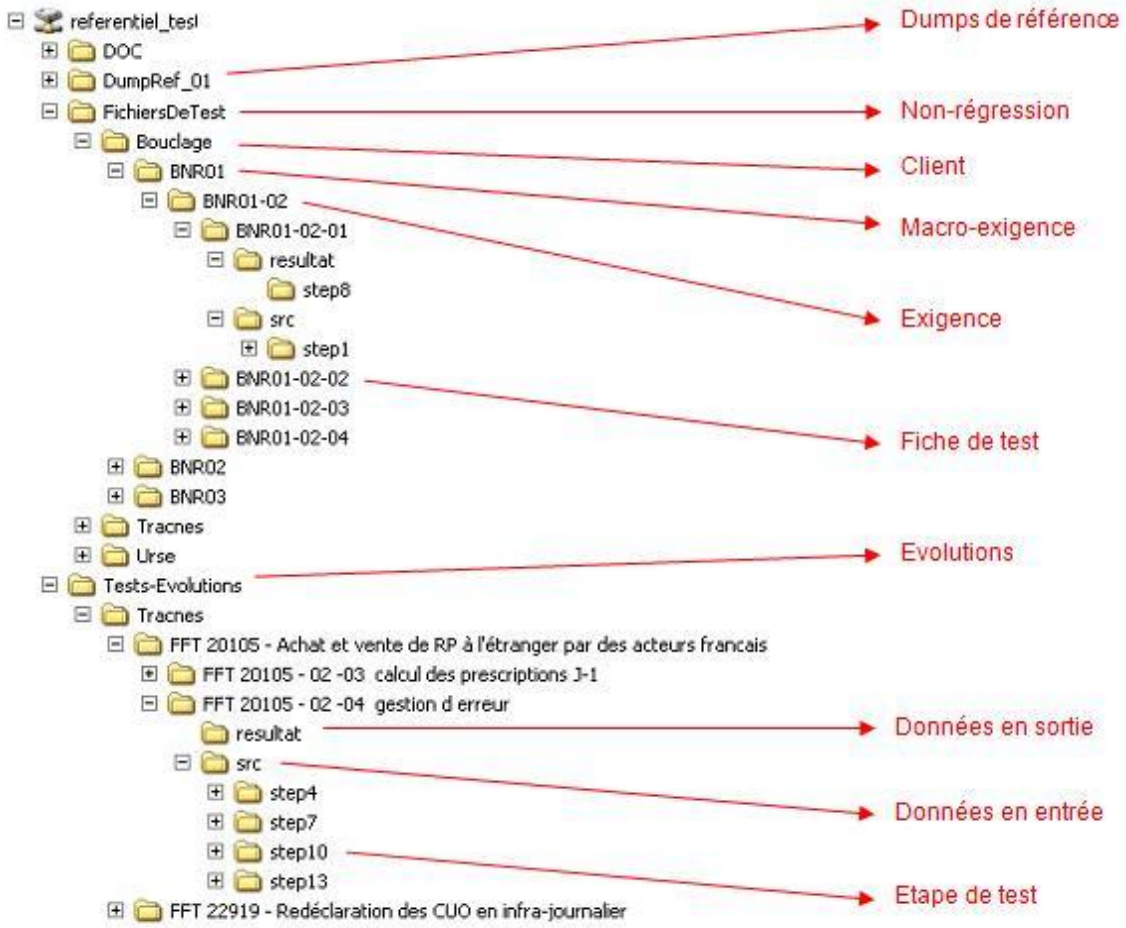


Figure 26 : Serveur des données

Les fichiers d'entrée et de sortie sont des fichiers CSV (Comma-separated values) ou des fichiers XML (Extensible Markup Language).

Les fichiers d'entrée sont soit à déposer sur un FTP soit à copier sur le serveur de l'application EMMA.

4.3.2 Bases de données

Pour gérer les données d'entrée d'EMMA, il a été décidé, bien avant l'utilisation des outils QC et QTP, de créer plusieurs dumps car les tests se déroulent à des périodes différentes de la journée. Ces dumps correspondent donc à un instant t de la journée.

Un dump pour EMMA contient des données de référence et des données dynamiques.

Les données de référence sont par exemple la liste des EDP (Entités de production), celle des acteurs,… Ce sont des éléments statiques.

Les données dynamiques sont des données à injecter dans la base de données, comme les programmes, les offres,...

Ces données dynamiques permettent d'avoir des dumps à chaque étape de la journée.

Pour l'instant, voici les dumps que l'équipe projet a besoin pour les tests :

- le dump J-1 vide
- le dump J-1 après le calcul des prescriptions
- le dump J-1 avant feu vert
- le dump J-1 avant basculement
- le dump J après basculement

Cette liste peut évoluer car les jeux de données doivent être adaptés aux cas de tests.

Il faut créer un dump à chaque étape majeure de la vie de l'application.

4.3.2.1 Gestion des bases de données

Pour créer les dumps de référence de la V2.3, un dump avant basculement de production a été récupéré (Dump Prod Vn-1).

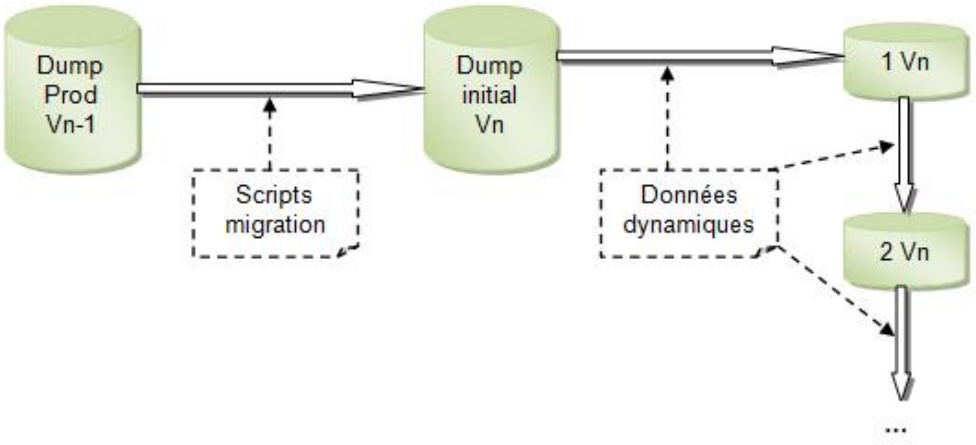
Différentes étapes ont ensuite été effectuées pour créer les dumps de référence.

Tout d'abord, un script de migration a été exécuté à partir du dump de production. Ce script est à lancer seulement s'il y a des changements au niveau de la structure de la base de données entre les deux versions, ce qui a été le cas entre les V2-2 et V2-3.

Il s'agit d'un script en PL/SQL à lancer sur la base de données. Ce fichier peut contenir des requêtes de sélection, de modification ou de suppression de données ainsi que l'ajout ou la suppression d'une table. Nous pouvons voir un exemple de script de migration en « Annexe D ».

Ensuite les données dynamiques, connues et adaptées aux tests, sont ajoutées dans la base de données soit en réalisant des actions directement dans l'application EMMA soit en injectant des fichiers.

Puis à chaque fin d'étape nous créons un dump. Nous retrouvons ainsi les différents dumps de données utilisés pour les tests (1 Vn correspond au dump J-1 vide, 2Vn est le dump J-1 après le calcul des prescriptions,…)



4.3.2.2 Problèmes rencontrés

Suite à l'automatisation des tests de non-régression, je me suis aperçue que cette solution posait plusieurs problèmes.

- Migration de la base de données

Tout d'abord le fait d'avoir migré la base dès la première étape est une erreur car ensuite nous construisons les dumps de référence (1 Vn,..) sur une version qui n'a pas encore été testé et qui est potentiellement en erreur. Si c'est le cas, les dumps de référence seront aussi en erreur et de ce fait les tests ne pourront pas se dérouler correctement.

- Utilisation de la base de production

Ensuite, je me suis rendue compte lors du passage des tests de non-régression de la V2.3 (tests conçus à partir de la V2.2) que cette méthode ne pouvait pas être utilisée pour les tests de non-régression.

En créant un dump initial à partir de la production, je me suis retrouvée avec des données différentes entre la V2.2 et la V2.3.

En production des nouvelles EDPs ont été ajoutées, certaines ont changé d'acteur, d'autres ont été supprimées, ...

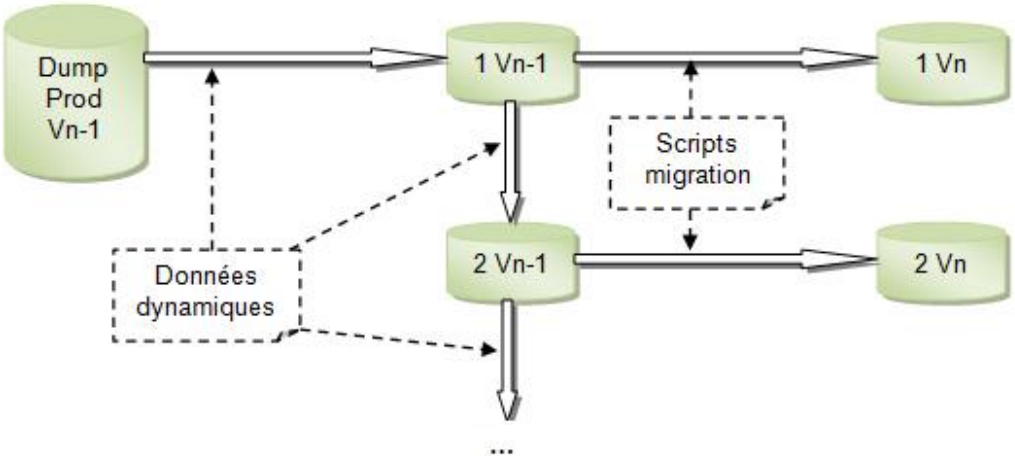
De nombreux tests de non-régression se sont donc retrouvés en erreur car les données en entrée étaient différentes.

4.3.2.3 Solutions envisagées

Avec l'équipe projet, nous avons réfléchi à plusieurs solutions pour résoudre ces problèmes.

- Migration de la base de données

Pour le premier problème rencontré, c'est très simple, il suffit de lancer le script de migration à la fin de la création de l'ensemble des dumps.

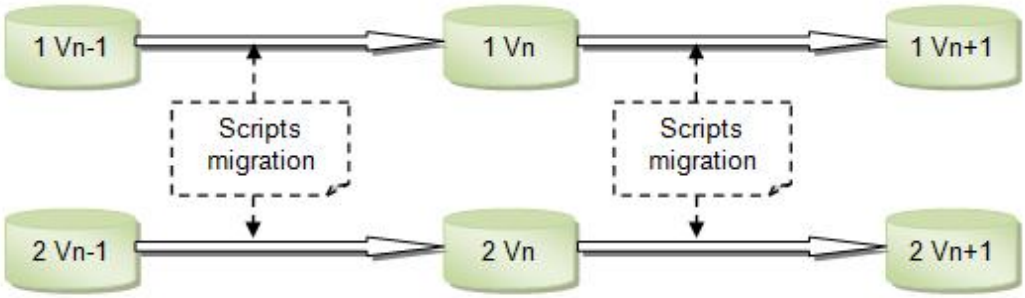


Par contre cette solution ne résout pas le second problème rencontré.

- Utilisation de la base de production

  - Première solution

Pour remédier à celui-ci, j'ai décidé qu'il ne fallait plus passer par un dump de production. Les dumps de référence 1 Vn-1, 2Vn-1,… seront migrés à chaque nouvelle version à tester.



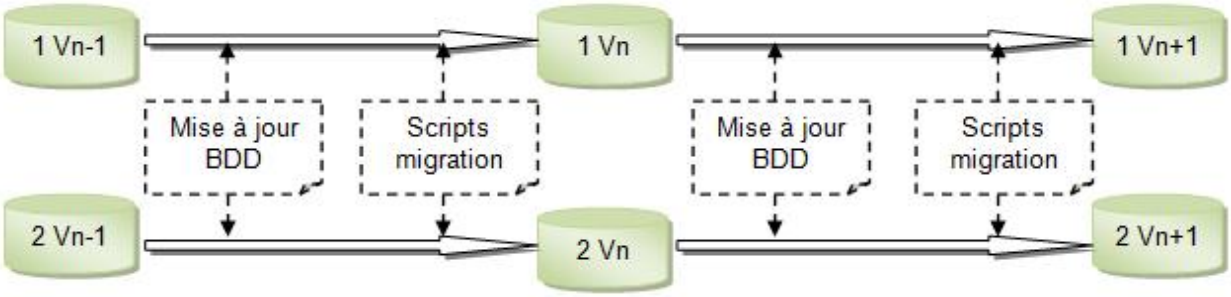
Cela pose encore un problème lorsque certaines évolutions vont passer en non-régression, les dumps de référence ne correspondront plus forcément aux nouvelles fonctionnalités de l'application. Il faudra donc ajouter ou modifier les données de référence.

J'ai proposé de modifier les dumps de référence si cela s'avère indispensable.

C'est à la fin de la recette validée, lors du passage des tests d'évolution en tests de non-régression dans QC qu'il faudra vérifier si les tests nécessitent un changement dans la base de données. Si c'est le cas, il faudra mettre à jour la base de données.

Cette solution est assez contraignante et ne pourra pas être automatisée car les changements ne sont pas récurrents.

Pour qu'elle puisse être réalisable, il est nécessaire de détailler dans QC les changements apportés à la base de données, s'il y en a. Cette mise à jour peut être, soit apportée directement dans l'application EMMA, soit transmise par des fichiers de référence.



La solution de migrer les dumps de référence à l'infini risque de poser un problème au bout d'un certain temps.

EMMA travaillant avec des données J et J-1 et les scripts de migration étant lancé à n'importe quel moment de la journée, certaines données risquent d'être perdu.

  - Seconde solution

La solution que j'ai mise en place est de créer à chaque fin de recette un dump initial pour être certain d'avoir des données correctes. Ce dump initial correspond à un dump après basculement c'est-à-dire que J-1 devient J et J-1 se retrouve sans données.

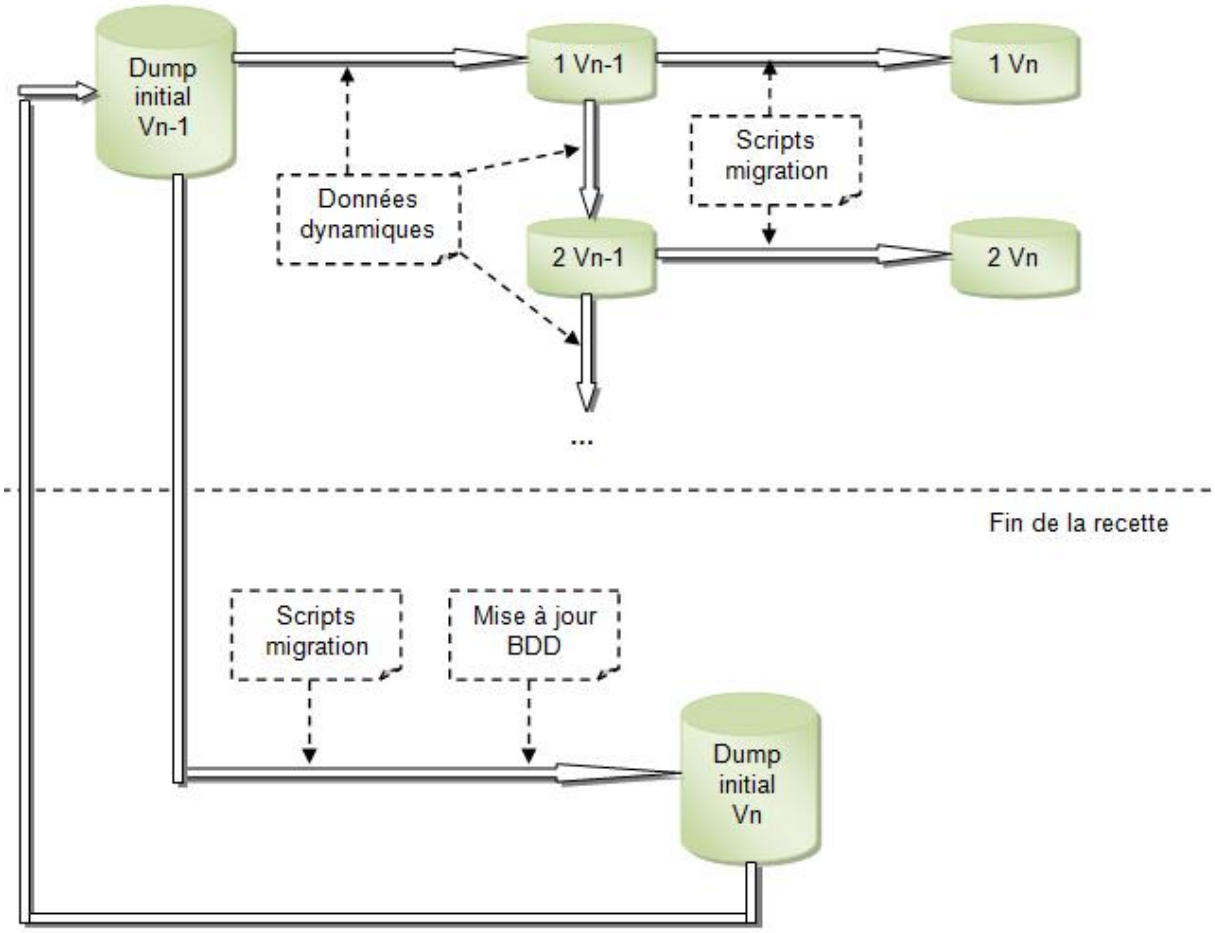
Ce dump initial sera mis à jour, comme vu précédemment, suivant les tests qui passeront en non-régression.

Ensuite les dumps de référence seront crées à partir du dump initial. Puis nous lancerons le script de migration sur tous les dumps de référence.

Lors de la prochaine recette, le dump final Vn deviendra le dump initial Vn-1.

La migration du Dump Initial Vn-1 vers le Dump Initial Vn se fait une fois que tous les tests sont réalisés et que la version est validée.

Les données intégrées lors du Dump Initial Vn vers le Dump Final Vn sont des modifications ou des nouvelles données de référence lors du passage des évolutions vers les tests de non-régression.



4.4 Mise en œuvre de Quick Test Professional

4.4.1 Création des fichiers Batch

Pour effectuer les tests manuels de l'application, de nombreuses commandes Shell ont été créées.

Ces commandes permettent :

- D'arrêter ou de redémarrer le serveur Weblogic ;
- De stopper ou de lancer le Scheduler ;
- De charger un dump ;
- De changer la date ou l'heure du PC.

Pour l'automatisation des tests, il a fallu trouver une méthode simple pour que QTP exécute ces commandes.

Avec l'équipe projet, nous avons créé des fichiers Batch sur le serveur de test d'EMMA qui appellent celles-ci.

Dans le répertoire où se trouvent les fichiers, nous avons placé « plink.exe ». Plink (PuTTY Link) est un outil de connexion en ligne de commande. Nous avons besoin de cet outil car QTP est sous Windows et EMMA sous Linux.

4.4.2 Création de bibliothèques de fonctions

L'enregistrement du test dans QTP, vu dans le paragraphe « 3.3.3 Enregistrement d'un script », ne suffit pas à automatiser tous les tests. J'ai du créer des fonctions réutilisables en Visual Basic Script. Ces scripts permettent d'exécuter des fichiers Batch, de copier des fichiers dans un répertoire,…

J'ai regroupé ces fonctions dans des bibliothèques. Je les ai créées dans QTP et elles sont visualisables dans l'onglet « Test Ressources » de QC.

4.4.2.1 La bibliothèque ExecuterBat

Cette bibliothèque contient les fonctions permettant d'exécuter un fichier Batch.

Il existe deux fonctions dans cette bibliothèque.

- Exécuter un fichier Batch

Dans QTP, j'ai crée une fonction qui exécute n'importe quel fichier Batch en rentrant en paramètres son adresse et une variable (utilisée pour le nom d'un dump ou pour renseigner une date).

- Se connecter à l'application EMMA

Les exécutables, pour se connecter aux différents clients de l'application EMMA, sont des fichiers Batch appelant les fichiers Java. J'ai repris la fonction précédente et je l'ai légèrement modifié au niveau de la méthode « WshShell.Run ».

La méthode « Run » dispose de plusieurs arguments à renseigner :

- « strCommand » qui est le chemin du fichier Batch ;
- « intWindowStyle » est optionnel, il indique l'apparence de la fenêtre qui va s'ouvrir ;
- « bWaitOnReturn » est optionnel, il permet de savoir s'il faut attendre que le programme se termine avant de continuer le script QTP (« True ») ou non (« False », par défaut).

Pour la fonction précédente, « Exécuter un fichier Batch », j'ai du mettre l'argument « bWaitOnReturn » à « True » car il faut attendre que le script est fini de s'exécuter avant de passer à l'étape suivante.

```
'La variable rep correspond au chemin du fichier .bat
'3 = intWindowStyle = active et affiche la fenêtre au maximum
res = WshShell.Run (rep,3,true)
```

Pour la connexion à l'application EMMA, il faut au contraire laisser l'argument « bWaitOnReturn » à « False ».

```
WshShell.Run rep
```

4.4.2.2 La bibliothèque Fichiers

Dans cette bibliothèque se trouvent les fonctions que j'ai écrites qui sont liées à la manipulation de fichiers. L'application EMMA dispose de nombreux fichiers d'entrée qu'il faut copier dans des répertoires spécifiques et des fichiers de sortie qu'il faut comparer avec ceux se trouvant sur le répertoire de référence des données.

- Copier un fichier

Cette fonction permet de copier un fichier d'un répertoire à un autre. Cette fonction vérifie que le fichier à copier existe. Si un fichier du même nom existe déjà dans le répertoire destinataire, la fonction écrase ce fichier.

Avec cette fonction nous voyons bien l'intérêt d'avoir un espace de référence de données comme décrit dans la partie « 4.3.1 Jeux de données » plutôt que de tout stocker sous QC.

- Copier tous les fichiers d'un même répertoire

Cette fonction permet de copier tous les fichiers d'un même répertoire vers un autre répertoire. Cette fonction vérifie que le répertoire source existe. Si un fichier du même nom existe déjà dans le répertoire destinataire, la fonction écrase ce fichier.

- Modifier un fichier texte (.txt, .csv, .properties,…)

Cette fonction est utilisée pour remplacer dans un fichier texte une chaîne de caractère par une autre.

- Vérifier qu'un fichier existe

Cette fonction renvoie la valeur booléenne « True » si le fichier rentré en paramètre existe.

- Comparer deux fichiers texte (.txt, .csv, .properties,…)

Cette fonction permet de comparer ligne par ligne deux fichiers texte. Elle renvoie une variable texte qui décrit toutes les différences trouvées entre ces fichiers.

- Vérifier la date de création d'un fichier

Cette fonction sert à vérifier si la date de création d'un fichier correspond à la date entrée en paramètre. Elle vérifie dans un premier temps si le fichier existe.

- Vérifier la date de modification d'un fichier

Cette fonction ressemble à la précédente mais elle vérifie cette fois-ci si la date de modification d'un fichier correspond à la date entrée en paramètre.

- Lister les fichiers d'un répertoire

Cette fonction permet de lister les fichiers d'un répertoire.

- Rechercher le dernier fichier modifié d'un répertoire

Cette fonction recherche le dernier fichier modifié dans un répertoire. Je l'ai crée car certains noms de fichiers générés par EMMA contiennent l'heure de création du fichier sous cette forme « hhmmss ». Il est difficile de connaître les minutes de création du fichier et impossible de savoir les secondes exactes. Pour retrouver le nom du fichier à comparer avec celui du serveur de référence j'ai choisi de rechercher le dernier fichier modifié dans un répertoire.

- Supprimer un fichier

Cette fonction permet de supprimer un fichier.

4.4.2.3 La bibliothèque FTP

La bibliothèque FTP contient deux fonctions liées à la manipulation de fichiers sur le FTP.

- Envoyer un fichier sur un FTP

Cette fonction permet d'envoyer un fichier sur un compte FTP. Elle est utilisée régulièrement pour effectuer des guichets.

- Télécharger un fichier depuis un FTP

Cette fonction permet de télécharger un fichier depuis un compte FTP. Elle n'est pas utilisée pour le moment sur la recette d'EMMA mais elle pourra servir un jour pour celle-ci ou pour une autre application.

4.4.2.4 La bibliothèque SQL

Cette bibliothèque contient les fonctions liées à la base de données Oracle.

- Sélectionner des données

Cette fonction dispose d'une requête générique de sélection et renvoie le résultat attendu du select.

- Modifier des données

Cette fonction permet de renseigner une requête de modification des données.

4.4.2.5 La bibliothèque Tableau

Cette bibliothèque contient les fonctions liées à la manipulation de tableaux sur l'application EMMA.

- Remplir un tableau

Cette fonction permet de remplir ou de modifier certaines valeurs d'un tableau de l'application EMMA. Je l'ai crée suite à un problème rencontré entre l'application EMMA et QTP. Ce problème est détaillé dans la partie « 4.4.6.3 Remplissage des tableaux ».

- Rechercher une valeur dans un tableau

Cette fonction recherche une valeur dans un tableau de l'application EMMA et renvoie le numéro de la ligne où se trouve cette valeur.

4.4.2.6 La bibliothèque Imprimante

Cette bibliothèque contient les fonctions liées à la manipulation des imprimantes. Elle ne contient pour le moment qu'une seule fonction

- Modifier l'imprimante par défaut

Cette fonction permet de modifier l'imprimante par défaut de la machine où les tests vont se dérouler. Je l'ai ajouté après avoir rencontré un problème entre l'application EMMA et QTP. Ce problème est détaillé dans la partie « 4.4.6.6 Impression du guichet ».

4.4.3 Utilisation de variables d'environnement

J'ai ajouté des variables d'environnement dans les scripts QTP. Elles sont rassemblées dans un seul fichier.

Il s'agit d'un fichier XML. Celui-ci contient :

- Les noms des dumps de référence de la version à tester ;
- Les noms des dumps de référence de la version précédente ;
- Les accès vers le compte FTP : adresse, login mot de passe, répertoires où doivent être déposés les fichiers,… ;
- Le chemin vers le répertoire de l'application ;
- Les noms des fichiers Batch (chargement d'un dump, arrêt du serveur Weblogic,…) ;
- Les noms des fichiers Batch permettant la connexion vers les clients d'EMMA ;
- Le login et le mot de passe pour les clients EMMA.

L'utilisation de ses variables permet en cas de modification de mettre à jour uniquement le fichier XML et non l'ensemble des scripts QTP.

Ces variables contiennent un nom et une valeur.

```
- <Variable>
    <Name>p_Login1</Name>
    <Value>CF00046T</Value>
  </Variable>
```

Dans le script QTP, l'appel se fait très simplement en renseignant le nom de la variable.

```
'Saisie du login pour se connecter à l'application
JavaDialog("Connexion à l'application").JavaEdit("Utilisateur :").Set
Environment("p_Login1")
```

Il faut bien évidement ajouter le fichier XML contenant les variables à chaque test en ayant besoin.

4.4.4 Connexion aux clients d'EMMA

4.4.4.1 Clients Traçabilité et Bouclage

Pour se connecter aux clients Traçabilité et Bouclage, il suffit de lancer respectivement les fichiers « lance_tracnes.bat » et « lance_bouclage.bat ».

Pour scripter les tests de connexion dans QTP, j'ai donc commencé par sélectionner, lors de l'enregistrement du script, le fichier Batch nécessaire.

Le problème c'est que pour la connexion à ces deux clients, une console DOS s'ouvre et invite l'utilisateur à saisir une date d'exploitation, comme nous pouvons le voir sur l'image ci-dessous. QTP ne reconnait pas la console et n'enregistre pas la saisie de la date.

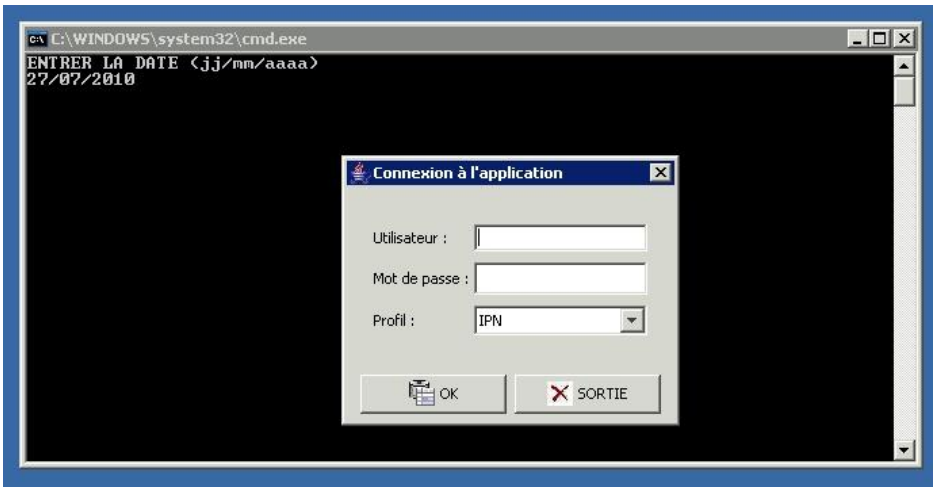


Figure 27 : Connexion clients Traçabilité et Bouclage

Pour pouvoir automatiser ces scripts, j'ai crée un nouveau fichier Batch de connexion et j'ai utilisé le langage Visual Basic Script pour automatiser ces scripts.

- Création d'un fichier Batch :

J'ai crée un fichier de connexion Batch spécifique à QTP.

J'ai copié le fichier « lance_tracnes.bat » et je lui ai apporté quelques modifications afin d'annuler la saisie de la date par l'utilisateur.

```
rem === echo ENTRER LA DATE (jj/mm/aaaa)
rem === for /F "tokens=1 delims= " %%i in ('date_t') do set _Z_DATE=%%i

rem === call java -Xms128M -Xmx640M -DREP_TRACE=%rep_trace% -DTRACE=REFPERF -
DDATE="%_Z_DATE%" -jar %appli%tracnes.jar 1>%fichierlog% 2>%fichiererr%

call java -Xms128M -Xmx640M -DREP_TRACE=%rep_trace% -DTRACE=REFPERF -DDATE="%1"
-jar %appli%tracnes.jar 1>%fichierlog% 2>%fichiererr%
```

J'ai effectué la même démarche pour le fichier « lance_bouclage.bat ».

- Automatisation du test
    - Sélection de la date

J'ai commencé le script par une recherche dans la base de données. La date de connexion à l'application se trouve dans la base et est modifié en fonction du dump qui est chargé.

```
'Sélectionne la date qui se trouve dans la BDD
reqSelectDate = "select valeur from etat_application where code = 11"
ReDim champsReq(0)
champsReq(0) = "valeur"
nbChampsReq = 1
'Appel la fonction générique de sélection dans la BDD
resDateJ = reqSelect(reqSelectDate, champsReq, nbChampsReq)
dateJbdd = left(resDateJ(0),10)
'Le format de dateJbdd est yyyy-mm-dd hh:mm:ss
'nous avons besoin de la date au format dd/mm/yyyy
tabDate = Split(dateJbdd,"-")
dateJ = " " & tabDate(2) & "/" & tabDate(1) & "/" & tabDate(0)
```

- Appel du fichier Batch

Après avoir récupéré la date de connexion dans la variable « dateJ », j'ai appelé la fonction permettant de se connecter aux différents clients d'EMMA et je lui ai envoyé en paramètre cette valeur.

```
'Lancement du fichier .bat
repBat = Environment("p_RepAppli") & Environment("p_RepTracnes")
connexionEmma repBat, dateJ
'Temps d'attente de 10 secondes
wait(10)
```

- Enregistrement de la connexion

Au moment de l'enregistrement du test, j'ai sélectionné l'option « Record and run test on any open Java application ». De cette manière, QTP ne lance pas lui-même l'application.

Ensuite j'ai saisi le login, le mot de passe et le type de profil. Pendant ce temps QTP a enregistré mes différentes actions.

```
JavaDialog("Connexion à l'application").JavaEdit("Utilisateur :").Set
Environment("p_Login1")
JavaDialog("Connexion à l'application").JavaEdit("Mot de passe :").SetSecure
Environment("p_Mdp1")
JavaDialog("Connexion à l'application").JavaList("Profil :").Select "IPN"
JavaDialog("Connexion à l'application").JavaButton("OK").Click
'Checkpoint permettant de vérifier que le client s'est bien ouvert
JavaWindow("Application EMMA Tracnes").Check CheckPoint("AppliOK")
```

4.4.4.2 Client URSE

Jusqu'à la V2.3 d'EMMA, la connexion au client URSE se faisait de la même manière que pour les clients Traçabilité et Bouclage.

Le processus de déploiement des clients URSE était lourd et ne permettait pas de gérer le déploiement de patchs urgents dans des conditions satisfaisantes. Une des évolutions de la V2.3

était de pouvoir déployer les clients URSE avec l'outil Java Web Start afin de pouvoir faire face aux retours arrière et aux patchs urgents.

Désormais au lieu d'appeler un fichier Batch, il faut saisir une ligne de commande lors de l'enregistrement du script de connexion au client URSE.

4.4.4.3 Client Validation

Le client Validation s'ouvre à l'aide d'un fichier Batch, comme pour les clients Traçabilité et Bouclage. La seule différence c'est au niveau de la saisie de la date de connexion, l'utilisateur doit sélectionner la date sur un calendrier.

Figure 28 : Connexion client Validation

Pour commencer à automatiser ce script, j'ai recherché la date d'exploitation dans la base de données, comme dans la partie « 4.4.3.1 Clients Traçabilité et Bouclage ».

Je dispose donc d'une variable « dateJ » contenant la date du dump.

```
dateJ = " " & tabDate(2) & "/" & tabDate(1) & "/" & tabDate(0)
'tabDate(2) = le jour, tabDate(1) = le mois et tabDate(0) = l'année
```

Au moment de l'enregistrement, QTP scripte le code suivant :

```
JavaDialog("Choix de la date d'exploitatio").JavaList("JComboBox").Select
"juillet"
JavaDialog("Choix de la date d'exploitatio").JavaSpin("JSpinner").Set "2010"
JavaDialog("Choix de la date d'exploitatio").JavaButton("27").Click
```

J'ai remplacé les valeurs enregistrées par les variables provenant de la base de données. Pour le nom du mois j'ai utilisé la fonction « MonthName » qui renvoie le nom du mois en fonction du numéro rentré en paramètre.

```
JavaDialog("Choix de la date d'exploitatio").JavaList("JComboBox").Select
MonthName(tabDate(1))
JavaDialog("Choix de la date d'exploitatio").JavaSpin("JSpinner").Set
tabDate(0)
```

Pour le jour du mois, comme il y a un bouton pour chaque jour, cela a été plus compliqué.

- Variable

J'ai essayé de remplacer le nom du bouton par la variable mais QTP ne retrouvait plus le bouton.

```
JavaDialog("Choix de la date d'exploitatio").JavaButton(tabDate(2)).Click
```

- Expression régulière

Ensuite j'ai voulu mettre une expression régulière (.*) à la place du label du bouton. Ce qui était une très mauvaise idée puisque QTP ne pouvait pas savoir quel bouton choisir.

- DataTable

Pour finir j'ai remplacé la valeur du label par un paramètre de la DataTable. J'ai testé cette solution avec différents jours et cela fonctionne. Nous pouvons voir ci-dessous la création d'un paramètre dans la DataTable.

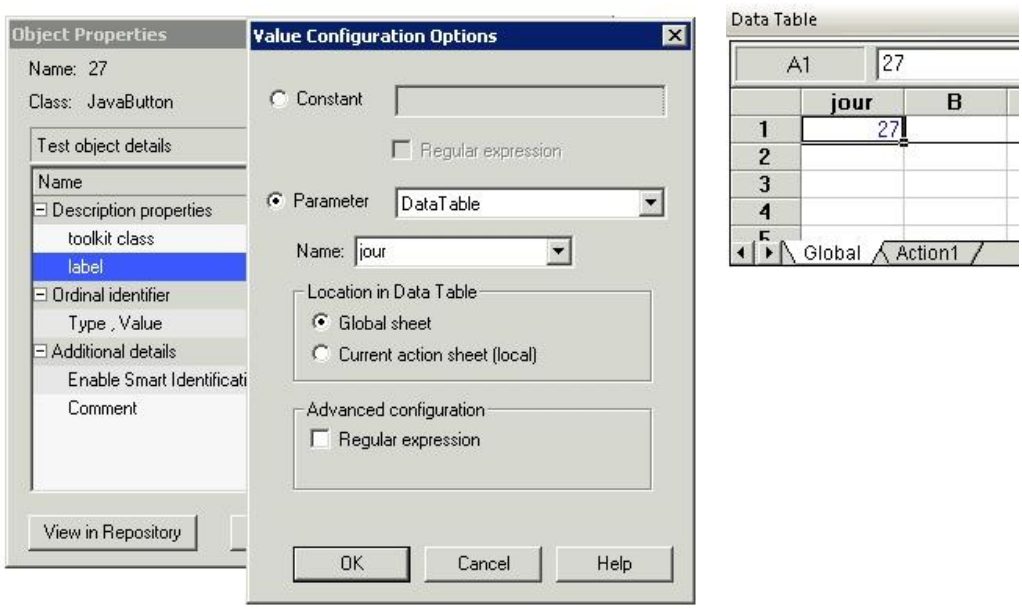


Figure 29 : Utilisation de la DataTable pour la connexion au client Validation

Dans le script, après avoir récupéré la date d'exploitation, j'ai ajouté la ligne ci-dessous pour indiquer à QTP qu'il faut modifier la valeur de la DataTable avec le jour de la base de données.

```
DataTable.Value("jour") = tabDate(2)
```

4.4.5 Gestion des dumps de référence

La création des dumps de référence se faisait manuellement. Cela prenait beaucoup de temps car les étapes sont nombreuses, comme nous l'avons vu dans la partie « 4.3.2.3 Solutions envisagées ». Pour remédier à ce problème j'ai mis en place des scripts automatiques pour créer les dumps de référence.

4.4.5.1 Scripts de migration SQL

Les scripts de migration SQL sont stockés sur le serveur dans plusieurs répertoires en fonction du numéro de version d'EMMA (« EMMA_2.2.6 », « EMMA_2.3.0 »,…) puis de l'environnement de tests (« TR », « VALID », « BO »).

Dans l'exemple ci-dessous, nous pouvons voir la liste des répertoires pour chaque version d'EMMA et les scripts se trouvant dans l'environnement « TR » de la version 2.4. Il y a deux scripts de migration pour cette version. Les scripts contenant le mot « Tables » doivent être exécutés en premier.

```
[webadm@PF9SOSMA2 TR]$ cd /appli/emma/livraison/FOURNISSEUR
[webadm@PF9SOSMA2 FOURNISSEUR]$ ls -ltr
total 56
drwxr-xr-x  6 webadm webgrp 4096 jun  3  2010 EMMA_2.2.6
drwxr-xr-x  4 webadm webgrp 4096 jun  9  2010 EMMA_2.2.6b
drwxr-xr-x  6 webadm webgrp 4096 jui 26  2010 EMMA_2.3.0a
drwxr-xr-x  6 webadm webgrp 4096 aoû 26  2010 EMMA_2.3.0b
drwxr-xr-x  6 webadm webgrp 4096 sep 13 15:46 EMMA_2.3.0
drwxr-xr-x  7 webadm webgrp 4096 sep 16 14:54 EMMA_2.2.7
drwxr-xr-x  6 webadm webgrp 4096 oct 11 14:10 EMMA_2.3.1
drwxr-xr-x  6 webadm webgrp 4096 nov 16 10:45 EMMA_2.2.8
drwxr-xr-x  6 webadm webgrp 4096 nov 16 17:49 EMMA_2.3.2
drwxr-xr-x  6 webadm webgrp 4096 nov 26 10:31 EMMA_2.3.3
drwxr-xr-x  6 webadm webgrp 4096 déc  3 10:15 EMMA_2.3.4
drwxr-xr-x  6 webadm webgrp 4096 déc  9 10:57 EMMA_2.3.5
drwxr-xr-x  4 webadm webgrp 4096 jan 18 16:30 EMMA_2.3.6
drwxr-xr-x  6 webadm webgrp 4096 fév 14 14:49 EMMA_2.4.0
[webadm@PF9SOSMA2 FOURNISSEUR]$ cd EMMA_2.4.0
[webadm@PF9SOSMA2 EMMA_2.4.0]$ cd oracle/migration/TR/
[webadm@PF9SOSMA2 TR]$ ls -ltr
total 32
-rwxr-xr-x  1 webadm webgrp  3208 fév 14 14:41 Migration_EMMA_v2.3.x_v2.4.0_Tables.sql
-rwxr-xr-x  1 webadm webgrp 25513 fév 14 14:41 Migration_EMMA_v2.3.x_v2.4.0.sql
```

Figure 30 : Arborescence des scripts de migration SQL

Pour rendre plus facile l'automatisation du lancement des scripts de migration, j'ai crée un répertoire spécifique pour QTP. Ce répertoire contient les scripts renommés avec un numéro d'ordre de passage et doit être mis à jour à chaque nouvelle version.

```
[webadm@PF9SOSMA2 TR]$ cd /appli/emma/livraison/QTP/TR/
[webadm@PF9SOSMA2 TR]$ ls -ltr
total 32
-rwxr-xr-x  1 webadm webgrp  3208 fév 17 12:10 01.sql
-rwxr-xr-x  1 webadm webgrp 25513 fév 17 12:10 02.sql
```

Figure 31 : Arborescence des scripts de migration SQL pour l'automatisation

4.4.5.2 Script Shell lançant les scripts SQL

Ensuite j'ai écrit le script Shell « scriptMigration.sh » permettant de lancer les scripts SQL se trouvant dans le répertoire spécifique QTP. Ce script effectue une boucle sur le répertoire, récupère les fichiers dans l'ordre croissant et les exécute.

```
# Les scripts de migration se trouvent dans le répertoire TR
# Le premier script à passer doit se nommer 01.sql, le deuxième : 02.sql,...
cd /appli/emma/livraison/QTP/TR
# On recherche tous les fichiers se trouvant dans le répertoire TR
# et on classe les fichiers dans l'ordre croissant
for i in `find $repertoire -type f | sort -n`
do
  whoami
  # $i est le nom du fichier (./01.sql,...)
  var=$i
  # on supprime les deux premiers caractères du nom de fichier (./)
  var2=${var#*./}
  echo "------ SCRIPT " $var2 "-------"
  # Passage du script
  sqlplus emma_tr/emma@emma @$var2
  echo "------ FIN DU SCRIPT " $var2 "-------"
done
```

Pour lancer ce script Shell à partir de l'outil QTP, j'ai crée un fichier Batch qui appelle celui-ci.

```
echo OFF
cd S:\Application\Lame2\ScriptAUTO
call S:\Application\Lame2\ScriptAUTO\SetEnv.bat
S:\Application\Lame2\ScriptAUTO\plink -ssh -pw %ORACLE_PWD% oracle@%IP% export
ORACLE_HOME=%ORACLE_HOME%; export PATH=%PATH%; %MIGR_DIR%/scriptMigration.sh
```

4.4.5.3 Automatisation des étapes

Il y a cinq dumps de référence (J-1 vide, J-1 calcul prescription, J-1 avant feu vert, J-1 avant basculement, J après basculement), j'ai donc crée cinq nouveaux tests pour chacun des dumps.

A la fin des tests, je passe les scripts de migration SQL puis j'exporte les nouveaux dumps.

Dans la partie « Test Lab » de QC j'ai crée un Test Set pour la création des dumps de référence où j'appelle l'ensemble de ces tests, ainsi que les tests de chargement de dump, de connexion et de déconnexion aux clients EMMA.

4.4.5.4 Pré-requis avant de lancer ces scripts

Il y a certains pré-requis à prendre en compte avant de lancer la création des dumps de référence.

- Sauvegarder les dumps de la version précédente

Tout d'abord, il faut sauvegarder les différents dumps de la version précédente sur le serveur de ressources. En cas de problème, il vaut mieux avoir une sauvegarde des anciens dumps.

Il faut se connecter sur le serveur via l'outil Putty et se déplacer sur le répertoire où les dumps doivent être copié :

```
cd /mnt/Referentiel_test/DumpRef_01/xx.yy.zz/Dumps_de_reference/Non-
regression/x.y.zz/
```

Puis, il faut copier les dumps :

```
cp -R /appli/emma/oracle/dump/QTP_DUMP_REF_01_*-Vx.y.zz*.dmp .
```

- Modifier les variables d'environnement

Comme indiqué dans la partie « 4.4.3 Utilisation de variables d'environnement », j'ai utilisé des variables d'environnement dans les scripts QTP.

Ces variables contiennent le nom des dumps de la version à tester et le nom des dumps de la version précédente. Il faut modifier les noms des dumps avant de lancer les scripts automatiques de création de dumps car ils contiennent la version et la date des données.

- Copier les scripts de migration

Les scripts de migration SQL doivent se trouver dans un répertoire spécifique avant de lancer la création des dumps de référence.

Il faut se connecter sur le serveur via l'outil Putty et se déplacer sur le répertoire où les dumps doivent être copié :

```
cd /appli/emma/livraison/QTP/TR/
```

Puis supprimer les scripts se trouvant dans ce répertoire :

```
rm *.sql
```

Pour finir, il faut copier les nouveaux scripts qui seront exécutés :

```
cp
/appli/emma/livraison/FOURNISSEUR/EMMA_x.y.z/oracle/migration/TR/Migration_EMMA_
v2.2.x_v2.3.2_Tables_TR.sql /appli/emma/livraison/QTP/TR/01.sql

cp
/appli/emma/livraison/FOURNISSEUR/EMMA_x.y.z/oracle/migration/TR/Migration_EMMA_
v2.2.x_v2.3.2.sql /appli/emma/livraison/QTP/TR/02.sql
```

4.4.6 Problèmes rencontrés

Dans l'ensemble les tests automatisés QTP pour EMMA fonctionnent bien mais j'ai quand même rencontré quelques problèmes lors de la mise en œuvre de cet outil pour les tests de l'application EMMA.

4.4.6.1 Nom des fenêtres de l'application

- Client URSE

J'ai commencé l'automatisation des tests d'EMMA sur le client URSE car il s'agit du plus simple et du moins critique des clients.

Le premier test que j'ai effectué était très basique, il permettait uniquement de choisir une EDP dans la liste déroulante et de visualiser les programmes en courbe de cette EDP.

L'enregistrement du script s'est bien passé mais il était impossible de le rejouer.

Je me suis aperçue que le nom de la fenêtre du client URSE contient la date et l'heure d'affichage de la fenêtre.

Ces informations ne sont pas visibles au premier abord ni dans QTP ni sur la fenêtre du client URSE.

C'est en visualisant les propriétés de la fenêtre que j'ai découvert que le titre comporte la date et l'heure, il était donc impossible pour QTP d'exécuter le script car il ne retrouvait pas la fenêtre enregistrée.

Pour contourner ce problème il a fallu placer une expression régulière à la suite de « … Profil : IEC ». L'expression régulière « .* » signifie que QTP recherche une fenêtre avec n'importe quelle chaîne de caractère après « … Profil : IEC ». Nous pouvons voir la procédure à suivre ci-dessous.

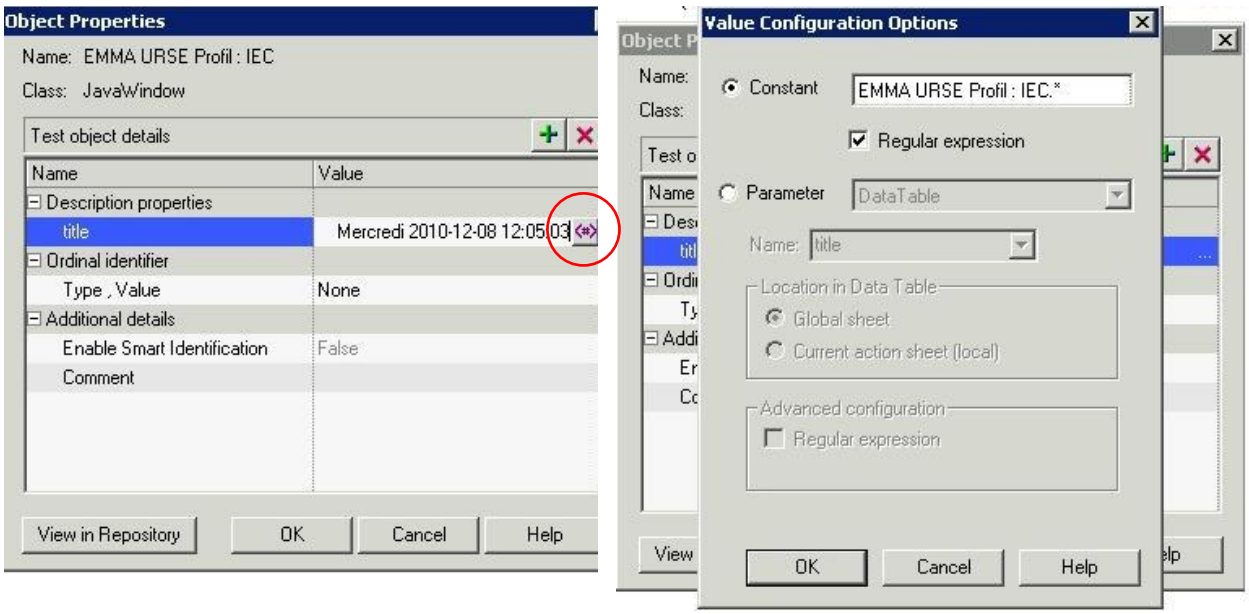


Figure 32 : Ajout d'une expression régulière

- Les autres clients

En ce qui concerne les autres clients d'EMMA, les titres disposent du numéro de version et de la date de mise à disposition de l'application. A chaque test il faut placer une expression régulière au niveau du titre sinon le test ne pourra pas s'exécuter lors de la prochaine livraison.

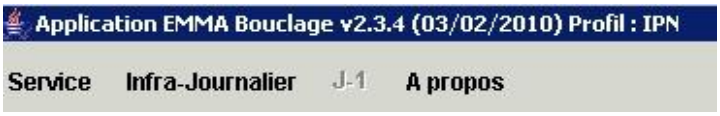


Figure 33 : Titre du client Bouclage

4.4.6.2 Nom des objets de l'application

Certains objets de l'application posent également problèmes car leur nom contient la date J ou J-1 du dump de données. A chaque version de mise en production, la date est modifiée, QTP ne peut pas retrouver l'objet enregistré.

Il faut, là aussi, ajouter des expressions régulières. Le problème c'est que les dates ne sont pas souvent visibles. C'est lors du passage des tests de non-régression sur la V2.3 que je me suis aperçue de quelques oublis.

4.4.6.3 Remplissage des tableaux

Un autre problème rencontré avec l'application EMMA, c'est le remplissage des tableaux. Dans EMMA les colonnes des tableaux correspondent aux heures de la journée par pas de demi-heure.

La première colonne n'a pas de nom, elle correspond à la plage horaire : 0h00 -> 0h30.

La seconde colonne se nomme « 1 », elle correspond à la plage horaire : 0h30 -> 1h00…

Plage		1		2		3		4		5		6		7		8		9		10		11		12		13
Puissance	850	700	700	700	700	700	700	700	700	700	700	700	750	750	850	850	750	750	750	750	750	750	750	750	750	750

Lorsque QTP enregistre le script pour remplir le tableau, les colonnes impaires (celles qui n'ont pas d'en-tête) sont appelées par l'outil « #n » dont n correspond au numéro de la colonne. Nous pouvons voir un exemple dans le code ci-dessous ligne 4.

Pour les colonnes paires elles sont nommées comme leur en-tête (ligne 7).

```
2:  'colonne 1 : 0h00 à 0h30, puissance = 850
3:  JavaWindow("Saisie des contraintes").JavaTable("TableAppli").ActivateCell "#0","#1"
4:  JavaWindow("Saisie des contraintes").JavaTable("TableAppli").SetCellData "#0","#1","850"
5:  'colonne 2 : 0h30 à 1h00, puissance = 700
6:  JavaWindow("Saisie des contraintes").JavaTable("TableAppli").ActivateCell "#0","1"
7:  JavaWindow("Saisie des contraintes").JavaTable("TableAppli").SetCellData "#0","1","700"
8:  'colonne 3 : 1h00 à 1h30, puissance = 700
9:  JavaWindow("Saisie des contraintes").JavaTable("TableAppli").ActivateCell "#0","#3"
10: JavaWindow("Saisie des contraintes").JavaTable("TableAppli").SetCellData "#0","#3","700"
11: 'colonne 4 : 1h30 à 2h00, puissance = 700
12: JavaWindow("Saisie des contraintes").JavaTable("TableAppli").ActivateCell "#0","2"
13: JavaWindow("Saisie des contraintes").JavaTable("TableAppli").SetCellData "#0","2","700"
14: 'colonne 5 : 2h00 à 2h30, puissance = 700
15: JavaWindow("Saisie des contraintes").JavaTable("TableAppli").ActivateCell "#0","#5"
16: JavaWindow("Saisie des contraintes").JavaTable("TableAppli").SetCellData "#0","#5","700"
17: 'colonne 6 : 2h30 à 3h00, puissance = 700
18: JavaWindow("Saisie des contraintes").JavaTable("TableAppli").ActivateCell "#0","3"
19: JavaWindow("Saisie des contraintes").JavaTable("TableAppli").SetCellData "#0","3","700"
```

Au moment de l'exécution du script QTP n'arrive pas à retrouver les colonnes du tableau. Il confond les colonnes « #n » et « n ». Dans l'exemple ci-dessus, après avoir rempli correctement la première colonne avec la première valeur, il efface celle-ci pour insérer la seconde valeur.

Il est possible de modifier manuellement les indices du tableau pour que QTP s'y retrouve.

```
2:  'colonne 1 : 0h00 à 0h30, puissance = 850
3:  JavaWindow("Saisie des contraintes").JavaTable("TableAppli").ActivateCell "#0","1"
4:  JavaWindow("Saisie des contraintes").JavaTable("TableAppli").SetCellData "#0","1","1"
5:  'colonne 2 : 0h30 à 1h00, puissance = 700
6:  JavaWindow("Saisie des contraintes").JavaTable("TableAppli").ActivateCell "#0","2"
7:  JavaWindow("Saisie des contraintes").JavaTable("TableAppli").SetCellData "#0","2","2"
8:  'colonne 3 : 1h00 à 1h30, puissance = 700
9:  JavaWindow("Saisie des contraintes").JavaTable("TableAppli").ActivateCell "#0","3"
10: JavaWindow("Saisie des contraintes").JavaTable("TableAppli").SetCellData "#0","3","3"
11: 'colonne 4 : 1h30 à 2h00, puissance = 700
12: JavaWindow("Saisie des contraintes").JavaTable("TableAppli").ActivateCell "#0","4"
13: JavaWindow("Saisie des contraintes").JavaTable("TableAppli").SetCellData "#0","4","4"
14: 'colonne 5 : 2h00 à 2h30, puissance = 700
15: JavaWindow("Saisie des contraintes").JavaTable("TableAppli").ActivateCell "#0","5"
16: JavaWindow("Saisie des contraintes").JavaTable("TableAppli").SetCellData "#0","5","5"
17: 'colonne 6 : 2h30 à 3h00, puissance = 700
18: JavaWindow("Saisie des contraintes").JavaTable("TableAppli").ActivateCell "#0","6"
19: JavaWindow("Saisie des contraintes").JavaTable("TableAppli").SetCellData "#0","6","6"
```

Dans ce cas, le tableau est correctement rempli mais cette méthode est loin d'être pratique lorsqu'il faut remplir plusieurs tableaux de quarante-huit colonnes.

C'est pour cette raison que j'ai mis en place la fonction « rempliTableau » permettant de saisir correctement les colonnes du tableau (« 4.4.2.5 La bibliothèque Tableau »), qui se trouve ci-dessous.

```
'Fonction de remplissage d'une ligne d'un tableau
'Tableau : le tableau Java à remplir
'Ligne : le numéro de la ligne du tableau à remplir
'Donnees : le tableau des valeurs servant à remplir le tableau
'debut : l'indice de la première case à modifier
'nbElement : le nombre de cas consécutive à mettre à jour
Function rempliTableau(Tableau, Ligne, Donnees, debut, nbElement)
      For i = 1 to nbElement
            Tableau.SetCellData Ligne,i+debut-1,Donnees(i-1)
      Next
End Function
```

Voici un exemple d'appel de la fonction de l'exemple précédent.

```
'Valeurs à mettre dans le tableau
chaine="850;700;700;700;700;700"
'Transformation des valeurs en tableau pour les passer en paramètre
mesDonnees=Split(chaine,";")
'Appel de la fonction rempliTableau
rempliTableau JavaWindow("Saisie des contraintes").JavaTable("TableAppli"), 0,
mesDonnees, 1 , 6
```

4.4.6.4 Checkpoint bitmap

J'ai rencontré plusieurs problèmes en utilisant le point de contrôle sur image de QTP.

- Fond noir

Pour les tests du client URSE j'ai utilisé de nombreux checkpoints bitmap car la plupart des données sont représentées par des graphiques.

Le problème c'est qu'un même test peut passer à cinq minutes d'intervalle du statut ok au statut échec.

Parfois QTP essaye de comparer l'image enregistrée (Expected bitmap) avec l'image présente dans EMMA URSE (Actual bitmap) mais celle-ci se trouve sous un fond noir.

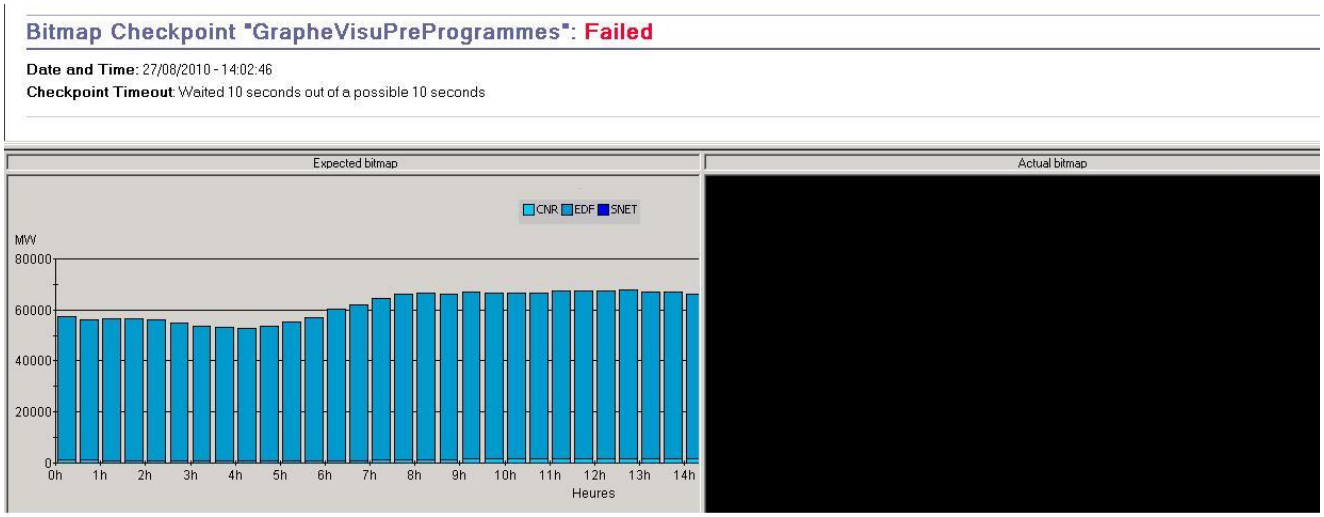


Figure 34 : Problème checkpoint Bitmap : fond noir

- Modification de l'écran

Un autre problème est survenu lors des tests de non-régression sur la V2.3.

Ci-dessous nous pouvons voir un rapport en erreur. Dans la partie « Expected bitmap » se trouve le graphique enregistré avec la V2.2 d'EMMA URSE. A droite se trouve le graphique de la V2.3. Le test est passé en erreur. A première vue, il est difficile d'apercevoir une différence entre les deux graphiques. C'est en cliquant sur le bouton « View Difference » que j'ai pu constater que le graphique de la V2.3 est plus tassé que celui de la V2.2, certainement à cause d'une légère modification dans l'IHM.

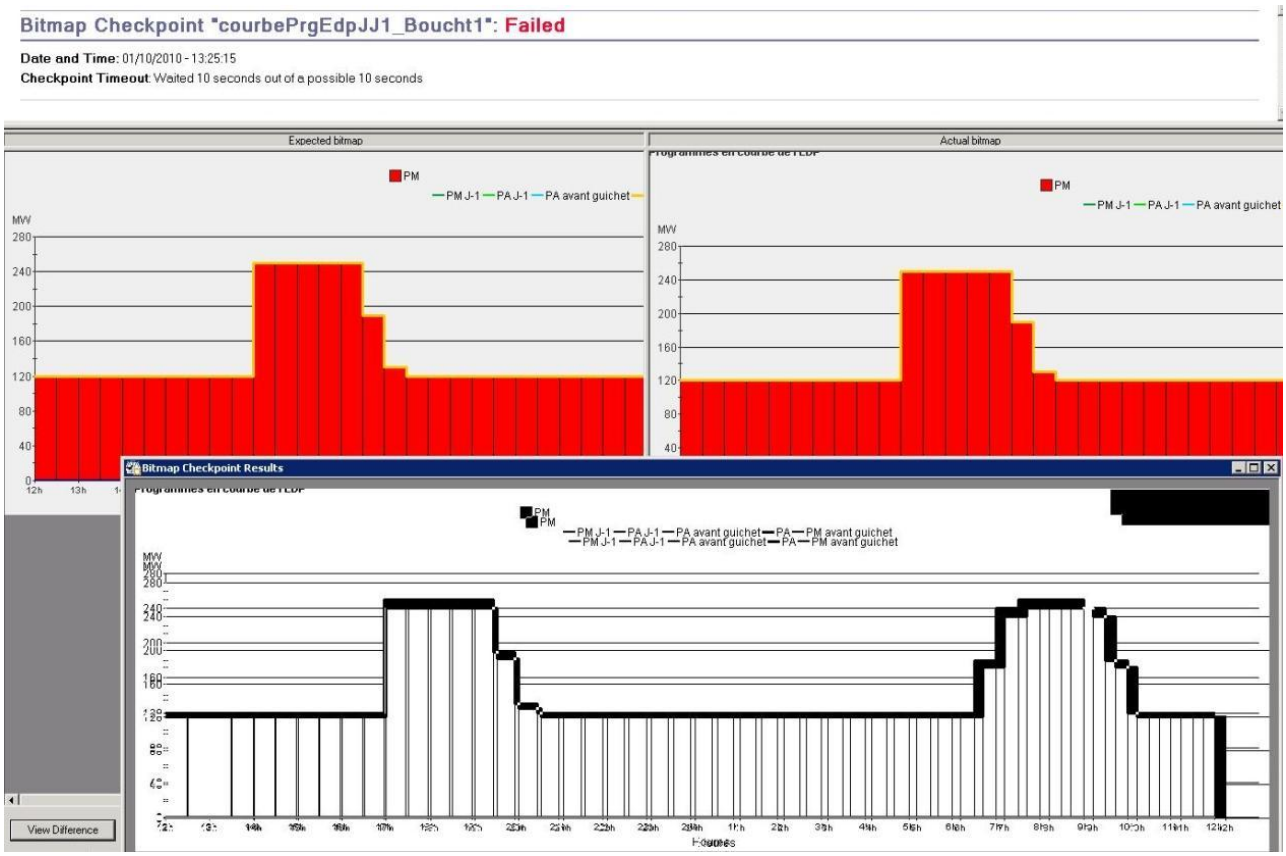


Figure 35 : Problème checkpoint Bitmap : modification

Ce checkpoint n'est pas très fiable. Pour la suite des tests j'ai essayé de l'utiliser le moins possible.

4.4.6.5 Passage d'EMMA de la V2.2 à la V2.3

Le passage d'EMMA de la V2.2 à la V2.3 a été assez laborieux car l'équipe projet et moi-même avions peu d'expérience sur l'automatisation des tests et il s'agissait de la première fois que les tests de non-régression crées en V2.2 étaient exécutés sur une nouvelle version.

Lorsque j'ai exécuté quelques tests de non-régression sur cette nouvelle version je me suis retrouvée avec tous les tests en erreur.

Dans l'exemple ci-dessous, les entités d'ajustement MAREGH et SSPMASH (lignes deux et trois) n'avaient pas été créées. Toutes les lignes suivantes étaient donc erronées.

	EDA	Catég-*	Offre dé	Heure li	Type*d'	Volume*	Prix*J :*	Prix*J :*	Prix*J :*	Région
1	DASK1TCD	E			H	100*121	---	---	---	NATION
2	MAREGH (vallée MAREGH)	P			H	25*123.	---	---	---	SESO
3	SSPMAH (vallée SSPMAH)	P			H	15*125.	---	---	---	SESO
4	DASK2TCD	E			H	100*126	---	---	---	NATION
5	NOK1 TCH	E			H	50*130.	---	---	---	NATION
6	DASK3TCD	E			H	150*132	---	---	---	NATION
7	CRUA5T 2	P			H	110*137	---	---	---	SESE
8	MTPEZH (vallée MTPEZH)	P			H	3*137.8	---	---	---	SERAA
9	VINTRH (vallée VINTRH)	P			H	8*137.9	---	---	---	SESO
10	LUZI2H (vallée LUZI2H)	P			H	3*137.9	---	---	---	SESO
11	HOSPIH (vallée HOSPIH)	P			H	28*139				SESO
12	ZZZCHA (vallée ZZZCHA)	P			H					SESE
13	GRANDH (vallée GRANDH	P			H					SESO
14	VINONH (vallée DURESH)	P			H					SESE
15	ZZZJOU (vallée DURESH)	P			H					SESE
16	ZZZF2H (vallée DURESH)	P			H					SESE
17	ZZZF4H (vallée DURESH)	P			H					SESE
18	NOK2 TCH	E			H					NATION
19	CHEY6H (vallée CHEY6H)	P	STEP T		H					SERAA
20	BISSOH (vallée BISSOH)	P			H	66*141.	---	---	---	SERAA
21	L.OO H (vallée L.OO H)	P			H	6*141.8	---	---	---	SESO
22	BATHIH (vallée BATHIH)	P			H	64*142	---	---	---	SERAA

Test Results

Expected Value: MAREGH (vallée MAREGH)

Actual Value: DASK2TCD

OK

Figure 36 : Erreur données

Après une première analyse de l'équipe projet d'EMMA, il s'est avéré que les scripts de migration étaient en erreur.

L'automatisation des tests a permis de se rendre compte plus rapidement de ces erreurs.

Une fois les corrections apportées sur les scripts de migration, j'ai relancé les tests de non-régression mais quelques-uns étaient toujours en erreur.

Cette fois-ci nous nous sommes aperçus que les dumps de référence de la V2.2 et de la V2.3 n'étaient pas identiques. Nous avons pu voir les causes de ces différences dans la partie « 4.3.2.1 Gestion des bases de données ».

Il a fallu vérifier chaque cas pour savoir si l'erreur était du à une non-régression ou à une erreur dans les données en entrée.

4.4.6.6 Impression du guichet

Un certain nombre de tests nécessite une validation du guichet de redéclarations. Pour cela il faut passer par une impression de ce guichet.

J'ai rencontré différents problèmes à ce niveau.

- Affichage d'une pop-up

Lorsque l'on valide le guichet, l'impression de celui-ci se lance. Puisque sur la machine où se trouve QTP l'imprimante par défaut est « Microsoft XPS Document Writer », une pop-up apparaît pour enregistrer le fichier.

Lors de l'enregistrement, QTP reconnaît correctement la pop-up.

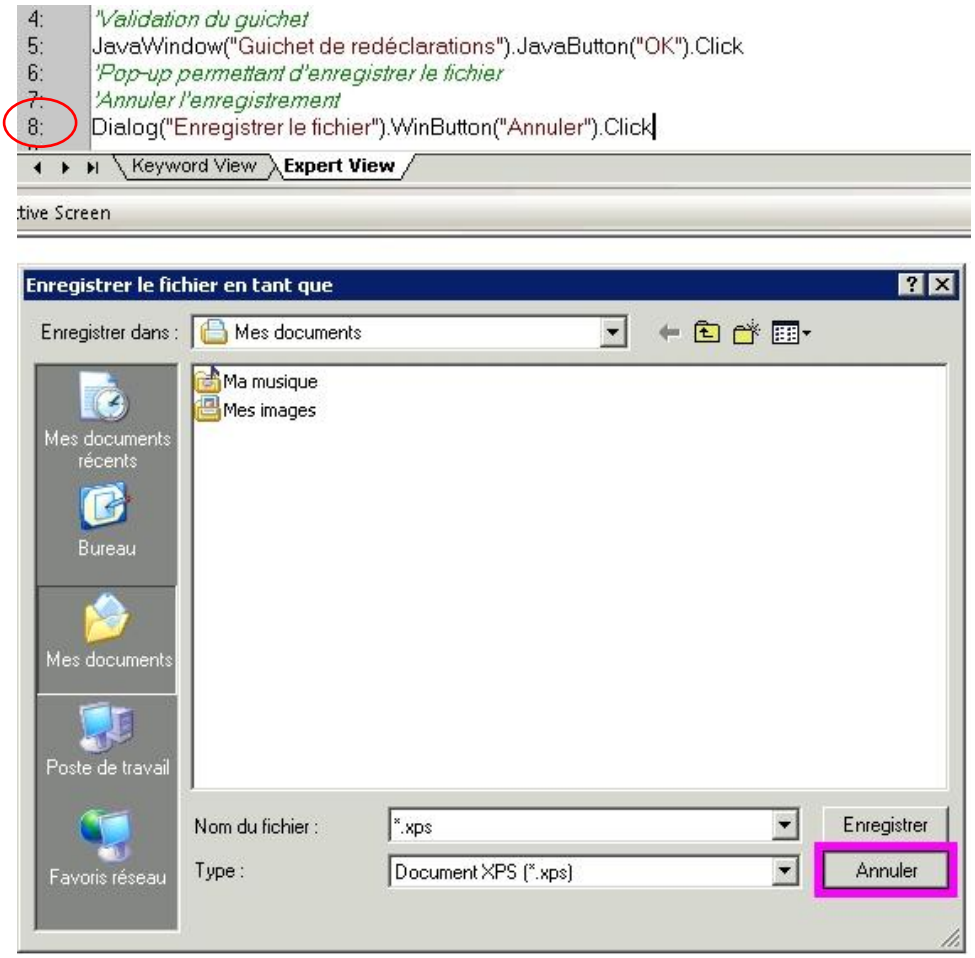


Figure 37 : Enregistrement du guichet

Au moment de l'exécution, le script passe de façon très aléatoire. La plupart du temps QTP ne retrouve plus la pop-up alors qu'elle est bien ouverte.

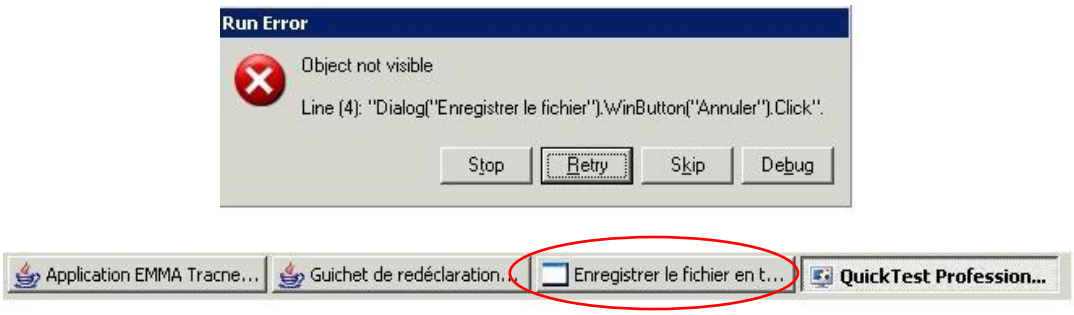


Figure 38 : Erreur lors de l'enregistrement du guichet

Il a donc fallu trouver une solution pour que le script ne bug pas.

J'ai utilisé les différents types d'enregistrement vus dans le paragraphe « Les types d'enregistrement ».

Tout d'abord j'ai testé l'enregistrement du guichet à l'aide du mode d'enregistrement « Low level ».

```
'Validation du guichet
Window("Bilan").Click 1036,60
'Pop-up permettant d'enregistrer le fichier
'Annuler l'enregistrement
Window("Enregistrer le fichier").WinObject("Annuler").Click 31,8
```

L'exécution du test s'est correctement déroulée. J'ai quand même testé avec le mode « Analog ». Dans ce mode, avant d'exécuter le script, il faut renseigner la fenêtre de l'application qui doit être testé.

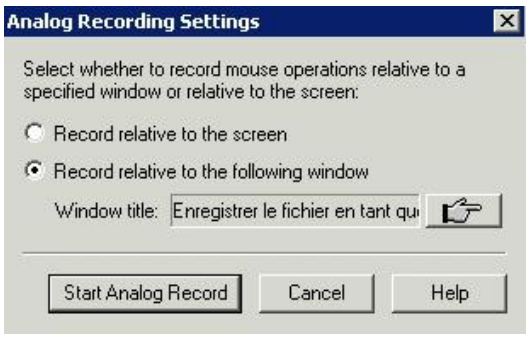


Figure 39 : Enregistrement Analog du guichet

```
'Validation du guichet
Window("Bilan").RunAnalog "Track1"
'Pop-up permettant d'enregistrer le fichier
'Annuler l'enregistrement
Window("Enregistrer le fichier").RunAnalog "Track2"
```

Le script généré n'est pas très explicite, j'ai malgré tout choisi ce mode pour l'enregistrement du guichet de redéclarations plutôt que le mode « Low level » qui enregistre les coordonnées de l'écran.

Lors du lancement des tests automatisés pour la V2.3 et la V2.4 d'EMMA, je n'ai pas rencontré de problème avec cette méthode.

- Choix de l'imprimante par défaut

Au moment d'enregistrer les scripts faisant appel à l'impression du guichet, l'imprimante par défaut était « Microsoft XPS Document Writer ».

Si l'imprimante par défaut est modifiée, l'exécution des scripts risque de passer en erreur car les pop-up seront différentes en fonction du choix d'impression.

Par exemple, si l'imprimante par défaut est « PDFCreator », à la place de la pop-up « Enregistrer le fichier en tant que », nous trouvons cet écran :

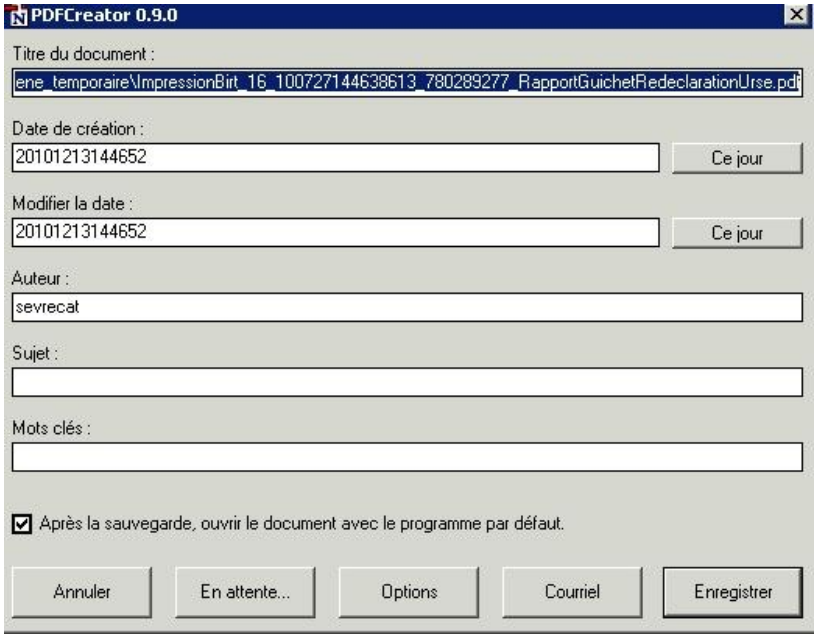


Figure 40 : Impression du guichet avec PDFCreator

Pour éviter toute erreur, j'ai crée une fonction en Visual Basic Script qui modifie l'imprimante par défaut (« 4.4.2.6 La bibliothèque Imprimante ») et je l'ai appelé au début des tests le nécessitant.

```
'Modifier l'imprimante par défaut
'Si l'imprimante n'est pas "Microsoft XPS Document Writer", le test sera en
'erreur 'à l'impression du guichet
nomImprimante = "Microsoft XPS Document Writer"
res = modifImprimanteDefaut(nomImprimante)
If res = false Then
      'Afficher un message d'erreur
      Reporter.ReportEvent micFail, "Erreur : imprimante par defaut", "Erreur
      : l'imprimante " & nomImprimante & " n'a pas été trouvée"
End If
```

4.4.6.7 Perte des add-ins

Lors de l'exécution des tests automatisés à partir de QC, j'ai perdu plusieurs fois les add-ins QTP liés au test (« 3.3.1.1 Sélection des add-ins »). Seul l'add-in « Web » était présent ce qui évidement posait problème pour l'application EMMA qui est en Java. QTP ne retrouvait plus des objets et le test passait en erreur.

Ce bug survenait de manière très aléatoire et sur n'importe quel test.

Après quelques recherches en collaboration avec l'équipe MAM, nous avons trouvé un problème ressemblant au notre sur le forum de l'éditeur HP.

Dans cette discussion, il est conseillé de modifier le « Template Test » (un modèle de test), deux manières différentes de procéder sont décrites.

- Première méthode

La première solution est d'ouvrir le « Template Test » se trouvant sur la machine QTP à l'adresse suivante :

« D:\Program Files\HP\Quick Test Add-in For Quality Center\bin\Templates\Template10 ».

Puis de modifier les add-ins qui lui sont associés en ajoutant ceux qui nous intéressent pour le projet EMMA et d'enregistrer ces changements.

Après avoir effectué ces modifications, j'ai lancé plusieurs scripts à la suite à partir de QC. J'ai eu de nouveau une perte d'add-in pour l'un des tests.

J'ai donc mis en place la deuxième solution.

- Deuxième méthode :

La deuxième méthode apportée sur le forum est de créer soit même un « Template Test ».

Il faut ouvrir QTP avec les add-ins que l'on souhaite lancer puis enregistrer ce test vide.

Dans QC il faut définir que le test qui vient d'être crée est un « Template Test ». Pour ce faire nous devons aller dans le module « Test Plan », rechercher le test crée, cliquer droit sur celui-ci et choisir l'option « Mark as Template Test ».

A chaque fois qu'un nouveau test automatisé est crée, il faut passer par l'outil QC et sélectionner le « Template Test » associé.

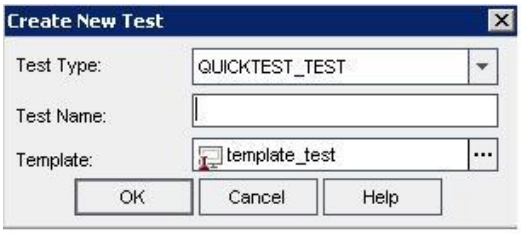


Figure 41 : Sélection d'un Template Test

Le problème avec cette solution c'est qu'il n'est pas possible d'associer un « Template Test » à un test déjà crée.

Pour vérifier que cette solution fonctionne j'ai du recréer les quelques tests qui avaient posé problème. Je n'ai pas pu faire cela pour tous les tests car il y avait déjà plus d'une centaine de tests automatisés qui avaient été crées.

Par contre, j'applique cette méthode dès qu'il y a un nouveau test. Je n'ai pas rencontré de nouveau ce problème sur les tests disposant du « Template Test ».

J'ai ajouté dans ce « Template Test » les variables d'environnement. De cette manière, les nouveaux tests disposent des variables d'environnement.

J'ai hésité à ajouter également les bibliothèques de fonctions mais contrairement aux variables, les fonctions ne sont pas utilisées dans tous les tests.

4.4.7 Modification des scripts QTP

Lors d'un changement de version il peut y avoir des scripts à mettre à jour. Cela m'est arrivé lors du passage de la V2.2 à la V2.3 d'EMMA.

Pour l'un des tests, j'obtenais des erreurs de checkpoints lors du passage en V2.3 car l'IHM avait été changé.

Le tableau de l'application de la V2.2 dispose de nombreuses colonnes qui ont été supprimées en V2.3.

Voici le tableau présentant les offres et conditions à la hausse de la V2.2 :

EDA	Type d'offre	00H00 - 06H00 Prix	00H00 - 06H00 Rang	PMD (MW)	Min.te (MW)	RPF (MW)	RSFP (MW)	P.max (hydrau) (MW)	Durée (Heures)	Vol.max (MWh)	Délai Appel (Minutes)	Délai Mob (Minutes)	Remarque Opérateur
AET1 TCH	-	N.O	205										
AIGLEH	-	72.28	102	#	#	#	#	309	#	3137		13	
ARAMOT 1	-	101.96	140	685	200	70	100		#	#	480	55	
ARAMOT 1	D	60548.0	199	685	200	70	100		#	#	480	55	

Figure 42 : Tableau des offres et conditions de la V2.2

Et celui de la V2.3 :

EDA	Type d'offre	00H00 - 06H00 Prix	Délai Mob (Minutes)	Remarque Opérateur
AVEN1TCH	-	0.0		
AVEN2TCH	-	0.0		
AVEN3TCH	-	0.0		
BALITTNG	-	38.61		
BLAYAT 2	-	45.27	15	

Figure 43 : Tableau des offres et conditions de la V2.3

Si l'application a légèrement été modifiée, la fonctionnalité « Update Run Mode » de QTP peut être utilisée.

Cette fonctionnalité permet d'exécuter le script tout en mettant à jour celui-ci en modifiant les checkpoints et/ou les objets. Cela évite de refaire l'enregistrement du test manuellement.

Il faut bien évidemment lancer cette fonction après avoir correctement vérifié qu'il n'y a pas de régressions dans l'application.

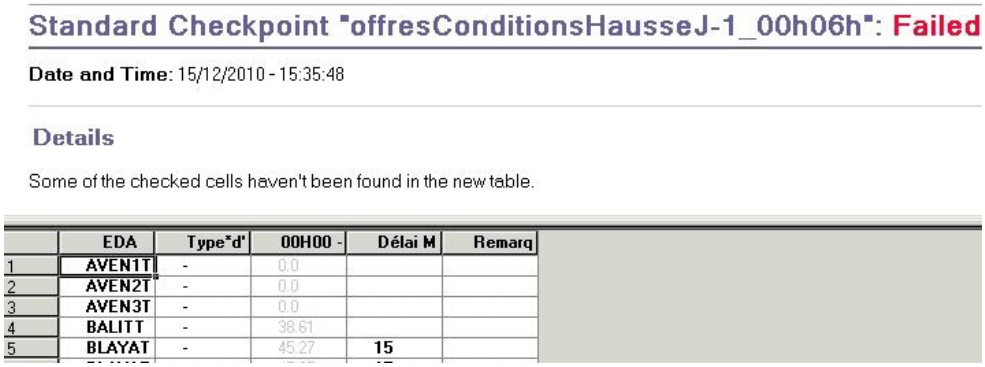
Dans l'exemple précédent sur les offres et conditions à la hausse, nous pouvons constater qu'en plus de la modification de l'IHM, il y a des erreurs au niveau des données de l'application.

J'ai testé la fonctionnalité « Update Run Mode » pour voir si QTP retrouvait des erreurs ou mettait simplement à jour le tableau avec les nouvelles données.

Pour utiliser cette fonction, il faut commencer par cocher ce que l'on souhaite mettre à jour (checkpoints, objets, copies d'écran,…).

Ensuite QTP exécute le script.

J'ai été rassurée de voir dans le rapport crée par QTP que l'outil avait bien trouvé des erreurs dans les données.



	EDA	Type"d'	00H00 -	Délai M	Remarq
1	AVEN1T	-	0.0		
2	AVEN2T	-	0.0		
3	AVEN3T	-	0.0		
4	BALITT	-	38.61		
5	BLAYAT	-	45.27	15	

Figure 44 : Rapport d'erreur suite à l'Update Run Mode

QTP précise qu'il a trouvé des incohérences dans les données. Il a ensuite fallu rechercher d'où venaient les erreurs. Il s'est avéré qu'il n'y avait pas les mêmes données en entrée dans la V2.2 et dans la V2.3.

J'ai remarqué un autre point avec la fonction « Update Run Mode ». Si elle est exécutée sur des scripts contenant des expressions régulières, QTP supprime ces expressions et les remplace par les nouvelles valeurs. Ce problème est un peu gênant avec l'application EMMA car nous avons du utiliser un certain nombre d'expression régulière.

4.5 Bilan

Le bilan pour la réalisation de la recette fonctionnelle outillée sur l'application EMMA est plutôt positif.

En ce qui concerne, la gestion de la recette avec Quality Center, j'ai mis en place un guide de bonnes pratiques et de nombreuses exigences et fiches de test sont désormais renseignées dans l'outil.

Ce qui m'a posé le plus de problème c'est l'organisation des exigences et donc des fiches de test puisque j'ai décidé de trier ces deux domaines de la même façon.

Le fait de séparer les évolutions et la non-régression demande un travail supplémentaire à l'équipe projet. A la fin de chaque recette, elle doit passer les évolutions les plus critiques dans la partie non-régression. De plus, une FFT peut contenir des évolutions ou des corrections d'anomalies dans plusieurs clients d'EMMA. De ce fait, l'équipe devra séparer l'évolution en plusieurs parties. Dans certains cas, des exigences de non-régression déjà existantes devront être modifiées pour contenir des évolutions. Cette méthode est loin d'être pratique mais elle me semble la plus adaptée à la complexité des évolutions d'EMMA. Heureusement pour la majorité des applications, cette solution n'est pas nécessaire. L'application EMMA est un cas assez particulier puisqu'elle dispose de nombreuses recettes au cours de l'année et elles sont toutes plus ou moins longues.

Un autre problème avec cet outil c'est qu'il demande du temps pour réaliser le suivi de la recette. Il faut, entre autre, renseigner les liens entre les différents éléments, décrire correctement les informations sur la recette (date, anomalies,…). Ces données ne sont pas toujours renseignées ou mises à jour, par conséquent le suivi de la recette ne peut pas se faire correctement. Ce point est dommage car il s'agit d'un des avantages majeurs de QC. Le module « Dashboard », qui permet de créer des graphiques et des rapports sur l'avancement de la recette, n'est absolument pas utilisé pour le moment.

Je pense que la solution serait de permettre au métier d'accéder à l'outil. QC permet cette réalisation avec les différents rôles utilisateurs. Le métier pourrait renseigner les exigences et suivre l'avancement des campagnes de tests.

Un autre problème rencontré dans la mise en place de la recette outillée d'EMMA est la création des dumps de référence. Il s'agit encore d'un cas assez particulier car EMMA travaille avec des données J et J-1. Pour la recette d'EMMA, il est plus simple d'utiliser plusieurs dumps de référence correspondant à un instant de la journée. Le choix de l'équipe projet était de travailler à partir d'un dump de production mais nous avons vu que cette solution n'est pas envisageable avec l'automatisation car il y a de nombreuses différences entre les données. J'ai choisi d'utiliser un dump initial à partir de la version précédente de test. Le souci avec cette solution c'est qu'elle demande encore un travail supplémentaire pour l'équipe projet. Celle-ci devra modifier ce dump initial pour prendre en compte les évolutions qui passeront dans la non-régression pour que les données soient en adéquation avec les tests.

Pour la mise en place des tests de non régression automatisés, contrairement au premier outil choisi par RTE (Rational Functional Tester d'IBM) Quick Test Professional a montré qu'il fonctionnait correctement sur plusieurs versions de l'application. J'ai crée plus d'une centaine de tests automatisés et de nombreuses fonctions Visual Basic Script.

L'une des fonctions que j'ai crée, celle pour la comparaison de deux fichiers texte, est à améliorer. J'ai écrit cette fonction pour comparer ligne par ligne deux fichiers texte. Celle-ci est très utile pour l'application EMMA car il y a de nombreux fichiers qui sont générés, il faut donc vérifier les résultats avec les fichiers se trouvant sur le serveur des données de référence. Cette fonction renvoie, dans le rapport d'erreur QTP, pour chaque ligne en erreur : le numéro, le texte du fichier généré et le texte du fichier source. Le problème avec ce résultat c'est qu'il est très difficile d'identifier clairement les différences entre les fichiers. Le plus problématique c'est que si la première ligne du fichier est différente toutes les lignes sont considérées en erreur.

L'un des problèmes que j'ai découvert pendant mon stage est l'utilisation du checkpoint Bitmap. Je ne conseille pas d'utiliser celui-ci ou alors le moins possible. Ce checkpoint n'est pas fiable puisque dès qu'il y a une légère modification dans l'IHM, celui-ci risque d'être en erreur. Lors de l'exécution de certains tests, je me suis retrouvée, dans le rapport d'erreurs, avec un fond noir à la place de l'image obtenue. Pour régler ce problème, j'ai préconisé à l'équipe projet d'utiliser soit un deuxième écran soit un autre ordinateur pour l'exécution des tests automatisés

Pendant ce projet, j'ai également donné une formation à l'équipe d'EMMA pour qu'elle puisse mettre à jour les tests et en créer de nouveaux après la fin de mon stage mais, contrairement à QC, QTP est plus technique. Certains tests peuvent être simples à créer mais il faudrait un expert QTP pour intervenir dans les cas les plus complexes. Les tests automatisés ont permis à l'équipe d'EMMA de découvrir des erreurs plus rapidement, cela serait regrettable que ces tests ne soient plus utilisés à cause de problème de maintenance.

5. AUTOMATISATION DES TESTS DE L'APPLICATION DECOMPTES

D'autres équipes projet du département DPCM ont souhaité mettre en œuvre les outils de tests. Je leur ai présenté les outils HP et je leur ai expliqué le travail effectué avec l'application EMMA. Puis je les ai aidés à mettre en place ces outils. Pour l'application DECOMPTES, j'ai tout d'abord testé la compatibilité avec l'outil QTP.

5.1 Présentation

5.1.1 Description fonctionnelle

L'application DECOMPTES permet les décomptes des écarts de responsables d'équilibre et des pertes du RTE.

DECOMPTES a pour objectifs de :

- Calculer les agrégats constitués de données de flux Physique (télérelèves) et du Marché (bourse...) afin de les publier aux clients et au CIREF (Centre d'Information du Réseau Electrique Français) ;
- Valoriser les Ecarts énergétiques des Responsables d'Equilibres afin de procéder à leur facturation ;
- Déterminer les pertes énergétiques du RTE en calculant des Bilans en énergie France et Régions transmis à une autre application de RTE.

5.1.2 Architecture technique

L'application DECOMPTES est basée sur une architecture client/serveur qui fonctionne avec un package télé-diffusable sur tout PC de RTE permettant l'installation de raccourcis de connexion au serveur distant (connexion bureau à distance paramétré) et l'utilisation du réseau local de l'utilisateur notamment pour les imprimantes.

Ce package communique avec :

- Un serveur applicatif Windows 2003 Server SP1 en utilisant une session Terminal Server (TSE) initiée par le poste client à la connexion distante. Cette dernière permet d'exécuter le « client DECOMPTES » qui échange des informations avec :
- Les bases Oracle des données modifiables (« en ligne ») et archivées (hébergées sur un serveur Linux sous RedHat 5.2) qui mettent à disposition les données vers chaque session TSE utilisateur ou sur les serveurs UNIX et Windows 2003 TSE pour exports applicatifs via EAI ou FTP « en direct ».

Les logiciels et les outils utilisés pour l'application DECOMPTES sont :

- IHM : ILOG Views et Objecteering
- Moteur de calcul : PL/SQL et Langage C++
- Base de données : Oracle 10.2

5.2 Mise en œuvre de QTP

Pour débuter la mise en place de QC et QTP sur DECOMPTES, le chef de projet de cette application m'a fourni un cas de test très simple.

Ce test permet de se connecter à l'application, faire une recherche sur un consommateur et quitter l'application.

J'ai rencontré plusieurs problèmes pour automatiser ce test.

5.2.1 Connexion à l'application

Un premier problème s'est posé pour la connexion à l'application.

Il faut tout d'abord lancer la connexion au serveur TSE de recette de DECOMPTES à l'aide d'une « Connexion Bureau à distance ».

J'ai voulu tester la connexion d'un bureau à distance à partir de QTP.

```
4:  'Lancer la connexion au bureau à distance
5:  SystemUtil.Run "C:\WINDOWS\system32\mstsc.exe","","",""
6:  'Saisir l'adresse IP du serveur Tse utilisé pour les tests
7:  Dialog("Connexion Bureau à distance").WinEdit("Ordinateur :").Set ip
8:  'Se connecter
9:  Dialog("Connexion Bureau à distance").WinButton("Connexion").Click
10: 'Message d'alerte RTE, cliquer sur OK
11: Window("Bureau à distance").WinObject("Input Capture Window").Click 801,556
12: 'Saisir le login
13: Window("Bureau à distance").WinObject("Input Capture Window").Type login
14: 'Passer au champ suivant
15: Window("Bureau à distance").WinObject("Input Capture Window").Type  micTab
16: 'Saisir le mot de passe
17: Window("Bureau à distance").WinObject("Input Capture Window").Type mdp
18: 'Renseigner le champ : se connecter à
19: Window("Bureau à distance").WinObject("Input Capture Window").Click 875,418
20: Window("Bureau à distance").WinObject("Input Capture Window").Click 758,460
21: 'Cliquer sur Ok
22: Window("Bureau à distance").WinObject("Input Capture Window").Click 669,479
23: 'Nous sommes connectés sur le serveur
24: 'Affichage d'un message RTE
25: Window("Bureau à distance").WinObject("Input Capture Window").Click 916,554
26: 'Fermer la session
27: Window("Bureau à distance").WinObject("Input Capture Window").Click 31,1040
28: Window("Bureau à distance").WinObject("Input Capture Window").Click 50,973
29: Window("Bureau à distance").WinObject("Input Capture Window").Click 753,544
30:
```

A partir de la ligne 11, nous pouvons constater que le seul objet reconnu par QTP est « Input Capture Window » et que QTP est passé par les coordonnées de l'application.

Après plusieurs re-jeux, le test s'exécute correctement jusqu'à la connexion sur le serveur (ligne 23). Cependant ce test ne pourra plus fonctionner lorsqu'il y aura une modification au niveau de l'écran car les coordonnées enregistrées se seront plus correctes.

A partir de la ligne 25, une alerte s'affiche au bout de quelques secondes. Le problème c'est que QTP n'attend pas ce message et l'exécution du test se termine avant l'arrivée de celui-ci.

Il serait possible d'ajouter dans le script un « Wait » de quelques secondes mais de toute façon il n'est pas envisageable d'automatiser correctement des tests en passant par la connexion du bureau à distance à cause de l'utilisation des coordonnées.

Nous avons pu contourner le problème en installant l'application DECOMPTES directement sur la machine où se trouve QTP.

Il suffit tout simplement de lancer un fichier Batch à partir du disque local pour lancer la connexion à l'application DECOMPTES.

5.2.2 Librairie ILOG Views

Après avoir installé l'application sur la machine QTP, j'ai enregistré un premier test permettant de se connecter à DECOMPTES.

Lorsque j'ai arrêté le script j'ai découvert que QTP était passé par les coordonnées de l'application.

```
'Saisir le login
Window("Window").Drag 338,104
Window("Window").Drop 78,86
Window("Window").Type login
'Saisir le mot de passe
Window("Window").Drag 342,135
Window("Window").Drop 83,119
Window("Window").Type mdp
'Se connecter
Window("Window").Click 165,177
'Affichage de l'écran d'accueil
Window("MGAccueilDecomptelhm").Click 743,594
```

Afin de comprendre pour quelles raisons QTP n'arrive pas à retrouver les objets de DECOMPTES, j'ai utilisé l'outil « Object Spy » de QTP.

Cet outil permet de sélectionner un objet de l'application et de visualiser les propriétés et les opérations qui lui sont associés.

J'ai utilisé l'outil « Object Spy » sur la page de connexion à l'application et je me suis aperçue que quel que soit l'objet sélectionné (le champ de login, le bouton de connexion,…), QTP trouve un seul objet : la page de connexion.

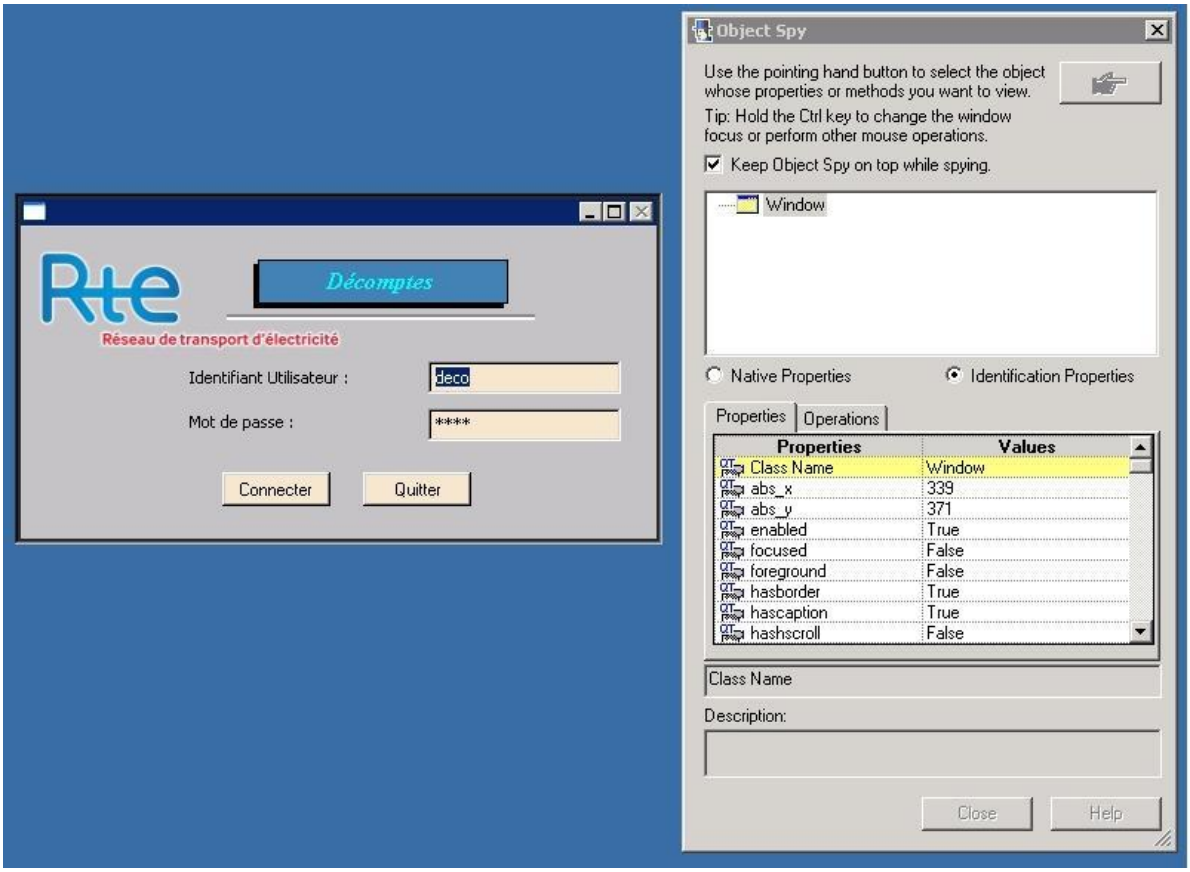


Figure 45 : Utilisation de la fonction « Object Spy »

Après quelques recherches, j'ai découvert que c'est l'utilisation de la librairie ILOG Views dans l'application DECOMPTES qui pose problème.

Les interfaces ILOG Views de la société ILOG (racheté par IBM en 2009) permettent aux développeurs C++ de créer rapidement de puissantes interfaces graphiques utilisateur ciblant de très nombreuses plates-formes.

ILOG Views encapsule les classes graphiques système par des classes spécifiques ce qui ne permet pas à la plupart des robots de test automatisé d'accéder aux objets ILOG Views et à leurs caractéristiques.

L'éditeur Compuware et Pacte Novation, partenaire d'ILOG, se sont associés afin de créer la première solution de tests pour ILOG Views, QAViews.

Cette solution s'appuie sur la suite standard QACenter Enterprise Edition. Elle est complétée par un module d'automatisation pour Views, intégré aux outils Compuware.

5.2.3 Bilan

La réalisation d'une recette fonctionnelle outillée pour l'application DECOMPTES est un échec.

Les fiches de tests sont toujours renseignées dans un document Word et il n'y a pas de réel suivi de la recette. Le chef de projet de cette application n'a pas souhaité utilisé QC pour le moment car son objectif principal était la création de tests automatisés.

L'automatisation des tests de non-régression de l'application DECOMPTES est le véritable problème. QTP ne retrouve pas les objets de l'application. Lors de l'enregistrement d'un script de test, QTP utilise les coordonnées de l'écran.

Il est inenvisageable d'utiliser, pour tous les tests de DECOMPTES, les coordonnées de l'écran ou les méthodes spécifiques de QTP pour l'enregistrement des scripts (3.4.3.1 Les types d'enregistrement) car, comme nous l'avons vu auparavant, ces solutions ne sont pas assez fiables.

D'après mes recherches, c'est la librairie ILOG Views utilisée dans le développement de DECOMPTES qui pose problème pour la plupart des automates de test.

Je n'ai pas trouvé de solution pour faire fonctionner QTP sur ce type d'application. J'ai interrogé le support d'HP mais je n'ai pas obtenu de réponse. La réalisation d'une recette fonctionnelle outillée pour l'application DECOMPTES n'a donc pas pu se faire.

CONCLUSION

Mon travail fournit sur la recette d'EMMA a permis à l'équipe de ce projet de réaliser des recettes fonctionnelles outillées avec les outils HP. Cette équipe continue à utiliser ces outils en prenant en compte les différents conseils que j'ai pu leur transmettre.

Mes présentations sur les outils ont permis aux autres équipes projet du département DPCM de se rendre compte de l'importance d'un outil de suivi de recette et des avantages procurés par l'automatisation des tests de non-régression.

Un chef de projet m'a d'ailleurs dit qu'il regrettait de ne pas avoir eu le temps de mettre en place ces outils pendant mon stage car cela lui aurait permis de découvrir de la non-régression avant que son projet soit mis en production.

Mon seul regret est de ne pas avoir pu aider d'autres projets qu'EMMA à mettre en place une recette outillée. Malgré les nombreux avantages de cette méthode, cela demande beaucoup de temps à consacrer à l'organisation et le retour sur investissement est loin d'être immédiat ce qui a un peu découragé certains chefs de projet.

J'espère qu'avec le succès de la mise en place de la recette outillée d'EMMA, d'autres projets vont passer très prochainement vers la recette outillée. Pour cela, ils pourront s'aider des documentations de bonnes pratiques que j'ai rédigées pour l'utilisation de QC et de QTP. Les bibliothèques de fonctions Visual Basic Script devront également être mises à leur disposition pour leur faire gagner du temps sur la création des scripts QTP.

Mon stage au sein de RTE m'a énormément apporté sur le plan personnel et professionnel.

Pendant ces neuf mois, j'ai mis en œuvre les outils HP pour la recette de l'application EMMA. J'ai formé les utilisateurs à ces outils et j'ai présenté le travail effectué à d'autres équipes.

Tout d'abord, je n'avais aucune expérience dans l'environnement de tests. Ce stage m'a permis de découvrir d'autres métiers informatique que je ne connaissais pas et j'ai pu voir les différentes phases de la recette applicative.

De plus j'ai appris à utiliser les outils HP Quality Center et Quick Test Professional qui sont très demandés sur le marché du travail.

Il m'a fallu également trouver des solutions pour mettre en œuvre ces outils sur la recette de l'application EMMA et proposer mes choix à l'équipe projet.

La mise en place d'une recette outillée doit être une décision mûrement réfléchie. Celle-ci demande du temps, il faut donc que l'application à tester dispose de nombreuses évolutions. Les tests automatisés ne doivent pas être passés au détriment des tests manuels. Ces deux types de tests sont indispensables pour une recette complète.

Le principal problème de l'industrialisation des tests est la maintenance. De nombreux projets ont abandonné cette pratique car la mise à jour des exigences et des fiches de tests est parfois laborieuse.

Une solution proposée dans le livre « Industrialiser le test fonctionnel » [LEGEARD2009] est d'utiliser un outil pour la génération des tests. Un générateur de tests génère automatiquement les cas et scripts de test à partir modèle représentant les comportements attendus de l'application.

Je pense que c'est une solution qui peut être mise en place pour assurer le suivi des évolutions fonctionnelles et accélérer la production du référentiel de tests automatisés.

GLOSSAIRE

CNER = Centre national d'expertise réseaux

CNES = Centre National de l'Exploitation du Système électrique

CSV = Comma-Separated Values (format représentant des données tabulaires sous forme de « valeurs séparées par des virgules »)

DPCM = Département Programme Clients Marché

EDA = Entité D'Ajustement (regroupement au niveau national, d'EDP, de consommateurs et d'échanges)

EDP = Entité De Production

FFT = Fiche de Fait Technique

FTP = File Transfer Protocol (protocole de transfert de fichiers)

GRT = Gestionnaire du Réseau de Transport

QC = Quality Center

QTP = Quick Test Professional

MAM = Mission Appui Méthodes

Offres = Proposition d'augmentation ou de diminution de l'énergie présente sur le réseau en jouant sur la production ou le soutirage, en contrepartie d'une rémunération.

PA = Programmes d'Appel

PARC = Programmation Ajustement Référentiels et Clients

Pertes électriques = la différence entre les énergies injectées sur le réseau RTE et les énergies soutirées sur ce réseau. Les Pertes sont rachetées par RTE.

PM = Programme de marche

Produits = Désigne les offres élaborées et diffusées par un vendeur à destination d'acheteurs potentiels.

RP = Responsable de Programmation

RTE = Réseau de Transport d'Electricité, GRT Français

SI = Système d'Information

SSY = Services Système

TR = Temps Réel

URSE = Unité Régionale du Système Electrique.

XML = Extensible Markup Language (Langage à balises étendu)

BIBLIOGRAPHIE

[BACH1996] BACH J., octobre 1996. *Test Automation Snake Oil*, Revue Windows Technical Journal.

[BACH1999] BACH J., 2003. *Troubleshooting Risk-Based Testing*, Revue Software Testing & Quality Engineering.

[COURTY2006] COURTY B., 2006, *Choix et mise en place d'un outil de test fonctionnel à RTE*, Mémoire ingénieur, CNAM.

[DEFAYE2008] DEFAYE J., 2008, *Organisation de la phase de recette, méthodes et outils dans le contexte du groupe de développement et de maintenance informatique d'EDF*, Mémoire ingénieur, CNAM.

[HP] : forum internet sur les outils HP

[INDUS] Portail de l'industrialisation du test fonctionnel
http://www.portaildutest.fr/

[LEGEARD2009] LEGEARD B., BOUQUET F., PICKAERT N., 2009, *Industrialiser le test fonctionnel : Des exigences métier au référentiel de tests automatisés*, Dunod.

[LORTHIOIR2005] LORTHIOIR-SOOCKALLINGUM T., 2005, *Les tests de non-régression en développement orienté objet (mise en œuvre de l'outil Rational Robot en environnement J2EE)*, Mémoire ingénieur, CNAM.

[MACIAK2001] MACIAK V., 2001, *L'automatisation des procédures de recette d'un progiciel de gestion commerciale (mise en oeuvre de l'outil Cyrano)*, Mémoire ingénieur, CNAM.

[NIST2002] TASSEY G., mai 2002, *The Economic Impacts of Inadequate Infrastructure for Software Testing*, Rapport du NIST (National Institute of Standards & Technology).

[PRADAT2009] PRADAT-PEYRE J.-F., PRINTZ J., 2009, *Pratique des tests logiciels Texte imprimé : concevoir et mettre en œuvre une stratégie de tests, préparation à la certification ISTQB*, Dunod.

[RTE] : Documentation interne RTE

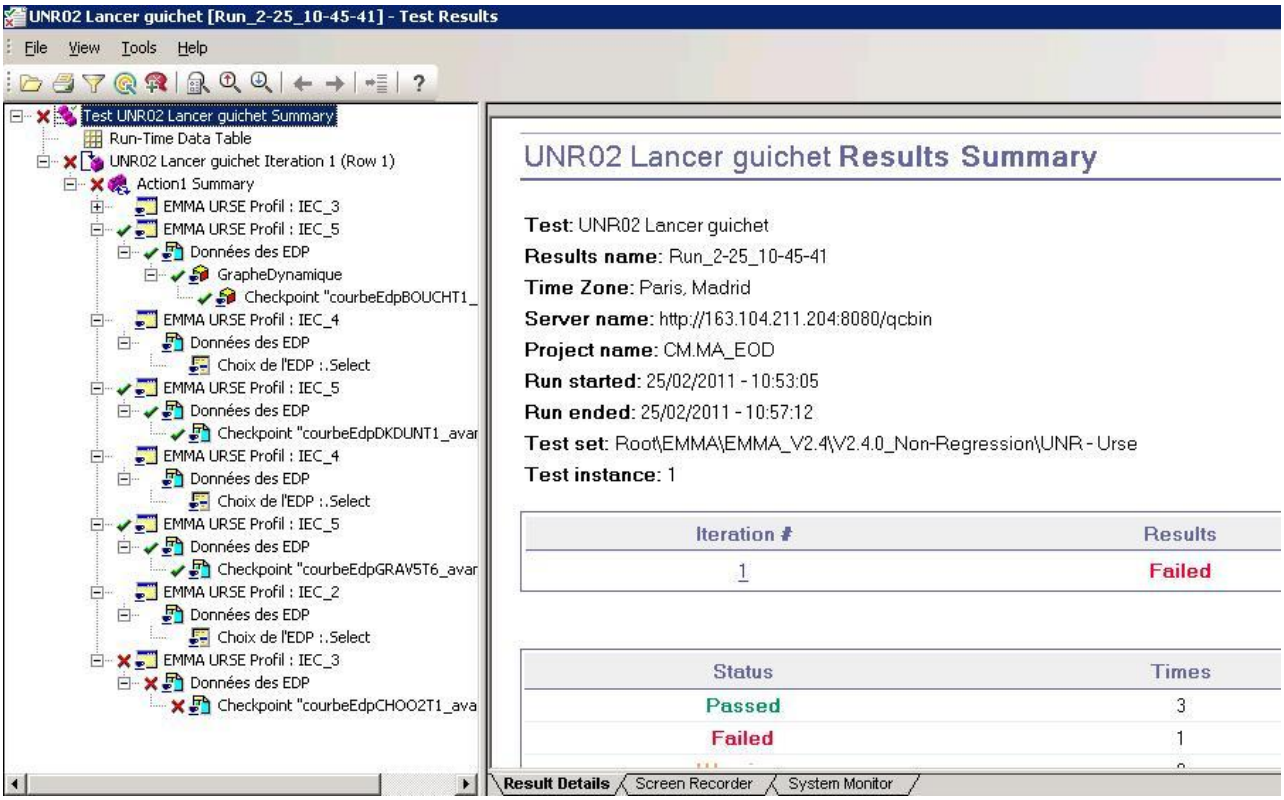
[TESTISSIMO] Testissimo, le site du test logiciel
http://www.testissimo.com/

[VALTECH2006] MANTEL G., BOULLIER P., février 2006, *Automatisation des recettes fonctionnelles : un levier pour la conduite du changement*, Article de la société Valtech

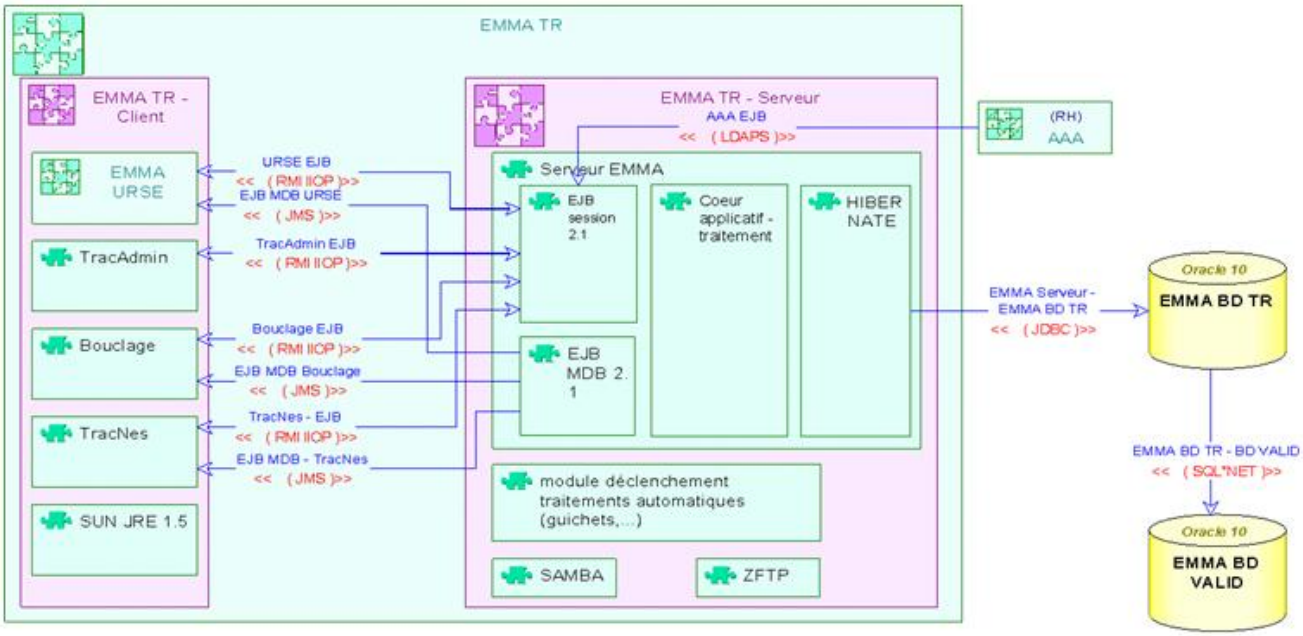
[YPHYSE1998] YPHYSE, 1998, *Mettre en place un banc de test pour maîtriser les tests fonctionnels et techniques*, Savoir-Faire.

ANNEXES

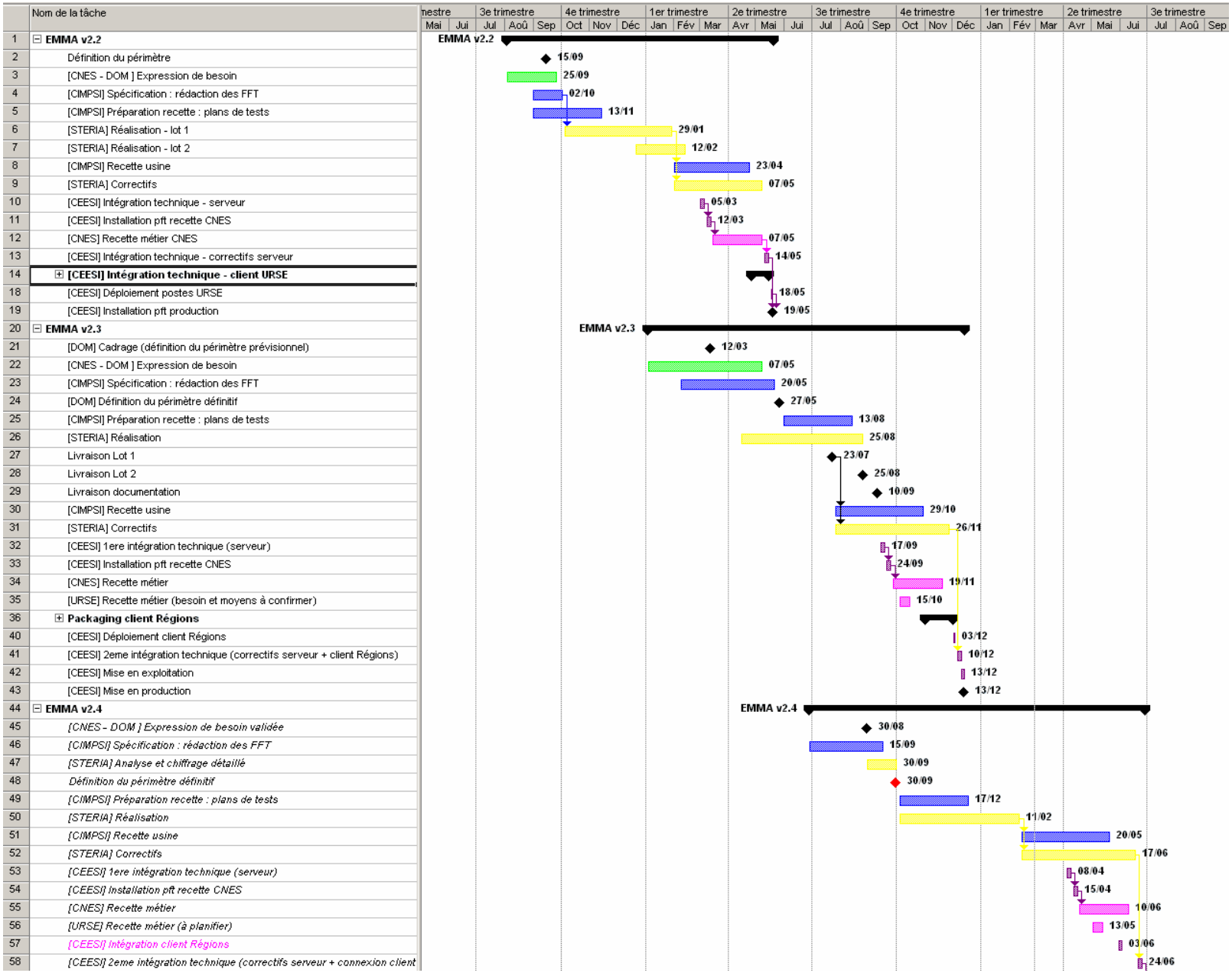
A. Rapport test QTP



B. Architecture applicative d'EMMA TR



C. Planning des versions 2.2, 2.3 et 2.4 d'EMMA



	Nom de la tâche
1	**EMMA v2.2**
2	Définition du périmètre
3	[CNES - DOM] Expression de besoin
4	[CIMPSI] Spécification : rédaction des FFT
5	[CIMPSI] Préparation recette : plans de tests
6	[STERIA] Réalisation - lot 1
7	[STERIA] Réalisation - lot 2
8	[CIMPSI] Recette usine
9	[STERIA] Correctifs
10	[CEESI] Intégration technique - serveur
11	[CEESI] Installation pft recette CNES
12	[CNES] Recette métier CNES
13	[CEESI] Intégration technique - correctifs serveur
14	**[CEESI] Intégration technique - client URSE**
18	[CEESI] Déploiement postes URSE
19	[CEESI] Installation pft production
20	**EMMA v2.3**
21	[DOM] Cadrage (définition du périmètre prévisionnel)
22	[CNES - DOM] Expression de besoin
23	[CIMPSI] Spécification : rédaction des FFT
24	[DOM] Définition du périmètre définitif
25	[CIMPSI] Préparation recette : plans de tests
26	[STERIA] Réalisation
27	Livraison Lot 1
28	Livraison Lot 2
29	Livraison documentation
30	[CIMPSI] Recette usine
31	[STERIA] Correctifs
32	[CEESI] 1ere intégration technique (serveur)
33	[CEESI] Installation pft recette CNES
34	[CNES] Recette métier
35	[URSE] Recette métier (besoin et moyens à confirmer)
36	**Packaging client Régions**
40	[CEESI] Déploiement client Régions
41	[CEESI] 2eme intégration technique (correctifs serveur + client Régions)
42	[CEESI] Mise en exploitation
43	[CEESI] Mise en production
44	**EMMA v2.4**
45	*[CNES - DOM] Expression de besoin validée*
46	*[CIMPSI] Spécification : rédaction des FFT*
47	*[STERIA] Analyse et chiffrage détaillé*
48	*Définition du périmètre définitif*
49	*[CIMPSI] Préparation recette : plans de tests*
50	*[STERIA] Réalisation*
51	*[CIMPSI] Recette usine*
52	*[STERIA] Correctifs*
53	*[CEESI] 1ere intégration technique (serveur)*
54	*[CEESI] Installation pft recette CNES*
55	*[CNES] Recette métier*
56	*[URSE] Recette métier (à planifier)*
57	*[CEESI] Intégration client Régions*
58	*[CEESI] 2eme intégration technique (correctifs serveur + connexion client*

D. Scripts de migration SQL

Script SQL : Migration_EMMA_v2.3.x_v2.4.0_Tables

```
-------------------------------------------------------------------------------
-------------
-- FFT 28377 - Suppression code PCCP
-------------------------------------------------------------------------------
-------------
ALTER TABLE TYPE_REDECLARATION drop column dnfortuit;
ALTER TABLE REDECLARATION_GUICHET drop column MODIFAPRESDN;
commit;
-------------------------------------------------------------------------------
-------------
-- FFT 28088 - Augmenttation du nombre de redéclaration
-------------------------------------------------------------------------------
-------------
ALTER TABLE GUICHET     ADD (APTEAAUGMENT NUMBER(1));
commit;
-------------------------------------------------------------------------------
-------------
-- FFT 28558 - MAJ des programmes et des CT.
-- Debut
-------------------------------------------------------------------------------
-------------

-- Creation de la table MODIFICATION_PROG_CT
create table MODIFICATION_PROG_CT  (
   ID                   NUMBER                          not null,
   IDENTITE             NUMBER                         not null,
   TYPEMODIF            NUMBER(1)                       not null,
   TYPECHRONIQUE        NUMBER                                    ,
   SENS                             NUMBER(1)
           ,
   constraint PK_MODIF_PROG_CT primary key (ID)
         using index
       tablespace TBS_EMMA_TR_IDX
       storage
       (
           initial 10M
           next 10M
       )
)
  storage
(
    initial 10M
    next 10M
)
tablespace TBS_EMMA_TR_DAT;

create index FK_ENTITE_PCE on MODIFICATION_PROG_CT (
   identite ASC
)
storage
(
    initial 10M
    next 10M
)
tablespace TBS_EMMA_TR_IDX;
-------------------------------------------------------------------------------
-------------
-- FFT 28558 - MAJ des programmes et des CT.
-- Fin.
-------------------------------------------------------------------------------
```

- Dump initial, J-1 vide

```
'Step 1 : modification de l'heure
'Il faut modifier l'heure du PC car le basculement est interdit entre 12h et 24h
'On sauvegarde l'heure du PC avant la modif (pour remettre l'heure correcte à la fin du script)
varHeureDebScript = hour(now) & ":" & minute(now) & ":" & second(now)
'On modifie l'heure du PC
rep = Environment("p_RepAppli") & Environment("p_RepScriptChangeHeureSys")
var = "10:00:00"
executeBat rep, var

'Step 2 : ajout du fichier Bilans_j1
FicSource = "R:\FichiersDeTest\Dump\DNR01\DNR01-01\src\step2"
'bilans_J1_20100727_201007261832.csv
FicDestination = "S:\jmoins1\DONNEES_GENEREES\Outil_Bouclage"
CopieFichiersRepertoire FicSource, FicDestination

'Step 4 : Saisir le minimum requis en RP : 608
'Se connecter à Tracadmin
RunAction "Action1 [TANR00-01 Connexion tracadmin]", oneIteration
JavaWindow("Application Administration").JavaMenu("Paramètres").JavaMenu("Saisie de RP").Select
JavaDialog("Saisie de RP").JavaEdit("RP=").Set "608"
JavaDialog("Saisie de RP").JavaButton("Enregistrer").Click
JavaDialog("Confirmation des modifications").JavaButton("Oui").Click
JavaWindow("Application Administration").JavaMenu("Paramètres").JavaMenu("Paramètre Guichet en J-1").Select
JavaWindow("Paramétrage de l'heure").JavaList("HH :").Select "16"
JavaWindow("Paramétrage de l'heure").JavaList("MM :").Select "30"
JavaWindow("Paramétrage de l'heure").JavaButton("Valider").Click
JavaWindow("Paramétrage de l'heure").JavaDialog("Confirmation").JavaButton("OK").Click
JavaWindow("Application Administration").JavaMenu("Service").JavaMenu("Quitter").Select
JavaWindow("Application Administration").JavaDialog("Application Administration").JavaButton("Oui").Click

'Step 3 : Basculement
JavaWindow("Application EMMA Tracnes").JavaMenu("Importation des données").JavaMenu("Basculement / export des").Select
JavaDialog("Application EMMA").JavaButton("Oui").Click
wait(240)
JavaDialog("Message").JavaButton("OK").Click

'Step 4 : Changement de l'heure
'Pour repasser à l'heure correcte, à la fin du script :
'On récupère l'heure de fin
varHeureFinScript =hour(now) & ":" & minute(now) & ":" & second(now)
'On calcule en secondes la différence entre les 2 heures
secondeDiff = DateDiff("s", var, varHeureFinScript)
'On ajoute le temps d'exécution en secondes à la variable varHeureDebScript (contient l'heure du PC avant la modif)
varFin = DateAdd("s", secondeDiff, varHeureDebScript)
'On modifie l'heure du PC
rep = Environment("p_RepAppli") & Environment("p_RepScriptChangeHeureSys")
var = varFin
executeBat rep, var

'Step 5 : Export du dump
rep = Environment("p_RepAppli") & Environment("p_RepScriptExportDump")
var = Environment("p_DumpPrecVersionJmoins1Vide")
executeBat rep, var

'Step 6 : Importer le dump que l'on souhaite migrer dans TR lame1
rep = Environment("p_RepAppli") & "ScriptAUTO\chargeDump_test_trl1.bat "
var = " " & var
executeBat rep, var

'Step 7 : Migrer le dump
rep = Environment("p_RepAppli") & "ScriptAUTO\migrationDump.bat "
var = " "
executeBat rep, var

'Step 8 : Exporter le dump dans TR lame1
rep = Environment("p_RepAppli") & "ScriptAUTO\exportDump_test_trl1.bat "
var = Environment("p_DumpJmoins1Vide")
executeBat rep, var
```

- Dump J-1 après le calcul des prescriptions

```
'Step 1 :  Il faut déposer les fichiers sources necessaires au calcul des Prescriptions dans les répertoires du S (Service_Systeme).
RepSource = "R:\FichiersDeTest\Dump\DNR01\DNR01-02\src\step1\fourniture_SS_15h30"
RepDestination = "S:\Services_Systeme\fourniture_SS_15h30"
CopieFichiersRepertoire RepSource, RepDestination
RepSource = "R:\FichiersDeTest\Dump\DNR01\DNR01-02\src\step1\Production_totale"
RepDestination = "S:\Services_Systeme\Production_totale"
CopieFichiersRepertoire RepSource, RepDestination

'Step 2 : Se placer dans l'écran « Saisie des contraintes minimales de surete»
'dans le menu « Services systeme --> Saisie des contraintes minimales de surete»
'Saisir les minimum requis en RS
'1 pas à 850 , 11 pas à 700,  2 pas à 750,   2 pas à 850, 14 pas à 750,    2 pas à 850,   2 pas à 750,   4 pas à 800,   7 pas à 750, 3 pas à 700
JavaWindow("Application EMMA Tracnes").JavaMenu("Services systeme").JavaMenu("Saisie des contraintes").Select
JavaWindow("Saisie des contraintes").JavaTable("TableAppli").SelectCell "#0","#1"
chaine="850;700;700;700;700;700;700;700;700;700;700;700;750;750;850;850;750;750;750;750;750;750;750;750;750;750;750;750;750;850
;850;750;750;800;800;800;800;750;750;750;750;750;750;750;750;700;700;700"
mesDonnees=Split(chaine,";")
rempliTableau JavaWindow("Saisie des contraintes").JavaTable("TableAppli") , 0 , mesDonnees , 1, 48

'Step 3 : Mettre les coef normatifs aux valeurs suivantes
'CNR 94  : 0 , EDF 94 : 90 , SNET 85  : 74
JavaWindow("Application EMMA Tracnes").JavaMenu("Services systeme").JavaMenu("Prescriptions J-1").Select
JavaWindow("Visualisation des prescription").JavaButton("Coefficients normatifs").Click
JavaWindow("Visualisation des prescription").JavaDialog("Coefficients normatifs").JavaTable("TableAppli").ActivateCell "#0","RP"
JavaWindow("Visualisation des prescription").JavaDialog("Coefficients normatifs").JavaTable("TableAppli").SetCellData "#0","RP","94"
JavaWindow("Visualisation des prescription").JavaDialog("Coefficients normatifs").JavaTable("TableAppli").ActivateCell "#0","RS"
JavaWindow("Visualisation des prescription").JavaDialog("Coefficients normatifs").JavaTable("TableAppli").SetCellData "#0","RS","0"
JavaWindow("Visualisation des prescription").JavaDialog("Coefficients normatifs").JavaTable("TableAppli").ActivateCell "#1","RP"
JavaWindow("Visualisation des prescription").JavaDialog("Coefficients normatifs").JavaTable("TableAppli").SetCellData "#1","RP","94"
JavaWindow("Visualisation des prescription").JavaDialog("Coefficients normatifs").JavaTable("TableAppli").ActivateCell "#1","RS"
JavaWindow("Visualisation des prescription").JavaDialog("Coefficients normatifs").JavaTable("TableAppli").SetCellData "#1","RS","90"
JavaWindow("Visualisation des prescription").JavaDialog("Coefficients normatifs").JavaTable("TableAppli").ActivateCell "#2","RP"
JavaWindow("Visualisation des prescription").JavaDialog("Coefficients normatifs").JavaTable("TableAppli").SetCellData "#2","RP","85"
JavaWindow("Visualisation des prescription").JavaDialog("Coefficients normatifs").JavaTable("TableAppli").ActivateCell "#2","RS"
JavaWindow("Visualisation des prescription").JavaDialog("Coefficients normatifs").JavaTable("TableAppli").SetCellData "#2","RS","74"
JavaWindow("Visualisation des prescription").JavaDialog("Coefficients normatifs").JavaButton("OK").Click

'Step 4 : Puis faire le cycle de calcul/envoi des prescriptions ( Cliquer sur tous les boutons à la suite de la gauche vers la droite).
JavaWindow("Visualisation des prescription").JavaButton("Injection Pré-programmes").Click
JavaWindow("Visualisation des prescription").JavaDialog("Application EMMA").JavaStaticText("Le fichier /mnt/tracabiliteVAL")
.Check CheckPoint("Fichier Service Systeme GDF Erreur")
JavaWindow("Visualisation des prescription").JavaDialog("Application EMMA").JavaButton("OK").Click
JavaWindow("Visualisation des prescription").JavaButton("Calcul prescriptions").Click
JavaWindow("Visualisation des prescription").JavaButton("Envoi prescription").Click
JavaWindow("Visualisation des prescription").JavaDialog("Application EMMA").JavaStaticText("La génération du fichier")
.Check CheckPoint("Fichier CNR Ok")
JavaWindow("Visualisation des prescription").JavaDialog("Application EMMA").JavaStaticText("La génération du fichier_2")
.Check CheckPoint("Fichier EDF Ok")
JavaWindow("Visualisation des prescription").JavaDialog("Application EMMA").JavaStaticText("La génération du fichier_3")
.Check CheckPoint("Fichier GDF Ok")
JavaWindow("Visualisation des prescription").JavaDialog("Application EMMA").JavaStaticText("La génération du fichier_4")
.Check CheckPoint("Fichier SNET Ok")
JavaWindow("Visualisation des prescription").JavaDialog("Application EMMA").JavaButton("OK").Click
JavaWindow("Visualisation des prescription").JavaButton("Injection retour producteurs").Click
JavaWindow("Visualisation des prescription").JavaDialog("Application EMMA").JavaStaticText("Le fichier /mnt/tracabiliteVAL_2")
.Check CheckPoint("Fichier servsys_GDF Erreur")
JavaWindow("Visualisation des prescription").JavaDialog("Application EMMA").JavaButton("OK").Click
JavaWindow("Visualisation des prescription").JavaButton("Envoi prescription finale").Click
JavaWindow("Visualisation des prescription").JavaDialog("Application EMMA").JavaStaticText("La génération des fichiers")
.Check CheckPoint("Generation Fichiers Finie")
JavaWindow("Visualisation des prescription").JavaDialog("Application EMMA").JavaButton("OK").Click
JavaWindow("Visualisation des prescription").JavaButton("Fermer").Click
JavaWindow("Saisie des contraintes").JavaButton("Fermer").Click

```

```
'Step 5 : Export du dump
rep = Environment("p_RepAppli") & Environment("p_RepScriptExportDump")
var = Environment("p_DumpPrecVersionJmoins1CalculPresc")
executeBat rep, var

'Step 6 : Importer le dump que l'on souhaite migrer dans TR lame1
rep = Environment("p_RepAppli") & "ScriptAUTO\chargeDump_test_trl1.bat "
var = " " & var
executeBat rep, var

'Step 7 : Migrer le dump
rep = Environment("p_RepAppli") & "ScriptAUTO\migrationDump.bat "
var = " "
executeBat rep, var

'Step 8 : Exporter le dump dans TR lame1
rep = Environment("p_RepAppli") & "ScriptAUTO\exportDump_test_trl1.bat "
var = Environment("p_DumpJmoins1CalculPresc")
executeBat rep, var
```

- Dump J-1 avant basculement

```
'Step 1 : Faire l'Init bouclage
JavaWindow("Application EMMA Bouclage").JavaMenu("J-1").JavaMenu("Bouclage (F5)").Select
JavaWindow("J-1").JavaTab("JTabbedPane").Select "Bouclage"
JavaWindow("J-1").JavaButton("Init").Click
JavaWindow("J-1").JavaDialog("Choix des données d'entrée").JavaButton("Valider").Click
wait(50)
JavaDialog("Initialisation des données").JavaButton("OK").Click

JavaWindow("J-1").JavaTable("TableAppli").Check CheckPoint("TableAppli")
JavaWindow("J-1").JavaTable("TableAppli_2").Check CheckPoint("TableAppli_2")
JavaWindow("J-1").JavaTable("TableAppli_3").Check CheckPoint("TableAppli_3")
JavaWindow("J-1").JavaTable("TableAppli_4").Check CheckPoint("TableAppli_4")
JavaWindow("J-1").JavaTable("TableAppli_5").Check CheckPoint("TableAppli_5")
JavaWindow("J-1").JavaTab("JTabbedPane").Select "Données"
JavaWindow("J-1").JavaTable("TableAppli").Check CheckPoint("TableAppli_6")
JavaWindow("J-1").JavaTable("TableAppli_2").Check CheckPoint("TableAppli_7")
JavaWindow("J-1").JavaTab("JTabbedPane").Select "Bouclage"

'Step 2 : Faire le Feu Vert
JavaWindow("J-1").JavaTab("JTabbedPane").Select "Simulation"
JavaWindow("J-1").JavaButton("Feu vert").Click
JavaDialog("Confirmation").JavaButton("Oui").Click
JavaWindow("J-1").JavaDialog("Choix du mode pour l'action").JavaButton("Valider").Click
'Affichage des points de fonctionnement
JavaDialog("Création des points de").JavaDialog("création terminée").JavaButton("OK").Click
JavaDialog("Export").JavaButton("OK").Click

'Step 3 : Fermer bouclage
RunAction "Action1 [BNR00-02-01 Fermeture bouclage]", oneIteration

'Step 10 : Exporter le dump dans TR lame1
rep = Environment("p_RepAppli") & "ScriptAUTO\exportDump_test_trl1.bat "
var = Environment("p_DumpJmoins1AvantBascule")
executeBat rep, var

'Step 3 bis : ouvrir le client bouclage avec le profil OMA, faire l'init J-1 du bouclage TR et fermer le client bouclage
RunAction "Action1 [BNR00-01-02 Connexion profil OMA]", oneIteration
```

```
39:  JavaWindow("Application EMMA Bouclage").JavaMenu("Infra-Journalier").JavaMenu("Aléas (F2)").Select
40:  JavaWindow("Infra-Journalier").JavaButton("Initialisation des données").Click
41:  JavaWindow("Infra-Journalier").JavaDialog("Importation").JavaButton("Oui").Click
42:  JavaDialog("Init J-1").JavaButton("OK").Click
43:  JavaWindow("Infra-Journalier").JavaButton("Fermer").Click
44:  RunAction "Action1 [BNR00-02-01 Fermeture bouclage]", oneIteration
45:
46:  'Step 4 : Se connecter en REPART au client Tracnes
47:  RunAction "Action1 [TNR00-01-03 Connexion profil REPART]", oneIteration
48:  'Step 5 : Faire les taches de fin de journée.
49:  JavaWindow("Application EMMA Tracnes").JavaMenu("Importation des données").JavaMenu("Tâches de fin de journée").Select
50:  wait(30)
51:  JavaDialog("message").JavaButton("OK").Click
52:  wait(30)
53:  OptionalStep.JavaWindow("Application EMMA Tracnes").JavaDialog("Message").JavaButton("OK").Click
54:
55:  'Step 6 : Fermer le client Tracnes
56:  RunAction "Action1 [TNR00-02-01 Fermeture Tracnes]", oneIteration
57:
58:  'Step 7 : Export du dump
59:  rep = Environment("p_RepAppli") & Environment("p_RepScriptExportDump")
60:  var = Environment("p_DumpPrecVersionJmoins1AvantBascule")
61:  executeBat rep, var
62:
63:  'Step 8 : Importer le dump que l'on souhaite migrer dans TR lame1
64:  rep = Environment("p_RepAppli") & "ScriptAUTO\chargeDump_test_trl1.bat "
65:  var = " " & var
66:  executeBat rep, var
67:
68:  'Step 9 : Migrer le dump
69:  rep = Environment("p_RepAppli") & "ScriptAUTO\migrationDump.bat "
70:  var = " "
71:  executeBat rep, var
72:
73:  'Step 10 : Exporter le dump dans TR lame1
74:  rep = Environment("p_RepAppli") & "ScriptAUTO\exportDump_test_trl1.bat "
75:  var = Environment("p_DumpJmoins1AvantBascule")
76:  executeBat rep, var
```