La recette fonctionnelle d'un système d'information : enjeux, méthodes, outils et exemple d'application

Yanis Okba

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

MEMOIRE

présenté en vue d'obtenir

le DIPLOME D'INGENIEUR CNAM

***spécialité* : INFORMATIQUE**

***option* : Systèmes d'information**

par

Yanis OKBA

La recette fonctionnelle d'un Système d'Information : enjeux, méthodes, outils et exemple d'application.

Soutenu le 25 septembre 2012

JURY

Président

Nicolas Trèves – Professeur titulaire de la chaire Génie logiciel – Conservatoire national des arts et métiers.

Membres

Olivier Flauzac – Professeur des Universités – Université de Reims Champagne-Ardenne,

Florent Nolot – Maître de conférences – Université de Reims Champagne-Ardenne,

Arnaud Renard – Ingénieur de recherche – CreSTIC – Université de Reims Champagne-Ardenne – Reims (51),

Daniel Kownacki – Responsable du pôle études – Directeur adjoint à la Direction des Systèmes d'Information – Région Champagne-Ardenne – Châlons-en-Champagne (51).

Remerciements

Mon aventure avec le CNAM a débuté en 2007 lorsque j'ai souhaité transformer mes acquis universitaires et mon expérience professionnelle en un diplôme d'ingénieur en informatique.

J'ai commencé par effectuer une demande de VAE qui a débouché sur une validation partielle du diplôme d'ingénieur. Je remercie d'ailleurs Rosanna DUBREUILLE du CNAM Champagne-Ardenne qui m'a accompagné dans cette démarche longue et fastidieuse.

Puis, j'ai suivi plusieurs unités d'enseignements avec le CNAM de Paris et de Reims entre 2008 et 2010 grâce à l'outil *Plei@d* qui offre des possibilités d'enseignement à distance modernes et efficaces. A cette occasion, je tiens à remercier mes professeurs de l'époque, Isabelle COMYIN-WATTIAU, Jacky AKOKA et Rosanna DUBREUILLE pour la qualité de leur enseignement, ainsi que Catherine MONNIN et Isabelle BITZ qui ont facilité mes démarches administratives auprès du CNAM Champagne-Ardenne.

Depuis maintenant un an et demi, j'effectue le stage de mon mémoire dans le cadre de mon emploi actuel d'ingénieur territorial au sein du pôle Études de la direction des systèmes d'information de la Région Champagne-Ardenne.

Dans ce contexte, je remercie mon employeur et tout particulièrement Alain GOBERT, directeur des systèmes d'information de la Région Champagne-Ardenne, et Daniel KOWNACKI, son adjoint et chef du pôle Études. Grâce à eux, j'ai pu concilier mes activités professionnelles et mon projet de mémoire dans d'excellentes conditions.

De même, je tiens à remercier Olivier FLAUZAC, mon directeur de mémoire, pour les précieux conseils qu'il m'a donné tout au long de mon mémoire. Pareillement, je voudrais témoigner ma gratitude aux membres du jury et à son Président pour le temps qu'ils m'ont consacré.

Enfin, je terminerai par une pensée affectueuse pour mon épouse, sans qui cette aventure n'aurait pas été possible.

Liste des abréviations

AMOA	Assistance à Maîtrise d'Ouvrage
CAO	Commission d'Appels d'Offres
CCTP	Cahier des Clauses techniques Particulières
CdP	Chef de Projet
CFTL	Comité Français des Tests Logiciels
CMP	Code des Marchés Publics
CQFD	Coût Qualité Fonctionnalités Délais
CRCA	Conseil Régional de Champagne-Ardenne
DSI	Direction des Systèmes d'Information
ISTQB	International Software Testing Qualifications Board
MOA	Maîtrise d'Ouvrage
MOE	Maîtrise d'Oeuvre
MOM	Mise en Ordre de Marche
OSIDI	Ouverture du Système d'Information aux Demandeurs Internautes
RCA	Région Champagne-Ardenne
RFSI	Recette Fonctionnelle d'un Système d'Information
SI	Système d'Information
VABF	Vérification d'Aptitude au Bon Fonctionnement
VSR	Vérification de Service Régulier

Glossaire

Anomalie	Une anomalie correspond à toute condition qui dévie des attentes basées sur le cahier des charges, le dossier de spécifications des exigences, les documents de conception, les manuels utilisateurs, les standards, etc., ou des perceptions ou expériences de quelqu'un. Elle est caractérisée par sa nature (non-conformité, évolution d'une exigence...) et sa sévérité (bloquante, majeure, mineure).
Automate de test	Logiciel permettant de réaliser automatiquement un cas de test.
Besoins	La norme AFNOR NF X50-150 définit un besoin comme *« une nécessité ou un désir éprouvé par l'utilisateur et non le volume du marché. Un besoin peut être exprimé ou implicite, avoué ou inavoué, latent ou potentiel. Dans tous les cas, il constitue le besoin à satisfaire pour lequel l'utilisateur est prêt à faire un effort».*
Cahier des charges	Un cahier des charges est un document visant à définir exhaustivement les spécifications de base d'un produit ou d'un service à réaliser. Outre les spécifications de base, il décrit ses modalités d'exécution. Il définit aussi les objectifs à atteindre et vise à bien cadrer une mission. En interne, il sert à formaliser les besoins et à les expliquer aux différents acteurs pour s'assurer de l'accord de tout le monde. Il sert ensuite à sélectionner un prestataire et à organiser la relation tout au long du projet. Il est considéré comme un référentiel contractuel partagé par le prestataire et l'équipe interne.
Cahier de recette	Le cahier de recette consigne les résultats des différents cas de test exécutés par la MOA. On parlera de cahier de recette d'usine pour la MOE.
Campagne de test	Une campagne de tests est un ensemble de cas de test à exécuter pour le test d'une version ou d'une partie d'un logiciel.
Cas de test	Ensemble de valeurs d'entrée, de préconditions d'exécution, de résultats attendus et de postconditions d'exécution, développées pour un objectif ou une condition de tests particulier, tel qu'exécuter un chemin particulier d'un programme ou vérifier le respect d'une exigence spécifique.
Cas de test non-passant	Un cas de test est dit non-passant lorsqu'il implique un comportement du logiciel testé générant un message d'erreur attendu.
Cas de test passant	Un cas de test est dit passant lorsqu'il correspond à un usage normal du logiciel testé.
Conformité	La norme ISO 9126 définit la conformité comme la *« capacité d'un produit logiciel à adhérer à des standards, conventions ou consignes dans des lois ou prescriptions similaires. ».*
Défaut	Un défaut est une imperfection qui peut conduire le logiciel à ne pas exécuter les fonctions requises, par exemple une instruction ou une définition de données

	incorrecte. Un défaut, s'il est rencontré lors de l'exécution peut causer la défaillance du logiciel.
Dossier de spécifications	Ensemble de documents qui spécifient, idéalement de façon complète, précise et vérifiable, les exigences, conceptions, comportements et autres caractéristiques d'un composant ou système, et souvent, les procédures pour déterminer si ces stipulations ont été satisfaites.
Exigence	La norme IEEE 610 définit une exigence comme *« une condition ou capacité requise par un utilisateur pour résoudre un problème ou atteindre un objectif qui doit être tenu ou possédé par un système ou composant pour satisfaire à un contrat, standard, spécification ou autre document imposé formellement »*. Cette même norme définit une exigence fonctionnelle comme *« une exigence qui spécifie une fonction qu'un composant ou système doit remplir »*. Par opposition, l'ISTQB et le CFTL définissent une exigence non-fonctionnelle comme *« une exigence qui ne se rapporte pas aux fonctionnalités, mais à des attributs tels fiabilité, rendement, utilisabilité, maintenabilité et portabilité. »*
Jeu de données de test	Ce sont les données nécessaires pour l'exécution d'un cas de test. Ces données affectent ou sont affectées par le logiciel testé. Elles sont stockées dans la base de données de test (ou de recette).
Logiciel	Un logiciel ou une application est un ensemble de programmes, qui permet à un ordinateur ou à un système informatique d'assurer une tâche ou une fonction en particulier (p. ex. logiciel de gestion de la relation client, logiciel de production, logiciel de comptabilité, logiciel de gestion des prêts...).
Maîtrise d'œuvre	Le maîtrise d'œuvre (MOE) est l'entité retenue par le maître d'ouvrage pour réaliser l'ouvrage, dans les conditions de délais, de qualité et de coût fixées par ce dernier conformément à un contrat. La MOE est donc responsable des choix techniques inhérents à la réalisation de l'ouvrage conformément aux exigences de la MOA. Le maître d'œuvre a ainsi la responsabilité dans le cadre de sa mission de désigner une personne physique chargée du bon déroulement du projet, il s'agit du chef de projet MOE.
Maîtrise d'ouvrage	On appelle maîtrise d'ouvrage (MOA) l'entité porteuse du besoin, définissant l'objectif du projet, son calendrier et le budget consacré à ce projet. Le résultat attendu du projet est la réalisation d'un produit, appelé ouvrage. La maîtrise d'ouvrage maîtrise l'idée de base du projet, et représente à ce titre les utilisateurs finaux à qui l'ouvrage est destiné. Ainsi, le maître d'ouvrage est responsable de l'expression des besoins mais n'a pas forcément les compétences techniques liées à la réalisation de l'ouvrage.
Modèle de développement en V	La norme IEEE 610 définit le modèle en V comme *« une structure décrivant les activités du cycle de développement logiciel, depuis la spécification des exigences jusqu'à la maintenance. Le modèle en V illustre comment les activités de tests peuvent être intégrées dans chaque phase du cycle de développement. ».*
Plan de test	La norme IEEE 829 définit le plan de test comme *« un document décrivant l'étendue,*

	l'approche, les ressources et le planning des activités de test prévues. Il identifie entre autres les éléments et caractéristiques à tester, qui fera chaque tâche, le degré d'indépendance des testeurs, l'environnement de test, les techniques de conception des tests et les techniques de mesure des tests à utiliser, et tout risque nécessitant des plans de contingence. C'est un document reprenant les processus de planification des tests ».
Qualification	La norme ISO 9000 définit le terme qualification comme *« le processus consistant à démontrer la capacité à satisfaire les exigences spécifiées. Note : le terme 'qualifié' est utilisé pour désigner le statut correspondant. ».*
Qualité	La norme NF Z67-130 définit la qualité de la manière suivante : *« un produit ou service de qualité est un produit dont les caractéristiques lui permettent de satisfaire les besoins exprimés ou implicites des consommateurs ».*
Qualité logicielle	La norme ISO 9126 définit la qualité logicielle comme *« la totalité des fonctionnalités et caractéristiques d'un produit logiciel qui influent sur sa capacité à satisfaire des besoins déclarés ou implicites ».*
Spécification d'exigences de logiciel	La norme IEEE 830 définit une Spécification d'Exigences de Logiciel (SEL) comme servant *« à spécifier les exigences d'un logiciel, d'un programme ou d'un progiciel en particulier, qui exécute certaines fonctions dans un environnent précis. Elle peut être rédigée par des représentants du fournisseur, du client ou les deux à la fois. ».*

Une SEL comprend :

- les fonctionnalités : services et fonctions que le système doit fournir,
- les interfaces externes : identification des interactions avec les utilisateurs et les autres systèmes avec lesquels le nouveau système doit s'intégrer,
- la performance : contraintes d'opération du système en disponibilité et en temps réponse,
- les attributs du système : caractéristiques intrinsèques du système (portabilité, maintenabilité, sécurité...),
- les contraintes de conception : contraintes sur la façon de développer le système.

Recette	La recette d'un SI peut être définie comme un ensemble d'actions permettant au client d'effectuer la vérification de la conformité du SI livré aux besoins qu'il a exprimé dans sa commande, avec l'objectif de valider celui-ci. La recette fonctionnelle d'un SI désigne les actions permettant de vérifier la conformité du SI aux besoins fonctionnels par un ensemble de tests dont les résultats sont appréciés en fonction de critères d'acceptation aboutissant à la validation ou non des aspects fonctionnels du SI.
Scénario de test	Un scénario de test détaille la séquence d'actions et de vérifications des résultats attendus d'un ou plusieurs cas de test (p. ex. une manipulation de l'interface graphique à effectuer et le message correspondant à constater).
Système d'Information	Un Système d'Information (SI) est un ensemble de composants inter-reliés qui recueillent l'information nécessaire au fonctionnement de l'organisation, la traitent, la stockent et la diffusent afin de soutenir la prise de décision et le contrôle au sein de

	l'organisation. Concrètement, un SI se matérialise par un ensemble de systèmes informatiques (matériels et logiciels) supportant tout ou partie des processus métiers d'une organisation et les flux d'information associés.
Téléservice	Services professionnels sécurisés, fournis à distance, via internet ou un extranet, par une administration publique, au bénéfice des professionnels et des usagers.
Test	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.
Test d'acceptation	La norme IEEE 610 définit le test d'acceptation comme le *« test formel en rapport avec les besoins, exigences et processus métier, conduit pour déterminer si un système satisfait ou non aux critères d'acceptation et permettre aux utilisateurs, clients ou autres entités autorisées de déterminer l'acceptation ou non du système »*.
Test fonctionnel	L'ISTQB et le CFTL définissent le test fonctionnel comme *« un test basé sur une analyse des spécifications d'une fonctionnalité d'un composant ou d'un système. »*.
Usager	Personne physique ou morale qui utilise un service public. Le terme usager est générique et invite à mettre en avant l'usage, l'utilisation de la prestation publique.
VABF	C'est la recette provisoire du SI nommée Vérification d'Aptitude au Bon Fonctionnement (VABF).
VSR	C'est la recette définitive du SI nommée Vérification de Service Régulier (VSR).
Validation	La norme ISO 9000 définit la notion de validation comme la *« confirmation par l'examen et la fourniture de preuves objectives que les exigences, pour un usage ou une application voulue, ont été remplies. »*.
Vérification	La norme ISO 9000 définit la notion de vérification comme la *« confirmation par l'examen et la fourniture de preuves objectives que des exigences spécifiées ont été remplies. »*.

Sommaire

Introduction

La recette fonctionnelle d'un Système d'Information (SI) peut être définie comme un ensemble d'actions permettant d'effectuer la vérification de la conformité du SI livré par un fournisseur aux besoins exprimés dans une commande client, avec l'objectif de valider ce SI.

Cette étape clé dans le contrat liant le client au fournisseur du SI est en général difficile et plus ou moins floue en fonction de la maturité des organisations en présence. Le manque de maîtrise de l'étape de recette fonctionnelle se traduit le plus souvent par des surcoûts financiers, des dysfonctionnements logiciels, des retards importants et des réductions du périmètre fonctionnel des logiciels mis en œuvre.

Ainsi, la mise en production d'un SI dont la recette fonctionnelle n'a pas été réalisée selon les règles de l'art implique des conséquences plus ou moins graves pour le client et son fournisseur.

Pour une collectivité territoriale telle qu'une Région par exemple, les dysfonctionnements du SI se concrétisent par une perte d'efficacité dans sa gestion opérationnelle, par un déficit d'image auprès de ses usagers et partenaires ou par un défaut dans le service rendu à ses usagers.

Le fournisseur, quant à lui, s'expose à une inflation de ses coûts de maintenance et à des risques en termes de sécurité, de retours produits, de relations contractuelles difficiles.

La recette fonctionnelle est donc une étape déterminante dans le cycle de vie d'un SI et peut être considérée comme le fondement de la relation contractuelle entre le client et son fournisseur ; d'autant plus que la validation du SI est en général un jalon incontournable, déclenchant par exemple l'ouverture du SI aux utilisateurs et les derniers paiements prévus au contrat dans le cas de l'externalisation de la réalisation du SI.

Dans cette perspective, la problématique majeure pour les organisations, client et fournisseur, est de savoir comment obtenir un SI répondant aux besoins exprimés dans le respect des coûts, de la qualité et des délais fixés initialement par le contrat.

Un des moyens de répondre à ces enjeux consiste à industrialiser le processus de test du logiciel et en particulier l'étape de recette fonctionnelle. Par industrialisation, nous désignons un ensemble constitué par les référentiels de bonnes pratiques, l'organisation, les méthodes, les techniques et les outils ainsi que la formation des personnes conduisant à une meilleure maîtrise de la recette fonctionnelle et des risques associés.

Comme la plupart des organisations, mon employeur, la Région Champagne-Ardenne (RCA) est confrontée à cette problématique. La RCA est une collectivité territoriale instituée par les lois de décentralisation de 1982 et 1983, et son existence s'est concrétisée par la révision constitutionnelle de 2003. Les conseillers régionaux sont élus au suffrage universel depuis 1986. Ils élisent à leur tour le Président du Conseil régional qui gère le budget, dirige le personnel et conduit la politique de la Région.

Les compétences de la RCA concernent l'aménagement du territoire, le développement économique, la formation professionnelle et l'apprentissage, les lycées, les transports ferroviaires, l'enseignement supérieur et la recherche...

C'est dans ce contexte que le pôle Études de la Direction des Systèmes d'Information (DSI) prend en charge les projets d'informatisation des processus métier de la RCA avec comme but l'alignement des SI mis en œuvre sur les besoins des utilisateurs.

Ces projets sont systématiquement gérés par une maîtrise d'ouvrage (MOA) composée d'une ou plusieurs directions métier et d'une assistance à maîtrise d'ouvrage (AMOA) assurée par la DSI. La maîtrise d'œuvre (MOE) est généralement externalisée sauf pour certains projets, où celle-ci est assurée par la DSI (p. ex. les sites extranets collaboratifs, les infocentres décisionnels).

Un projet possède un cycle de vie qui couvre un certain nombre d'étapes allant des analyses préliminaires (étude préalable et détaillée) jusqu'au bilan du projet en passant par les phases d'acquisition (appels d'offres) et de réalisation des SI.

C'est ainsi que les équipes projets, MOA, AMOA et MOE confondues, sont amenées à mettre en place des SI répondant aux besoins des utilisateurs de la RCA tout en respectant les contraintes de coûts, délais et qualité assignés au projet.

Dans le but de maîtriser ces contraintes, le pôle Études, dont je fais parti en tant que chargé de projet, a entrepris de rationaliser ses activités de gestion de projet depuis maintenant plusieurs années. Aujourd'hui, un des derniers chantiers en cours, vise à formaliser la recette des SI et organiser les tests correspondants.

C'est pourquoi, nous envisageons aujourd'hui, dans le cadre du présent mémoire, l'industrialisation de la recette d'un SI comme un moyen efficace contribuant à l'atteinte des objectifs des projets informatiques de la RCA.

Dans un premier temps, nous commencerons par expliciter le contexte et les enjeux de l'administration régionale ainsi que l'organisation du système d'information de la Région. Cette première partie permettra également d'introduire mon projet de mémoire et sa concrétisation dans les projets de la DSI. Nous expliquerons également en quoi la recette fonctionnelle est une étape cruciale dans la relation entre un client et son fournisseur.

Dans un deuxième temps, nous détaillerons les différents aspects du management d'un projet dans le contexte du pôle Études de la DSI. Nous verrons comment sont traités l'ingénierie des besoins, l'estimation des charges et la planification des délais, le pilotage d'un projet, la gestion financière, la composante humaine et la conduite du changement.

Dans un troisième temps, nous donnerons un aperçu de l'état de l'art du test logiciel appliqué à la recette fonctionnelle et décrierons les bonnes pratiques, les méthodes, les techniques et les outils disponibles sur le marché.

Dans un quatrième temps, nous proposerons une solution adaptée aux besoins de la Région. Nous présenterons les différentes phases du projet menant à cette solution : les étapes d'étude, de conception, de réalisation et mise en œuvre de la solution.

Enfin, dans un cinquième temps, il s'agira de mettre en pratique la solution mise en place précédemment afin de pouvoir mesurer concrètement sa pertinence et ses effets sur les objectifs d'un projet informatique. Le but étant de valider l'utilité et l'usage de la solution proposée.

1 Une administration à l'échelon régional

Une Région est une collectivité territoriale instituée par les lois de décentralisation de 1982 et 1983, et dont l'existence s'est concrétisée par la révision constitutionnelle de 2003. La décentralisation s'est essentiellement traduit dans les faits par des transferts réguliers de compétences, accompagnés de moyens financiers, de l'administration centrale étatique vers les Régions, les Départements et les Communes. Pour faire face à ces nouvelles missions, la Région Champagne-Ardenne a due moderniser son administration en s'appuyant, en autre, sur les technologies de l'information et de la communication.

1.1 La Région Champagne-Ardenne

1.1.1 Le Conseil régional et les élus

Le Conseil Régional de Champagne-Ardenne (CRCA) est composé de 49 conseillers régionaux élus pour 4 ans en 2010, formant ainsi l'assemblée régionale. Elle se réunit en séance plénière en moyenne 5 à 6 fois par an pour se prononcer sur le budget et les grandes orientations de la politique régionale. Le Président du CRCA, lui-même élu par les conseillers régionaux, incarne l'exécutif de la RCA. Il prépare et exécute les délibérations du CRCA, il ordonne les dépenses et prescrit les recettes, il dirige les services administratifs et représente la RCA dans ses relations avec l'État et les autres collectivités.

Le CRCA délègue certaines de ses compétences à la commission permanente. Élue par les conseillers régionaux, elle comprend, sous la présidence du Président de Région, les 12 Vice-Présidents et 16 autres membres. Elle se réunit une fois par mois et assure la permanence du CRCA entre les séances plénières. Elle décide de la répartition des financements votés par l'assemblée plénière et suit les affaires courantes.

Les conseillers régionaux sont répartis au sein de 7 commissions thématiques, correspondant aux compétences du CRCA. Elles se réunissent avant les séances plénières. Elles examinent les dossiers qui leur sont soumis et préparent les délibérations qui seront votées en commission permanente.

Les 7 commissions thématiques sont les suivantes :

-Finances – Stratégie – Relations Internationales Communication – Égalité Femme / Homme

-Lycées – Apprentissage – Culture – Patrimoine

-Formation professionnelle – Insertion

-Emploi – Développement économique – Économie sociale et solidaire – Enseignement supérieur – Recherche – Innovation

-Infrastructures – Transport et mobilité durable;

-Territoires – Aménagement de l'espace – Agriculture – Forêt – Développement durable – Tourisme;

-Solidarités – Vie associative – Citoyenneté – Sport – Santé – Handicap.

1.1.2 La structure administrative sous-jacente

Du fait du transfert aux régions de la gestion des personnels de l'Éducation Nationale et des lycées agricoles, chargés des missions d'accueil, d'hébergement, de restauration et d'entretien général et technique des lycées, le nombre d'agents de la RCA est passé de 251 au 31 décembre 2005 à 1815 au 1er janvier 2010 (345 agents situés au siège et 1470 agents répartis dans les lycées).

Dans ce contexte, l'administration de la RCA assure la mise en œuvre des décisions du CRCA. Elle est organisée en directions :

- 10 directions opérationnelles se consacrent au développement du territoire et à la formation, correspondant aux différents domaines d'intervention relevant de la compétence de la RCA. Elles constituent les interlocuteurs directs des acteurs et partenaires régionaux.

- 5 directions fonctionnelles ont la charge des moyens nécessaires au bon fonctionnement de la collectivité. Elles assurent également la gestion des opérations transversales.

- 3 directions transversales : la direction de la communication, la direction des affaires européennes et internationales et le contrôle de gestion.

L'organigramme de la RCA est donné ci-après.

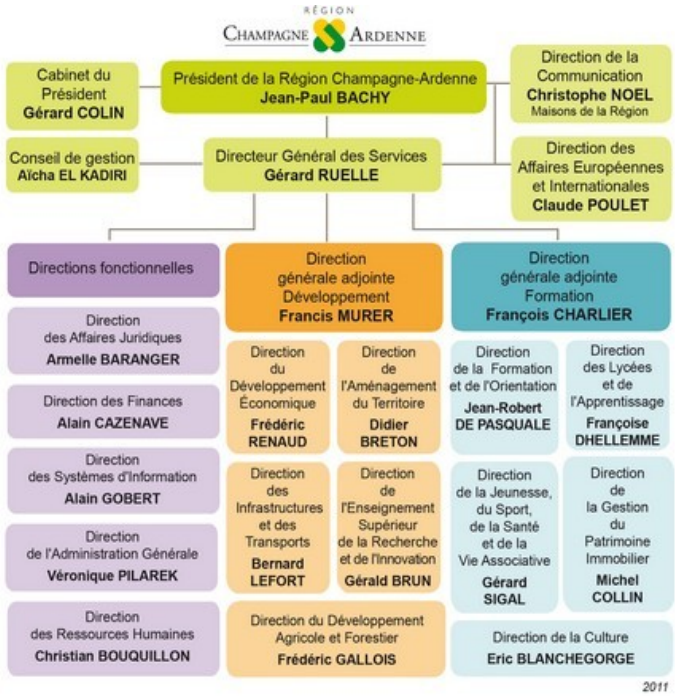


Illustration 1: Organigramme de la RCA

C'est dans ce cadre organisationnel que j'effectue mon stage de mémoire d'ingénieur au sein du pôle Études de la Direction des Systèmes d'Information.

1.1.3 La direction des systèmes d'information

La direction des systèmes d'information (DSI) est chargée de proposer et de piloter la stratégie des services de la RCA en matière de systèmes d'information. Elle accompagne les directions dans le choix et la mise en œuvre de logiciels. Elle est responsable du choix des matériels et des infrastructures. Elle veille au maintien d'un haut niveau de sécurité des systèmes informatiques et anticipe les évolutions dans les domaines technologiques, juridiques ou métiers. La DSI se décompose en 5 pôles de compétences couvrant l'ensemble des missions confiées au directeur des systèmes d'information qui peut s'appuyer sur 12 agents (2 secrétaires, 4 techniciens et 6 ingénieurs).

Systèmes et réseaux

Le bon fonctionnement des équipements techniques et des applications exige une exploitation et une maintenance quotidiennes. Les éléments clés du dispositif de sécurité (pare-feu, antivirus...) sont constamment actualisés. La qualité des liaisons inter sites est surveillée régulièrement. Des axes d'amélioration tant quantitatifs que qualitatifs sont étudiés et proposés. Le renouvellement de ces moyens est programmé.

Études techniques et fonctionnelles (= pôle Études)

Les applications et les équipements sont constamment mis à jour. Le développement et la mise en production exigent des études préalables à leur mise en place. Les outils spécifiques aux services de la RCA sont développés et maintenus. Les projets dont la maîtrise d'œuvre est sous-traitée sont pilotés et supervisés par les ingénieurs de la DSI.

Administration et gestion du parc informatique

Les ressources mises à disposition des agents des services de la RCA exigent une gestion quotidienne. Les applications et les équipements sont mis à jour en fonction de plans pluriannuels, les droits d'accès et les paramètres personnalisés sont administrés en permanence. Un suivi quotidien de chacune des ressources et de leur utilisation est nécessaire à leur bonne gestion.

Assistance aux utilisateurs et interventions

Les demandes d'intervention d'utilisateurs sont traitées continuellement. La maintenance et l'installation du parc des postes de travail et des imprimantes s'effectuent selon les urgences et les plans de renouvellements.

Système d'Information Géographique

La mise à disposition pour les services de la RCA de documents cartographiques et d'analyse géographique exige la capitalisation, le traitement et la diffusion de l'ensemble de ces informations.

Ainsi, la DSI gère environ 300 postes de travail, 300 téléphones sur IP (Internet Protocol), 30 serveurs et environ 50 applications métier mis à disposition des 1800 agents de la RCA. Pour cela, le budget annuel de la DSI est d'environ 1 M€ (fonctionnement et investissement confondus).

Dans ce contexte, j'occupe un poste de chargé de projets au sein du pôle Études dans le cadre d'emploi des ingénieurs territoriaux.

Mes principales missions consistent à :

- consolider le SI de la RCA en proposant des axes d'amélioration par la réalisation d'études fonctionnelles, techniques et financières dans le respect de l'urbanisme du SI et du schéma directeur informatique,
- assurer aux utilisateurs la mise à disposition d'applications répondant aux besoins de traitements optimisés de leurs processus métier,
- garantir le suivi et l'adaptation des applications et de leurs interfaces,
- fournir aux agents assurant l'exploitation quotidienne et aux administrateurs système les fiches et cahier d'exploitation nécessaires à leur métier.

Je gère des projets plus ou moins importants en fonction du budget, du périmètre et du nombre d'acteurs... A titre d'exemple, j'ai été amené à manager le renouvellement et l'évolution de l'extranet de la formation professionnelle (budget de 280 K€ sur 4 ans) et l'acquisition et le paramétrage d'une application de gestion des contacts (80 K€ sur 18 mois). En parallèle, j'ai conduit de petits projets comme la migration de l'application des marchés publics vers la dernière version majeure (1500 € sur 3 mois), l'évolution fonctionnelle de l'extranet de la formation professionnelle (22 K€ sur 5 mois) tout en assurant la maintenance des applications mises en œuvre.

1.2 Les enjeux de l'administration régionale

1.2.1 La modernisation de l'administration

La RCA est une collectivité territoriale qui a grandi au rythme des transferts de compétences de l'État dans le cadre des lois de décentralisation. Elle assume aujourd'hui de nombreuses missions dans des domaines très variés et travaille avec de nombreux partenaires, qu'ils soient privés (p. ex. les entreprises locales) ou institutionnels (p. ex. les communes de la région et certains ministères de l'État...).

La modernisation continue de son organisation et de ses méthodes de travail collaboratif avec ses partenaires est une conséquence directe de ces transferts de compétences successifs tout au long de ces 30 dernières années. Historiquement, cette modernisation s'est concrétisée par une informatisation massive : câblages des locaux, mise en réseaux des postes de travail, mise en œuvre d'une infrastructure pour héberger les serveurs informatiques, mise en place d'applications supportant le fonctionnement (finances et comptabilité, ressources humaines...) et le cœur de métier de la RCA (gestion des subventions et des lycées, formation professionnelle...).

Actuellement, les objectifs de modernisation poursuivis par la RCA peuvent se résumer de la manière suivante :

- améliorer des outils existants,
- valoriser le poste de travail,
- rationaliser les coûts,
- partager les informations en interne et avec les partenaires externes.

1.2.2 L'impact sur le système d'information

Concrètement, cette modernisation continue est en partie rendue possible par le développement et la maintenance de systèmes d'information de plus en plus complexes. En effet, un SI peut être considéré comme *« un ensemble de composants inter-reliés qui recueillent l'information nécessaire au fonctionnement de l'organisation, la traitent, la stockent et la diffusent afin de soutenir la prise de décision et le contrôle au sein de l'organisation »* [23]. De façon concrète, un SI se matérialise par un ensemble de systèmes informatiques (matériels et logiciels) supportant tout ou partie des processus métiers d'une organisation et les flux d'information associés (voir les détails en annexe au §6.1).

Pour faire face à ces enjeux et maîtriser le cycle de vie des systèmes d'information de la RCA, la DSI s'appuie sur un outil incontournable : le Schéma Directeur Informatique (SDI). D'après [36], le SDI correspond à une opération de prospective et de planification stratégique, à moyen ou long terme, effectuée dans une approche systémique menant à une politique d'informatisation, d'organisation et de communication. Le SDI est donc un outil au service d'une stratégie des systèmes d'information alignée sur les objectifs de la RCA. De façon pragmatique, le SDI se concrétise par un portefeuille de projets (planification, plan de charge, ressources nécessaires, méthode de management des projets, interactions avec le plan d'urbanisation des SI...) permettant :

- d'aligner les objectifs du SI sur ceux de la RCA,
- d'informatiser les processus métier exécutés par les agents et leurs partenaires,
- d'acquérir et de mettre en œuvre des solutions informatiques qui répondent aux besoins des utilisateurs internes (agents) et externes (partenaires).

1.2.3 Les conséquences sur les projets informatiques

Les projets identifiés par le SDI sont ensuite affectés aux différents pôles de la DSI en fonction de leur contenu : projets d'infrastructure, applicatif... Quelque soit le type de projet, l'organisation mise en place pour les conduire doit garantir le respect des objectifs assignés à chaque projet informatique en termes de coûts, de qualité, de fonctionnalités et de délais.

Au niveau du pôle Études, cela passe par :

- une politique et une stratégie d'assurance qualité,
- une ingénierie des besoins efficace,
- la maîtrise du processus d'acquisition des applications,
- la standardisation et la professionnalisation du management des projets pour assurer un bon de niveau de prédictibilité (p. ex. pour décider de la mise en production d'une nouvelle application), de performance et de maîtrise des risques,
- le contrôle de la recette des applications livrées.

1.3 Les projets système d'information de la Région

1.3.1 L'organisation d'un projet

Nous distinguons deux catégories d'entités organisationnelles dans un projet piloté par le pôle Études. Les entités internes qui englobent la MOA (les directions métiers, les contributeurs, les utilisateurs finaux) et l'AMOA (la DSI), et les entités externes qui comprennent la MOE (généralement l'entreprise qui a remporté le marché public), les partenaires de la RCA (contributeurs et/ou utilisateurs finaux) et éventuellement des fournisseurs intermédiaires (pour les achats complémentaires).

Les acteurs qui participent au projet sont tous issus de ces entités et sont regroupés au sein de structures organisationnelles en fonction de leur mission.

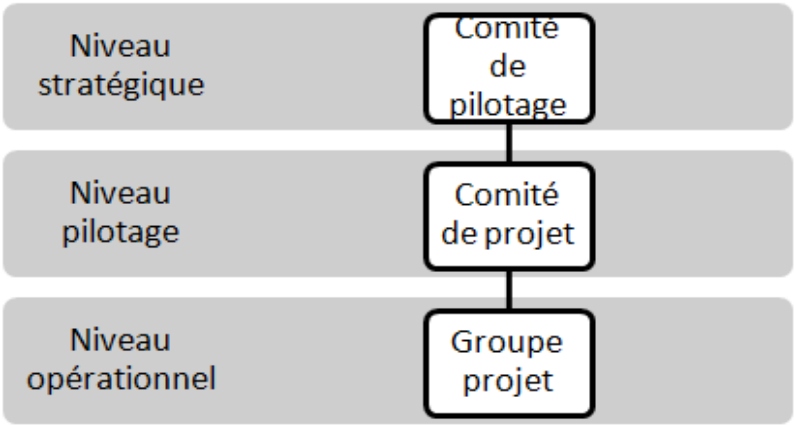


Illustration 2: Structure organisationnelle d'un projet Études

Comité de pilotage

Il regroupe la direction générale des services, le directeur des systèmes d'information, la directrice du conseil de gestion et le ou les directeurs métiers qui représentent la MOA. Ce comité définit la stratégie et les orientations du projet.

Comité de projet

Il regroupe le directeur des systèmes d'information, la directrice du conseil de gestion et le ou les directeurs métier qui représentent la MOA ainsi que les chefs de projet MOA (direction métier) et AMOA (DSI). Au moment de la phase de réalisation, il inclut également les représentants de la MOE : chef de projet, directeur projet et ingénieur commercial de l'entreprise retenue. Ce comité valide les livrables et les jalons du projet et entérine le passage d'une phase/étape à l'autre. Il suit également l'avancement du projet, les coûts, les délais, la qualité, les actions en cours et les risques.

Groupe projet

Il regroupe les chefs de projet MOA (direction métier) et AMOA (DSI) ainsi que des experts techniques (métier et DSI) et des utilisateurs clés (métier, partenaires). Ce groupe contribue et participe activement à toutes les phases du projet : analyse de l'existant, expression des besoins, recette fonctionnelle...

1.3.2 Les méthodes de management de projets

Un projet, selon la norme ISO 10006 [50], *« est un processus unique qui consiste en un ensemble d'activités coordonnées et maîtrisées, comportant des dates de début et de fin, entrepris dans le but d'atteindre un objectif conforme à des exigences spécifiques, telles que les contraintes de délais, de coûts et de ressources. ».* Partant de cette définition, une majorité d'organisations constatent encore aujourd'hui lors du bilan des projets, qu'un ou plusieurs des objectifs en termes de Coût, Qualité, Fonctionnalités et Délais (CQFD) n'ont pas été atteints de manière satisfaisante [26].

Ces insatisfactions se traduisent fréquemment par des surcoûts financiers, des dysfonctionnements logiciels, des retards importants ou des réductions de périmètre fonctionnel. Ces conséquences sont liées à des difficultés et des exigences inhérentes au travail en « mode projet ». En effet, les caractéristiques d'un projet sont parfois difficiles à définir, sa faisabilité n'est pas toujours démontrée, il consomme beaucoup de ressources et concurrence souvent d'autres projets; son déroulement reste à inventer, il nécessite une organisation spécifique qui concurrence l'organisation métier et la prise de décision peut se révéler particulièrement difficile à obtenir [24].

Ce constat global est également vrai pour certains projets de la RCA, ce qui a amené le pôle Études à standardiser ses activités de management de projet depuis maintenant plusieurs années. Concrètement, cette modernisation visant la maîtrise des objectifs CQFD d'un projet s'est effectuée autour de trois grands axes : les méthodes, les outils et les bonnes pratiques. Il existe de nombreuses méthodes de gestion de projet apparues dans les années 80 (p. ex. Merise[1]). Historiquement, ces méthodes visaient à clarifier le déroulement d'un projet, à professionnaliser les processus exécutés, à améliorer les relations et la communication entre les informaticiens et les utilisateurs. Elles proposaient également des outils graphiques permettant de représenter les futurs systèmes informatiques.

Toutes ces méthodes ont permis d'organiser le travail des équipes informatiques et d'introduire une culture méthodologique au sein des organisations. Néanmoins, elles n'ont pas tenues toutes leurs promesses, l'application pratique de cadres théoriques, souvent considérés comme trop rigides, restait malaisée du fait de la complexité et de la diversité des projets informatiques.

1 Merise est une méthode d'analyse, de conception, de modélisation et de gestion de projet informatique.

C'est pourquoi, la notion d'agilité est apparue au début des années 2000[2] en réaction à la persistance des problèmes de coût, de qualité et de délais dans les projets informatiques. La force annoncée de ces méthodes agiles (p. ex. XP[3]) résidait dans leur capacité à s'adapter aux différents types de projets. Pour cela, elles ont valorisée les dimensions humaines et organisationnelles d'un projet.

Finalement, toutes ces méthodes reposent sur une organisation impliquant les différents acteurs (décideurs, utilisateurs, développeurs...) et sur des outils pour contrôler le processus projet (atelier de génie logiciel, prototypage, diagramme de Gantt[4]...) ainsi que sur la systématisation des bonnes pratiques qui ont fait leur preuve dans le temps.

Cependant, la singularité de chaque projet informatique ne permet pas d'aboutir aujourd'hui à une méthode universelle, même s'il est vrai que des référentiels généralistes sont apparus pour permettre de répondre à tous les besoins en termes de gestion de projet (p. ex. le PMBOK[5]). Il faut donc nécessairement adopter une approche contingente pour aboutir et respecter les objectifs CQFD d'un projet dans une organisation donnée [25]. Dans la pratique, cela se traduit par l'utilisation de tout ou partie de plusieurs méthodes qui sont adaptées en fonction des besoins et des problématiques rencontrées.

1.3.3 Le cycle de vie des projets des Études

Le pôle Études, composé de 4 chargés de projet et d'un responsable, prend en charge les projets d'informatisation des processus métier de la RCA. Ces projets sont systématiquement gérés par une maîtrise d'ouvrage (MOA) composée d'une ou plusieurs directions métier et d'une assistance à maîtrise d'ouvrage (AMOA) assurée par la DSI. La maîtrise d'œuvre (MOE) est généralement externalisée. Ces projets ont un cycle de vie qui couvre un certain nombre d'étapes allant des analyses préliminaires jusqu'au bilan du projet en passant par les phases d'acquisition et de réalisation des applications informatiques mises en œuvre.

Dans ce contexte, les équipes projets (MOA, AMOA et MOE confondues) sont amenées à développer les applications qui répondront aux exigences des utilisateurs dans le respect des coûts, des délais et de la qualité définis par les décideurs. Pour faciliter l'atteinte de ces objectifs, le pôle Études de la DSI a développé sa propre démarche projet, adaptée au contexte de la RCA en fonction de l'expérience acquise et largement inspirée des méthodes classiques et agiles.

2 En sachant que les fondements de ces méthodes ont été élaborés dans les années 90, comme RAD (Rapid Application Developpement) par exemple.

3 eXtreme Programming (XP) est une méthode de gestion/réalisation de logiciel dite agile dans le sens où elle capable d'absorber les évolutions de besoins et les changements inhérents à tous projets informatiques.

4 Un diagramme de Gantt est un outil graphique qui permet d'ordonnancer les tâches d'un projet en fonction du temps. MS Project et GanttProject sont des logiciels permettant de réaliser et gérer des diagrammes de Gantt.

5 Project Management Body Of Knowledge : corpus de connaissances largement reconnues par les acteurs du management de projet. Les processus et techniques présentées sont censées être applicables à la majorité des projets.

Celle-ci se décompose en 4 phases découpées en étapes :

- phase n°1 : études
  - étape n°1.1 : étude de cadrage (faisabilité, opportunité...)
  - étape n°1.2 : étude détaillée (existant, processus, besoins, solutions...)
- phase n°2 : marché d'acquisition (logiciels, matériels, prestations)
  - étape n°2.1 : écriture du cahier des charges
  - étape n°2.2 : publication de l'avis d'appel à la concurrence
  - étape n°2.3 : analyse des réponses des entreprises
  - étape n°2.4 : notification du marché public à l'entreprise retenue
- phase n°3 : réalisation (mise en œuvre de la solution retenue)
  - étape n°3.1 : lancement du projet
  - étape n°3.2 : spécifications détaillées et conception
  - étape n°3.3 : mise en ordre de marche (développements, tests, installation MOE)
  - étape n°3.4 : vérification d'aptitude au bon fonctionnent (recette MOA)
  - étape n°3.4 : vérification de service régulier (mise en production)
- phase n°4 : bilan
  - étape n°4.1 : évaluation des résultats du projet par rapport aux objectifs fixés
  - étape n°4.2 : mise en exploitation (transfert vers la maintenance)
  - étape n°4.3 : capitalisation (partage des bonnes pratiques)

Cette démarche s'apparente aux méthodes classiques car au niveau macroscopique le cycle de vie du projet correspond à un modèle en cascade pour l'enchaînement des phases et en V en ce qui concerne les étapes de la phase de réalisation. Néanmoins, au niveau microscopique la démarche adoptée est beaucoup plus pragmatique et en cela elle se rapproche des méthodes agiles : forte implication des utilisateurs, développement itératif et incrémental en phase n°3, prototypage et tests utilisateurs.

Il faut également noter qu'à chaque étapes un certain nombre d'activités standardisées (p. ex. la rédaction d'un cahier des charges) permettent d'organiser le travail des équipes projet en termes d'objectifs concrets à atteindre, de livrables à produire, d'outils et de techniques à utiliser (p.ex. le diagramme de Gantt). Ces activités sont en générale gérées selon le principe de la roue de Deming ou cycle PDCA (Plan Do Check Act)[6].

1.4 Mon projet de mémoire

1.4.1 Mon sujet de mémoire

La démarche de management de projet présentée dans les chapitres précédents fait partie d'un processus d'amélioration continue des pratiques du pôle Études de la DSI. Dans ce contexte, l'industrialisation de la recette fonctionnelle des SI mis en œuvre par la RCA participe de cette amélioration continue. Par industrialisation, nous désignons un ensemble constitué par les référentiels de bonnes pratiques, l'organisation, les méthodes, les techniques et les outils ainsi que la formation des personnes conduisant à une meilleure maîtrise de la recette fonctionnelle et des risques associés. De manière synthétique, l'industrialisation de la recette peut être vue comme une succession d'étapes conduisant à gestion efficace des ressources de la RCA dans un contexte projet donné.

C'est dans cet esprit que mon projet de mémoire d'ingénieur CNAM s'intitule « La recette fonctionnelle d'un Système d'Information - Enjeux, méthodes, outils et exemple d'application ».

Le travail à réaliser se décompose en quatre phases :

1. Étude des enjeux, des méthodes et des solutions (état de l'art : processus, méthodes, logiciels...) permettant de maîtriser la recette fonctionnelle d'un projet informatique, et conception d'une solution adaptée au contexte de la RCA.

2. Réalisation, mise en œuvre et test de la pertinence de la solution retenue.

3. Bilan : analyse des résultats, conclusions et propositions d'améliorations.

4. Rédaction du mémoire, préparation de la soutenance.

Concrètement, mon sujet de mémoire s'inscrit parfaitement dans mes activités professionnelles à travers deux projets du pôle Études que je manage en tant que chef de projet informatique :

- le projet Recette Fonctionnel d'un Système d'Information (RFSI) qui couvre la majeure partie du travail de mémoire à réaliser,

- le projet d'Ouverture du Système d'Information aux Demandeurs Internautes (OSIDI) pour la partie « test de la pertinence de la solution retenue ».

6 Le cycle de vie PDCA ou roue de Deming est une méthode permettant d'assurer l'amélioration continue de la qualité d'un projet définie par William Edward Deming, statisticien américain du 20ème siècle.

En résumer, la solution définie dans le cadre du projet RFSI sera éprouvée dans le cadre de la recette du projet OSIDI.

1.4.2 La recette fonctionnelle d'un SI

Le projet RFSI vise à organiser et simplifier la recette et les tests des projets fonctionnels[7] menés par le pôle Études. De fait, la recette est aujourd'hui une étape clé souvent difficile dans un projet, car il faut pouvoir vérifier et valider correctement les applications livrées afin de minimiser l'apparition d'anomalies en production, tout en maîtrisant les coûts et les délais contractuels. En effet, une recette non maîtrisée conduit généralement à des glissements de planning, des surcoûts financiers, des réductions de périmètre fonctionnel et à des dysfonctionnements de l'application en production. Ce qui se traduit par un coût global du projet élevé et difficilement amortissable sur le moyen terme.

Dans ce sens, le projet RFSI vise les objectifs présentés par le diagramme ci-après.

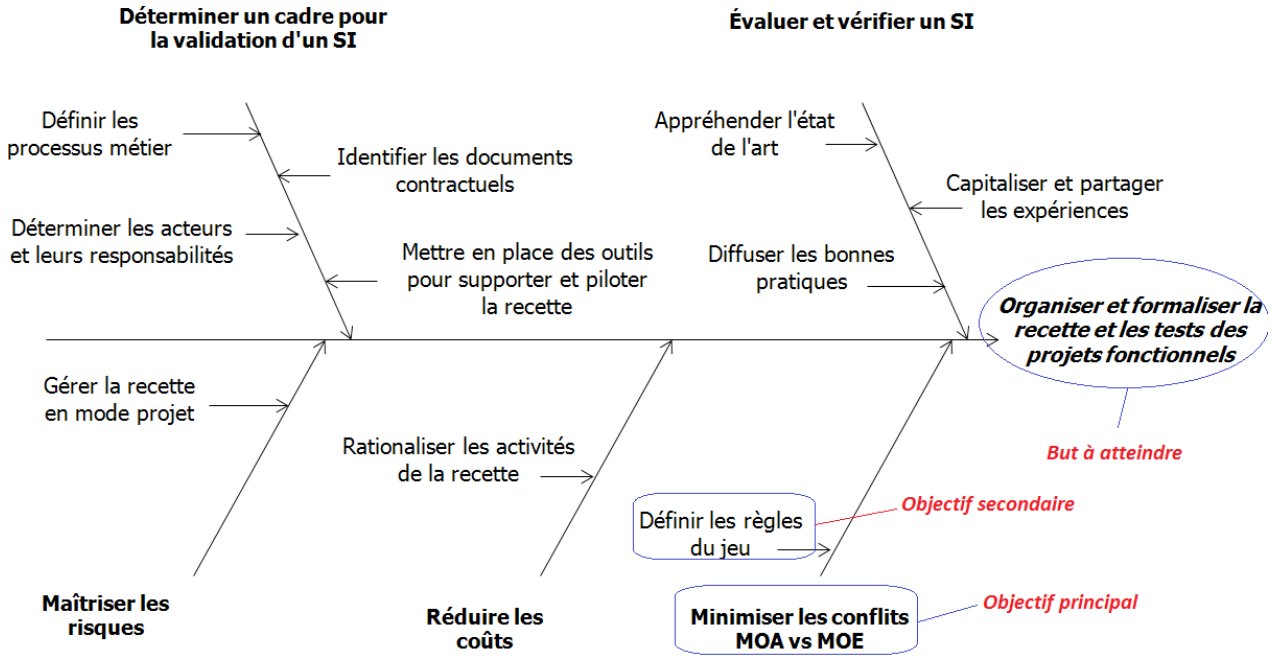


Illustration 3: Objectifs du projet RFSI

Le périmètre du projet RFSI se limite au traitement de la phase de réalisation (cf. §1.3.3), la phase de maintenance est donc hors périmètre, tout en sachant que celle-ci pourra être traitée dans le cadre d'un projet ad hoc ultérieur en s'appuyant sur les résultats du projet RFSI. En termes de contraintes, le budget alloué au projet est de 25 K€ et les délais à respecter dépendent essentiellement des principaux jalons du projet OSIDI, sachant qu'il est prioritaire par rapport à RFSI (i.e les ressources disponibles sont affectées en priorité à OSIDI).

7 Par opposition aux projets dits techniques (infrastructure, réseau, exploitation...).

1.4.3 Le portail des aides régionales

Le projet OSIDI vise à uniformiser, simplifier et dématérialiser la gestion des demandes d'aides régionales pour répondre aux besoins des utilisateurs en s'appuyant sur des outils capables d'intégrer les différents dispositifs d'aides (partage des informations) à moindre coût (mutualisation de l'infrastructure technique). Dans cette optique, le projet OSIDI doit se concrétiser par la mise en place d'un portail de saisie et de suivi des demandes d'aides régionales permettant de centraliser la publication des aides et de mutualiser les moyens informatiques mis en œuvre.

Dans ce sens, le projet OSIDI vise les objectifs présentés par le diagramme ci-après.

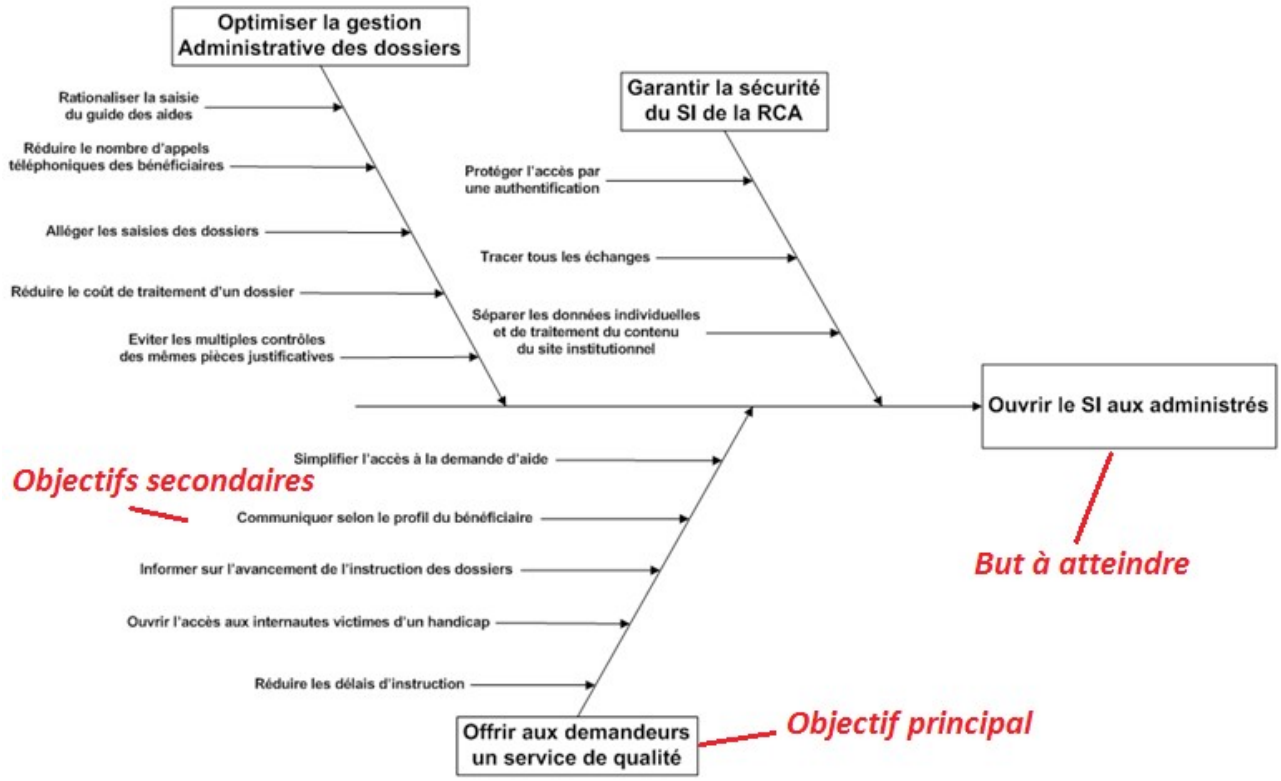


Illustration 4: Objectifs du projet OSIDI

A court terme, le périmètre du projet OSIDI se limite au traitement des aides à destination des étudiants (dispositif StudéO[8]) et aux bourses aux élèves en formations paramédicales et en travail sociale[9].

Le dispositif StudéO comprend les aides suivantes : l'aide à la mobilité internationale, l'aide pour la réalisation d'un stage en entreprise, Pass' Études et Libre Études, l'aide à l'achat d'un ordinateur portable, l'aide « complémentaire santé ».

A moyen et long terme, l'objectif du portail est de couvrir l'ensemble des aides régionales (environ 200 actuellement).

8 Voir la description du dispositif sur le site institutionnel de la RCA (http://www.cr-champagne-ardenne.fr/?SID=57).
9 Idem, voir http://www.cr-champagne-ardenne.fr/?SID=817.

En termes de contraintes, le budget alloué au projet est de 270 K€ et les délais à respecter lors de la phase réalisation (cf. §1.3.3) sont les suivants :

- lancement du projet → septembre 2011
- spécifications détaillées et conception → septembre 2011-décembre 2011
- mise en ordre de marche (développements, tests, installation MOE) → janvier-mars 2012
- vérification d'aptitude au bon fonctionnent (recette MOA) → mars-avril 2012
- vérification de service régulier (mise en production) → mai-juillet 2012

1.4.4 Mon rôle et mes missions

En tant qu'ingénieur territorial (mon cadre d'emploi), j'occupe un poste de chargé de projet au sein du pôle Études de la DSI depuis le 1er janvier 2008. Dans les projets qui me sont confiés, comme RFSI et OSIDI, j'assume un rôle de chef de projet informatique (AMOA).

Cela se traduit par un certain nombre de missions qui me sont attribuées :

- étude d'opportunité avec la MOA,
- ingénierie des besoins avec la MOA,
- acquisition des solutions avec la MOA : cahier des charges, appel d'offres, analyse et choix,
- mise en œuvre des solutions retenues avec la MOA/MOE : spécifications fonctionnelles et techniques,
- recette fonctionnelle des solutions avec la MOA/MOE,
- gestion de projet : coordination des acteurs, planification, suivi des coûts, des délais et de la qualité, encadrement de la MOE/MOA, conduite du changement.

La RCA est donc la structure administrative qui permet de mettre en œuvre les politiques des élus du CRCA dans les limites des compétences qui lui sont assignées.

Dans ce contexte, les systèmes d'informations participent à la modernisation de l'organisation et des méthodes de travail de la RCA en supportant les processus métier et les échanges d'informations entre les différents acteurs. Cela passe essentiellement par la mise en place de systèmes informatiques (matériels et logiciels) dans le cadre de projets ad hoc.

En conséquence, le management de projet devient une activité cruciale dans la réussite des projets informatiques menés par la DSI.

C'est dans ce sens que le pôle Études, en charge des projets fonctionnels des directions métier de la RCA, a développé sa propre démarche de gestion de projet en s'inspirant et en adaptant des éléments des méthodes de gestion de projet existantes.

La deuxième partie du présent mémoire se propose donc d'expliciter les éléments fondamentaux constitutifs de la démarche projet du pôle Études.

Ainsi, je présenterai le processus d'ingénierie des besoins et son importance dans la qualité globale d'un projet. Puis, je décrirai les principales méthodes et techniques d'estimation des charges et de planification des délais avant d'expliquer comment le pôle Études pilote ses projets.

Enfin, j'aborderai les problématiques liées à la dimension relationnelle d'un projet et la nécessaire conduite du changement induit par tout projet informatique.

2 Une démarche projet pour réussir la mise en œuvre d'un SI

Un système d'information est un ensemble *« d'informations organisées, d'évènements ayant un effet sur ces informations et d'acteurs qui agissent sur ces informations ou à partir de ces informations, selon des processus visant une finalité de gestion et utilisant les technologies de l'information »* [25]. La mise en œuvre d'un SI consiste donc à identifier les acteurs, les processus et les informations ainsi que les besoins de traitement afin de mettre en place les systèmes informatiques (matériels et logiciels) permettant de les supporter. Dans ce contexte, une démarche de « projet SI » est l'ensemble des activités permettant d'organiser et de surveiller le déroulement du projet dans le but de le mener à son terme [25], en respectant les coûts, la qualité et les délais fixés.

2.1 Les difficultés spécifiques d'un projet SI

2.1.1 Le contexte de la Région

En plus des problèmes classiques de gestion de projet (caractère abstrait d'un SI, instabilité des exigences, modification des plannings…), le chef de projet Études rencontre des difficultés spécifiques liées au contexte et à l'environnement de la RCA.

Celles-ci sont essentiellement dues à deux facteurs :

1. l'organisation et l'écosystème des projets de la RCA,
2. le cadre réglementaire imposée à la RCA.

Les projets gérés par le pôle Études peuvent être qualifiés de complexes, car chaque changement (organisation, technologies, contraintes légales...) se répercute généralement sur l'équilibre du projet (coûts, qualité, délais) et/ou sur le SI en cours de mise en œuvre, déclenchant tout un système de rétroactions plus ou moins complexes à maîtriser.

2.1.2 L'organisation et l'écosystème d'un projet

Comme brièvement évoqué dans le §1.3.1, on retrouve dans un projet informatique de nombreux acteurs, diverses entités organisationnelles et différents niveaux de décision.

De nombreux acteurs

Le nombre important d'acteurs est justifié par le large spectre de compétences nécessaires à la réalisation des différentes tâches du projet (définition des besoins, réalisation, pilotage, gestion financière et marchés publics...). De ce fait, le chef de projet Études est amené à piloter, coordonner, contrôler et participer à la réalisation d'une variété de tâches techniques et administratives impliquant des acteurs internes (agents) et externes (prestataires, partenaires) sur lesquels il n'a aucun pouvoir hiérarchique. D'autant plus, qu'il faut le plus souvent paralléliser les activités pour garantir des délais acceptables. Dans ce contexte, les outils et les pratiques du management transversal et de la négociation sont les principales armes du chef de projet Études.

Diverses entités organisationnelles

L'externalisation de la réalisation des développements informatiques est un choix politique dicté par des contraintes de coût. Il en effet plus rentable sur le moyen terme pour la RCA d'externaliser la fonction « développements » compte tenu du coût d'une équipe de développeurs internes (recrutement, formation, intégration...). En contre-partie, la réalisation d'un projet implique en général plusieurs sociétés informatiques, le chef de projet Études doit alors gérer la coordination des actions et la communication entre les différents acteurs.

De même, la multiplicité des entités organisationnelles (cf. §1.3.1) implique de communiquer de façon ciblée (niveaux stratégique, pilotage, opérationnel) et de piloter les projets de manière collégiale afin d'engager et de responsabiliser tous les acteurs. En conséquence, un travail de reporting important est demandé au chef de projet Études.

Différents niveaux de décision

Enfin, la chaîne de décision de la RCA (élus, direction générale, conseil de gestion, DSI...) peut impacter significativement les délais de réalisation d'un projet. Par exemple, dans le cadre du projet OSIDI, le quorum[10] n'a pas été atteint lors de la Commission d'Appels d'Offres (CAO) initialement prévue. La décision de retenir la société SWORD a donc été repoussée à la CAO suivante, ce qui s'est traduit par un glissement du planning initial de deux mois.

Pour conclure et à titre d'exemple, le projet OSIDI implique actuellement 5 directions, 3 sociétés informatiques et compte environ 25 acteurs travaillant sur le projet :

- la MOA est représentée par la direction de l'enseignement supérieur, de la recherche et de l'innovation et par la direction de la jeunesse, du sport, de la santé et de la vie associative (experts métiers, utilisateurs finaux, testeurs),

- l'AMOA est assurée par la DSI (directeur et chef de projet, les équipes exploitation, réseaux, systèmes et sécurité, testeurs),

- la MOE est confiée à la société SWORD (directeur et chef de projet, responsable commercial, architecte/concepteur, graphiste, développeurs, testeurs),

- d'autres partenaires sont également concernés comme les sociétés BERGER-LEVRAULT, qui intervient dans la mise en place des interfaces entre les front-office et back-office du SI, ou encore Microsoft comme fournisseur de logiciel d'infrastructure.

Sans oublier le public de la RCA en tant qu'utilisateurs finaux : étudiants, établissements d'enseignement supérieur...

10 Nombre minimal de membres d'un corps délibératif nécessaire à la validité d'une décision.

2.1.3 Le cadre réglementaire

En tant que collectivité territoriale, la RCA est soumise à un certain nombre de contraintes réglementaires touchant d'une part ses achats et ses dépenses et d'autre part, les outils informatiques qu'elle met en œuvre.

Code des marchés publics

La RCA doit respecter un certain nombre de règles pour satisfaire ses besoins en matière de travaux, de fournitures ou de services. Ces règles sont définies dans le Code des Marchés Publics (CMP) [55] et ont pour objectif de faire respecter un certain nombre de grands principes [56] :

- la liberté d'accès des entreprises à la commande publique,
- l'égalité de traitement des candidats,
- le contrôle de l'usage des deniers publics (publicité des offres, mise en concurrence des fournisseurs, transparence des choix effectués, contrôles externes).

Pour faire respecter ces principes et permettre aux collectivités territoriales de satisfaire leurs besoins, le CMP prévoit différentes procédures permettant d'aboutir à la signature d'un contrat administratif entre la RCA et un fournisseur.

Les principaux concepts définis par le CMP sont les suivants [56] :

- la catégorie du marché permet d'adapter la réglementation à l'objet du marché : par exemple les marchés de relatifs aux techniques de l'information et de la communication sont soumis à des dispositions spécifiques[11],
- la forme du marché permet d'apporter plus ou moins de souplesse dans l'exécution du marché ; il y quatre forme principales, qui peuvent être par ailleurs combinées, le marché ordinaire quand tous les paramètres sont connus à l'avance (durée, rythme d'exécution...), le marché fractionné à bons de commande ou à tranches conditionnelles pour moduler le rythme des besoins à satisfaire et l'accord-cadre pour séparer la procédure de choix des fournisseurs de celle des futurs commandes (*via* des marchés ad hoc),
- la durée du marché n'est pas imposée par le CMP mais les bonnes pratiques la limite en générale à quatre années maximum,
- le prix du marché conditionne le type de procédure (appel d'offres ouvert, marché à procédure adaptée, dialogue compétitif...) en fonction d'un certain nombre de seuils[12].

11 Pour plus de détails, voir notamment le site http://www.marche-public.fr/Marches-publics/Marches-publics.htm/.
12 Voir le détail des différents seuils à cette adresse http://www.marche-public.fr/Marches-publics/Definitions/Entrees/Publicite-tableaux-obligations-pa.htm

Le processus « marché public »

Dans le cadre de la démarche projet du pôle Études, une phase « marché d'acquisition » est systématiquement nécessaire avant de pouvoir acheter des prestations, des logiciels et des matériels. Pour cela, il faut tout d'abord formaliser les besoins à satisfaire, c'est le rôle du cahier des charges, appelé Dossier de Consultation des Entreprises (DCE) dans le jargon juridique. Puis, la consultation des entreprises commence par une publicité incitant les candidats potentiels à répondre au marché public. La forme de cette publicité dépend du montant du marché. Par exemple, pour un marché de fournitures et de services[13] dont le montant est compris entre 90 et 130 K€ un publicité nationale dans le Bulletin Officiel des Annonces des Marchés Publics (BOAMP) ou dans un journal d'annonces légal (p. ex. l'Union dans la Marne) est obligatoire[14].

Concrètement, le DCE est déposé sur la plate-forme des marchés publics de la RCA[15], dont je m'occupe par ailleurs dans le cadre des mes missions. Les candidats peuvent alors le télécharger DCE et proposer leur offre via la plate-forme des marchés publics. Pour cela, ils disposent d'un délai réglementé par le CMP qui dépend du montant du marché et du type de procédure (appel d'offres ouvert, marché à procédure adaptée, dialogue compétitif...).

Une fois que le délai de réponse est écoulé, les candidatures et les offres remises à la RCA sont analysées puis classées par une Commission d'Appel d'Offres[16] (CAO) en fonction des critères de jugement des offres consignés dans le règlement de consultation. Dans la plupart des cas, le marché est attribué puis notifié au candidat retenu après signature du Président du CRCA ; néanmoins, si les offres ne répondent pas de manière satisfaisante aux besoins de la RCA alors le marché peut être déclaré infructueux.

L'exécution du marché pourra commencer à compter de la date de retour de l'accusé-réception de la notification au candidat retenu. Les dépenses se feront tout au long de la réalisation du projet en respectant le principe du service fait qui consiste à vérifier la réalité de la dette ; autrement dit toute facture ne pourra être réellement payée par le Payeur régional que sur présentation des justificatifs associés : c'est le principe de séparation de l'ordonnateur et du payeur. Une fois la durée du marché échue, ce dernier est soldé puis clôturé.

Les référentiels de l'administration électronique

En 2010, les services l'État (DGME et ANSSI[17]) ont publié trois référentiels de l'administration électronique ayant pour objectifs de rendre interopérables (RGI), de sécuriser (RGS) et de rendre accessible (RGAA, CE) les SI mis en œuvre par les administrations publiques. Ces trois référentiels, qui s'imposent en partie à la RCA, recensent les bonnes pratiques et les technologies à employer pour atteindre les objectifs de l'administration électronique.

13 Cas des marchés informatiques.

14 Voir le modèle obligatoire à suivre à cette adresse http://www.journal-officiel.gouv.fr/documents/externe/aapc_2006.pdf.

15 https://marchespublics.cr-champagne-ardenne.fr/

16 La Commission d'Appel d'Offres de la RCA est une instance composée d'élus régionaux à voix délibérative qui sont issus du Conseil régional.

17 Agence Nationale de la Sécurité des Systèmes d'Information et Direction Générale de la Modernisation de l'État.

Le Référentiel Général d'Interopérabilité (RGI) fixe les règles minimales d'interopérabilité concernant la messagerie (protocole, sécurisation), les profils de services web et les infrastructures (annuaire LDAP, protocoles, sécurisation, horodatage et synchronisation).

Le Référentiel Général de Sécurité (RGS) fixe les règles minimales de sécurité : authentification (personnes et machines), confidentialité (données échangées), signature (personnes et machines) et horodatage (transactions) électroniques. Les processus d'accusé de réception et d'enregistrement électroniques sont également soumis au RGS.

Le Référentiel Général d'Accessibilité pour les Administrations (RGAA) fixe les conditions de publication d'information sur une application web afin de rendre le contenu perceptible, utilisable, compréhensible et robuste.

Enfin, la Charte Ergonomique (CE) donne toutes les recommandations nécessaires à la mise en œuvre d'une application web ergonomique (interfaces conviviales, facilité d'utilisation).

Pour le chef de projet Études, cela implique de prévoir dans le cahier des charges les moyens de faire respecter ces référentiels par la MOE (inscription dans les contrats, vigilance concernant les technologies utilisées) et de contrôler leur respect effectif au moment de la recette fonctionnelle, notamment en prévoyant éventuellement des audits ad hoc réalisés par un tiers.

2.2 L'ingénierie des besoins : comment comprendre les attentes des utilisateurs

2.2.1 Des projets complexes à maîtriser

Le rapport *Chaos* du Standish Group [26] montre que plus de la moitié des projets informatiques sont abandonnés ou ne respectent pas les contraintes de coûts ou de délais fixées initialement. Dans environ 50% des cas, l'échec du projet est imputable directement ou indirectement au processus d'ingénierie des besoins (p. ex. un cahier des charges incomplet) [15] et plus précisément à un défaut dans la gestion des besoins. Par gestion, nous entendons tout ce qui permet de collecter, de définir, de formaliser et de communiquer les besoins des utilisateurs aux différents acteurs (MOA, AMOA, MOE). Nous incluons également dans cette définition les moyens permettant de maîtriser l'intégration des inévitables évolutions des besoins au cours du projet.

Par ailleurs, la norme AFNOR NFX 50150 [53] définit un besoin comme *« une nécessité ou un désir éprouvé par l'utilisateur et non le volume du marché. Un besoin peut être exprimé ou implicite, avoué ou inavoué, latent ou potentiel. Dans tous les cas, il constitue le besoin à satisfaire pour lequel l'utilisateur est prêt à faire un effort »*.

Dans ce contexte, la définition des besoins est au cœur de la relation contractuelle liant la MOA à la MOE et une activité importante du processus de développement d'un logiciel, car elle influence directement son coût et sa qualité. Par exemple, on sait depuis maintenant plusieurs années que le coût de la correction d'une anomalie est environ 200 fois plus élevé après livraison du logiciel qu'en phase d'expression des besoins [34].

Le schéma ci-après illustre la relation entre MOA et MOE dans le cadre d'un processus d'ingénierie des besoins.

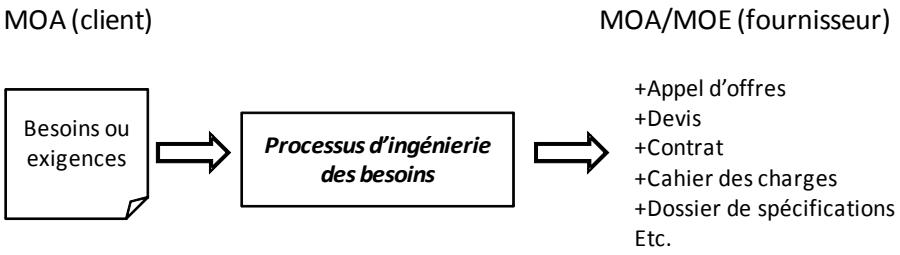


Illustration 5: La relation MOA/MOE et le processus d'ingénierie des besoins

2.2.2 Des besoins difficiles à identifier

Un certain nombre de difficultés sont inhérentes à la nature même du besoin qui peut être implicite, latent ou potentiel et donc non exprimé clairement par la MOA. Il faut donc aller au-delà des besoins explicites, qui représentent la partie émergée de l'iceberg, pour identifier l'ensemble des besoins à satisfaire. De plus, le manque de langage commun entre la MOA et la MOE est un facteur important de déformation des besoins. D'autant plus important, quand on considère la chaîne de transmission des besoins et la multitude des acteurs impliqués (voir schéma ci-après).

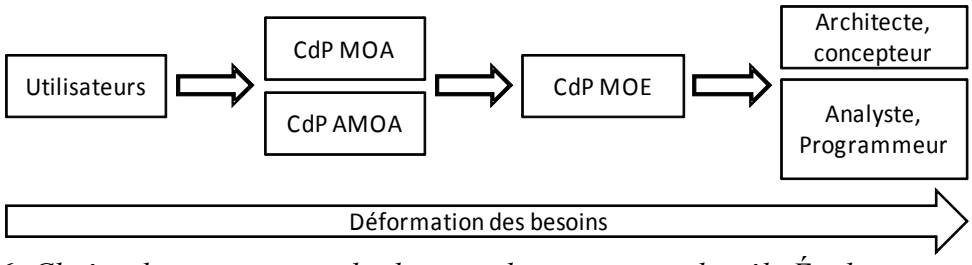


Illustration 6: Chaîne de transmission des besoins dans un projet du pôle Études

De même, la formalisation de l'expression des besoins joue un rôle primordial afin de pouvoir différencier *a posteriori* dans le discours de la MOA des éléments de nature différente, comme les besoins, les attentes, les objectifs, les contraintes, les solutions techniques...

Enfin, les surspécifications sont un autre écueil rencontré lors de l'expression des besoins. En effet, la surabondance des besoins exprimés par la MOA sans analyse critique, impactent fortement l'équilibre entre les coûts, la qualité et les délais d'un projet. Dans ce cas, le dimensionnement des besoins et la prise en compte du retour sur investissement sont complètement négligés.

L'exemple d'un processus d'ingénierie des besoins décrit ci-après, se propose de répondre à cette problématique.

2.2.3 Le processus d'ingénierie des besoins

L'ingénierie des besoins est le domaine qui regroupe les activités liées à l'élicitation[18], l'analyse, la spécification, la validation et la gestion des besoins (ou exigences) à travers leur documentation, leur traçabilité et la prise en compte des évolutions dans le cycle de vie du logiciel. L'objectif étant de rédiger des spécifications des besoins de qualité, c'est-à-dire possédant les caractéristiques suivantes [54] : exacte, non-ambiguë, complète, cohérente, hiérarchisée, vérifiable, modifiable et traçable. Le diagramme ci-après schématise le processus en spirale d'ingénierie des besoins.

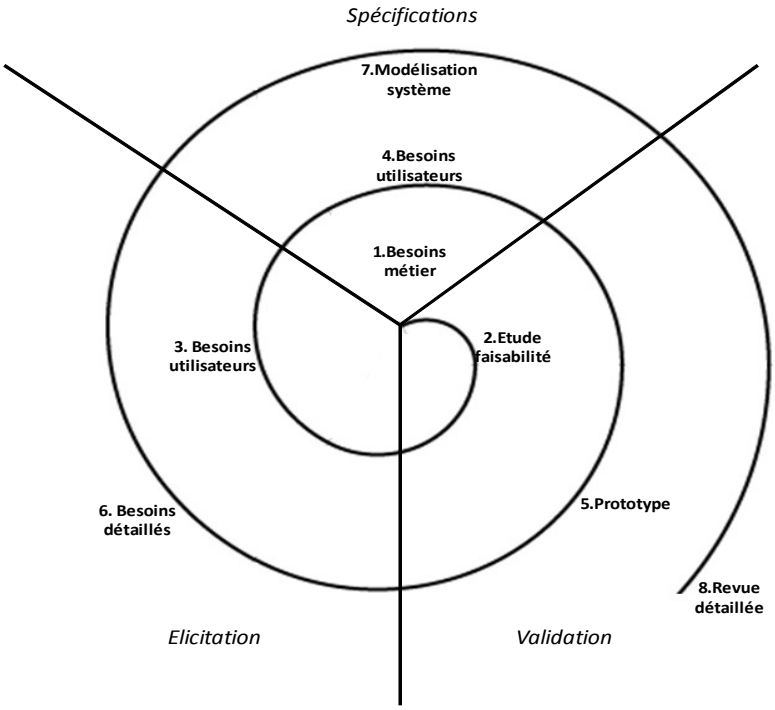


Illustration 7: Processus d'ingénierie des besoins en spirale selon [15]

La spécification et l'élicitation des besoins consistent à préciser et à détailler les besoins des utilisateurs en allant plus loin que la simple récolte d'informations. En effet, il s'agit d'écouter activement les utilisateurs et de les questionner afin de faire émerger les besoins latents et de clarifier les besoins implicites en utilisant systématiquement des techniques de reformulation et en alternant les questions ouvertes et fermées. En parallèle, la définition et la mise à jour régulière d'un référentiel des concepts et du vocabulaire métiers permettent de faciliter la communication entre les différents acteurs. Dans ce sens, l'AMOA, rôle assuré par le pôle Études, qui cumule des compétences métier et informatiques est le facteur essentiel contribuant à l'efficacité de la communication entre la MOA et la MOE.

Enfin, la validation itérative et incrémentielle des besoins est une étape charnière permettant de fixer progressivement les besoins en vue de la rédaction d'un cahier des charges ou d'un dossier de spécifications. En général, l'utilisation d'une maquette et d'un prototype est préconisée pour faciliter la communication et la compréhension des différents acteurs, pour favoriser l'élicitation des besoins, pour vérifier l'ergonomie du logiciel ou pour soulever des problèmes jusque là insoupçonnés.

18 L'élicitation est le fait d'expliciter les besoins des utilisateurs et notamment ceux qui restent implicites ou sont latents. C'est le principe de la « maïeutique » : c'est-à-dire amener un utilisateur « à accoucher » de ses besoins par le questionnement.

Dans le cadre de la démarche projet du pôle Études, c'est le cahier des charges qui décrit les besoins et spécifie les contraintes de réalisation. C'est le principal livrable de la phase « marché d'acquisition ». Il est écrit par l'AMOA en partenariat avec la MOA. De même, le dossier de spécifications des exigences fonctionnelles et techniques, qui détaille la réponse aux besoins, est le principal livrable de la phase de « réalisation ». Il est écrit par la MOE en partenariat avec la MOA et l'AMOA.

2.3 La planification des charges et des délais pour organiser la réalisation des objectifs

2.3.1 Définitions et enjeux

Une fois que les objectifs d'un projet sont clairement définis, la question qui se pose tout naturellement est de savoir quels moyens vont permettre d'atteindre ces objectifs. Cela passe principalement par l'estimation des coûts, des délais et des ressources humaines. Ces données permettrons ainsi de prendre des décisions concernant le projet (enveloppe budgétaire, durée prévisible, effort à fournir) ou la stratégie à adopter (mobiliser des ressources internes, sous-traiter tout ou partie du projet, établir un planning de réalisation...). Concrètement, l'estimation des charges va permettre de calculer une estimation approximative du coût et des délais du projet et des ressources humaines nécessaires par l'application de règles simples que nous allons voir dans le chapitre suivant.

2.3.2 Les techniques d'estimation et de planification

Dans un projet, la charge exprimée en jour*homme représente une quantité de travail indépendamment du nombre de personnes qui vont réaliser effectivement ce travail [31]. La charge de travail estimée permet de déduire, via l'application de règles définies par les différentes méthodes d'estimation, le coût et la durée prévisibles d'un projet. La précision et la fiabilité de ces estimations dépendent avant tout du choix d'une méthode adaptée au contexte d'un projet. Il existe différentes familles de méthodes d'estimation de charge plus ou moins efficaces en fonction de l'avancement du projet [25].

Le jugement d'experts

Cette démarche permet d'estimer la charge globale d'un projet ou d'une activité quand peu d'éléments sont disponibles, par exemple lors d'une étude de faisabilité. La méthode DELPHI est une démarche de ce type qui implique plusieurs experts du domaine du projet. Les étapes à suivre sont les suivantes :

1. chaque expert estime une première fois la charge sur la base de son expérience,
2. les résultats anonymisés sont alors rendus publics,
3. chaque expert procède à une 2ème estimation en tenant compte des résultats précédents,
4. chaque expert doit justifier publiquement son estimation,
5. une révision des différentes estimations permet d'établir un consensus.

La limite de cette démarche réside dans la pertinence des experts impliqués : leur domaine de compétences, leur niveau d'expertise et leur expérience acquise.

L'estimation par analogie

Cette démarche permet d'estimer la charge globale d'un projet et de ses différentes phases ou d'une activité en s'appuyant sur l'expérience acquise sur d'autres projets ou activités similaires. Par exemple, la méthode de la répartition proportionnelle permet de répartir proportionnellement la charge globale, qui doit donc être estimée au préalable *via* une autre méthode, en fonction des différentes phases du projet. Dans le même esprit, la méthode des ratios est utilisée pour estimer les charges complémentaires à une activité principale, dont la charge a été estimée au préalable *via* une autre méthode, par exemple la charge de l'encadrement d'un projet correspond à 10% de la charge de programmation. Pour appliquer ces méthodes, il faut pouvoir disposer de données fiables, représentatives et en nombre suffisant pour pouvoir établir les règles de répartition et les ratios applicables. C'est la principale limite de ces méthodes.

L'estimation ascendante

Cette démarche permet d'estimer la charge globale d'un projet en identifiant les composants d'un logiciel à développer. La charge de travail associée à chaque composant est alors calculée en pondérant une charge de base en fonction de la typologie, la nature, la difficulté...du composant à développer. La somme des charges de chaque composant donne la charge globale du projet. Cette méthode est particulièrement adaptée au chiffrage approximatif d'un appel d'offres en analysant son cahier des charges. Évidemment, la précision de l'approximation dépend de la qualité du cahier des charges (complet, non-ambiguë...) et de l'analyse qui en est faite.

L'estimation paramétrique

Cette démarche permet d'estimer la charge globale d'un projet à partir des paramètres pertinents qui le structure. Par exemple, la méthode des points de fonction mesure la taille fonctionnelle d'un projet et est basée sur le dénombrement et la complexité des services rendus par un logiciel : saisies, mises à jour et restitution des données, possibilités d'interroger, d'importer et d'exporter des données...

Les étapes à suivre sont les suivantes :

1. définir le périmètre fonctionnel du logiciel,

2. identifier les composants (Groupe de Données Internes, Groupe de Données Externes, Entrées, Sorties, Interrogations) et peser les composants selon leur complexité ce qui donne un nombre de points de fonction bruts,

3. évaluer l'influence des caractéristiques générales du logiciel (p. ex. les performances attendues, l'ergonomie) ce qui donne un nombre de points de fonction nets,

4. transformer les points de fonction nets en charge de travail en appliquant une règle issue de la pratique (p. ex. 2 jours*homme par point de fonction net pour un petit projet et 4 pour un grand).

Les points de fonction peuvent être utilisés très tôt dans un projet en se basant sur un cahier des charge par exemple. Évidemment, la précision de l'approximation dépend de la qualité de ce dernier (complet, non-ambiguë...) et de l'analyse qui en est faite. Il faut enfin noter que cette méthode n'est plus applicable aux projets dont la taille fonctionnelle est inférieure à 100 points de fonction [31].

L'estimation probabiliste

Cette démarche permet d'estimer la charge globale d'un projet ou d'une activité quand règne une forte incertitude sur leur déroulement.

Par exemple, la méthode d'estimation à trois points se déroule de la façon suivante :

3. on réalise trois estimations de charge : optimiste, pessimiste et vraisemblable,

4. on calcule la moyenne, qui peut être pondérée, de ces trois estimations pour obtenir la charge finale.

Cette démarche se rapproche assez nettement de la méthode DELPHI.

Il existe encore bien d'autres méthodes (p. ex. COCOMO[19], Monte-Carlo[20]) mais dans le cadre de la démarche projet du pôle Études, nous sommes essentiellement amenés à utiliser les méthodes décrites précédemment. Par exemple, quand je dois estimer les coûts et les délais d'un projet avant de lancer un appel d'offres, je m'appuie sur la méthode des points de fonctions associée à la répartition proportionnelle car je dispose alors du cahier des charges. Une fois la charge du projet connue en fonction des différents intervenants de la MOE (chef de projet, architecte concepteur, développeur, formateur), il suffit de multiplier le nombre de jours*homme par profil pour obtenir le coût total approximatif du projet (p. ex. 1 jour*homme pour un profil développeur est en général facturé entre 500 et 600 € HT).Par contre, quand il s'agit d'estimer la charge et la durée d'une activité spécifique, je m'appuie plutôt sur une combinaison de la méthode DELPHI et des estimations analytique ou à trois points.

Concernant le calcul de la durée du projet, je me base sur les contraintes internes (p. ex. les rentrées scolaires, la disponibilité des acteurs métier...) et sur l'expérience acquise sur des projets similaires, bien qu'il existe des formules empiriques permettant de déduire la durée de la charge (p. ex. 0,5 fois la racine cubique de la charge en homme*année [31]).

Enfin, les délais sont systématiquement traduit par un diagramme de Gantt, outil graphique qui permet d'ordonnancer les tâches d'un projet en fonction du temps. Pour ma part, j'utilise GanttProject, qui est alternative intéressante à Microsoft Project, pour réaliser et gérer ce type planning.

19 Méthode d'estimation paramétrique basée sur la connaissance *a priori* du nombre de milliers d'instructions sources du logiciel à développer.

20 Méthode d'estimation probabiliste basée sur la connaissance de la loi de distribution de la charge d'une activité.

4.1 La dimension humaine d'un projet

4.1.1 Définitions et enjeux

La réussite d'un projet dépend essentiellement des dimensions technique et humaine dans des proportions égales. C'est un adage bien connu des chefs de projet. Dans le cadre de ses activités, le chef de projet Études est amené à piloter, coordonner, contrôler et participer à la réalisation d'une variété de tâches techniques et administratives impliquant des acteurs internes (agents) et externes (prestataires, partenaires) sur lesquels il n'a aucun pouvoir hiérarchique. Dans ce contexte, les outils et les pratiques du management transversal et de la négociation sont les principales armes du chef de projet Études.

Par ailleurs, la mise en place d'un nouveau SI provoque un changement qui correspond au passage d'une situation existante vers une situation future, et ce changement impacte inévitablement les différents acteurs du projet et l'organisation du travail (outils utilisés, façons de travailler, responsabilités). En conséquence, le chef de projet Études doit faciliter ce passage : c'est l'objectif de la conduite du changement.

4.1.2 Spécificités du management transversal

Pour faire face à son manque d'autorité hiérarchique sur les membres de l'équipe projet, le chef de projet Études met en œuvre un certain nombre de techniques de management pour faire progresser régulièrement son projet et le faire aboutir dans le respect des objectifs fixés (coût, délais, satisfaction des utilisateurs...). Toutefois, l'efficacité de ces techniques dépend en partie du chef de projet (personnalité, expérience...), du contexte du projet (sponsor du projet...) et de sa capacité à influencer les différents acteurs.

Adapter son style de management

Une des missions du chef de projet Études est de coordonner et de contrôler la réalisation des tâches du projet confiées aux différents acteurs. En interne, le spectre des agents concernés est très large. Il est en relation avec des directeurs, des chefs de pôles, des ingénieurs, des techniciens et des agents administratifs. En externe, il interagit avec ses fournisseurs et autres partenaires de la RCA. Par conséquent, il doit nécessairement adapter son style de management et négocier (moyens, délais, résultats...), voire contractualiser, avec son interlocuteur.

Outre le type d'interlocuteur, le chef de projet Études choisit également son style en fonction d'un certain nombre de paramètres comme :

- l'impact d'une décision ou d'une solution sur le projet,
- l'acceptation potentielle de cette décision/solution et l'effet prévu en cas de non adhésion,
- son niveau d'information, une contrainte forte sur les délais, etc.

Le tableau ci-après donne un aperçu des différents styles de management du Chef de Projet (CdP) en fonction des situations rencontrées.

Style	Description	Cas d'application et limites
Directif	Le CdP décide seul	+Contraintes forte sur les délais +Situation de crise +Acteurs peu expérimentés -Manque d'information du CdP
Consultatif	Le CdP prend des avis et décide seul	+Manque d'information du CdP -Risque de conflit
Participatif	La décision est collégiale	+Acteurs expérimentés qui adhérent au projet -Acteurs peu expérimentés et/ou qui n'adhèrent pas franchement au projet
Persuasif	Le CdP cherche à convaincre en explicitant les tenants et les aboutissants	+Expérience des acteurs hétérogènes -Risque : les arguments ne sont pas entendus
Par délégation	Le CdP délègue des tâches, fournit les moyens et contrôle l'avancement et les résultats	+Acteurs expérimentés qui adhérent au projet -Acteurs peu expérimentés et/ou qui n'adhèrent pas franchement au projet

Gérer les conflits

Les projets sans conflits sont rares, autre adage bien connu. Pour y faire face, la littérature regorge de techniques et autres méthodes. A titre d'exemple, je mets en pratique des méthodes simples qui ont fait leur preuve : DESC pour gérer un conflit et CRAC pour traiter les objections.

La méthode DESC consiste à :

- Décrire des faits concrets et observables de la situation de conflit,
- Exprimer ses émotions, ses sentiments par rapport à la situation et les problèmes causés,
- Solutionner les problèmes,
- Conclure (p. ex. en définissant un plan d'actions à court terme).

La méthode CRAC consiste à :

- Creuser pour comprendre les raisons et les motivations profondes de son interlocuteur,
- Reformuler l'objection de manière positive pour faire apparaître les besoins sous-jacents,
- Argumenter face à l'objection,
- Conclure.

En plus de ces méthodes, un certain nombre de comportements types contribuent à résoudre une situation de conflit. Le tableau ci-après donne un aperçu des différents comportements pouvant être adoptés par le CdP en fonction des situations rencontrées.

Style	Description	Cas d'application et limites
Le retrait	Le CdP se retire du conflit pour attendre que les choses se calment	+Stratégie à court terme +Conflit de faible intensité qui n'a pas un impact important sur le projet
L'apaisement	Le CdP rencontre son interlocuteur pour minimiser les oppositions et mettre en valeur les rapprochements Le CdP mise sur les effets du temps pour relativiser les oppositions	+Stratégie temporaire +Efficace si l'interlocuteur ne souhaite pas aborder le conflit de front
La force	Le CdP utilise une position de pouvoir : la sienne ou celle d'un tiers (sponsor projet, hiérarchie...) favorable pour imposer son point de vue et la solution qu'il préconise	+Arbitrage d'une tierce autorité -Stratégie ayant des conséquences potentiellement négatives sur le long terme
Le compromis	Le CdP entame une négociation avec son interlocuteur : chacun cède partiellement pour atteindre une solution acceptable	+Intervention d'un médiateur -Stratégie ne menant pas toujours à le meilleure solution quand il s'agit d'option technique ou d'exigences de qualité
La collaboration	Le CdP reconnaît la valeur de la position de son interlocuteur	+Efficace s'il ne s'agit pas d'un conflit de personne -Dangereux si le temps est restreint

Influencer pour obtenir des résultats

Chaque acteur poursuit les objectifs métier de son entité organisationnelle qui ne sont pas forcément ceux du projet, voire ceux de la RCA. Par exemple, le chef de projet MOE visera avant tout la satisfaction client et le respect de ses coûts internes (notamment dans le cadre de prestations forfaitaires) tandis que le chef de projet MOA s'attachera à la satisfaction des utilisateurs finaux et au respect des délais. Dans ce contexte, le chef de projet Études doit vérifier un certain nombre de conditions pour mobiliser efficacement un acteur du projet et éviter son acquiescement, voire sa résistance.

Elle sont principalement liées à :

- la capacité de l'acteur à accomplir une tâche (compétences, moyens nécessaires, délais...),
- son intérêt et ses motivations personnelles (recherche d'autonomie, de reconnaissance, de pouvoir...),
- la crédibilité du chef de projet Études,
- la perception du coût et du risque par l'acteur (s'il est élevé il paralysera son action).

4.1.3 La conduite du changement

Le but de la conduite du changement consiste à faciliter le passage d'une situation actuelle vers une situation future en minimisant les impacts sur le projet (coût, qualité, délais, résistances, rejet...), et en facilitant l'acceptation des changements provoqués par la mise en œuvre de nouveaux outils informatiques [33].

Les facteurs de rejet

Les principaux facteurs de rejet liés à la mise en place de nouveaux outils informatiques peuvent être appréhendés selon les aspects humains, socio-culturels liés à l'usage des Technologies de l'Information et de la Communication (TIC) et financiers.

L'utilisateur final est concerné en premier lieu car (aspect humain) :

- il va devoir utiliser les nouveaux outils,
- ses processus métier et les activités associées, ses habitudes de travail, son environnement professionnel sont modifiés, voire remis en cause, ce qui nécessite une capacité d'adaptation importante dans des délais très court.

Le chef de projet Études doit particulièrement faire attention à la résistance au changement qui se traduit par :

- une cristallisation des appréhensions sur les inconvénients liés au changement imposé sans pour autant voir les avantages de la nouvelle situation (Quel est l'intérêt de changer ?),
- un manque de visibilité sur la situation future qui encourage ce type comportement.

De plus, la démocratisation des TIC dans l'entreprise est relativement récente, de ce fait, nous ne sommes pas tous égaux devant l'outil informatique, notamment en matière d'expérience, d'acquis fonctionnel et technique.

Du point de vue financier, les changements (politique des élus, cadre réglementaire...) peuvent avoir un impact indirect conséquent pour les partenaires institutionnels (p. ex. les administrations, les établissements publics) et privés (p. ex. ses fournisseurs, ses usagers) de la RCA. En effet, la mise à niveau de leurs propres outils informatiques est souvent nécessaire pour assurer la continuité de service. Par exemple, l'obligation légale d'internaliser le paiement des primes aux employeurs d'apprentis, jusque là externalisé, a obligé notre fournisseur à mettre en place une interface d'export (format imposé par le SI financier de la RCA) pour nous fournir les données nécessaires au paiement.

Le processus de changement

Le schéma ci-après illustre le processus du changement qui traduit concrètement les différents facteurs de rejet. Ce schéma est inspiré du processus du deuil (déni, colère, marchandage, découragement, acceptation).

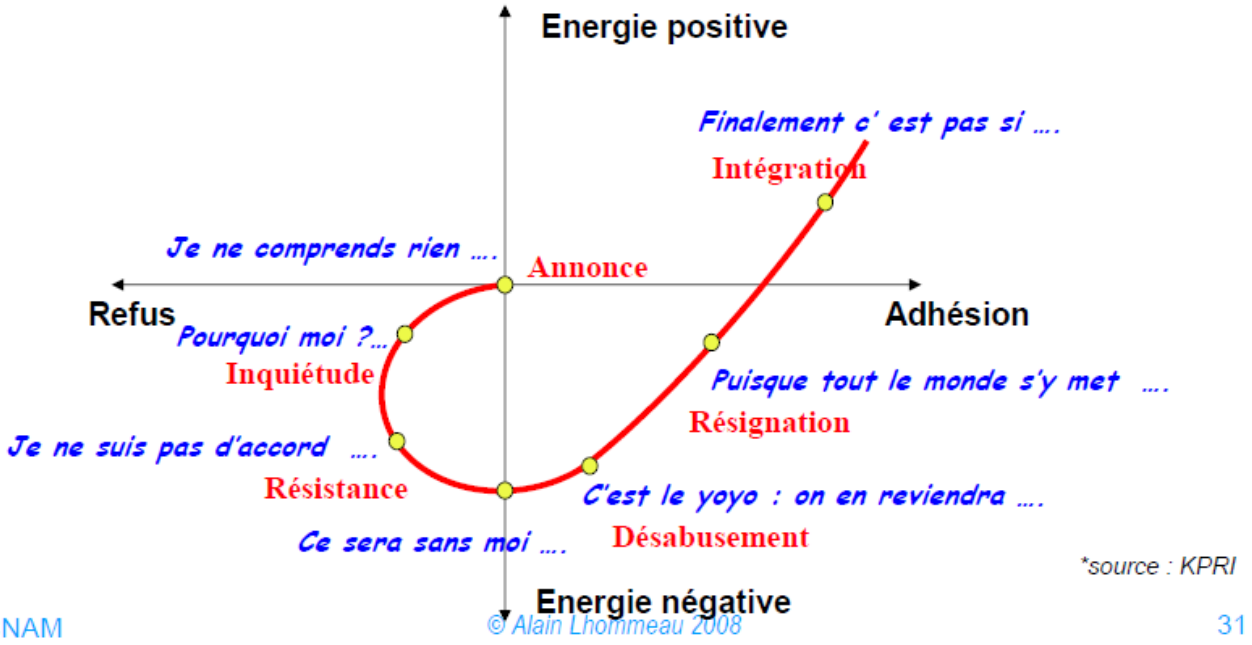


Illustration 8: Le processus du changement d'après [33]

La stratégie du projet latéral

Pour faire face au processus de changement, il faut faire le constat de départ suivant :

- les problèmes rencontrés sont dus aux difficultés relationnelles plus qu'à la technique
- il faut travailler avec des acteurs opposés au projet
- tous les projets génèrent une opposition parmi les acteurs
- il faut valoriser les acteurs moteur du projet

Une fois ce constat posé, la stratégie de conduite du changement repose alors sur quatre grands principes [33] :

- recenser tous les acteurs concernés par le projet de changement et identifier les alliés (ils déploient de l'énergie pour le projet), les opposants (ils déploient de l'énergie contre le projet) et les acteurs neutres (ils déploient peu d'énergie pour le projet se traduisant par un désintérêt ou du suivisme),
- définir avec les alliés un projet qui recueille leur intérêt (le projet latéral) et les encourager à convaincre les indécis,
- adapter au fur et à mesure le projet pour gagner une majorité à la cause (c'est « latéraliser » le projet),
- désamorcer les attaques des opposants sans entrer dans un conflit ouvert.

Dans ce sens, [33] propose une une démarche éprouvée basée sur la communication, l'anticipation des risques, la formation et l'accompagnement des acteurs (se concentrer sur les acteurs neutres et les alliés). Le tableau ci-après résume cette démarche vue par le chef de projet Études dans le contexte de la RCA.

Axe	**Description des actions**
La communication : vise la réduction des difficultés liées aux aspects humains	• Impliquer l'ensemble des acteurs du projet par une communication adaptée (prise en compte des avis, des besoins, des craintes…) • Faire comprendre aux acteurs les raisons et les facteurs du changement provoqués par le projet pour qu'ils puissent se l'approprier • Expliciter les enjeux, les caractéristiques politiques, économiques, sociales, techniques, environnementales, et légales ainsi que les objectifs de coûts, qualité et délais associées au projet • Informer régulièrement les acteurs (état d'avancement du projet) : c'est indispensable pour maintenir la motivation et l'implication de tous les acteurs
L'anticipation des risques permet de diminuer leur impact négatif sur le projet (actions de prévention)	• Ils doivent être identifiés, analysés et hiérarchisés selon leur criticité afin de prévoir les leviers d'action appropriés
L'accompagnement opérationnel des acteurs vise la réduction des problèmes liés aux aspects socio-culturels	• Formations en adéquation avec leurs besoins • Actions d'assistance au démarrage

Dans cette deuxième partie, nous avons vu qu'une démarche réfléchie était nécessaire à la réussite d'un projet SI. D'autant plus, qu'un projet des Études implique des difficultés spécifiques liées à l'organisation et l'écosystème des projets de la RCA (acteurs nombreux, diverses entités organisationnelles, chaîne de décision complexe) et au cadre réglementaire imposée au collectivités territoriales (codes marchés publics...).

Dans ce contexte, le chef de projet Études est responsable des activités d'ingénierie des besoins, de planification et d'estimation des charges dans le respect des règles de l'art adaptées selon ses besoins. De même, en tant que gestionnaire, il manage les acteurs de son projet et conduit le changement en fonction de ses interlocuteurs.

La troisième partie du présent mémoire se propose de zoomer sur une des activités majeure du chef de projet Études, à savoir la recette fonctionnelle. Tout d'abord, je présenterai ce qu'est le test logiciel et les différentes techniques associées ainsi que leurs limites.

Puis, j'aborderai les enjeux de l'outillage des tests, les principaux outils et les limites de l'automatisation. Enfin, j'aborderai les tests logiciels pratiqués dans le cadre de la recette fonctionnelle.

5 Le test logiciel appliqué à la recette fonctionnelle

Pour être efficace et contribuer à la qualité d'un SI, les tests entrepris doivent être effectués tout au long du processus de développement du SI et pas seulement après sa livraison. En conséquence, les activités de test doivent être dûment réparties entre la MOE et la MOA [5], de l'ingénierie des besoins à la recette fonctionnelle du SI en passant par la production des logiciels.

5.1 Qu'est-ce que le test ?

Le test logiciel peut être défini de plusieurs manières. D'un point de vue métier, il s'agit d'exécuter le logiciel dans un contexte normal d'utilisation pour vérifier le respect des besoins exprimés [2]. D'un point de vue plus technique, le test peut être considéré comme *« une activité destinée à déterminer si l'évaluation d'une caractéristique ou d'une aptitude d'un programme ou d'un système donne les résultats requis »* [5] ou bien encore comme *« l'action d'étudier et de comprendre le niveau des avantages et du danger liés à la livraison d'un système logiciel »* [5].

De ces définitions, il ressort que le test doit permettre d'une part, de vérifier la conformité du logiciel aux exigences spécifiées et d'autre part, de détecter les anomalies compromettant le logiciel ou la possibilité de l'utiliser (c-à-d. de le mettre en production).

Un processus de test simplifié peut être résumé par le schéma ci-après.

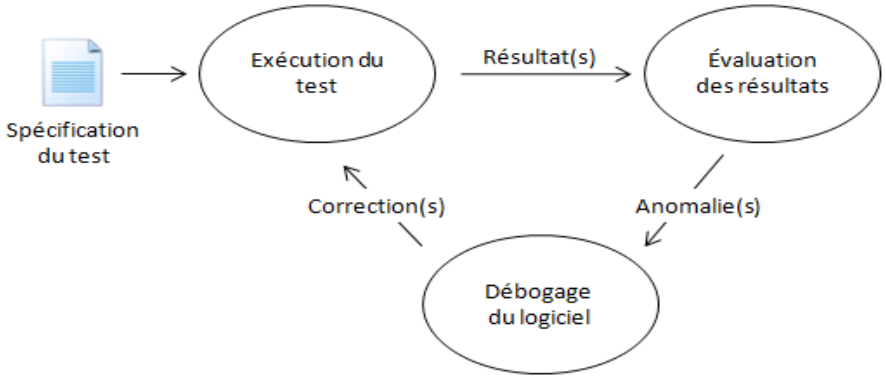


Illustration 9: Processus de test élémentaire

Ce schéma montre les principales tâches du test, à savoir préparer et exécuter un test couvrant une ou plusieurs exigences à vérifier, évaluer les résultats en comparaison avec ce qui était attendu, détecter des anomalies et procéder à leur débogage pour ensuite corriger le logiciel. Une nouvelle exécution du test permettra de valider la correction effective des anomalies.

Le tableau ci-après détaille les tâches réalisées par le testeur ou le développeur du logiciel en fonction des étapes du processus simplifié de test.

Étape	Tâches effectuées par le testeur	Acteur
Préparation du test	Spécifier le test, mettre en place l'environnement de test et planifier le test.	Testeur
Exécution du test	Dérouler le test et enregistrer les résultats obtenus.	Testeur
Évaluation des résultats	Comparer les résultats enregistrés avec les résultats attendus (par rapport au comportement normal du logiciel).	Testeur
Débogage du logiciel	Trouver, analyser et éliminer les causes des anomalies dans le logiciel.	Développeur
Correction du logiciel	Corriger le logiciel et livrer une nouvelle version au testeur.	Développeur

Pour obtenir des tests objectifs, une bonne pratique consiste à confier les rôles de testeur et de développeur à des personnes différentes. D'une manière générale, il est conseillé de mettre en place des équipes de développement et de test du logiciel indépendantes et bénéficiant de leurs propres ressources, notamment pour éviter les conflits d'intérêts.

5.2 Les principales techniques de test

Les techniques de test permettent d'élaborer des cas de test valables, c'est-à-dire un ensemble de tests spécifiques et pertinents qui visent à vérifier un aspect ou un besoin particulier du logiciel : c'est l'objectif du test. Il existe plusieurs types de tests plus ou moins complémentaires dont l'utilité et l'usage dépendent de l'objectif poursuivi. La typologie de ces différentes techniques de test et les définitions correspondantes sont données ci-après.

5.2.1 Choisir des données de test

Les techniques suivantes permettent de réduire l'explosion combinatoire inhérente aux tests en tentant de maîtriser les différentes combinaisons possibles des données en entrée du logiciel.

Partitionnement par classes d'équivalence

Le partitionnement du domaine de variation des données en entrée du logiciel en classes d'équivalence constitue le pivot des techniques de tests. Une classe d'équivalence regroupe un ensemble de valeurs de données en entrée du logiciel qui déclenchent le même comportement observable (c-à-d. le même résultat attendu). De même, les données invalides doivent également être prise en compte par l'analyse partitionnelle pour tester les comportements non-passants (c-à-d. qui provoquent des erreurs attendues).

Dans un premier temps, il s'agit de répartir les entrées du logiciel dans des catégories cohérentes (p. ex. à partir des règles de gestion métier), appelées classes, fondées sur le traitement identique par le logiciel de tous les éléments d'une même classe. Ainsi, le résultat du test d'un élément d'une classe vaut pour tous les autres éléments de cette classe.

Cette technique permet de réduire un espace de données illimité ou de très grande taille, due aux différentes combinaisons possibles des données en entrée du logiciel, en un nombre limité de combinaisons intéressantes. En pratique, cela permet de construire des bases de données de test de taille réduite car il n'y a que quelques représentants significatifs par classe d'équivalence (voir l'exemple du §6.2 en annexe).

Test aux limites

Ce test complète le partitionnement par classe d'équivalence. Il s'agit de retenir les valeurs extrêmes et aux bornes de chaque classe d'équivalence (le plus près possible de la borne), l'expérience montrant que les erreurs se situent souvent à la frontière de comportements différents du logiciel [9] (voir l'exemple du §6.2 en annexe).

Test combinatoire

Le test par combinaisons de valeurs de données en entrée du logiciel permet de maîtriser « l'explosion combinatoire » entre les valeurs des différentes classes d'équivalence (paires, triplets de valeurs...) dans la construction des données de test. Par exemple, le **test par paires** (pairwise testing en anglais) est la technique la plus simple. Il s'agit de sélectionner les données de test pertinentes[21] afin de couvrir toutes les paires de valeurs possibles, car l'expérience montre que la majorité des erreurs sont détectées par des combinaisons de deux valeurs de variables [2] (voir l'exemple du §6.3 en annexe). La limite de cette technique réside dans les interactions et dépendance entre les valeurs des différentes classes d'équivalence (p. ex. deux valeurs peuvent être contradictoires et donc à ne pas tester).

Test aléatoire

Ce test est basé sur des générateurs capables de créer automatiquement des valeurs valides d'entrées du logiciel de manière aléatoire. Les sorties étant enregistrées puis analysées par le testeur. Ce type de test est un bon complément aux autres types de test [5]. Cependant, il est consommateur de charge de travail car l'analyse des entrées/sorties doit être effectuée par un opérateur humain.

5.2.2 Tester la structure du logiciel

Le **test de type « boîte blanche »** permet de détecter les erreurs structurelles d'un logiciel en analysant sa structure interne. Pour cela, l'équipe de développement doit réaliser des revues de code pour vérifier les instructions, les branches de décision, les conditions et les différents chemins possibles...en s'appuyant sur des outils d'analyse syntaxiques ou de mesure de la complexité. Concrètement, le développeur s'appuie sur sa connaissance de la structure d'une portion de code (p. ex. une boucle si-alors-sinon) pour constituer le cas de test et son jeu de données.

5.2.3 Tester le comportement du logiciel

Le **test de type « boîte noire»** est utile lorsque le testeur n'a pas de connaissance sur la structure interne du logiciel. Dans ce cas, le testeur doit s'appuyer sur la description du comportement externe du logiciel pour concevoir les cas de test.

21 Un certain nombre de logiciels permettent d'effectuer ce choix, http://www.pairwise.org/tools.asp.

Le testeur exploite essentiellement le cahier des charges, le dossier de spécifications, voire le manuel utilisateur.

Le **test de transition d'état** s'appuie sur une représentation des différents états possibles (valides et invalides) du logiciel. Les cas de test et les données de test associés sont conçus à partir des spécifications des exigences pour provoquer des transitions d'état du logiciel (passage d'un état à un autre). Ce test est particulièrement utile en complément du test par affirmation (cf. §3.2.5) et permet de soulever certains défauts de spécifications (p. ex. un comportement insuffisamment décrit).

5.2.4 Tester selon une logique métier

Le **test de navigation ou d'exécution** vise l'évaluation d'une fonctionnalité ou d'une logique métier de bout en bout en simulant un utilisateur dans des conditions normales de travail. Pour concevoir les cas de test, le testeur s'appuie principalement sur la modélisation des processus métier, les cas d'utilisation et les règles de gestion décrits par le cahier des charges.

5.2.5 Vérifier les exigences spécifiées

Le **test par affirmation** sert à vérifier la conformité d'un logiciel aux exigences spécifiées tandis que le **test par négation** consiste à vérifier son comportement en dehors de la stricte portée des exigences spécifiées. Ces deux tests sont complémentaires dans le sens où le test par négation est une réponse plutôt satisfaisante au problème d'incomplétude des exigences. Cependant, il ne permet pas de maîtriser convenablement la charge de travail car, par définition, aucune limite n'est fixée pour ce type de test. Dans les deux cas, le testeur conçoit les cas de test à partir du dossier de spécifications. Les exigences clairement spécifiées sont la base des tests par affirmation, tandis que les exigences mal décrites ou manquantes (logiciel hérité) sont le point de départ des tests par négation.

5.2.6 Améliorer l'efficacité des tests

Le **test de prédiction d'erreurs** n'est pas un test en soit mais peut être appliqué aux autres tests pour améliorer leur capacité à détecter des anomalies. Cette technique s'appuie sur des informations capables d'orienter les tests comme par exemple :

- l'expérience acquise dans le test de logiciels similaires,
- les résultats de tests des étapes précédentes,
- la connaissance des erreurs types d'implémentation.

5.2.7 Diminuer la charge de test par automatisation

Un moyen de diminuer significativement la charge de travail liée aux tests consiste à automatiser une partie des tâches du test, principalement les tâches répétitives.

Le **test automatisé** implique l'utilisation de logiciels dédiés *« pour contrôler l'exécution des tests, comparer les résultats obtenus aux résultats attendus, mettre en place les pré-conditions de tests, et d'autres fonctions de contrôle et de reporting sur les tests »* [42]. Ce type de test est très efficace quand il s'agit d'effectuer des tests de non-régression[22] ou de performances (cf. chapitre ci-après). Par opposition, le test manuel est réalisé par le testeur en simulant le comportement normal d'un utilisateur via l'interface graphique du logiciel. Ce type de test est fastidieux et est très sensible aux erreurs humaines [9].

5.2.8 Tester les caractéristiques non-fonctionnelles

Les techniques de test non-fonctionnel s'attachent aux attributs du logiciel qui ne sont pas liés aux fonctionnalités (performance, convivialité...) [42]. De la même manière, il existe plusieurs familles de test plus ou moins complémentaires dont l'utilité dépend de l'objectif poursuivi.

Test de configuration et d'installation

Ce test permet de s'assurer de l'installation correcte du matériel (serveur...), du logiciel (fichiers, paramétrage des valeurs par défaut...), des fichiers de données (bases de données test/production) et des interfaces avec les autres systèmes de l'environnement de production[23] (connexions réseau...). Le testeur réalise les cas de test en s'appuyant sur la documentation d'installation et de paramétrage du logiciel. Cela peut se traduire simplement par une « checklist » des composants installés.

Test de compatibilité/interopérabilité

Le test de compatibilité permet de vérifier l'absence de répercussion néfaste du logiciel exécuté dans son environnement de production sur les autres systèmes et réciproquement. Le test d'interopérabilité consiste à contrôler les communications entre le logiciel et les autres systèmes de l'environnement de production (échanges de données, appels à des services web...). Le testeur réalise les cas de test en s'appuyant sur la documentation spécifiant les contrats d'interfaces, les interactions entre les systèmes...

Test de documentation

Ce test permet de vérifier la conformité de la documentation à destination des utilisateurs et du système d'aide proposé par le logiciel par rapport aux exigences spécifiées. Cela passe par l'étude de la documentation (niveau d'actualité des documents, vérification de la cohérence de la table des matières, de l'index...) et par la simulation d'utilisateurs en situation de recherche d'aide.

Test de recouvrement sur panne

Ce test permet de vérifier la restauration à l'état normal du logiciel et son bon fonctionnement suite à l'apparition d'une erreur ou d'une exception impliquant un « crash système ». Le testeur peut ,par exemple, concevoir un cas de test sur la base d'une coupure de courant...

22 Permet de vérifier le bon fonctionnement du logiciel après avoir été modifié ou étendu (modification de l'environnement de production, améliorations ou évolutions fonctionnelles...).

23 L'environnement technique où a été installé le logiciel avant sa mise à disposition des utilisateurs.

Test de performance

Ce test est basé sur un modèle des performances attendues et nécessite des outils spécialisés comme par exemple un injecteur de charge couplé à des sondes mesurant les réactions du logiciel testé (utilisation de la mémoire, des processeurs, des disques ou du réseau...). Un injecteur de charge (p. ex. *OpenSTA*[24] ou *Rational Test Performer*[25]) est un logiciel capable de simuler les actions d'un grand nombre d'utilisateurs et de ce fait, générer une charge importante pour le logiciel testé.

Test de sécurité

Ce test vise à s'assurer que les exigences de sécurité (p. ex. celles imposées par le RGS[26]) ont bien été prises en compte dans le développement du logiciel. Typiquement, il s'agit de vérifier la disponibilité, l'intégrité, la confidentialité et la traçabilité des données manipulées par le logiciel.

Test de stress (ou de charge)

Ce test cherche à examiner la capacité du logiciel à fonctionner correctement lors de pics de charge de travail. Ce type de test nécessite des outils spécialisés (cf. test de performance ci-dessus).

Test de convivialité

Ce test cherche à vérifier la conformité de l'interface graphique du logiciel aux normes reconnues dans ce domaine (p. ex. la charte ergonomique des sites internet de l'administration électronique[27] ou la norme ISO 9241-110 [49]). Ce type de test subjectif implique des test manuels effectués par des représentants des futurs utilisateurs.

Test de volume (ou de flux)

Ce test évalue la capacité du logiciel à fonctionner correctement avec une quantité de données très importante dans le but d'identifier les anomalies qui n'apparaissent que dans ces conditions.

5.3 Les limites et les difficultés du test

Les différents techniques exposées dans les chapitres précédents permettent d'orienter et de cadrer le travail du testeur. Néanmoins, celles-ci ne garantissent pas à elles seules la qualité des tests effectués car, certaines techniques sont limitées par la combinatoire (pairwise testing) et d'autres nécessitent des tests longs, fastidieux (test de non-régression) qui, la plupart du temps, demandent des actions manuelles (test de convivialité). De plus, il faut ajouter à cela les problèmes liés à la dimension humaine du test, comme une expression des besoins incomplète, voire instable (des besoins qui évoluent constamment au cours du développement du logiciel, voire en recette), une mauvaise compréhension du cahier des charges, des erreurs de conception et de programmation, etc.

24 *OpenSTA* est une solution d'injection de charge HTTP/HTTPS qui permet de tester les performances d'une application web développée en J2EE, .NET, PHP, etc.

25 *Rational Test Performer* est un logiciel de tests de performance et de charge édité par IBM Rational.

26 Référentiel Général de Sécurité (RGS) [45] publié par les services de l'État : l'Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI, http://www.ssi.gouv.fr) et la Direction Générale de la Modernisation de l'État (DGME, http://references.modernisation.gouv.fr).

27 Publiée par la DGME [46].

Enfin, les connaissances et le niveau d'expérience du testeur dans l'utilisation des techniques et des technologies du test influencent également la qualité des tests effectués (pertinence des choix des données en entrée, erreurs humaines...). Quoiqu'il en soit, les techniques de test n'ont de véritable utilité que si elles s'inscrivent dans une stratégie de test ad hoc.

5.4 L'outillage des tests fonctionnels

5.4.1 Pourquoi automatiser les tests de recette ?

D'après [42], un test manuel correspond à une vérification qui est un *« test d'un logiciel ou de spécifications par une simulation manuelle de son exécution. ».* Par nature, le test manuel est une activité longue, fastidieuse, coûteuse (explosion de la charge pour les tests de non-régression) et difficilement reproductible dans des conditions identiques (deux personnes peuvent appliquer un cas de test de deux manières différentes). De plus, selon la plupart des spécialistes (p. ex. [2] et [5]), les activités du test représentent une charge de l'ordre de 15 à 30% du coût total d'un projet de développement logiciel, et environ 25 à 35% pour la mise en place d'un progiciel (intégration, paramétrage, développements spécifiques...) supportant plusieurs processus métier. Ces ratios augmentant en fonction de la criticité (la probabilité d'occurrence et le niveau d'impact d'une anomalie sur l'organisation) et de la complexité (technologie, nombre de règles de gestion métier...) du logiciel à tester.

Dans ce contexte, l'automatisation est aujourd'hui considérée comme un élément de réponse permettant de faire face à ces difficultés. Cette automatisation résidant dans l'*« utilisation de logiciels pour exécuter ou supporter des activités de tests, p. ex. gestion des tests, conception des tests, exécution des tests ou vérification des résultats. »* [42]. En effet, l'utilisation de logiciels permettrait ainsi d'accélérer les cycles de vérification et de validation, d'améliorer la productivité des tests et la qualité du logiciel, de libérer du temps pour des tâches à plus forte valeur ajoutée (p. ex. l'analyse des résultats) et de gérer efficacement les tests de non-régression par le réemploi de tests existants.

5.4.2 Les principaux outils de test pour la recette

Par définition, les tests effectués lors de la recette fonctionnelle englobent les tests des fonctionnalités et des services offerts par le logiciel. Dans ce cadre, des outils spécialisés permettent de définir, d'exécuter et de suivre des campagnes de tests (référentiel des tests) couvrant plusieurs exigences (référentiel des exigences) et d'y associer les anomalies détectées (via un gestionnaire d'anomalies).

Comme beaucoup de domaines de l'informatique, il y a les suites de logiciels d'éditeurs importants et les logiciels libres[28]*»,* par exemple (voir également les exemples au §6.4 en annexe) :

28 Un logiciel libre est un logiciel distribué avec l'intégralité de ses programmes-sources, afin que l'ensemble des utilisateurs qui l'emploient, puissent l'enrichir et le redistribuer à leur tour. Un logiciel libre n'est pas nécessairement gratuit et les droits de la chaîne des auteurs sont préservés (d'après http://www.marche-public.fr/Terminologie/Entrees/logiciel-libre.htm).

- HP Mercury (*HP Quality Center, TestDirector...*), Borland (*SilkCentral Manager, SilkTest...*) ou IBM Rational (*Rational Test Manager, Rational Functional Tester...*),

- *RTMR* (*Requirements and Tests Management Repository*) et *Salomé-TMF* (*Test Management Framework*) qui peuvent s'interfacer avec des gestionnaires d'anomalies comme *Mantis* ou *BugZilla.*

Concernant l'automatisation des tests de la recette fonctionnelle, il est possible d'utiliser des robots ou automates de test qui simulent les interactions entre un utilisateur et l'interface graphique du logiciel à tester. Les évènements du clavier et de la souris (un clic sur un bouton, la valorisation d'un champ de saisie, la sélection dans une liste de choix...) sont soit capturés par l'outil de test (en créant un script ad hoc) pour être rejouées automatiquement par la suite, soit reproduits par des programmes exécutables développés via un langage de script. Dans les deux cas, les résultats et les affichages du logiciel testé sont enregistrés par l'automate dans le but de comparer les résultats obtenus avec ceux initialement attendus. C'est la technique la plus utilisée dans le test fonctionnel des SI [2]. Néanmoins, l'efficacité des automates dépend essentiellement de la stabilité et des évolutions de l'interface graphique. C'est pourquoi, elle est typiquement utilisée pour automatiser les tests de non-régression.

5.4.3 Les limites de l'automatisation

L'automatisation des activités de test ne constitue qu'un élément de réponse à la problématique de maîtrise des coûts, de la qualité et des délais d'un projet de test logiciel. En effet, les logiciels de test ne se substituent pas entièrement aux tests manuels. Par exemple, le testeur doit toujours créer les cas de test[29] qui seront automatisés, éventuellement par un développeur qui devra alors maintenir des scripts de test dépendant du logiciel à tester (c-à-d. gérer les anomalies, les corrections et les évolutions). De même, l'activité d'analyse des résultats de test reste en très grande partie humaine.

Enfin, il faut être conscient du fait que l'automatisation ne permet pas de détecter plus d'anomalies d'une part, ce sont les cas de test qui le permettent, et d'autre part, elle ne se substitue ni aux utilisateurs et ni aux testeurs.

5.5 Le test logiciel dans le cadre de la recette fonctionnelle

5.5.1 Le cycle de vie d'un SI

D'une manière générale, le cycle de vie d'un SI peut être scindé en deux grandes phases, d'une part la phase de conception et de développement, et d'autre part, la phase de maintenance et d'évolution [32]. Dans cette optique, le « modèle de développement en V » constitue une approche traditionnelle[30] du développement et du test logiciel [5] comme le montre le schéma ci-après.

29 Il existe des techniques et des outils permettant de générer automatiquement des cas de test à partir à partir d'un modèle de test définissant l'ensemble des comportements attendus à tester sur le logiciel, voir [2]. Cette méthode permet de réduire la maintenance des cas de test à la seule maintenance du modèle de test.

30 Par opposition aux méthodes agiles (itératives et incrémentales) plus récentes et censées résoudre les insuffisances des méthodes traditionnelles de développements logiciel (SCRUM, XP...). D. WATKINS montre dans [5] qu'une itération de développement d'une méthode agile peut être ramenée à un cycle en V.

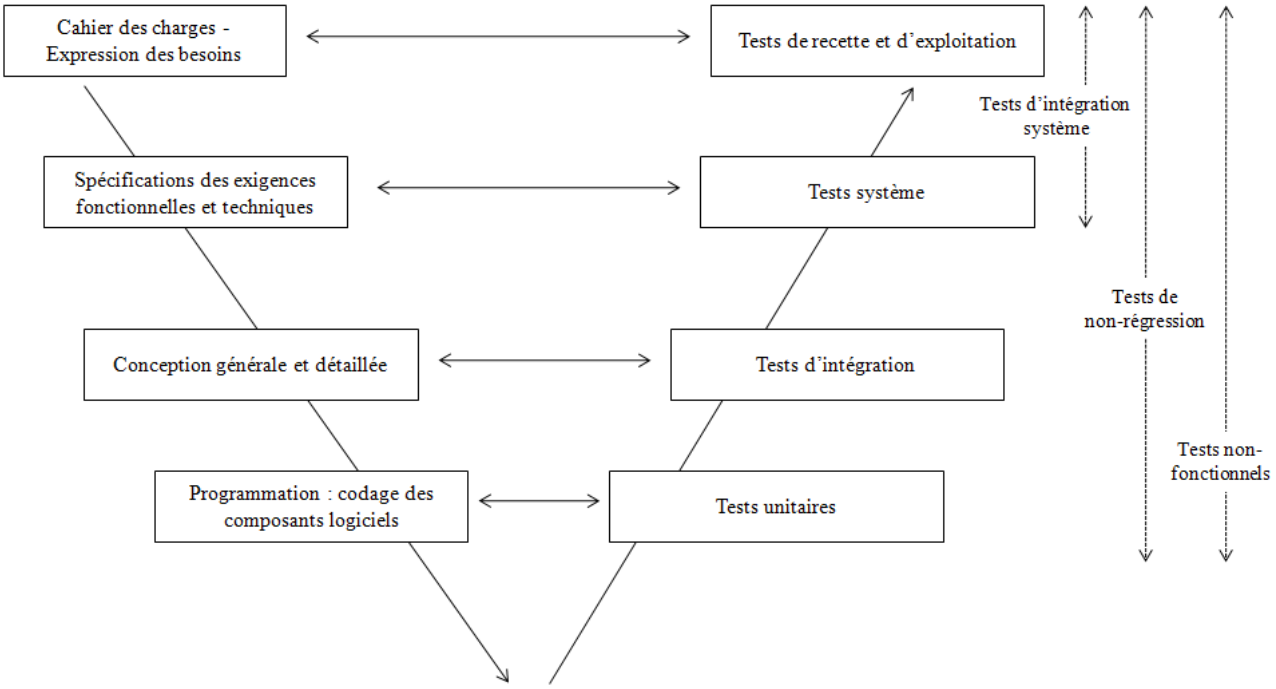


Illustration 10: Le modèle de développement en V

Selon la norme IEEE 610 [39], le modèle en V correspond à une *« une structure décrivant les activités du cycle de développement logiciel, depuis la spécification des exigences jusqu'à la maintenance. Le modèle en V illustre comment les activités de tests peuvent être intégrées dans chaque phase du cycle de développement. »*.

En général, les activités liées à la rédaction du cahier des charges et au test de recette sont prises en charge par la maîtrise d'ouvrage (MOA) qui représente le client. Les autres activités étant de la responsabilité de la maîtrise d'œuvre (MOE) choisie pour réaliser le SI (c-à-d le fournisseur), avec une implication plus ou moins importante de la MOA dans les activités de spécification, de conception et de test. Dans ce contexte, la MOA réalisera au minimum les tests de recette et d'exploitation ainsi que les test de non-régression. Les autres tests étant du ressort de la MOE, voire effectués conjointement (p. ex. les tests de performances).

5.5.2 Les tests de recette

Le test de recette (ou d'acceptation ou recette utilisateur) est effectué par la MOA et précisément par des représentants des utilisateurs et des experts métier, éventuellement assistés par la MOE. Ce test doit confirmer la réponse du logiciel aux exigences du cahier des charges et donner confiance dans le logiciel avant sa mise en production. Le représentant des utilisateurs teste le logiciel en reproduisant les tâches qu'un utilisateur accomplirait lors de l'utilisation normale du logiciel (p. ex. dérouler un processus de bout en bout).

Pour s'aider dans sa tâche, le testeur peut s'appuyer sur les techniques de test suivantes :

- tests de type boîte noire et d'exécution basés sur les exigences du cahier des charges,
- tests de convivialité pour vérifier l'interface graphique du logiciel (est-elle intuitive, cohérente, facile d'emploi...) et les fonctions d'aide en ligne (sont-elles satisfaisantes...).
- tests pour vérifier la documentation à destination des utilisateurs,
- tous les types de tests non-fonctionnels selon les objectifs de test poursuivis.

5.5.3 Les tests d'exploitation

Le test d'exploitation (ou d'acceptation par le service exploitation) est effectué par la MOA et précisément par des représentants des utilisateurs du service exploitation, éventuellement assistés par la MOE. Ce test doit confirmer la réponse du logiciel aux exigences d'exploitation du cahier des charges avant sa mise en production et son ouverture aux utilisateurs du service exploitation. Le représentant des utilisateurs teste le logiciel en reproduisant les tâches types qu'un utilisateur du service exploitation accomplirait lors de l'utilisation normale du logiciel.

Pour s'aider dans sa tâche, le testeur peut s'appuyer sur les techniques de test suivantes :

- tests de type boîte noire et d'exécution basés sur les exigences d'exploitation du cahier des charges,
- tests de convivialité de l'interface graphique,
- tests de la documentation d'exploitation du logiciel.

5.5.4 Les tests de non-régression

Le test de non-régression (ou de régression) est effectué par la MOA ou par la MOE en fonction du cycle de développement en V. Par exemple, les tests de non-régression peuvent s'appliquer aux tests unitaires ou de recette. Il permet de vérifier le bon fonctionnement du logiciel après avoir été modifié ou étendu (modification de l'environnement de production, amélioration ou évolution fonctionnelle...).

Pour s'aider dans sa tâche, le testeur peut s'appuyer sur les techniques de test suivantes :

- tests de type boîte noire basés sur la documentation existante et éventuellement sur la documentation des nouvelles exigences.

L'utilisation d'outils spécialisés qui permettent de simuler des actions d'un utilisateur sont particulièrement efficaces pour ce type de test (p. ex. *Selenium*[31] ou *HP Quality Center*[32]).

31 *Selenium* est une suite d'outils permettant d'effectuer des tests fonctionnels d'application web.
32 *HP Quality Center* est une suite de logiciels de tests éditée par HP Mercury.

Comme l'a montré cette troisième partie, le test permet d'une part, de vérifier la conformité du logiciel aux exigences spécifiées et d'autre part, de détecter les anomalies compromettant la possibilité de l'utiliser.

Dans ce contexte, la mise en œuvre d'un processus et de techniques spécifiques aux tests permet de contribuer à l'efficacité de l'étape de recette fonctionnelle en apportant des bonnes pratiques, des outils et des méthodes.

La quatrième partie du présent mémoire se propose de détailler les différents étapes du projet RFSI. Pour cela, j'exposerai les éléments de cadrage du projet et les travaux préliminaires qui ont mis en valeur la problématique et les difficultés rencontrées pendant l'étape de recette fonctionnelle d'un SI.

Enfin, je présenterai les activités d'ingénierie qui ont permis aboutir à la conception et à la réalisation d'une solution répondant à cette problématique et satisfaisant les besoins identifiés.

6 Une solution adaptée aux besoins de la Région

Dans le jargon de la DSI, le terme « solution » désigne un ensemble d'éléments hétérogènes qui répondent à une problématique donnée. Dans le cas du projet RFSI, il s'agit de trouver une solution permettant d'optimiser l'organisation et de simplifier l'étape de recette pendant la phase de réalisation des projets fonctionnels menés par le pôle Études. Dans la plupart des cas, une solution se concrétise par une organisation cible, de la documentation, des logiciels et des matériels mis en œuvre dans le cadre de prestations (installation, paramétrage, formation...) assurées par la MOE.

6.1 Étude du projet

6.1.1 Cadrage et travaux préliminaires

Une fois que l'intérêt du projet RFSI a été validé par le DSI en novembre 2010, un comité de projet, regroupant les chargés de projet et le responsable du pôle Études, a été mis en place en mars 2011 pour piloter le projet (cf. §1.3.1). Les premiers travaux réalisés ont permis d'établir un état de l'art concernant la recette fonctionnelle d'un SI, qui s'est matérialisé par une synthèse bibliographique [14] présentant le contexte, les enjeux et la problématique de la recette fonctionnelle ainsi que les solutions actuelles pour y faire face : processus, méthodes, logiciels, techniques de test, etc.

J'ai rédigé ce document entre janvier et février 2011. Ce travail a nécessité la recherche, le contrôle, la synthèse et la restitution des informations collectées sur internet et dans la littérature existante, en respectant les normes et les standards en la matière [51]. La synthèse bibliographique [14] a été présentée au comité de projet du 29 mars 2011. Dans le même temps, j'ai mené l'étude de cadrage qui comme son nom l'indique avait pour but de cadrer le projet en explicitant ses principales caractéristiques, à savoir :

- l'origine de la demande de projet,
- les faits, les difficultés rencontrées et leurs causes,
- les objectifs et les résultats attendus,
- le périmètre et les contraintes CQFD à respecter,
- les interactions avec les autres projets et les solutions envisageables aux problèmes posés.

Pour mener cette étude et rédiger le livrable correspondant (document interne standard de 9 pages), je me suis appuyé sur le contenu de la synthèse bibliographique et sur les résultats des entretiens individuels que j'ai eu avec mes collègues du pôle Études (4 réunions d'environ une heure). Ces entretiens ont mis en évidence un certain nombre difficultés rencontrées par les chargés de projet Études et nous avons travaillé sur leurs causes. A partir du constat établi, j'ai proposé des solutions pour éliminer, voire réduire dans certains cas, les effets de ces causes sur nos projets. Elles ont été ensuite discutées en comité de projet.

6.1.2 Causes des problèmes identifiés et solutions proposées

Les principales causes identifiées sont regroupées dans les quatre catégories suivantes :

1. Manque de formalisation et de contractualisation liées au processus de recette.
2. Activités de recette pas suffisamment rationalisées.
3. Faiblesses dans l'exécution des processus de test.
4. Insuffisance dans la maîtrise du processus de gestion des exigences.

Les solutions proposées s'articulent autour de trois principaux axes : l'organisation, les méthodes et les outils.

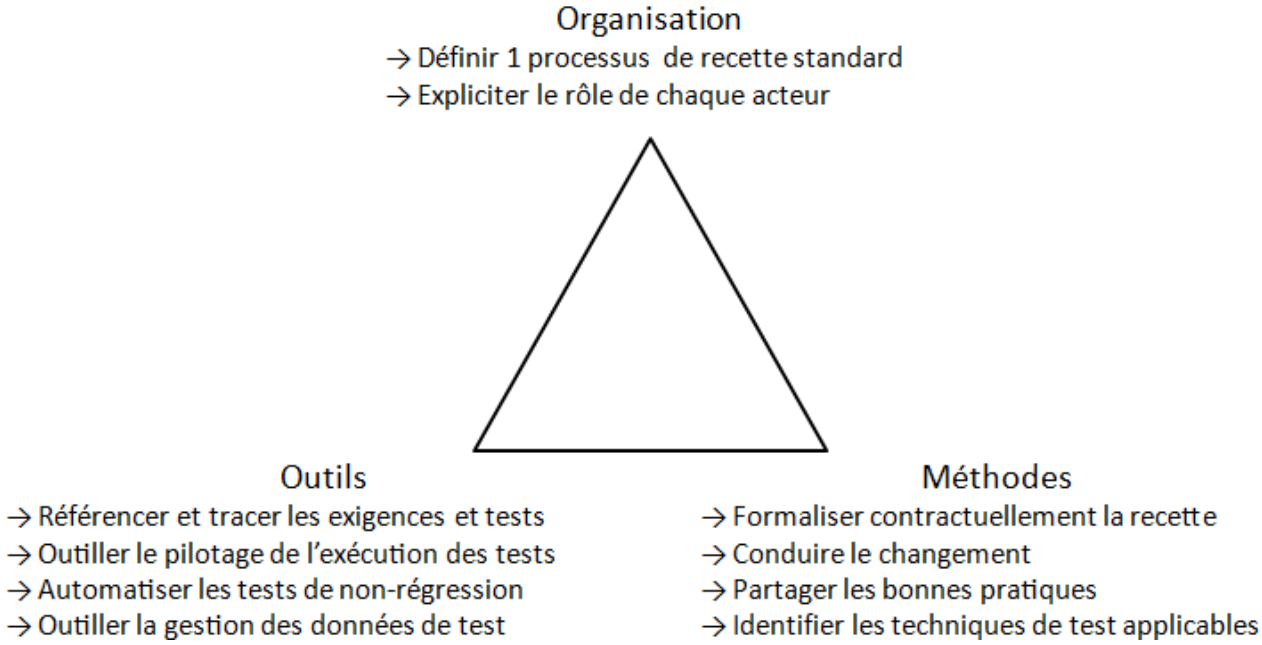


Illustration 11: Solutions proposées suite au cadrage du projet RFSI

L'organisation

Un processus de recette standard doit permettre de fixer un cadre de travail commun à chaque projet des Études en décrivant les étapes, les acteurs, les actions et les règles de gestion ainsi que les documents produits pendant l'étape de recette. Concrètement, ce processus de recette doit se présenter sous la forme d'un schéma BPMN[33] accompagné d'une description textuelle. Celle-ci comprenant l'organisation, les rôles et les responsabilités de chaque acteur pendant l'exécution du processus de recette sous la forme d'une grille RACI[34].

33 Business Process Modeling Notation (BPMN) est une notation graphique permettant de modéliser des processus d'entreprise. Elle a été standardisée en 2006 dans le cadre de l'Object Management Group (OMG).

34 Méthode qui permet d'identifier les rôles et responsabilités dans un processus selon 4 axes : l'acteur responsable

Les méthodes

Une fois arrêté, le processus de recette doit être intégré dans la documentation du projet sous la forme d'un document spécifique du cahier des charges. Ceci pour fixer les règles du jeu et les responsabilités respectives de la MOA et de la MOE. Concrètement, la contractualisation MOA/MOE doit être réalisée au travers d'un document spécifique appelé plan de test (cf. § 5.2.2), annexé au CCTP.

De plus, la professionnalisation des méthodes de test de la MOA passe principalement par l'identification des bonnes pratiques (p. ex. la mise en place d'un tableau de bord recette[35]) et des techniques de test du logiciel applicables au contexte de nos projets.

Enfin, comme tout projet informatique, une conduite du changement appropriée à l'étape de recette doit être définie par un ensemble d'actions de communication, de formation, d'accompagnement et d'assistance aux utilisateurs. Concrètement, cela se traduit par une liste d'actions à mettre en œuvre tout au long de la recette fonctionnelle de nos projets.

L'efficacité du processus de recette dépend en partie des logiciels permettant, d'une part de centraliser et de partager les documents (p. ex. le plan de tests, le tableau de bord recette) et les données liées à l'étape de recette (exigences, tests, anomalies), et d'autre part ils doivent permettre d'automatiser des tâches répétitives et fastidieuses, comme par exemple les tests de non-régression ou la génération automatique des données utiles pour réaliser un test.

6.1.3 Validation du projet

Le principal livrable à l'issue de ces travaux de cadrage est ce que nous appelons l'étude préalable du projet (document interne standard de 9 pages). Cette étude a été présentée et discutée par le comité de projet du 14 avril 2011. Elle a été définitivement validée, après intégration des diverses remarques, lors du comité de projet du 5 mai 2011. De fait, j'ai obtenu l'accord du comité de projet pour lancer l'étape suivante : l'étude détaillée du projet.

Lors de cette première étape de cadrage, j'ai consommé 18 jours*homme de charge de travail, la moitié étant dédiée à la recherche documentaire, l'analyse des documents collectés et la synthèse bibliographique correspondante. Le reste de la charge étant consacrée aux activités de gestion de projet (planification, suivi de l'avancement...) et à l'ingénierie des besoins (recensement et analyse des besoins, propositions d'orientations).

6.2 Étude détaillée et conception d'une solution

6.2.1 Méthodologie suivie

L'étude détaillée, comme son nom l'indique, a permis de préciser plus finement les différents aspects du projet, à savoir :

d'une activité, qui la réalise, qui est consulté ou informé avant ou pendant sa réalisation.

35 Le tableau de bord recette est un document qui regroupe un certain nombre d'indicateurs (p. ex. le nombre d'anomalies détectées) permettant de suivre le déroulement de la recette.

- le contexte, les enjeux, les objectifs et les acteurs du projet,
- l'analyse de l'existant (organisation, méthode, outils),
- la modélisation et la description des processus de recette cibles,
- le recensement et l'analyse des besoins,
- les différentes solutions envisageables pour satisfaire ces besoins.

La méthodologie appliquée durant cette étape était guidée par une « approche orientée processus ». Dans ce sens, 4 groupes de travail ont été identifiés à partir des conclusions de l'étude préalable :

- « processus de recette »,
- « outillage de l'exécution des processus de recette »,
- « pilotage de la recette »,
- « conduite du changement ».

Chaque groupe de travail a traité une problématique, ce qui a donné lieu à un certain nombre de réunions (environ une dizaine au total) qui ont permis d'identifier :

- Les processus de recette → Que-fait-on ? Qui le fait ? Quand ?
- Les besoins → De quoi a-t-on besoin pour le faire ?
- Les outils permettant d'automatiser une partie de ces processus → Comment le faire ?

Les différents acteurs ayant participé aux réunions pendant l'étude détaillée sont les suivants :

- les chefs de projet du pôle Études,
- les responsables des pôles Études, Exploitation informatique et Systèmes réseaux et sécurité,
- un chef de projet métier représentatif de la MOA.

Chaque acteur a participé à deux groupes de travail en fonction de ses préférences afin de répartir la charge de travail qui s'est ajoutée à leurs activités hebdomadaires. En contre-partie, j'ai assuré la cohérence de l'ensemble des travaux étant donné que les résultats d'un groupe de travail étaient utilisés dans le groupe de travail suivant.

6.2.2 Objectifs métier assignés au projet RFSI

La problématique qui s'est imposée à nous peut être résumée de la manière suivante : comment vérifier et valider la conformité du SI livré aux exigences spécifiées et détecter les anomalies compromettant le SI ou la possibilité de l'utiliser tout en maîtrisant les coûts et les délais initialement prévus ?

Comme nous l'avons vu tout au long des chapitres précédents, une façon de répondre à cette problématique consiste à définir l'organisation (processus recette), les méthodes (conduite du changement, technique de test) et les outils (logiciel de test) permettant de maîtriser l'étape de recette. Dans cette perspective, le but principal du projet a consisté à mettre en place les moyens permettant de vérifier, d'évaluer et de valider correctement un SI mis en œuvre dans le cadre d'un projet fonctionnel mené par le pôle Études.

Partant de ce constat, le diagramme ci-après récapitule les objectifs métier assignés au projet.

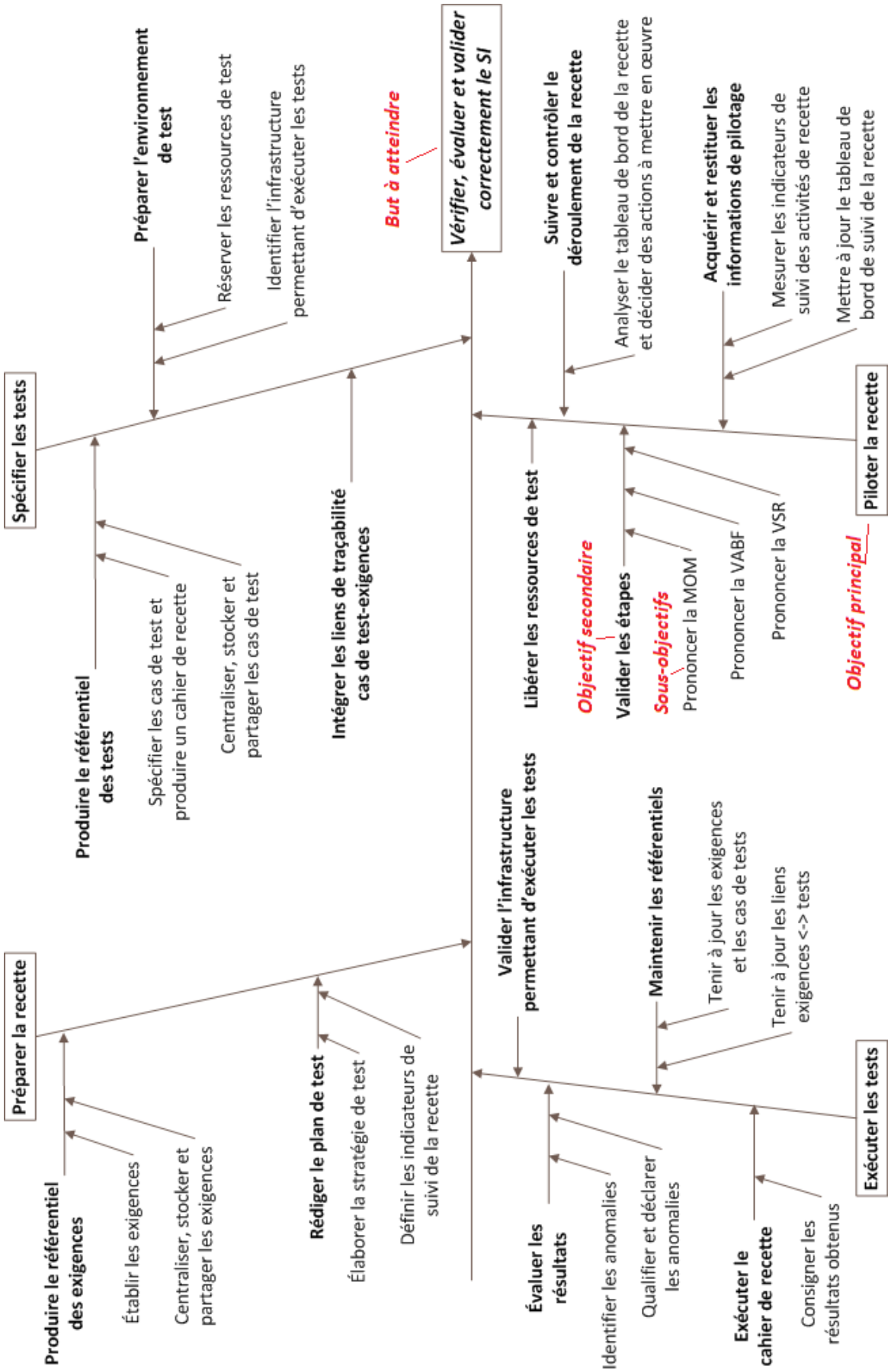


Illustration 12: Modèle des objectifs métier assignés au projet RFSI

Le diagramme ci-dessus est un des résultats du groupe de travail « processus de recette ». Il représente les quatre objectifs majeurs du projet : préparer et piloter la recette, spécifier et exécuter les tests.

6.2.3 Analyse de l'existant

L'analyse de l'existant est un préalable qui permet d'établir un diagnostic d'une situation donnée afin de mesurer l'effort à produire pour atteindre une situation cible. Dans le cadre du projet RFSI, l'analyse de l'existant peut être synthétisée par les éléments suivants.

Au début de la phase de réalisation d'un projet, le chef de projet Études dispose de deux documents contractuels traitant les aspects de l'étape de recette : le CCTP et le Plan d'Assurance Qualité (PAQ). Ces documents précisent les éléments concernant la MOE :

- elle réalise les tests unitaires et d'intégration des développements réalisés ainsi que les tests d'intégration dans un environnement similaire à celui de la production,
- elle installe sur site un environnement dédié aux tests en plus de l'environnement de production,
- elle livre un cahier de recette à la MOA,
- elle met un gestionnaire d'anomalies à disposition de la MOA.

D'un point vu méthodologique, ces deux documents de référence :

- placent les tests de la MOE au niveau de l'étape de Mise en Ordre de Marche (MOM),
- positionnent la recette lors de l'étape de Vérification d'Aptitude au Bon Fonctionnement (VABF),
- définissent les délais d'intervention pour la prise en compte et la résolution des anomalies détectées en VABF et Vérification de service régulier (VSR),
- déterminent les critères de passage de la VABF à la VSR.

Dans la pratique, les tests de recette sont manuels et effectués par la direction métier (chef de projet MOA, experts métiers, utilisateurs finaux) et la DSI (chef de projet et technicien DSI). Les testeurs sont formés au préalable et s'appuient sur le cahier de recette et sur certains documents pour réaliser les tests (support utilisateur, documentation d'installation).

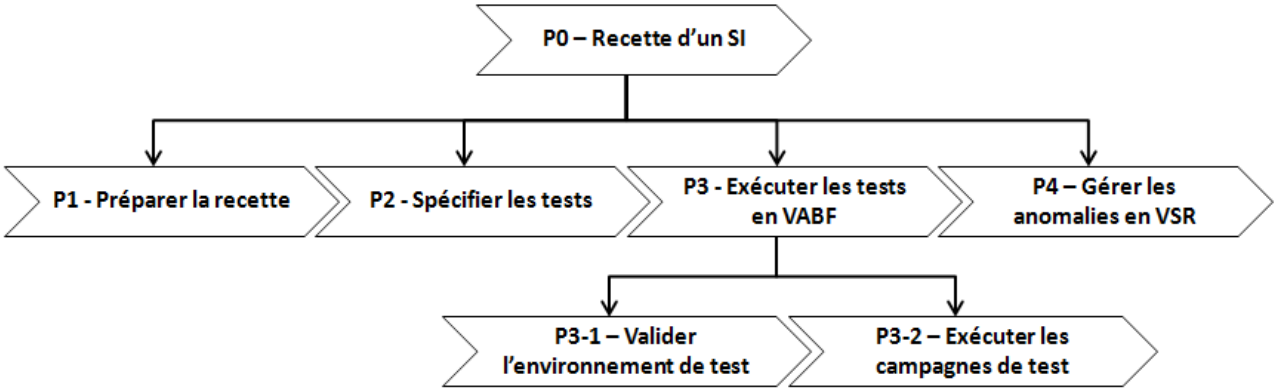
6.2.4 Identification des processus cibles

A partir de l'étude de l'existant et des objectifs métier assignés au projet, sept processus de recette cibles ont été identifiés par le groupe de travail « processus recette ».

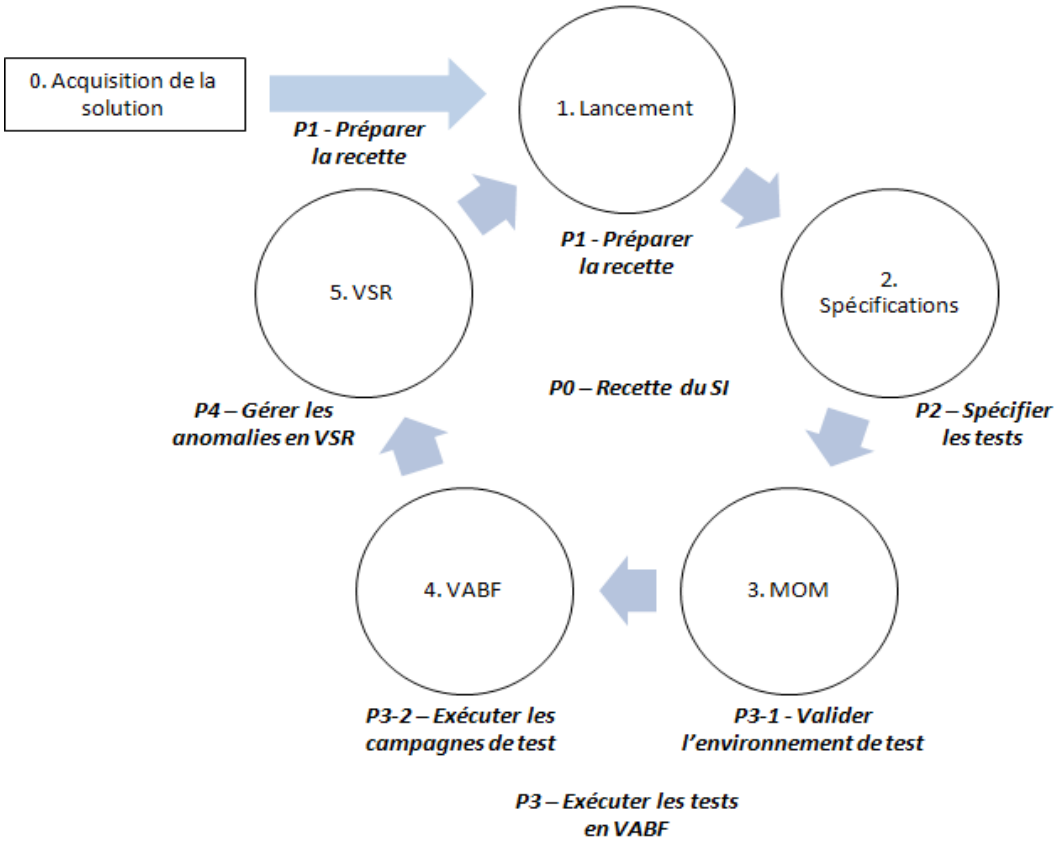
Il ont été ensuite nommés, hiérarchisés et numérotés de la manière suivante :

- P0 - Recette d'un SI,
  - P1 - Préparer la recette,
  - P2 - Spécifier les tests,
  - P3 - Exécuter les tests en VABF,
    - P3-1 - Valider l'environnement de test,
    - P3-2 - Exécuter les campagnes de test,
  - P4 – Gérer les anomalies en VSR.

Leur enchaînement temporel est donné par le schéma ci-après.



De même, leur positionnement dans la phase de réalisation de nos projets, qui peut être cyclique, est donné par le schéma ci-après.



La formalisation de ces sept processus de recette (modélisation BPMN + documentation textuelle associée) a fait l'objet d'un livrable ad hoc (7 fichiers Word + 7 schémas BPMN).

6.2.5 Identification des besoins des utilisateurs

Une fois les processus de recette cibles définis, le groupe de travail « outillage de l'exécution des processus de recette » a recensé les principaux besoins en s'appuyant sur l'analyse des processus de recette.

Ces besoins sont donnés dans le tableau ci-après.

Processus	Besoins identifiés
P0 - Recette d'un SI	Disposer d'un plan d'actions de la conduite du changement relative à la recette.
P1 - Préparer la recette	Gestion du référentiel des exigences • création, mise à jour, stockage et partage du référentiel, • définition, caractérisation, structuration des exigences, • production de la grille des exigences annexée au CCTP.

Processus	Besoins identifiés
	Gestion du plan de test : • création, mise à jour, stockage et partage du plan de test.
P2 - Spécifier les tests	Gestion du référentiel des tests • création, mise à jour, stockage et partage du référentiel, • organisation et rédaction des tests (cas de test, scénarios, et campagnes de test), • maintien des liens de traçabilité entre les exigences les cas de test, • production du cahier de recette, • interface avec le gestionnaire d'anomalie. Gestion « classique » des anomalies en lien avec le référentiel des tests (interface). Disposer d'une description des techniques de test selon l'état de l'art. Gestion des données de test (génération, échantillonnage et anonymisation des données de test).
P3 - Exécuter les tests en VABF	Disposer d'un manuel utilisateur ou d'un document similaire (p. ex. un support de formation détaillé) permettant d'assister les testeurs dans leurs tâches.
P3-1 - Valider l'environnement de test	Disposer d'une documentation d'installation à jour permettant d'assister les techniciens testeurs dans leurs tâches.
P3-2 - Exécuter les campagnes de test	Gestion du tableau de bord recette • collecte des données, mise à jour, stockage et partage Outiller les tests fonctionnels et de non-régression.
P4 – Gérer les anomalies en VSR	Gestion du tableau de bord recette • collecte des données, mise à jour, stockage et partage Outiller les tests fonctionnels et de non-régression.

6.2.6 Analyse des besoins

Chaque besoin référencé dans le tableau précédent a été traduit en une ou plusieurs exigences devant être satisfaites par la solution. De même, l'analyse des besoins a fait émerger les composants décrits ci-après.

Le gestionnaire des exigences et des tests

Il comprend :

- un référentiel des exigences et des tests permettant de :
  - gérer les exigences :
    - créer, modifier, consulter et supprimer une exigence,
    - définir et caractériser une exigence (projet, référence, dates de création/mise à jour, version, nom, description, catégorie),
    - organiser les exigences sous la forme d'un arbre (relation mère-fille(s)),
    - lier une exigence à un cas de test,
    - exporter toutes exigences vers Word/Excel pour produire la grille des exigences annexée au CCTP d'un projet,
    - importer des exigences,

- de gérer les cas de test :
  - créer, modifier, consulter et supprimer un cas de test,
  - définir et caractériser un cas de test (projet, référence, dates de création/mise à jour, version, nom, objectif, activité testée, pré-requis techniques et fonctionnels, pas de test (numéro, description de la tâche, modalité d'exécution, comportement attendu, résultat obtenu)),
  - organiser les cas de test sous la forme d'un arbre (relation père-fil(s)),
  - lier un cas de test à une ou plusieurs exigences,
  - associer une ou plusieurs pièces jointes (p. ex. un manuel utilisateur) au cas de test (Word, Excel, PDF…),
  - importer/exporter des cas de test,

- de gérer les campagnes de test :

  - créer, modifier, consulter et supprimer une campagne de test,
  - définir et caractériser une campagne de test (projet, référence, dates de création/mise à jour, version, nom, objectif, pré-requis techniques et fonctionnels),
  - lier une campagne de test à un ou plusieurs cas de test,
  - exporter toutes les campagnes de test vers Word/Excel pour produire le cahier de recette du projet,
  - exécuter manuellement une campagne de test et consigner les résultats obtenus :
    - date d'exécution et nom du testeur,
    - cas de test OK, KO, non exécuté,
    - lier une ou plusieurs anomalies aux cas de test en échec en indiquant leurs références dans le gestionnaire d'anomalies,

- un référentiel d'anomalies permettant de :
    - lier une anomalie à un cas de test, déclarer, qualifier et suivre la correction de celles-ci.

L'infrastructure des tests de non-régression

Elle comprend un outil permettant :

- de modéliser/« scripter » des tests fonctionnels,
- d'exécuter automatiquement les tests fonctionnels de non-régression,
- de suivre les résultats de l'exécution automatisée.

Le gestionnaire des données de tests

Il comprend un outil permettant :

- de générer des données types (personnes, adresses, etc.), voire de les charger/décharger de la base de données de recette à la demande,
- d'échantillonner et de rendre anonyme des données de production.

Le pilotage de la recette

Le pilotage de la recette se concrétise par un infocentre[36] permettant :

- de relever les valeurs des indicateurs de suivi de la recette,
- d'éditer des d'états de sortie permettant de faire le suivi au fil de l'eau de la recette,
- publier un tableau de bord recette actualisé.

Les autres ressources

Les ressources recouvrent :

- les principaux modèles de livrables de la recette :
    - grille des exigences, plan de test, cahier de recette, tableau de bord de recette,
- une liste des actions d'accompagnement de la MOA pendant la recette,
- une liste des techniques de tests applicables
- les logiciels permettant d'outiller le gestionnaire des tests, les tests de non-régression et le gestionnaire des données de tests.

36 Un infocentre comprend un ensemble d'outils permettant d'interroger des données métier gérées par des applications informatiques différentes.

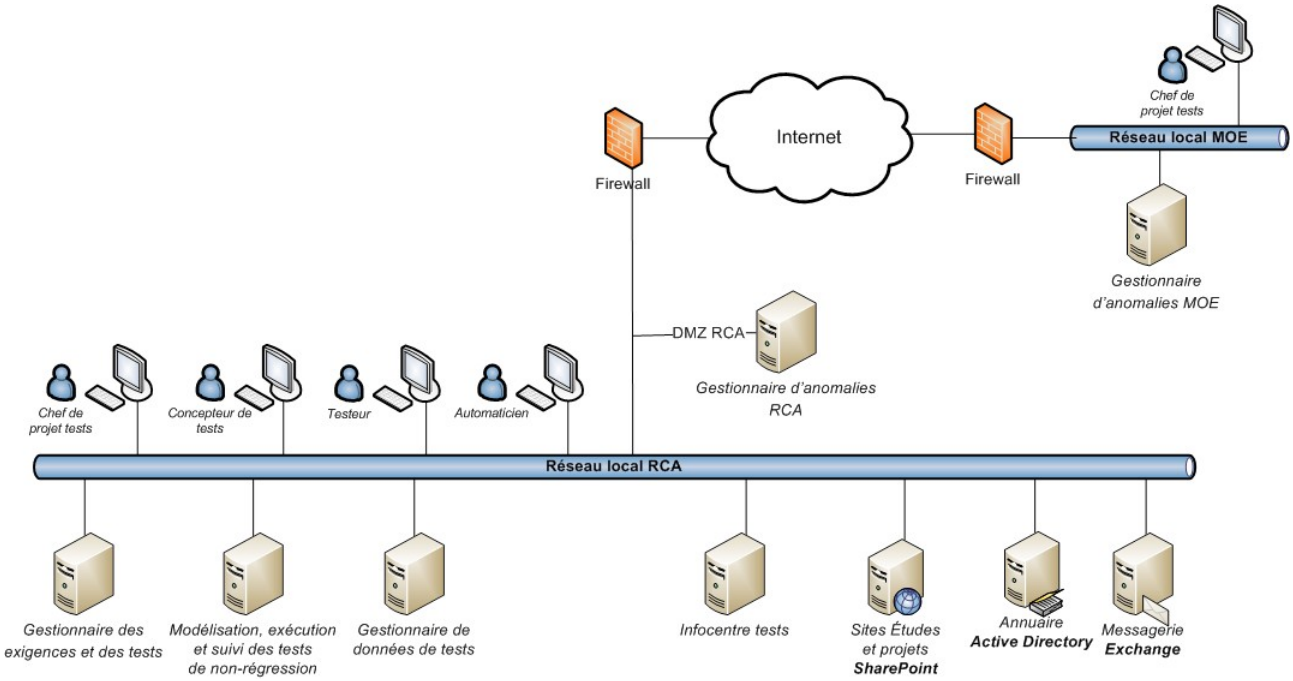
6.2.7 Typologie des utilisateurs

Ces outils sont destinés à des utilisateurs ayant des rôles différents. Ceux-ci sont présentés dans le tableau ci-après et ont été dérivés de l'analyse des processus de recette et des besoins.

Rôle de l'utilisateur	**Activités gérées**
Chef de projet tests (MOA, DSI, MOE)	En charge du cycle de vie des livrables de la recette (grille des exigences, plan de test, cahier de recette, tableau de bord recette). Responsable de la création et de la maintenance du référentiel des exigences. Il déclare, qualifie et suit la correction des anomalies en lien avec les tests. Il lie les cas de test aux anomalies. Il met en œuvre les actions d'accompagnement de la MOA pendant la recette. Il suit l'avancement des tests au fil de l'eau et prépare le reporting auprès des instances de pilotage.
Concepteur de tests	Concepteur et rédacteur des scénarios et des cas de test regroupés en campagnes de test. Il lie les cas de test aux exigences. Il utilise les techniques de tests applicables pour concevoir les cas de test et les outils nécessaires pour créer des données de test.
Testeur	Il exécute manuellement les campagnes de test et consigne les résultats obtenus.
Automaticien	Concepteur et rédacteur des tests de non-régression automatisés. Il exécute automatiquement les campagnes de test de non-régression et consigne les résultats obtenus.

6.2.8 Modélisation de la solution cible

Compte tenu des besoins exprimés et de l'organisation de la recette adoptée pour la plupart de nos projets (processus, acteurs, responsabilités…), j'ai proposé l'architecture technique suivante qui permet de couvrir l'ensemble des besoins recensés et fait apparaître les différents profils utilisateurs.



6.2.9 Analyse des outils du marché

Partant de cette architecture cible, j'ai identifié les principaux logiciels du marché et contacté des éditeurs et des fournisseurs « open source » pouvant implémenter cette solution technique. Les résultats de mes recherches sont donnés dans le tableau ci-après.

Composant de l'architecture cible	Logiciels candidats
Gestionnaire des exigences et des tests	***Éditeurs importants*** SilkCentral Test Manager (Borland) QADirector et QACenter (Compuware) TestDirector (HP Mercury) Rational Test Manager, Rational ClearQuest Test Management (IBM Rational) Visual Studio 2010 Test Manager (Microsoft) ***Éditeur français*** ReferTest (Referty) ***Open source*** Salomé-TMF, RTMR, Fitnesse, Testlink, RTH, Squash TM (open source)
Modélisation, exécution et suivi des tests de non-régression	***Éditeurs importants*** SilkTest (Borland) HP Quality Center (HP Mercury) Rational Functional Tester (IBM Rational) Visual Studio 2010 Test Manager (Microsoft) ***Open source*** Cubic Test, Selenium, Watir, Sahi, Squash AT (open source)
Gestionnaire de données de tests	***Éditeur*** DataGen (e-n@xos) ***Open source*** *Jailer, Benerator, Squash Data (open source)*

Cas particulier du gestionnaire d'anomalies

Le gestionnaire d'anomalies est généralement mis en œuvre par l'entité qui assure la MOE de la phase de réalisation de nos projets. Cette entité peut être un prestataire quelconque ou la DSI. Par ailleurs, le gestionnaire d'anomalies est une source de données indispensable permettant la construction d'un tableau de bord recette par projet *via* l'infocentre test. Dans ce contexte, les outils mis en place peuvent fortement varier d'un prestataire à l'autre (p. ex. Mantis, Excel, site SharePoint…) ce qui implique autant d'interfaces entre l'infocentre test et les gestionnaires d'anomalies de la MOE pour récupérer les données nécessaires à l'élaboration d'un tableau de bord recette. Pour résoudre ce problème, j'ai proposé de mettre en place un référentiel unique et centralisé des anomalies pour l'ensemble des projets des Études. Ce référentiel devant être interopérable avec les gestionnaires d'anomalies des prestataires par l'exposition de services web ou *via* un autre moyen technique.

6.2.10 Proposition de solutions

Compte tenu de l'architecture cible visée et des outils disponibles sur le marché (éditeur et open source), j'ai identifié trois orientations principales :

1. mise en œuvre d'une suite de logiciels de tests d'un éditeur reconnu du marché (Borland, Compuware, HP Mercury, Microsoft), voire celle d'un éditeur moins renommé (p. ex. R*eferTest* édité par Referty),

2. mise en place de la suite de logiciels de tests open source *Squash*[37] qui semble être la seule alternative française open source aux grands éditeurs (c.-à-d. une suite complète de logiciels intégrés),

3. mise en place de plusieurs outils open source complémentaires et plus ou moins interopérables (p. ex. *RTMR, Mantis, Selenium*).

L'alternative du développement spécifique ne m'a pas semblé pertinente au vu de la maturité et de la richesse de l'offre du marché. Enfin, quelque soit l'orientation envisagée, celle-ci devait être complétée par un outil de reporting spécialisé dans la restitution et la mise en forme des données (p. ex. *Business Objects* ou *QlikView*) permettant d'établir des états de suivi des tests et le tableau de bord recette.

6.2.11 Comparatif des différentes orientations

J'ai ensuite établi un comparatif des différents scénarios possibles en fonction de quatre critères majeurs afin d'aider le comité de projet à prendre une décision. Cette analyse est exposée ci-après.

- ✓ ***Réponses aux besoins***

Les suites de logiciels proposées par les éditeurs importants du marché du test sont complètes, voire trop riche d'un point de vue fonctionnel compte tenu des besoins recensés (p. ex. couverture totale du cycle de développement logiciel).

37 *Squash* (http://www.squashtest.org/) est un projet open source français initié en 2010 et soutenu par des grands comptes (France Telecom, Renault…) et financé en partie par l'État et des collectivités territoriales (Région Île-de-France, CG92).

De plus, les modules composant une même suite de logiciels communiquent entre eux le plus souvent par des interfaces d'import/export ce qui nécessite de maîtriser plusieurs outils. Chez les éditeurs plus modestes, le logiciel *ReferTest (Referty)* semble être le plus adapté aux besoins de la RCA en tant qu'elle est MOA/AMOA de ses projets, ce logiciel étant moins orienté MOE que ceux des grands éditeurs. Enfin, l'alternative de l'open source n'est pas négligeable. Il existe de nombreux logiciels susceptibles de répondre en grande partie aux besoins. Le plus intéressant étant sans conteste *Squash*, une prometteuse suite de logiciels intégrés, suivi des logiciels complémentaires *RTMR* (gestionnaire exigences/tests), *Mantis* (gestionnaire d'anomalies), *Selenium* (exécution et suivi des tests de non-régression), *Jailer* et *Benrator* (gestionnaire de données de tests).

- ✓ ***Délais de mise en œuvre***

On peut considérer que les délais de mise en œuvre seront maîtrisés à partir du moment où la DSI pourra s'appuyer sur une MOE externe qui possède les compétences techniques spécifiques nécessaires à la mise en œuvre des logiciels de la solution. C'est évidemment le cas pour les éditeurs de logiciels de tests, mais c'est également vrai pour la suite open source *Squash* qui bénéficie du support de la société Henix (http://www.henix.com/). Ce prestataire est le principal contributeur du projet et est actuellement le seul à proposer des prestations autour de cette solution (assistance, maintenance, installation, migration de données, formation…). A contrario, la mise en place d'une solution open source sans soutien externe peut se révéler complexe, notamment en termes :

- de compétences techniques spécifiques (installation, paramétrage et interconnexion des logiciels),
- d'intégration des technologies utilisées dans le SI de la RCA (non respect de la charte technologique de la RCA),
- de ressources (pas de support ni de maintenance classiques, peu de documentation, pas de formation).

Le principal avantage étant bien entendu un coût de licence nul.

- ✓ ***Coût prévisionnel***

Les suites de logiciels des grands éditeurs semblent être très couteuses en termes de licences et de prestations de déploiement (installation, paramétrage). Pour donner un ordre de grandeur, d'après le cabinet Yphise[38], le seul logiciel *TestDirector Enterprise* couplé à *Oracle* reviendrait à environ 42 K€ TTC pour 5 utilisateurs nommés. Un éditeur moins important comme Referty est plus abordable. Sa politique de licences flottantes est tout à fait pertinente dans le contexte de la RCA. A titre d'exemple, il faut compter environ 16 K€ TTC pour 5 utilisateurs simultanés, le plugin *Mantis* et 3 jours de prestations. Pour une solution à base de logiciels open source, l'aspect financier se traduira essentiellement en termes de coûts internes induits : prise en charge de l'installation, du paramétrage, des formations, acquisition des compétences techniques… Dans ce sens, la charge de travail et donc le coût global de la solution peuvent facilement exploser.

38 Voir http://www.yphise.fr/award/mercury/570C.pdf.

Enfin, pour le cas particulier de *SquashTM*, la maîtrise des coûts peut passer par la commande de journées de prestations auprès de la société Henix. Son offre SaaS[39] semble appropriée à ce type de suite de logiciels encore expérimentale et destinée à fortement évoluer sur les 2 prochaines années. Le forfait annuel est de 18 K€ TTC. Celui-ci comprend la mise à disposition d'une instance SaaS de Squash TM, Mantis, le support technique et la maintenance. L'offre de support technique et de maintenance seule est de 7 K€ TTC par an.

- ✓ ***Pérennité, évolutivité et maintenance***

Dans le cas d'une solution d'un éditeur, la pérennité et l'évolutivité sont garanties par des contrats de support client et de maintenance classiques. Cette assertion étant également valable avec la solution *Squash* supportée par la société Henix (p. ex. l'offre SaaS a été retenue par le ministère de l'éducation nationale pour remplacer *Salomé-TMF*). Dans ce contexte, le principal inconvénient réside dans le lien plus ou moins fort entre l'éditeur et son client, bien qu'une offre SaaS semble apporter un peu de souplesse, si la migration des données vers un autre système est garantie. Dans ce dernier cas, un client peut alors changer d'éditeur plus facilement quand il n'a pas eu à réaliser un investissement trop important pour l'acquisition, l'hébergement et l'exploitation du logiciel.

La pérennité, l'évolutivité et la maintenance des solutions open source constituent aujourd'hui les principaux inconvénients de ce type d'approche. Les mises à jour des logiciels ne sont pas maîtrisées, la documentation est rare, pas de visibilité à moyen terme sur les futures évolutions, déficit de compétences… Pour une organisation sans expérience sur la technologie employée, les coûts internes peuvent très vite dépasser les investissements qui auraient été réalisés dans le cadre d'un contrat avec un éditeur.

Conclusions

Les suites de logiciels des grands éditeurs sont vraisemblablement hors-jeu à cause des coûts élevés induits par leur mise en œuvre relativement aux besoins et aux ressources allouées au projet RFSI (cf. §1.4.2 page). Quant aux autres options, elles se valent plus ou moins en fonction de l'importance accordée à chaque critère (besoins, coûts, délais, maintenance…). A titre d'exemple, concernant le gestionnaire des tests, les solutions envisageables pourraient se décliner de la manière suivante :

- *ReferTests* si on privilégie l'expérience d'un éditeur spécialisé,
- *SquashTM* si on décide de parier sur l'avenir,
- *RTMR* et *Mantis/Bugzilla* si on souhaite tester les possibilités de l'open source à moindre frais.

Pour les autres composants de la solution (gestion des données, tests de non-régression), les logiciels doivent être choisis en prenant en compte la technicité requise par ces outils, en plus des autres critères (besoins, coûts, délais, maintenance…). Une étude ad hoc en temps opportun sera menée dans ce sens.

39 Software as a Service, équivalent à un abonnement/location d'un logiciel externalisé, hébergé et exploité par un tiers.

6.2.12 Validation des résultats

Pour rédiger le principal livrable correspondant à l'étude détaillée (document interne standard de 27 pages), je me suis appuyé sur les résultats produits par les différents groupes de travail, à savoir :

- le modèle des objectifs métier de la recette,
- la modélisation des processus de recette,
- les modèles de documents suivants : la grille des exigences, le plan de test et le cahier de recette (modèles Word),
- la liste des indicateurs de pilotage de la recette,
- la liste des actions de la conduite du changement en recette.

Les conclusions de l'étude détaillée, notamment l'architecture cible et les différentes orientations possibles pour l'implémenter, ont été présentées lors du comité de projet du 29 septembre 2011. Lors de cette réunion, il a été retenu que les différentes orientations envisagées (organisation, méthodes, outils) seraient soumises pour validation au DSI. Cette présentation a été effectuée le 14 décembre 2011 et le DSI a choisi de parier sur l'avenir et a retenu l'orientation préconisée par le comité de projet, à savoir implémenter l'architecture de la solution avec la suite de logiciels open source Squash TM associé au logiciel libre Mantis. D'autant plus que le budget initialement prévu de 25 K€ a dû être reporté à l'exercice 2013, les effets de la crise actuelle ayant impactés le budget de la DSI...

Lors de cette étape d'étude détaillée et de conception, j'ai consommé 37 jours*homme de charge de travail répartie de la manière suivante :

- 40 % de la charge investie dans les groupes de travail (organisation, préparation et animation des réunions, rédaction des compte-rendus, présentation et validation des résultats),
- 40 % de la charge dédiée aux activités de conception (modélisation des processus, analyse des besoins, définition d'une architecture, analyse et test des outils du marché),
- 20 % de la charge consacrée aux activités de gestion de projet (planification, suivi de l'avancement...).

La validation des résultats des groupes de travail, qui se sont déroulés de mai à juillet 2011, et de l'étude détaillée a constitué la principale difficulté rencontrée pendant cette étape. Les chargés de projet Études ont eu du mal a dégagé le temps nécessaire pour relire et valider les livrables produits, notamment ceux qui ont été élaborés par des groupes de travail auxquels ils n'ont pas participé. Cette dérive s'expliquait par des emplois du temps chargés (plusieurs projets à gérer), et par le fait que le projet RFSI, en tant que projet « interne », n'était pas prioritaire dans le plan de charge du pôle Études. Néanmoins, pour continuer à faire vivre le projet et minimiser les dérives de planning, j'ai dû re-négocier (en tête à tête) et contractualiser (par un système de gestion de tâches planifiées) les délais de validation avec chaque chargé de d'étude. Les leviers de la négociation et de la contractualisation se sont avérés efficaces une fois de plus.

6.3 Réalisation et mise en place de la solution retenue

6.3.1 Mise au point de la réalisation

Compte tenu des processus et des besoins identifiés, il m'a semblé pertinent de prioriser la réalisation des différents chantiers afin de maîtriser les délais de mise en œuvre des logiciels.

La mise en œuvre a donc été découpée en 4 chantiers :

- gestionnaire des exigences et des tests,
- infrastructure des tests de non-régression,
- gestionnaire des données de tests,
- pilotage de la recette.

J'ai alors proposé de prioriser les chantiers « gestionnaire des exigences et des tests » et « pilotage de la recette » par rapport aux chantiers « infrastructure des tests de non-régression » et « gestionnaire des données de tests ». Ces deux derniers chantiers étant conditionnés par la disponibilité des outils Squash TA (automatisation des tests) et Squash Data (générateur des données de test) fin avril 2012.

6.3.2 Lancement de la réalisation du projet

En amont du lancement du projet, j'ai monté une plate-forme de test regroupant les outils Squash TM et Mantis afin d'anticiper les problèmes d'installation et de paramétrage. Le lancement officiel de la réalisation des chantiers « gestionnaire des exigences et des tests » et « pilotage de la recette » a été effectué lors du comité de projet du 31 janvier 2012. A cette occasion, j'ai pu présenté la solution (document interne de 12 pages), le planning de mise en œuvre et les charges de travail en fonction des différents acteurs. Le compte-rendu du comité de projet a permis de valider cette étape.

6.3.3 Solution retenue

La solution présentée au comité de projet se décompose en deux volets, d'une part la documentation pour rationaliser les activités métier et d'autre part, l'outillage pour supporter celles-ci.

La documentation regroupe les éléments suivants :

- les 7 processus de recette concrétisés chacun par un schéma, une description textuelle et une matrice RACI :
    - P0 - Recette d'un SI,
        - P1 - Préparer la recette,
        - P2 - Spécifier les tests,
        - P3 - Exécuter les tests en VABF,
            - P3-1 - Valider l'environnement de test,
            - P3-2 - Exécuter les campagnes de test,
        - P4 – Gérer les anomalies en VSR,
- les 4 documents modèles ci-dessous:
    - la grille descriptive des exigences qui est systématiquement annexée au CCTP,
    - les livrables de la recette :
        - le plan de test, le cahier de recette et le tableau de bord recette,
- la liste des actions de conduites du changement à mener pendant l'exécution des processus de recette,
- une synthèse de l'état de l'art en matière de tests et de recette fonctionnelle d'un SI [14].

Les logiciels retenus pour supporter les différentes activités de recette sont les suivants :

- un gestionnaire des exigences et des tests → **Squash Test Management**,
- un infocentre recette → **Business Objects** couplé à un **« univers recette[40] »**,
- un gestionnaire d'anomalies → **Mantis Bug Tracker**,
- un ensemble d'outils permettant :
    - d'automatiser les tests de non-régression → **Squash Test Automation**,
    - de gérer les données de test → **Squash Data**.

40 Un univers BO est une représentation métier des objets gérés dans une base de données (champs, variables calculées..). Ces vues métier permettent aux utilisateurs d'interroger les données avec leur propre vocabulaire.

6.3.4 Pertinence de la solution

La pertinence de la solution retenue a été validée en fonction de sa capacité supposée à contribuer à l'atteinte des objectifs du projet RFSI définis au §1.4.2. Le tableau ci-après met en évidence cette contribution présumée.

<table>
<tr><th>Objectif</th><th>Sous-objectif</th><th>Moyens</th></tr>
<tr><td rowspan="4">Déterminer un cadre pour la validation d'un SI</td><td>Définir les processus métier</td><td>Documentation des 7 processus de recette.</td></tr>
<tr><td>Déterminer les acteurs et leurs responsabilités</td><td>La matrice RACI associée à chaque processus de recette.</td></tr>
<tr><td>Identifier les documents contractuels</td><td>Les modèles de grille des exigences, du plan de test et du cahier de recette.</td></tr>
<tr><td>Mettre en place des outils pour supporter et piloter la recette</td><td rowspan="3">Les gestionnaires d’exigences, de tests et d’anomalies.

L’infocentre et le tableau de bord recette.

La liste des actions de conduite du changement.

Les outils d’automatisation des tests de non-régression et de gestion des données de test.</td></tr>
<tr><td>Maîtriser les risques</td><td>Gérer la recette en mode projet</td></tr>
<tr><td>Réduire les coûts</td><td>Rationaliser les activités de la recette</td></tr>
<tr><td>Minimiser les conflits MOA vs MOE</td><td>Définir les règles du jeu</td><td>Le plan de test et le tableau de bord recette.</td></tr>
<tr><td rowspan="3">Évaluer et vérifier un SI</td><td>Appréhender l'état de l'art</td><td>La synthèse de l’état de l’art en matière de tests et de recette fonctionnelle d’un SI.</td></tr>
<tr><td>Diffuser les bonnes pratiques</td><td rowspan="2">Réunions et site Études.</td></tr>
<tr><td>Capitaliser et partager les expériences</td></tr>
</table>

Ainsi, d'après le tableau précédent nous pouvons constater que chaque objectif est couvert par la solution. En d'autres termes, celle-ci offre un ou plusieurs moyens d’atteindre chaque objectif fixé au projet RFSI.

6.3.5 Description des principaux composants de la solution

L'outillage de la solution repose avant tout sur un certain nombre de logiciels. Leur description est donnée ci-après.

Squash TM

C'est un gestionnaire de référentiel de tests multi-projets open source[41]. Il permet de gérer les étapes de la recette d'un projet informatique, de la gestion des exigences à l'exécution des campagnes de test en passant par la déclaration des anomalies détectées pendant les tests de recette (en lien avec Mantis). C'est un logiciel « full Web » qui permet de centraliser, d'organiser et de partager les exigences, les cas de test et les campagnes de test avec tous les membres d'un projet.

Pour rappel, les exigences d'un projet correspondent aux fonctionnalités et autres besoins (charte graphique, ergonomie, performances…) devant être satisfaits par le système informatique mis en place dans le cadre du projet (logiciel de gestion, site Intranet collaboratif, portail Internet…). En général, les exigences sont décrites par le cahier des charges du projet (p. ex. dans le CCTP).

Un cas de test correspond à une séquence d'actions et de vérifications pratiquées sur le système informatique à tester visant à vérifier que celui-ci satisfait effectivement une ou plusieurs exigences.

Une campagne de test est un regroupement de cas de test visant à tester une version ou une partie du système informatique.

Mantis

C'est un gestionnaire d'anomalies en contexte multi-projets open source[42].

Cet un outil éprouvé, complet et fortement paramétrable permet en autre :

- de gérer des utilisateurs, leurs profils, droits d'accès et notifications par emails,
- de déclarer, tracer et suivre l'ensemble du cycle de vie des demandes (anomalies, évolutions…),
- d'effectuer des synthèses statistiques des demandes et de produire des rapports,
- d'importer/exporter des anomalies dans un fichier XML,
- de communiquer avec des applications tierces via le service web[43] Mantis Connect[44].

41 Voir http://www.squashtest.org/index.php/fr/outilsfonctionnalites/squash-tm.
42 Voir http://www.mantisbt.org/, voir également http://www.projet-plume.org/fiche/mantis.
43 Un service web désigne un mécanisme de communication entre applications distantes permettant d'échanger des données fournies par un composant fournisseur à l'attention d'un composant consommateur.

44 Mantis propose son API à travers un web service SOAP, accessible à cette adresse http://[MANTIS_SITE]/api/soap/mantisconnect.php. La WSDL est disponible à http://[MANTIS_SITE]/api/soap/mantisconnect.php?wsdl.

Infocentre Business Object

L'infocentre recette s'appuie sur l'infrastructure logicielle et matérielle existante du SI de la RCA. Celle-ci est composée de l'outil de reporting Business Object qui exploitera un univers Oracle recette alimenté par l'ETL[45] Sunopsis qui se chargera d'exploiter les sources de données disponibles, essentiellement les données gérées par Squash TM, Mantis et SharePoint.

Squash TA

C'est le module d'automatisation des tests fonctionnels de la suite open source Squash[46]. Il se compose d'un designer permettant de créer les scripts automatisés et d'un lanceur pour gérer les campagnes de tests automatisés. Squash TA permet d'automatiser les tests des IHM Web, des services web et des batchs.

Squash Data

C'est le module de Squash chargé de la gestion des jeux de données des tests[47], qu'ils soient manuels ou automatisés. Le module regroupe des outils permettant la création des jeux de données (par peuplement ou échantillonnage), leur stockage/partage (au travers d'un repository centralisé) et leur utilisation (pour injecter un ou plusieurs jeu(x) de données dans une base avant un test, par exemple). Squash Data a été conçu pour pouvoir traiter efficacement des volumétries de données importantes, entre bases de données et/ou fichiers.

Sites projets Microsoft SharePoint

Les sites projets des Études publiés par le gestion de contenus SharePoint 2007 centralisent l'ensemble des informations relatives à chaque projet. Il paraît donc naturel de gérer les documents de recette (plan de test, cahier de recette…) de la même manière que les autres documents projet (CCTP, PAQ…). De plus, la page dédiée aux tests (GT-TESTS.aspx) contiendra également une liste d'hyperliens vers les différents outils de recette (Squash TM, Mantis, Infocentre recette…).

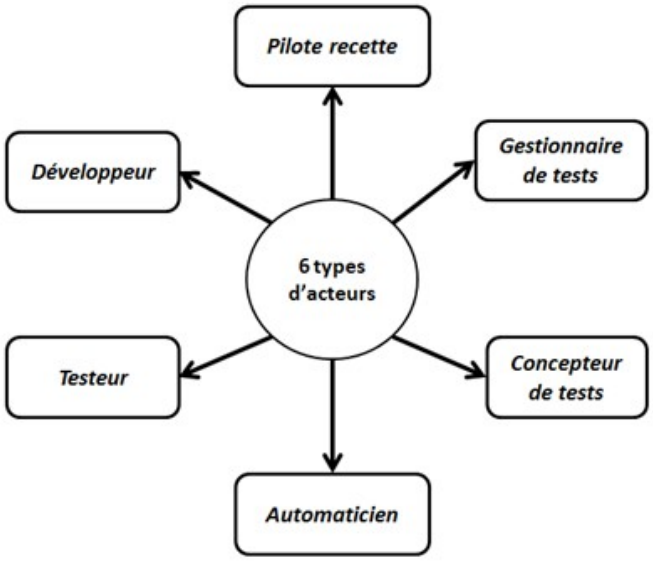
6.3.6 Rôles et profils des utilisateurs

Les acteurs amenés à utiliser les logiciels de la solution mis en œuvre sont décrits par le schéma ci-après.

45 Extract Transform Load (ETL) : logiciel destiné à extraire des données de diverses sources de données (base de données, fichiers), à les transformer et à les charger en général dans un entrepôt de données.

46 Voir http://www.squashtest.org/index.php/fr/outilsfonctionnalites/squash-ta.

47 Voir http://www.squashtest.org/index.php/fr/outilsfonctionnalites/squash-data.



Le rôle de pilote recette est potentiellement joué par tout membre d'une des entités organisationnelles du projet : comités de pilotage/projet/suivi, équipes projet/recette. Le pilote recette est sollicité en fonction de l'importance des problématiques à traiter et selon le niveau de décision requis.

Le rôle de gestionnaire de tests peut être joué par le chef de projet MOA, DSI et/ou MOE selon les cas. Il produit et met à jour les différents livrables de la recette (référentiels exigences/tests/anomalies, plan de test, tableau de bord recette). Il met également en œuvre les actions d'accompagnement de la MOA pendant la recette et il suit l'avancement et les résultats des tests au fil de l'eau tout en préparant le reporting auprès de l'instance de pilotage de la recette.

Le rôle de concepteurs de tests (orientés métier) est joué par les chefs de projet MOA et DSI accompagnés d'un ou plusieurs experts métier. Il utilise les techniques de tests applicables pour concevoir des cas de test et des données de test pertinentes.

Le rôle d'automaticien est joué par un expert technique formé à l'utilisation des logiciels d'automatisation de l'exécution de tests fonctionnels et techniques. Il exécute automatiquement les tests de non-régression et consigne les résultats obtenus.

Le rôle de testeur est joué par un expert métier, technique et/ou un futur utilisateur. Il exécute manuellement les campagnes de test et consigne les résultats obtenus.

Le rôle de développeur est joué par un expert technique de la MOE. Il corrige les anomalies détectées par les tests.

6.3.7 Installation et paramétrage des outils

Je me suis servi de l'expérience acquise sur la plate-forme de test des outils Squash TM et Mantis pour définir l'infrastructure logicielle et matérielle nécessaire à la mise en œuvre de la solution. Celle-ci a été modifiée par les responsables des pôles Exploitation et Réseaux, systèmes et sécurité

lors d'une réunion de travail le 5 janvier 2012. Je prévoyais initialement de faire installer le logiciel Mantis en « DMZ Sécurité »[48] car l'outil devait être accessible à des utilisateurs externes, en général, les utilisateurs de la MOE quand celle-ci est externalisée. Cependant, pour mutualiser les moyens existants[49], Mantis a été installé dans la zone Intranet (la même zone que le serveur MySQL) et publié sur Internet via le proxy-inverse ISA Server 2006. Ce dernier garantissant un niveau de sécurité tout à fait acceptable : sécurisation des échanges en HTTPS, filtrage sur IP (règles définies au niveau du pare-feu) et contrôle des accès par compte Active Directory. Le principal inconvénient concernait le confort d'utilisation des utilisateurs externes. Ceux-ci devaient s'authentifier deux fois, une première fois pour accéder au réseau Intranet, puis une seconde fois pour se connecter à Mantis.

A priori, à l'heure actuelle il n'existe qu'une solution pour régler de problème. Celle-ci consiste à modifier certaines pages de configuration de Mantis pour faire de l'authentification « Windows intégrée » *via* du code PHP et configurer Apache pour utiliser le module d'authentification « mod_ntlm2 ». Mais faute de compétence interne et de ressource sur le Web, cette solution n'a pas pu être mise en place, sachant qu'en parallèle la DSI mène un projet de gestion des accès au SI qui devrait permettre de solutionner ce problème à moyen terme.

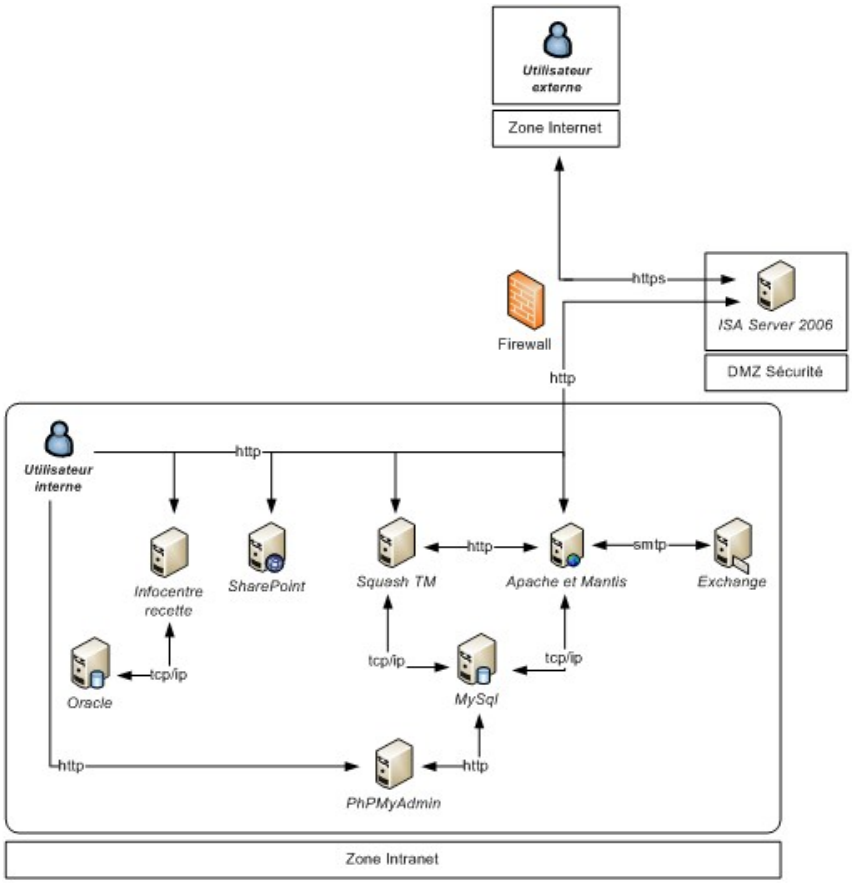


Illustration 17: Architecture technique et protocoles de communication

48 Zone du réseau isolée du réseau interne de la RCA et qui est accessible aux utilisateurs externes *via* Internet.

49 Le serveur de base de données MySQL et sa console d'administration PhpMyAdmin ainsi que le serveur Web Apache.

Une fois l'architecture arrêtée, les outils ont été installés par le pôle Réseau, système et sécurité. Quant à moi, je me suis chargé du paramétrage des outils Squash TM et Mantis (connexion aux bases de données, interface entre les deux outils, personnalisation...). L'ensemble de ces travaux a donné lieu à la rédaction d'une documentation technique (document interne de 10 pages).

6.3.8 Réalisation du tableau de bord recette

La réalisation du tableau de bord recette a été confiée à l'expert décisionnel du pôle Études qui a participé au groupe de travail « pilotage de la recette ». Ce groupe de travail avait pour but de définir les indicateurs et le tableau de bord de suivi des activités de recette (environ une douzaine d'indicateurs). Du point de vue de la DSI, un indicateur est une représentation chiffrée d'un phénomène que l'on veut mettre sous contrôle. Il permet d'objectiver une situation. Dans notre cas, il s'agit de contrôler le suivi de l'exécution des tests, les résultats obtenus et le traitement des anomalies détectées.

Dans ce contexte, une seule réunion de travail, qui s'est tenue le 1er février avec le responsable du pôle Études et moi-même, a permis de spécifier les besoins à partir des indicateurs identifiés précédemment. Autrement dit, nous avons clarifié la définition des indicateurs, défini les croisements possibles avec les axes d'analyse retenus (p. ex. le temps : année, mois, jour ou la sévérité d'une anomalie : bloquante, majeure, mineure), déterminé les modes de calcul des indicateurs et identifié les sources de données utilisables.

Suite à cette réunion, l'expert décisionnel du pôle Études a rédigé le dossier de spécifications (document interne de 14 pages) qu'il nous a soumis pour validation le 22 février 2012. Ce document décrit la matrice de croisement entre les indicateurs et les axes d'analyse, les modèles physiques des données des sources et cibles, les traitements d'extraction et de transformation des données sources ainsi que le chargement des données dans la base de données cible.

Suite à la validation du document, la réalisation des développements a été lancée. Durant cette phase, j'ai essentiellement assisté l'expert décisionnel dans son travail de reengineering des bases de données sources (analyse des modèles physiques des données[50], tests applicatifs...). Le tableau de bord recette (document de 3 pages, cf. §5.5) a été présenté en réunion de travail le 22 mars 2012. Les tests qui ont suivi ont permis de valider les données moyennant quelques évolutions.

6.3.9 Mise en production du pilote

La mise en production des logiciels de la solution s'est faite en deux temps. Dans un premier temps, les logiciels Squash TM et Mantis ont été déployés début mars. Fort de l'expérience acquise sur la plate-forme de test, nous n'avons pas eu de difficulté avec Squash TM. Par ailleurs, étant donné que le projet OSIDI était déjà en phase de réalisation depuis plusieurs mois, j'ai dû anticiper dès le mois de janvier la préparation et la spécification des tests de recette (processus P1 – Préparer la recette et P2 – Spécifier les tests) en travaillant sur une instance Squash TM installée en locale sur mon poste de travail. Ainsi, lorsque Squash TM a été installé sur le serveur de production, une simple migration de base de données avec l'outil PhpMyAdmin m'a permis de récupérer le travail réalisé.

50 Le travail d'analyse a été facilité par l'utilisation du logiciel PowerAMC de Sybase qui fourni des outils pour faire de la rétro-conception de base de données.

Par contre, Mantis nous a posé quelques problèmes pendant la période de tests d'accès effectués en collaboration avec la société SWORD, MOE du projet OSIDI. D'une part, les règles de filtrage sur l'IP publique de SWORD, configurées dans le pare-feu, était trop restrictives en interdisant l'accès à certains utilisateurs externes, et d'autre part, les emails générés automatiquement par Mantis faisaient référence à des URL (adresse Web) différentes en fonction du type d'utilisateur (interne vs externe) ce qui provoquait des problèmes de connexion.

L'autorisation d'une plage d'adresses IP a permis de résoudre le premier problème et la création d'un enregistrement DNS (liaison adresse IP/adresse Web) interne identique à l'adresse de Mantis publiée sur Internet a permis de résoudre le second problème. Dans un deuxième temps, le tableau de bord recette a été mis en production fin mars sous la forme d'un document PDF accessible depuis l'infocentre régional. Ce document produit à partir de données réelles, celles du projet OSIDI, a permis d'identifier quelques anomalies mineures et cosmétiques qui ont été rapidement corrigées.

Le projet RFSI a été lancé dans le cadre des activités de rationalisation et de professionnalisation du pôle Études. Nous avons pu constater à travers ce projet que les difficultés rencontrées par les différents acteurs peuvent se réduire à quatre catégories. Le manque de formalisation et de contractualisation liées au processus de recette, des activités de recette pas suffisamment rationalisées, des faiblesses dans l'exécution des processus de test, des insuffisances dans la maîtrise du processus de gestion des exigences.

Pour répondre à cette problématique, nous avons proposer trois pistes de réflexion, l'organisation, la méthodologie et l'outillage, qui nous ont amenées à imaginer et concevoir une solution originale au problème de la recette fonctionnelle d'un SI en définissant des processus de recette outillés par des logiciels adaptés aux besoins des utilisateurs : la suite de logiciels Squash et Mantis.

Cette quatrième partie a également permis d'illustrer par un exemple concret le déroulement d'un projet mené par le pôle Études et le travail réalisé par un chargé de projet.

La cinquième partie quant à elle se propose d'appliquer au projet OSIDI une partie de la solution définie dans le cadre du projet RFSI. Dans ce but, je présenterai le portail des aides régional mise en œuvre dans le cadre du projet OSIDI.

Puis, il s'agira d'utiliser certains éléments de la solution pour recetter le portail des aides, à savoir dérouler les processus de recette en s'appuyant sur les outils et la documentation de la solution mise au point dans le cadre du projet RFSI.

Enfin, j'exposerai mon opinion concernant la réalisation des objectifs fixés au projet RFSI ainsi que mon analyse des écarts de coûts et de planning constatés à la fin du projet. Mon mémoire se conclura ainsi par un premier bilan du projet RFSI sur la base de l'expérimentation de la recette du projet OSIDI.

7 La recette fonctionnelle du portail des aides régionales

La solution conçue et réalisée pendant le projet RFSI a été éprouvée durant l'étape de recette du projet OSIDI. Néanmoins, la solution définie dans les chapitres précédents n'a pas pu être entièrement testée dans le cadre de mon mémoire pour des raisons de planning. Concrètement, je n'ai pas pu exécuté le processus « P4-Gérer les anomalies en VSR » (prévue fin 2012) et je n'ai pas eu l'occasion d'utiliser les outils d'automatisation des tests de non-régression et de génération de données de test. Leur mise en œuvre étant prévue début 2013.

7.1 Remarques préliminaires

7.1.1 Déroulement de la réalisation du projet OSIDI

La phase de réalisation du projet OSIDI a débuté par une réunion de lancement avec la société SWORD, MOE du projet, début septembre 2011. Pour rappel, le projet OSIDI, que je gère au même titre que le projet RFSI, vise la mise en œuvre d'un portail web de saisie et de suivi des demandes d'aides régionales.

Après avoir mis au point les détails de la réalisation (PAQ, planning détaillé...), nous avons réalisé les spécifications et la conception du portail des aides régionales entre septembre 2011 et février 2012. Les développements, les tests unitaires et d'intégration ainsi que la recette d'usine ont été effectués par SWORD sur la période de décembre à mars 2012. L'installation technique du portail dans les locaux de la RCA a eu lieu fin mars 2012. La recette fonctionnelle réalisée par la RCA s'est déroulée sur les mois d'avril et de juin, la mise en production étant réalisée fin juin 2012.

7.1.2 Structure du portail des aides régionales

Le portail des aides régional est composé d'un site Internet/Extranet (front-office) qui regroupe un ensemble de page web permettant de :

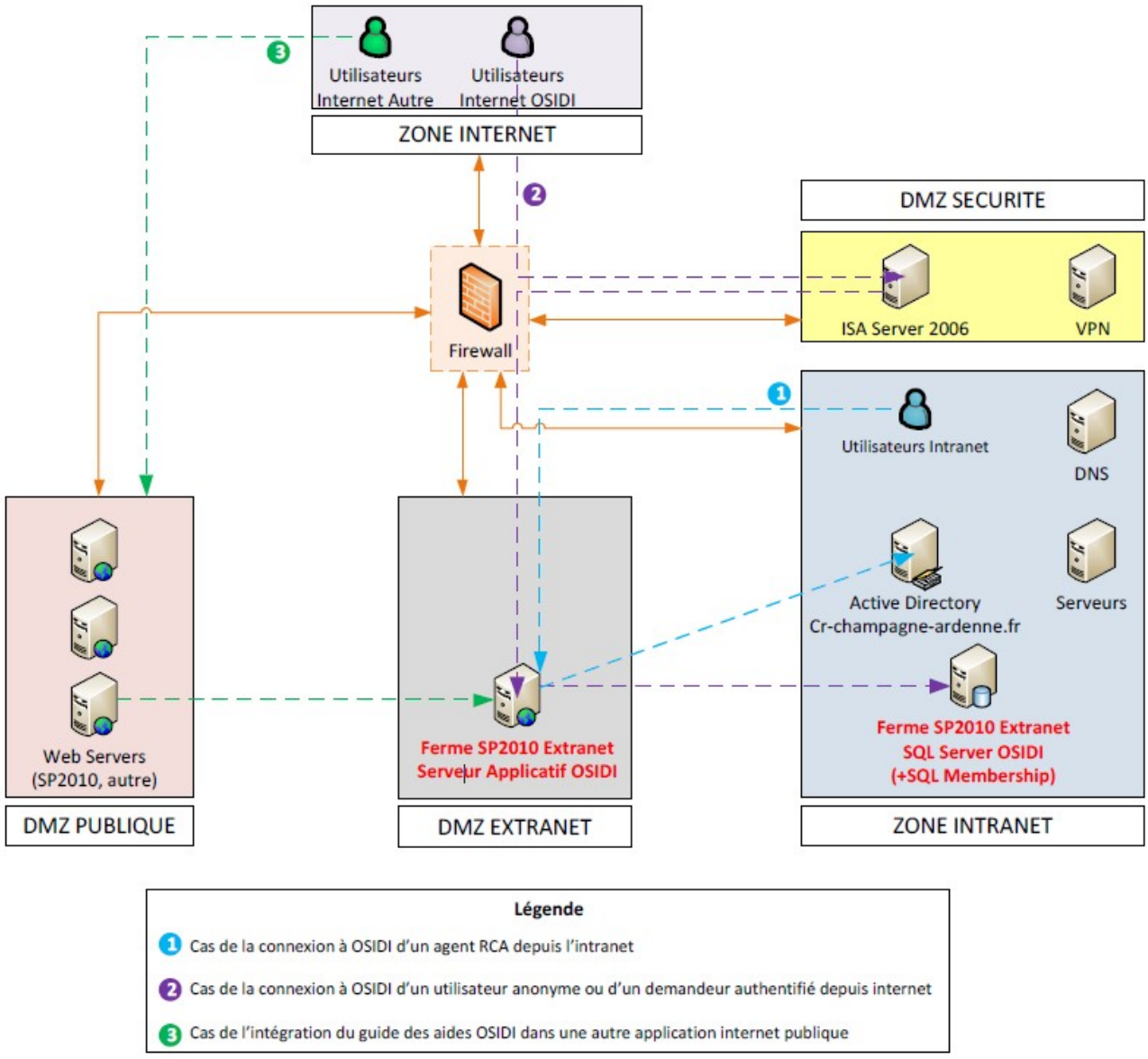
- consulter le guide des aides de la RCA (accès internaute),
- de saisir une demande d'aide en ligne (accès extranaute = internaute authentifié),
- de suivre l'instruction des demandes d'aides (accès extranaute).

Il comprend également un site Intranet (back-office) permettant :

- de consulter les demandes d'aide et de les exporter vers le logiciel d'instruction (agents instructeurs),
- de gérer les contenus publiés sur le portail (agents responsables du contenu éditorial),
- d'administrer le portail (agents administrateurs).

D'un point de vue technique, le portail des aides correspond à une application web SharePoint 2010 constituée d'une collection de sites et utilisant la technologie InfoPath 2010 pour la gestion des formulaires en ligne. Les données sont stockées dans une base de données SQL Server 2010 et les développements spécifiques ont été réalisés en C#.

Le schéma suivant présente son architecture physique.



7.1.3 Périmètre de la solution testée

Les éléments de la solution qui ont été utilisés durant l'étape de recette du projet OSIDI sont les suivants :

- les processus de recette « P1 - Préparer la recette », « P2 - Spécifier les tests », « P3 - Exécuter les tests en VABF », « P3-1 - Valider l'environnement de test », « P3-2 - Exécuter les campagnes de test »,
- les documents modèles : le plan de test, le cahier de recette, le tableau de bord recette,
- la liste des actions de conduites du changement à mener pendant l'exécution des processus de recette,
- la synthèse de l'état de l'art en matière de tests et de recette fonctionnelle d'un SI [14].

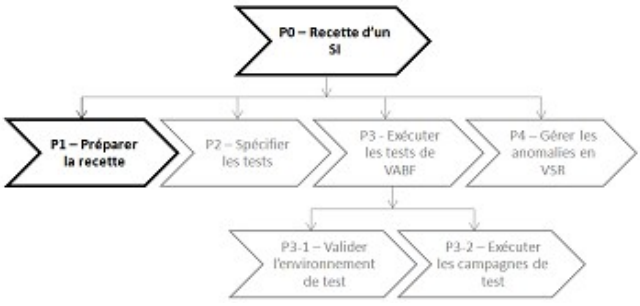
Les logiciels qui ont été utilisés pour supporter les différentes activités de recette sont les suivants :

- le gestionnaire des exigences et des tests → Squash Test Management,
- le gestionnaire d'anomalies → Mantis Bug Tracker,
- l'infocentre recette → Business Objects couplé à un « univers recette».

La suite du présent mémoire va s'attacher à décrire comment s'est déroulée l'exécution des différents processus de recette, quels résultats ont été obtenus et quelles difficultés ont été rencontrées. Le but étant de dresser un bilan de cette expérimentation de la solution.

7.2 Préparation de la recette

7.2.1 Production du référentiel des exigences



L'exécution des premières activités du processus **« P1 - Préparer la recette »** a consisté à produire le référentiel des exigences.

Pour rappel, les exigences d'un projet correspondent aux fonctionnalités et autres besoins (charte graphique, ergonomie, performances…) devant être satisfaits par le système informatique mis en place dans le cadre du projet informatique (logiciel de gestion, site Intranet collaboratif, portail Internet…). En général, les exigences sont décrites par le cahier des charges du projet (p. ex. dans le CCTP).

Pour cela, j'ai saisi dans Squash TM les informations de la grille des exigences annexée au CCTP du marché public OSIDI. En effet, la société SWORD, MOE du projet OSIDI, a été sélectionnée suite à un appel d'offres ouvert qui s'est déroulé du 11 mars au 2 mai 2011. Dans ce contexte, j'ai rédigé le CCTP du marché en collaboration avec les différents acteurs du projet. Ce CCTP comportait une annexe appelée « grille des exigences » qui récapitulait dans un tableau l'ensemble des exigences décrites dans le CCTP.

Pour les futurs projets, la procédure sera inversée, les exigences seront saisies directement dans Squash TM. La « grille des exigences » annexée au CCTP sera générée automatiquement, soit à partir d'un export Excel depuis Squash TM, soit *via* un état produit par l'infocentre recette.

La copie d'écran du logiciel Squash TM v1.0 ci-après montre l'arbre des exigences du référentiel associé au projet OSIDI.

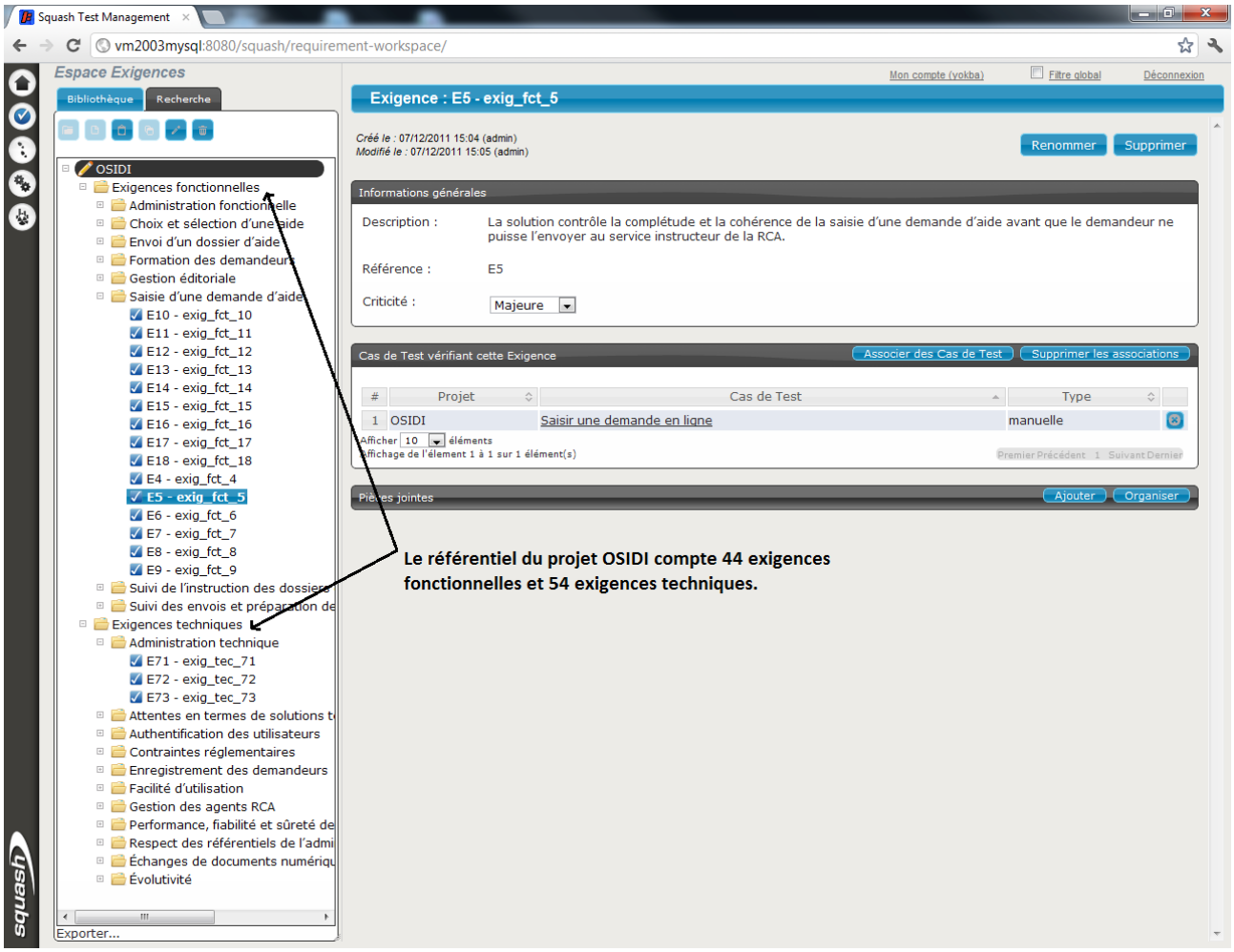


Illustration 19: Référentiel des exigences du projet OSIDI

La version v1.0 de Squash TM a été disponible en téléchargement à partir de fin septembre 2011 ce qui m'a permis de saisir les exigences courant décembre 2011 sur mon poste de travail, en anticipant la mise en production de janvier 2012. Une fois le serveur de production disponible, j'ai simplement effectué une migration du référentiel[51] pour ne pas perdre le travail réalisé.

51 Grâce à l'outil PhpMyAdmin qui permet d'exporter/importer une base de données MySql (fichier « dump »).

Squash TM en tant que référentiel des exigences du projet OSIDI a permis d'une part de centraliser, partager et diffuser la description des exigences auprès des acteurs du projet et d'autre part, de tracer l'évolution des exigences (modification, ajout, suppression).

7.2.2 Rédaction du plan de test

La norme IEEE 829 [47] définit le plan de test comme *« un document décrivant l'étendue, l'approche, les ressources et le planning des activités de test prévues. Il identifie entre autres les éléments et caractéristiques à tester, qui fera chaque tâche, le degré d'indépendance des testeurs, l'environnement de test, les techniques de conception des tests et les techniques de mesure des tests à utiliser, et tout risque nécessitant des plans de contingence. C'est un document reprenant les processus de planification des tests ».*

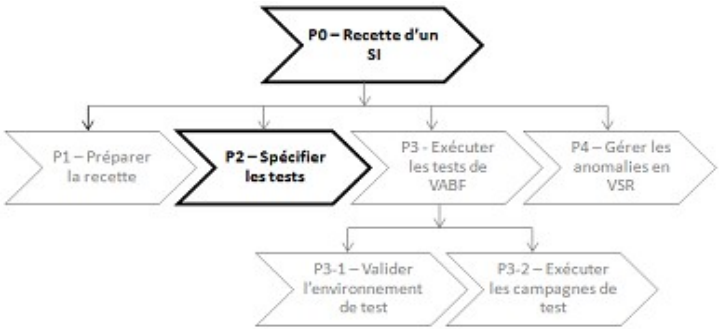
Le contenu type du plan de test du projet OSIDI, défini dans le cadre du projet RFSI en s'inspirant de la norme IEEE 829, est le suivant :

- les références des documents applicables (cahier des charges, dossier de spécifications, normes et standards...),
- le périmètre couvert par le plan de test (exigences, logiciels, interfaces, reprise de données...),
- les exigences non couvertes par le plan de test (lesquelles et pourquoi),
- la description du processus de recette (acteurs, activités, livrables),
- la description de l'environnement de test (outils, base de données de recette...),
- la définition de l'équipe de recette (responsabilités, compétences nécessaires, besoins en formation),
- le planning prévisionnel et la gestion des risques (qualification de la criticité du SI...),
- la liste des critères de démarrage de la recette (niveau de qualité minimal, nombre et nature des anomalies, installation des différents logiciels, disponibilité de la documentation, résultats des tests d'intégration système...),
- la liste des critères d'arrêt des tests (les conditions où les tests sont interrompus et celles permettant de reprendre les tests),
- la liste des critères d'acceptation qui permettent de valider ou non les résultats d'un test,
- la gestion des anomalies (comment rapporter, suivre, résoudre les anomalies et valider les corrections),
- la définition des conditions de sortie de la recette (plus aucune anomalie bloquante, charge de travail consommée...).

J'ai donc rédigé le plan de test du projet OSIDI (document interne standard de 20 pages) au fur et à mesure de l'évolution de la réalisation du portail des aides. Sa rédaction a débuté en janvier 2012, moment à partir duquel nous avions une bonne idée du résultat final puisque les spécifications et la conception du portail des aides étaient quasiment terminées. Puis, le plan de tests s'est progressivement enrichi et détaillé au fur et à mesure de la réalisation du prototype par la société SWORD. Ainsi, j'ai pu présenter le plan de test au groupe projet, dont le chef de projet SWORD, pour validation début mars 2012. Il a été validé mi-mars moyennant l'intégration des remarques effectuées par les différents acteurs et publié sur le site de travail collaboratif du projet.

7.3 Spécifications des tests

7.3.1 Production du référentiel des tests



Le recensement et la spécification des tests ont été dirigés par la stratégie de test décrite dans la plan de test, à savoir tester des ensembles de fonctionnalités qui concourent à l'atteinte d'un ou plusieurs objectifs métier, notamment en exécutant des sous-ensemble de tâches automatisées dans le cadre du projet OSIDI.

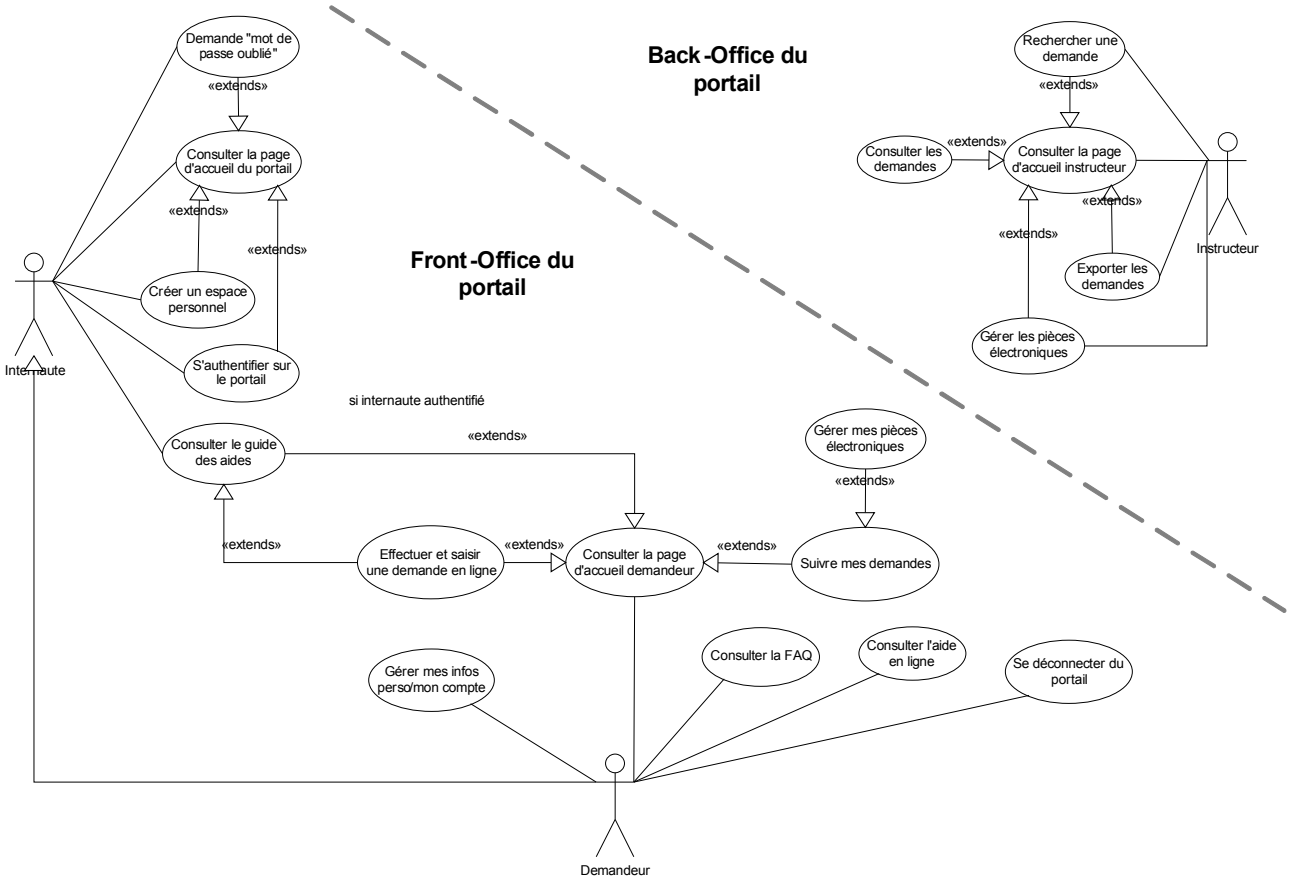
Ainsi, je me trouvais d'emblée dans le cadre des tests de type boîte noire (aucune connaissance de la structure interne des logiciels testés), de navigation/exécution (cf. §3.2.4 : tests orientés par la logique métier) et par affirmation (cf. §3.2.5 : vérification des exigences spécifiées). Je n'ai pas eu recours aux tests de type boîte blanche et de prédiction d'erreur, ceux-ci étant du ressort de la MOE conformément au plan de test.

Pour identifier ces sous-ensembles de fonctionnalités, je me suis appuyé sur le dossier de spécifications fonctionnelles[52] qui comme son nom l'indique, décrit en détail les fonctionnalités du portail des aides en fonction des différents profils utilisateurs. J'ai analysé ce dossier afin de faire apparaître les principaux scénarios d'utilisation dérivés des cas d'utilisation[53] UML[54] du portail des aides (voir le schéma ci-après).

52 Le dossier de spécifications fonctionnelles est un des livrables de la phase de réalisation d'un projet. Le contenu de ce dossier est produit lors des ateliers de travail (workshop) impliquant la MOA, l'AMOA et la MOE. Il livré par la MOE.

53 Le diagramme de cas d'utilisation UML permet de mettre en évidence les relations fonctionnelles entre les acteurs et le système étudié.

54 UML *(Unified Modeling Language*) est un langage de modélisation graphique particulièrement adapté à la modélisation des systèmes informatiques.

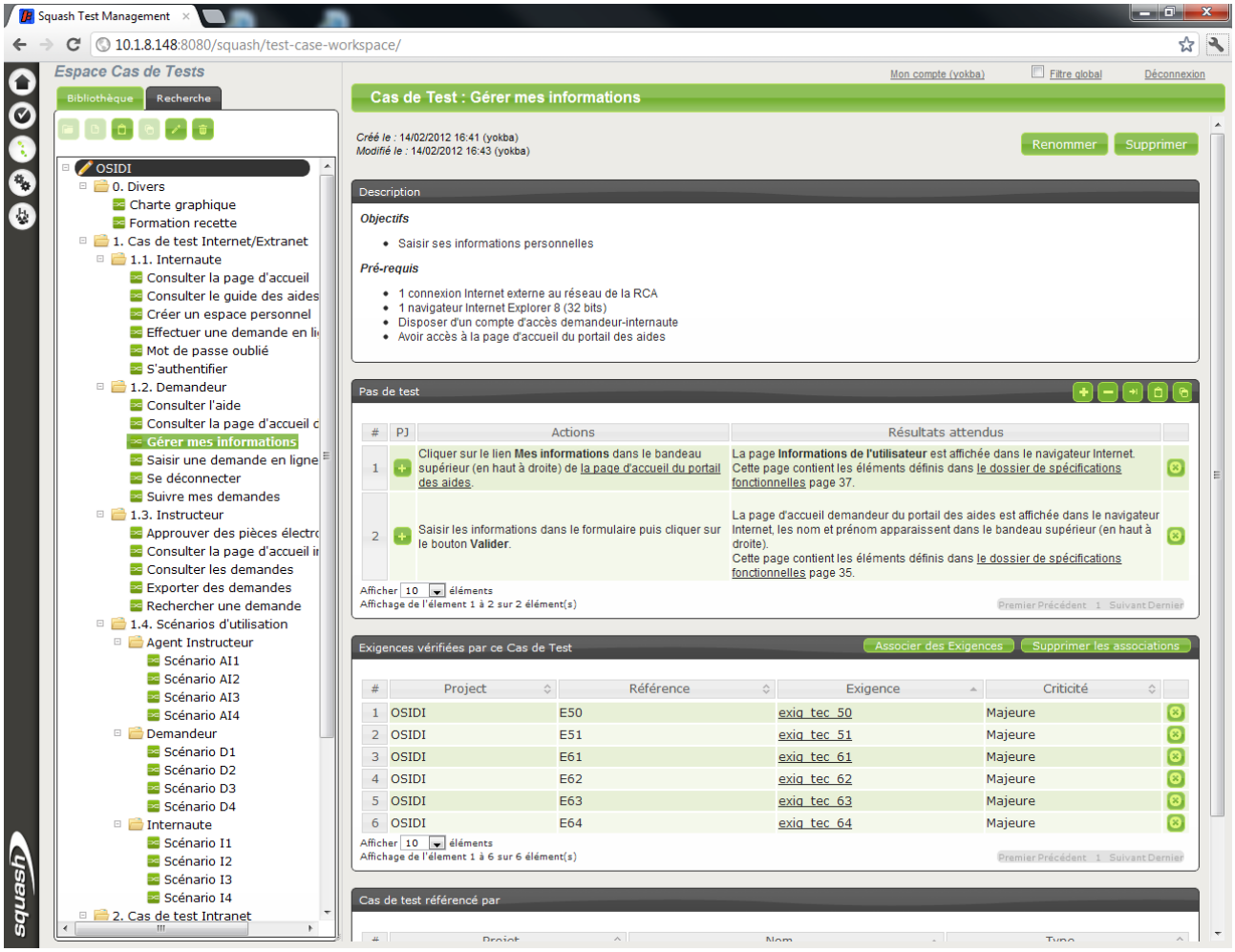


A partir du diagramme précédent, j'ai proposé de tester les principaux scénarios d'utilisation du portail en fonction des profils utilisateurs. Par exemple, j'ai identifié le scénario suivant qui concerne un utilisateur du type « Demandeur » (internaute authentifié sur le portail) :

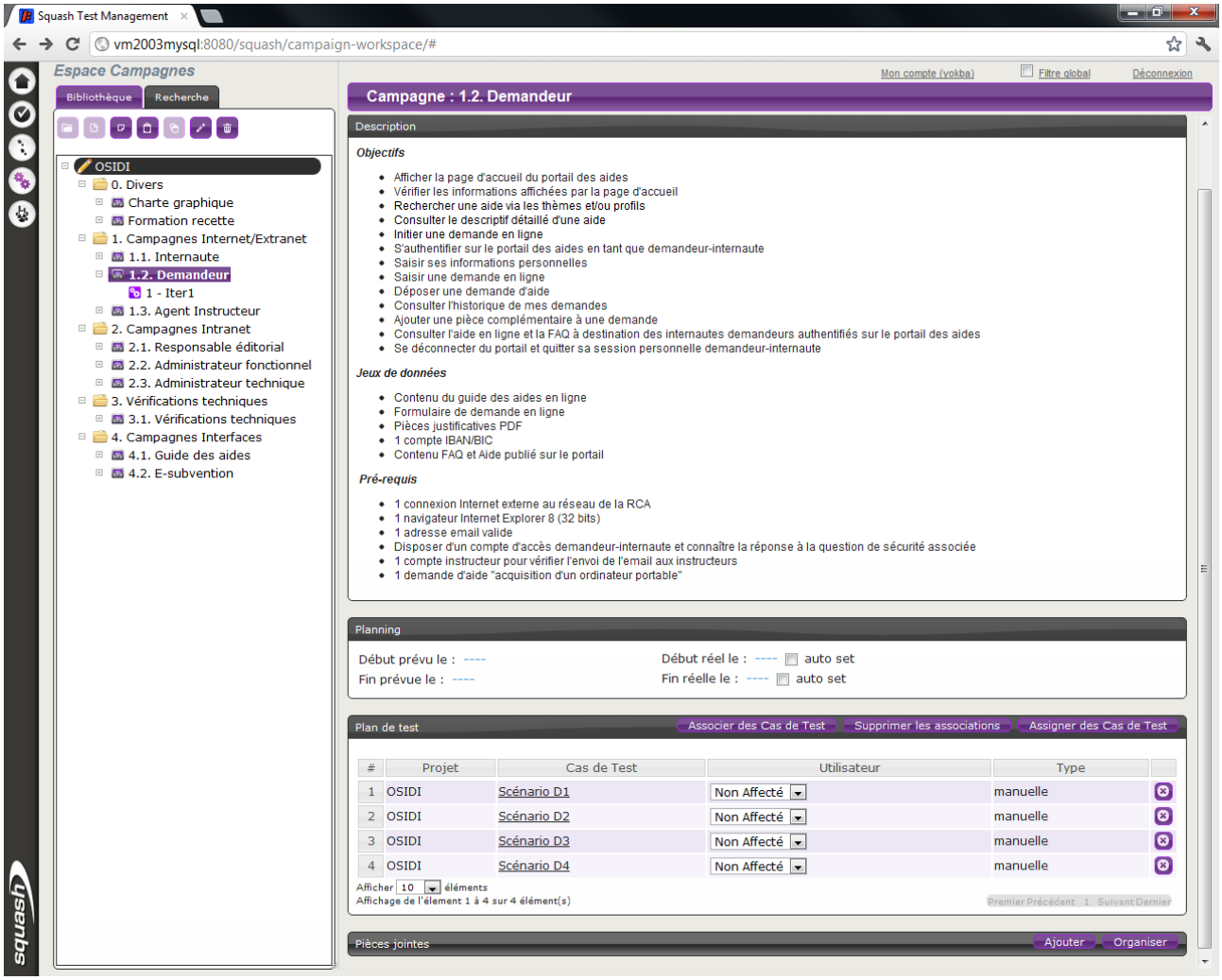
1. Consulter la page d'accueil du portail
2. Consulter le guide des aides
3. Faire une demande en ligne
4. S'authentifier
5. Saisir une demande en ligne
6. Se déconnecter

Ensuite, pour chaque item de chaque scénario, j'ai écrit un cas de test correspondant à une séquence d'actions et de vérifications pratiquées sur le portail des aides et visant à vérifier que celui-ci satisfait effectivement les exigences idoines.

40 cas de test ont été spécifiés et saisis dans Squash TM comme le montre la copie d'écran ci-après.



Squash TM en tant que référentiel des cas de test du projet OSIDI a permis d'une part de centraliser, partager et diffuser la description des tests auprès des acteurs du projet et d'autre part, de tracer l'évolution des tests (modification, ajout, suppression) en fonction de celles des exigences. Enfin, j'ai créé dans Squash TM, 12 campagnes de test regroupant des cas de test et permettant de tester les différents scénarios d'utilisation du portail des aides.



Squash TM en tant que référentiel des campagnes de test du projet OSIDI a permis d'outiller l'exécution des campagnes de test. En effet, dans Squash TM une campagne de test est exécutée par l'intermédiaire d'une itération qui correspond à l'exécution de la campagne par un testeur à un moment donné. Une campagne de test peut être constituée d'autant d'itérations que nécessaire (p. ex. une nouvelle itération à chaque nouvelle livraison).

7.3.2 Gestion des données de tests

Les données de tests sont nécessaires pour exécuter un cas de test. Ces données affectent ou sont affectées par le logiciel testé ; elles sont stockées dans une base de données de test (ou de recette). Ceci étant, le problème est de savoir comment définir des données de test pertinentes, c'est-à-dire permettant d'atteindre les objectifs du cas de test qu'elles alimentent, et efficaces, dans le sens où elles contribuent à détecter des anomalies du logiciel testé.

Dans cette optique, la gestion des données de tests consiste à :

- définir les données de tests pertinentes et efficaces en fonction des cas de test à exécuter,
- les produire, éventuellement à partir de données réelles (échantillonnage, anonymisation), à les stocker et les partager (accès donné aux testeurs),
- les injecter dans la base de données de recette (chargement/déchargement).

Définition des données de tests

Dans le contexte du projet OSIDI, j'ai identifié les données de test à partir de l'analyse des cas de test préalablement spécifiés. Ces données peuvent être regroupées dans les quatre catégories décrites dans le tableau ci-après.

Catégorie	Description	Exemple
Connexion utilisateur	Données permettant de se connecter au portail des aides en fonction des différents profils utilisateurs (demandeur, instructeur, administrateur, responsable éditorial)	Identifiant et mot de passe d'un demandeur : création d'un compte Windows live ou d'un agent : utilisation d'un compte test de l'active directory
Saisie de formulaire	Données nécessaires pour renseigner chaque formulaire disponible sur le portail : formulaire d'authentification, de création de compte, mes informations personnelles, formulaire d'aide à l'acquisition d'un ordinateur portable	Adresse d'un demandeur de type étudiant, son établissement supérieur, son niveau d'études...
Règles de gestion	Données permettant de vérifier le comportement des règles de gestion implémentées sur le portail des aides	Dates d'ouverture/clôture d'une campagne d'aide...
Contenu éditorial	Données publiées sur le portail des aides : textes et messages affichés, actualités, description des aides régionales (guide des aides), la faq, l'aide en ligne...	Le contenu d'une actualité publiée sur le portail...

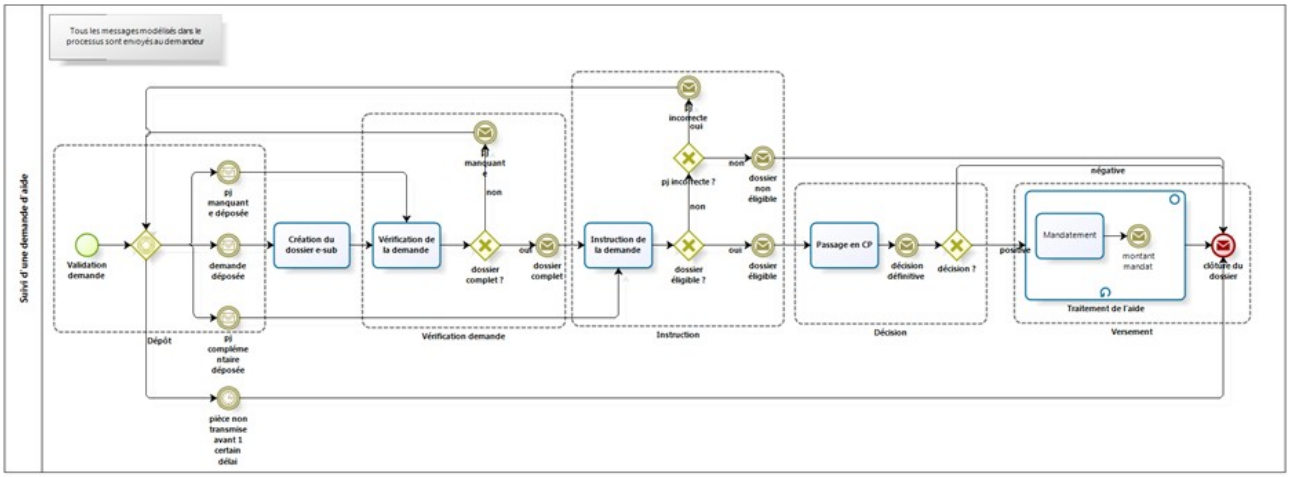
Puis, pour chaque cas de test et en fonction des règles de gestion à vérifier, j'ai défini les données de test nécessaires pour tester les cas passants et non-passants en appliquant les techniques de partitionnement par classes d'équivalence et de test aux limites (cf. §3.2.1) ; le but étant de minimiser le nombre de tests valables à réaliser (voir un exemple au §6.5 en annexe).

A titre d'exemple, les deux techniques précédentes appliquées à la règle de gestion suivante *« Si âge demandeur ≥ à 27 ans alors pas d'accès au formulaire de saisie de l'aide à l'achat d'un ordinateur portable »* ont permis de définir une partie des données de test pour le cas de test *« Gérer mes informations personnelles »* :

- cas passant : 1 demandeur ayant 27 ans - 1 jour,
- cas non-passant : 2 demandeurs ayant respectivement 27 ans et 27 ans + 1 jour.

Ainsi, seulement trois tests ont permis de couvrir l'ensemble des cas possibles, c'est-à-dire tous les âges envisageables pour un demandeur. D'ailleurs, l'exécution du test (demandeur ayant 27 ans) a permis de détecter une anomalie relative à l'application erronée de cette règle de gestion.

En ce qui concerne le cas particulier de l'interface bidirectionnelle entre le portail des aides et le logiciel d'instruction (échanges de fichiers XML), j'ai dû identifier les données de test à partir des cinq cas de test principaux dérivés du processus d'instruction d'une aide, préalablement modélisé avec les instructeurs (document interne standard de 8 pages). Cette modélisation permettant également de mettre en évidence les différents états d'une demande d'aide, ce qui a facilité les tests de transition d'états (cf. §3.2.3), eux-mêmes couplés à un test par négation[55] (cf. §3.2.5).



Par exemple, pour le cas de test n°1 qui correspond à un dossier passant par les étapes :

- vérification → le dossier est incomplet puis devient complet,
- instruction → une ou plusieurs pièces sont NOK puis toutes les pièces sont OK et le dossier devient éligible,
- décision → la décision est positive,
- versement → émission du paiement de l'aide,
- clôture du dossier,

la plupart des données de test nécessaires pour créer les dossiers de demande d'aide ont été extraites de la base de données de production du logiciel d'instruction.

55 Le test par négation a permis de révéler un comportement de l'interface non spécifié qui a pu être pris en charge par la MOE pendant la recette.

Production et injection des données de tests

La production et l'injection des données de test dans la base de recette ont été réalisées manuellement en utilisant l'interface graphique du portail des aides. Le déchargement (p. ex. la suppression d'un compte ou d'un formulaire d'aide) s'est fait également *via* l'interface du portail en utilisant les fonctionnalités d'administration. Évidemment, ce mode de fonctionnement est très consommateur de temps, principal inconvénient de l'approche manuelle. L'automatisation de cette gestion sera en partie prise en charge par la mise en œuvre prochaine des outils Squash Data et Squash TA (cf. §4.3.5).

7.3.3 Les techniques de test non-appliquées

Un certain nombre de techniques de test n'ont pas été appliquées faute de moyen ou parce qu'elles étaient plutôt du ressort de la MOE. Par exemple, les tests de type boîte blanche, de performance ou de fiabilité étaient à la charge de la société SWORD.

Pour la partie données, les techniques de test combinatoire et aléatoire (§3.2.1) n'ont pas été utilisées du fait de la charge importante demandée et du faible nombre de données en entrée du portail des aides et de leur relative indépendance (d'où un grand nombre de combinaisons non pertinentes pour les tests).

Concernant les tests non-fonctionnels (§3.2.8), les techniques liées aux tests de performance, de stress et de volume n'ont pas été utilisées car ces aspects ont été retirés du périmètre de la recette, la DSI n'ayant pas retenu l'architecture technique proposée par la MOE pour faire face aux besoins décrits dans le cahier des charges.

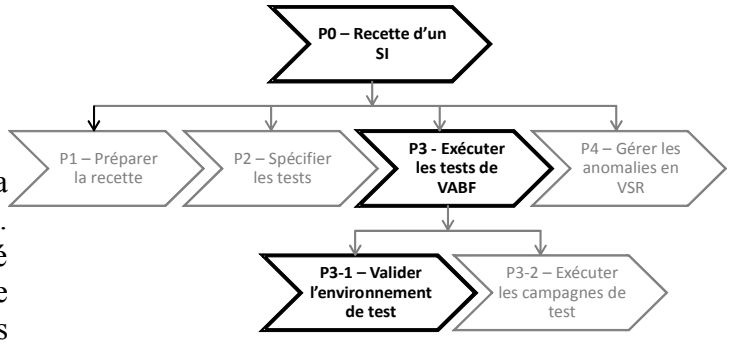
En effet, la mise en place d'une ferme de 2 serveurs frontaux SharePoint couplés à 2 serveurs SQL en cluster n'était pas nécessaire vue la faible sollicitation prévue à court terme et compte tenu des investissements à réaliser. Par contre, l'installation des outils s'est faite de telle manière qu'une évolution vers cette architecture soit facilement réalisable à moyen terme (ceci permet également de répartir dans le temps les investissements). De plus, la charge nécessaire pour prendre en main les outils (p. ex. JMeter[56]), spécifier et exécuter les tests de performance n'était pas compatible avec le planning du projet OSIDI.

Enfin, les tests de documentation et de recouvrement sur panne (§3.2.8) se sont fait au fil de l'eau de la recette selon les situation rencontrées pendant l'exécution des tests. Par exemple, le portail des aides à « crashé » pendant une séance de formation ce qui a obligé les techniciens de la DSI à « rebooter » les systèmes, en s'appuyant sur la documentation livrée par la MOE, afin de rétablir une situation normale.

56 Voir http://jmeter.apache.org/.

7.4 Exécution des tests

7.4.1 Validation de l'environnement de test



La période de VABF correspond à la recette fonctionnelle effectuée par la RCA. Elle commence lorsque le SI a été installé dans les environnements de test et de production, et une fois que les tests d'intégration ont été réalisés par la MOE. La recette fonctionnelle s'est déroulée sur les mois d'avril et de juin 2012.

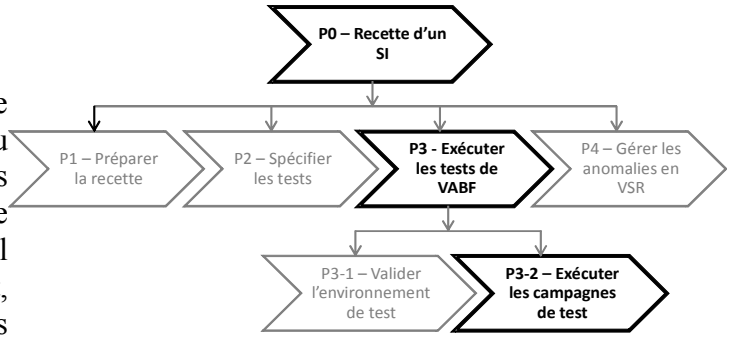
La validation de l'environnement de test a consisté à installer les outils de test, à former l'équipe recette (c-à.d les testeurs) et à vérifier que les techniciens de la DSI maîtrisaient effectivement les procédures d'installation et de paramétrage du SI installé.

Les outils de tests, Squash TM et Mantis (cf. §4.3.5), ont été installés dans le cadre du projet RFSI. En tant qu'administrateur de ces deux outils, j'ai créé les comptes utilisateurs et affectés les droits afférents (testeurs RCA, chef de projet et développeur MOE).

La maîtrise effective des procédures d'installation et de paramétrage du SI installé (cf. §3.2.8 : test de configuration/installation) par les techniciens de la DSI devait normalement se faire de la manière suivante. Le testeur DSI devait installer le SI (infrastructure logicielle SharePoint/SQL Server 2010 et l'application web OSIDI) dans un environnement ad hoc en se basant sur la documentation livrée par la MOE. Malheureusement, faute de disponibilité des techniciens DSI ce test n'a pas pu être réalisé comme initialement prévu. Le plan de test a dû être révisé pour prendre en compte cette modification.

La formation de l'équipe recette, constituée des testeurs du projet OSIDI (une dizaine de testeurs provenant de quatre directions différentes), a été effectuée par la MOE, assistée du chef de projet Études (c-à-d. moi-même), le 28 mars 2012. Cette formation avait pour but de présenter le contexte des tests et les fonctionnalités du portail des aides. Elle a également permis de lancer les premiers tests et d'évaluer l'ergonomie et la charte graphique du portail (cf. §3.2.8 : tests de convivialité).

7.4.2 Exécution des campagnes de test



Cette étape a consisté à exécuter chaque campagne de test, regroupant un ou plusieurs cas de test, en s'appuyant sur les spécifications des tests stockées dans le référentiel de test Squash TM (référentiel des campagnes). Pour chaque cas de test, le testeur a réalisé les actions décrites dans Squash TM (c-à-d. les pas de test) sur le portail des aides afin de confronter le comportement observé avec les résultats attendus.

En cas de divergence, il avait la possibilité de saisir une anomalie Mantis directement à partir de Squash TM. J'ai pré-qualifié chaque anomalie avec le testeur qui l'a détectée, avant de l'affecter au chef de projet MOE *via* Mantis. Ce dernier se chargeant de la ré-affecter au développeur de son équipe pour correction. L'ensemble des correctifs a fait l'objet d'une nouvelle livraison de l'application OSIDI. Ainsi, quatre versions du portail des aides ont été livrées et testées par la RCA (v1.0, v1.1, v1.2 et v1.3). Après chaque nouvelle installation, toutes les campagnes de test ont été rejouées manuellement afin de vérifier la correction effective des anomalies et pour constater l'absence de régression fonctionnelle. Encore une fois, ce mode de fonctionnement est très consommateur de temps, principal inconvénient de l'approche manuelle. L'automatisation de cette gestion sera en partie prise en charge par la mise en œuvre prochaine des outils Squash Data et Squash TA (cf. §4.3.5).

7.5 Pilotage de l'exécution des tests

Le pilotage de l'exécution des tests s'est principalement basée sur le tableau de bord recette du projet OSIDI, mis à jour quotidiennement et de manière automatique (cf. infocentre recette §4.3.5). Ainsi, chaque instance de pilotage du projet OSIDI (comité de projet, comité de suivi, groupe projet) a pu suivre l'avancement et l'exécution des tests ainsi que le traitement des anomalies. Concrètement, le tableau de bord recette est constitué de trois pages, la première est synthétique et permet d'avoir accès à toutes les informations utiles en un coup d'œil.

La première section de la page 1 du tableau de bord donne le taux d'avancement de l'exécution des campagnes et des cas test. Pour rappel, une campagne de test regroupe plusieurs cas de test permettant de vérifier un ensemble cohérent de fonctionnalités.

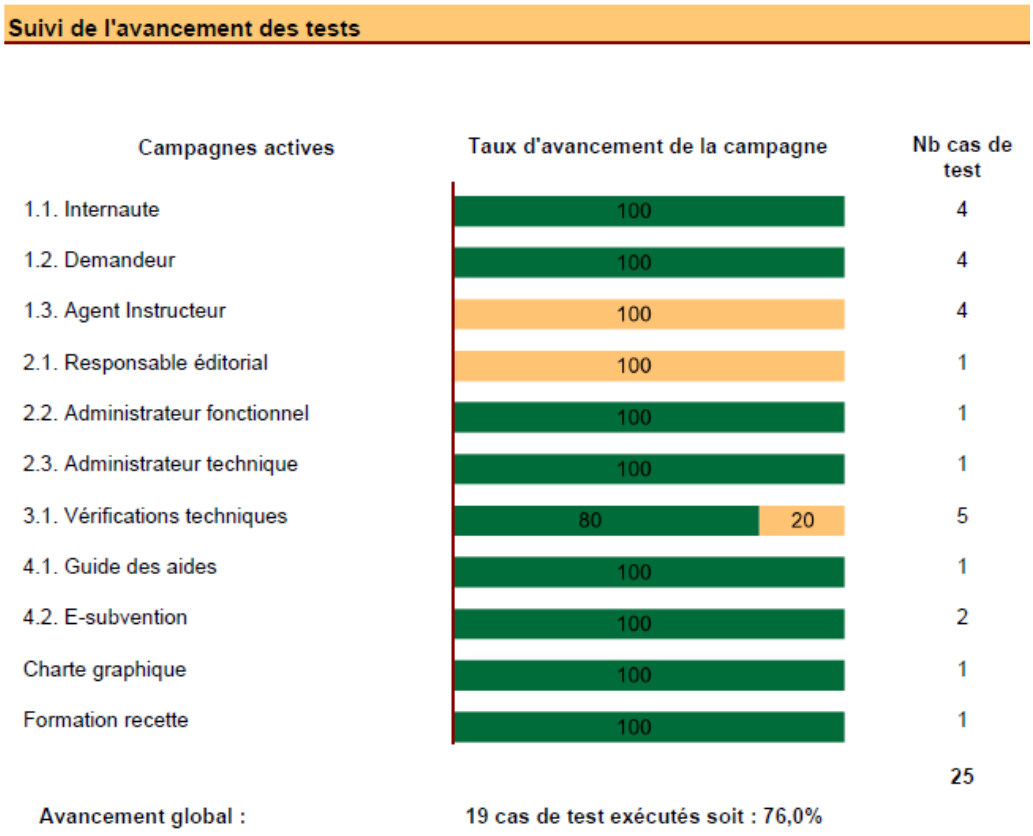
Illustration 24: Taux d'exécution des campagnes/cas de test

La deuxième section de la page 1 du tableau de bord présente les résultats de l'exécution des campagnes et des cas test selon les statuts suivants :

- ECHEC = au moins une anomalie a été détectée durant l'exécution de la campagne/cas de test,
- SUCCES = aucune anomalie détectée,
- BLOQUE = le test n'a pas pu aller jusqu'à son terme (p. ex. à cause d'une anomalie bloquante sans contournement possible).

La troisième section de la page 1 du tableau de bord est dédiée au suivi des anomalies/évolutions en fonction de leur sévérité (bloquante, majeure, mineure).

La page 2 du tableau de bord détaille l'avancement de l'exécution des campagnes et donne et le nombre de cas de test par campagne.



Suivi de l'avancement des tests

Campagnes actives	Taux d'avancement de la campagne	Nb cas de test
1.1. Internaute	100	4
1.2. Demandeur	100	4
1.3. Agent Instructeur	100	4
2.1. Responsable éditorial	100	1
2.2. Administrateur fonctionnel	100	1
2.3. Administrateur technique	100	1
3.1. Vérifications techniques	80 / 20	5
4.1. Guide des aides	100	1
4.2. E-subvention	100	2
Charte graphique	100	1
Formation recette	100	1
		25
Avancement global :	19 cas de test exécutés soit : 76,0%	

Enfin, la page 3 du tableau de bord détaille le statut de chaque campagne et cas de test.

Illustration 28: Statut des campagnes/cas de test

7.6 Bilan de l'expérimentation pilote

7.6.1 Remarque préliminaire

Étant donné les glissements du planning des projets OSIDI et RFSI, je n'ai pas eu le temps matériel pour faire le bilan officiel de l'expérimentation de la solution avec le comité de projet dans le cadre de mon mémoire d'ingénieur. La réalisation de ce bilan étant prévue fin septembre 2012. Dans ces conditions, je propose de présenter dans les paragraphes suivants mon appréciation personnelle concernant les principaux résultats du projet RFSI, mon analyse de ces résultats et les axes d'amélioration que j'envisage.

Par ailleurs, j'ai retenu les trois axes suivants pour apprécier les principaux résultats du projet RFSI : l'atteinte des objectifs métiers, les écarts entre les délais et les coûts prévus vs réalisés, les faits marquants, les problèmes rencontrés et les solutions mises en place.

7.6.2 Niveau de réalisation des objectifs métiers et analyse des écarts

Pour rappel, les objectifs métiers du projet RFSI sont récapitulés ci-après.

1. Déterminer un cadre pour la validation d'un SI,
    1.1. définir les processus métier,
    1.2. déterminer les acteurs et leurs responsabilités,
    1.3. identifier les documents contractuels,
    1.4. mettre en place des outils pour supporter et piloter la recette,
2. Maîtriser les risques :
    2.1. gérer la recette en mode projet,
3. Réduire les coûts :
    3.1. rationaliser les activités de la recette,
4. Minimiser les conflits MOA vs MOE :
    4.1. définir les règles du jeu,
5. Évaluer et vérifier un SI :
    5.1. appréhender l'état de l'art,
    5.2. diffuser les bonnes pratiques,
    5.3. capitaliser et partager les expériences.

Les objectifs 1.1, 1.2 et 1.3 ont été atteints par la modélisation des 7 processus de recette et les 3 documents modèles mis en place (grille des exigences, plan de test, cahier de recette). Les modèles BPMN favorisent la communication, la documentation est claire et les matrices RACI permettent de visualiser rapidement la distribution des responsabilités en fonction des tâches à réaliser. Néanmoins, l'expérience acquise lors de la recette du projet OSIDI a montré qu'il fallait apporter quelques modifications à ces processus.

Tout d'abord, les tâches de spécification des cas de test initialement du ressort de la MOE, doivent être finalement réalisées par la MOA pour plusieurs raisons.

Premièrement, la MOE n'a pas les connaissances nécessaires à la réalisation de tests orientés « processus métier RCA ». En effet, la stratégie retenue par la RCA consiste à tester des ensembles cohérents de fonctionnalités visant à atteindre un ou plusieurs objectifs métiers. Or, j'ai constaté que la MOE avait en général une vision « unitaire » du test, dans le sens où son objectif est de vérifier le bon fonctionnement de chaque fonctionnalité.

Deuxièmement, les bonnes pratiques préconisent de séparer les entités qui développement et testent un SI [2], d'autant plus que la MOA et la MOE poursuivent des objectifs qui ne pas forcément les mêmes (cf. §2.4.2).

Troisièmement, la différence de nature et la complémentarité des tests MOA/MOE renforcent la capacité des ressources de test (MOA et MOE) à détecter des anomalies avant la mise en production effective du SI.

En conclusion, je recommande la rédaction d'un cahier de recette d'usine écrit et exécuté par la MOE avant et après livraison du SI, et un cahier de recette métier électronique (logiciel Squash TM) écrit et exécuté par la MOA/AMOA au moment de la recette fonctionnelle du SI.

L'objectif 1.4 n'a été atteint que partiellement à l'heure actuelle. En effet, aujourd'hui les outils d'automatisation des tests de non-régression et de gestion des données de test non pas encore été mise en œuvre. Leur mise en place étant prévue pour début 2013. Toutefois, la mise en œuvre des gestionnaires des exigences, des tests et des anomalies (Squash TM et Mantis), du tableau de bord recette montrent qu'une partie importante de l'objectif est atteinte. Reste la liste des actions de conduite du changement qui nécessite quelques précisions. En fait, la liste actuelle n'est pas suffisamment concrète concernant les actions d'accompagnement. Il faudrait détailler la forme de ces actions (p. ex. des réunions de travail) avec les autres chefs de projet Études.

Les objectifs 2 et 3 sont complètement atteints étant donné que nous avons effectivement travaillé en mode projet dans le cadre de la recette du projet OSIDI (planifier, réaliser, contrôler, ajuster) et que les règles du jeu (c.-à-d. le plan de test, cf. §5.2.2) ont été définies au préalable et validées par la MOA et la MOE.

L'objectif 3 est difficilement appréciable faute de données permettant de comparer les charges internes consommées sur le projet OSIDI (seul poste de coût pour la RCA) avec des projets antérieurs. Cependant, nous pouvons tenter de faire quelques comparaisons entre le prévisionnel et le réalisé sur la période de recette.

Tout d'abord, concernant les charges d'exécution des tests, la MOE comptait 8 jours*homme (j*h) pour les chefs de projet MOA et AMOA, et 10 j*h par testeur. Or, d'après mes estimations, les consommations internes sont les suivantes :

- chef de projet AMOA : 14 j*h (séances de test + gestion des anomalies + maintenance du référentiel des exigences/tests),
- chef de projet MOA : 4*h (formation recette + séances de tests),
- testeur RCA : entre 3 et 5 j*h par testeur de l'équipe de recette (formation recette + séances de tests).

A la vue de ces chiffres, on constate donc que la charge consommée est finalement moins importante que prévue, de l'ordre de 20 à 50% selon les profils. Pour être complet, il faudrait faire le même exercice pour les charges des autres activités de recette (préparation et spécification) mais je ne dispose pas d'information suffisamment précise pour le faire. Par exemple, mon relevé d'activités par projet ne détaille pas mes activités de recette en tâches de préparation, spécification et exécution des tests…

Enfin, concernant les délais de réalisation de la recette, le planning initial prévoyait quatre mois de période de VABF contre un peu plus de trois finalement réalisés (du 20/03 au 25/06), soit une diminution de 20%.

Plus finement, cette période s'est décomposée de la manière suivante :

- tests de la version 1.0 : du 20/03 au 19/04, soit environ un mois,
- tests de la version 1.1 : du 14/05 au 07/06, soit environ trois semaines,
- tests de la version 1.2 : du 11/06 au 15/06, soit environ une semaine,
- tests de la version 1.3 : du 15/06 au 25/06, soit environ une semaine.

L'objectif 5 est à moitié atteint. En effet, la note de synthèse que j'ai rédigé au début du projet RFSI (cf. §4.1.1) permet effectivement d'appréhender l'état de l'art en matière de tests et de recette fonctionnelle mais je ne suis pas sûr que les autres chefs de projet Études se soient réellement appropriés ces informations.

Plus globalement, la conduite du changement du projet RFSI n'a pas été suffisante. C'est pourquoi, je vais écrire, sous l'impulsion du chef du pôle Études, une feuille de route visant à déployer les outils et les nouvelles pratiques de test au sein du pôle Études, puis au niveau DSI. Cette feuille de route sera mise en œuvre par chaque chef de projet et je les accompagnerai dans leur démarche.

7.6.3 Analyse des coûts

Le coût du projet se traduit essentiellement en charge de travail consommée en j*h. En effet, les outils mis en place sont *open source* et n'ont entrainé aucune dépense financière. Concrètement, la charge de travail globale correspond au temps passé par moi-même, les autres chefs de projet et l'expert décisionnel du pôle Études. La charge consommée par les autres acteurs impliqués dans le projet RFSI étant négligeable compte tenu de l'étalement de la réalisation du projet.

La charge globale prévisionnelle du projet RFSI a été estimée à environ 50 j*h, or la charge réelle constatée est d'environ 90 j*h, soit 4,5 mois*homme, ce qui correspond à une multiplication de la charge par un facteur de 1,8. Malgré cela, on reste toujours dans l'ordre de grandeur d'un petit projet [31].

Ce dépassement est essentiellement dû à la phase d'étude (cf. §4.1 et 4.2) où certaines activités ont consommé trop de charge (facteur multiplicateur de 1,6).

Par exemple, j'ai surinvestit dans la recherche, le contrôle, la synthèse et la restitution des informations collectées sur internet et dans la littérature existante pour aboutir à l'état de l'art en matière de test et de recette fonctionnelle d'un SI. Je devrais pouvoir faire des gains importants avec un peu plus d'expérience et en réduisant le périmètre des recherches au strict nécessaire. Par ailleurs, il est également possible de s'abonner à des services fournissant ce genre de travail de veille « clé en main », je pense notamment aux sociétés spécialisées Gartner et CXP… La charge interne est alors transformée en coût financier.

De même, j'ai probablement passé trop de temps sur la modélisation des processus de recette, ne maîtrisant pas encore le langage BPMN auquel j'ai été formé les 12 et 13 novembre 2009. Par contre, le retour sur cet investissement se fera sentir dans les autres projets. D'ailleurs, pour le projet OSIDI la modélisation du processus d'instruction (cf. §5.3.2) n'a nécessité qu'une journée, réunion de spécification incluse.

Enfin, il aurait fallu passer moins de temps sur la rédaction de l'étude détaillée car j'ai finalement su assez tôt qu'une phase marché ne serait pas nécessaire puisque que le budget du projet prévu en 2011 avait été reporté à 2012. En effet, un des intérêts d'une étude détaillée poussée consiste dans sa transformation rapide en cahier des charges pour un appel d'offres.

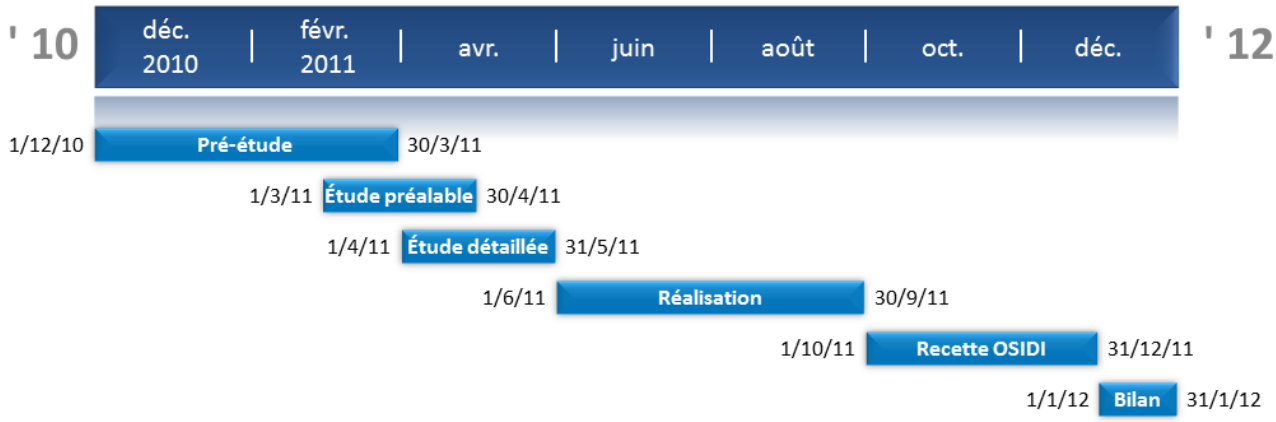
7.6.4 Analyse des écarts de planning

Le planning initial du projet RFSI a été établi lorsque j'ai proposé mon sujet de mémoire d'ingénieur au CNAM Champagne-Ardenne début décembre 2010. Ce planning est décomposé de manière classique pour un projet Études (§1.3.3), à savoir une phase d'études, la réalisation du projet et son bilan. A cela, j'ai ajouté une phase de pré-étude nécessaire pour appréhender l'état de l'art en matière de recette fonctionnelle d'un SI et donner une base de connaissances solide au projet RFSI. En décembre 2010, le planning devait répondre à deux contraintes principales :

1. la phase de réalisation de la solution retenue devait être achevée à temps afin de permettre son utilisation pendant la recette du projet OSIDI,

2. le projet devait être court et consommer un minimum de ressources humaines.

Ainsi, j'ai estimé la charge approximative pour chaque étape et les délais nécessaires en fonction des contraintes ci-dessus, en utilisant les techniques exposées au §2.3.2, pour aboutir au planning initial donné ci-après.



Ce planning initial a été modifié quatre fois au cours du projet RFSI pour répondre à un certain nombre de contraintes apparues en cours de projet et expliquées ci-après. Le planning final du projet RFSI est donné ci-dessous.

Illustration 30: Planning final du projet RFSI (juin 2012)

A la lecture de ce planning final, nous constatons que :

- la pré-étude s'est déroulée conformément au prévisionnel, ce qui n'est pas surprenant étant donné que j'étais la seule ressource sur cette tâche (= risque de dépassement maîtrisé),
- l'étude préalable a glissé d'une semaine, temps nécessaire pour réunir le comité de projet et préparer la validation de cette étape (= rien de préoccupant),
- la durée de l'étude détaillée a été prolongée de trois mois : c'est le premier impact important,

- la phase de réalisation a débutée avec deux mois de décalage par rapport à la fin de la phase précédente : c'est le deuxième impact important,

- la recette OSIDI s'est déroulée d'avril à juin 2012 et non sur le dernier trimestre 2011 comme initialement prévu : c'est le troisième impact important,

- le bilan se fera en septembre 2012 en conséquence des bouleversements de planning vus précédemment.

Tous ces glissements du planning du projet RFSI s'expliquent par les justifications suivantes :

- j'ai sous-estimé le nombre de réunions des différents groupes de travail et la charge des activités afférentes (animation, compte-rendu, modélisation des processus, rédaction de l'étude détaillée...), il a donc fallu solliciter des ressources humaines déjà mobilisées sur des projets prioritaires (résultat : environ deux mois de retard),

- la période de congés d'été du 15 juillet au 15 août 2012, soit un mois, a été « chômée » pour le projet RFSI,

- le planning initial ne tenait pas compte dans la phase de réalisation d'une étape supplémentaire de développement d'un infocentre recette puisqu'on pensait s'appuyer sur les rapports standards des outils Squash TM et Mantis qui se sont montrés insuffisants par rapport aux besoins (résultat : environ un mois de travail supplémentaire),

- le projet OSIDI ayant pris un retard important dû à l'annulation d'une CAO, la recette OSIDI n'a pas pu commencer avant avril 2012, d'où le décalage voulu de deux mois de la phase réalisation,

- par ailleurs, j'ai été extrêmement occupé par le projet OSIDI[57], quasiment à 100% sur la période de septembre à décembre 2011 (lancement de la réalisation + spécifications fonctionnelles et techniques) ce qui ne m'a pas permis d'avancer sur le projet RFSI.

57 Pour rappel, le projet OSIDI est un projet prioritaire de la DSI et est très important pour la RCA, contrairement au projet RFSI qui est un projet plutôt mineur du point de vue des enjeux de la collectivité.

Conclusion

La Région Champagne-Ardenne est une collectivité territoriale dont le cœur de métier consiste à définir et à mettre en œuvre des politiques pour répondre aux besoins de ses administrés dans l'intérêt général et dans les limites des compétences qui lui sont assignées par les lois de décentralisation. Dans cette optique, les élus du Conseil Régional s'appuient sur une organisation administrative pour réaliser leurs projets politiques : c'est la Région Champagne-Ardenne.

De fait, cette administration est en constante évolution pour supporter la politique des élus et répondre à ses propres objectifs de modernisation de son organisation et de ses méthodes de travail. Dans ce contexte, les systèmes d'informations participent à la réalisation de ces objectifs en supportant les processus métiers et les échanges d'informations entre les différents acteurs.

Cela passe essentiellement par des projets de mise en place de systèmes informatiques (matériels et logiciels) managés par la direction des systèmes d'information, et notamment par le pôle Études en charge des projets fonctionnels (par opposition aux projets d'infrastructure réseaux et systèmes).

Dans ce cadre, il est essentiel pour le pôle Études que les projets menés avec les directions métiers puissent aboutir en répondant, d'une part aux besoins métiers et en respectant d'autre part, les objectifs de coûts, de qualité et de délais fixés à chaque projet.

Pour cela, le pôle Études a développé sa propre démarche de gestion de projet en s'inspirant des éléments des méthodes de gestion de projet existantes tout en prenant en compte les aspects organisationnels, humains, techniques et financiers spécifiques de l'écosystème régional (acteurs nombreux, entités organisationnelles variées, chaîne de décision complexe, cadre réglementaire ad hoc).

Dans ce contexte, le chef de projet Études est confronté aux problématiques classiques de la gestion de projet (ingénierie des besoins, estimation des charges, planification, conduite du changement...) pour lesquels il dispose de moyens et de techniques adaptés. Néanmoins, il est beaucoup moins armé pour prendre en charge les activités de recette fonctionnelle.

La recette d'un système d'information peut être définie comme un ensemble d'actions permettant d'effectuer la vérification de la conformité du système livré avec l'objectif de valider celui-ci.

Cette une étape clé dans un projet et elle est en général difficile, d'autant plus que son manque de maîtrise se traduit le plus souvent par des surcoûts financiers, des dysfonctionnements logiciels, des retards importants, voire des conséquences plus ou moins graves quand des anomalies bloquent un système d'information en production.

Une solution pour répondre à ces enjeux consiste à mettre en place l'organisation, les bonnes pratiques, les techniques et les outils conduisant à une meilleure maîtrise de la recette fonctionnelle et des risques associés. C'est ce que nous avons tenté de faire avec le projet RSI.

Ce projet a été lancé pour répondre à un certain nombre de questions : comment faire face au manque de formalisation et de contractualisation de l'étape de recette, comment gérer des activités pas suffisamment rationalisées, comment corriger les faiblesses dans l'exécution des processus de test...

Pour répondre à cette problématique, nous avons proposer trois pistes de réflexion, l'organisation, la méthodologie et l'outillage, qui nous ont conduit à imaginer et concevoir une solution originale au problème de la recette fonctionnelle en définissant des processus de recette documentés et outillés par des logiciels adaptés aux besoins des utilisateurs : la suite de logiciels Squash, Mantis et l'infocentre recette.

Actuellement, une partie de cette solution a été mise en œuvre et expérimentée dans le cadre de la recette fonctionnelle du projet de mise en œuvre du portail des aides régionales (nom de code OSIDI). Les résultats de l'expérimentation ont conforté la solution définie par le projet RFSI. Toutefois, des adaptations seront nécessaires avant de pouvoir déployer celle-ci au sein du pôle Études, puis au niveau de la DSI.

Tout d'abord, la documentation doit être mise à jour en fonction de ce premier retour d'expérience, notamment pour affiner les responsabilités MOA/AMOA vs MOE dans certains processus de recette. Par exemple, les activités de spécifications des tests métiers seront désormais de la responsabilité de la MOA/AMOA contrairement à ce qui était initialement envisagé. De même, il est nécessaire de détailler concrètement les activités d'accompagnement du plan de conduite du changement, en instaurant par exemple des réunions de travail spécifiques tout au long de l'exécution des processus de recette.

Par ailleurs, le projet RFSI est loin d'être achevé. Aujourd'hui, seul la premier lot, qui couvre la documentation (processus de recette, modèles..), la mise en place des outils Squash TM, Mantis et l'infocentre recette a été réalisé . Pour être complet, il faut encore déployer les outils existants (pôle Études et DSI) et réaliser les deux lots suivants, à savoir le mise en place des outils d'automatisation des tests de non-régression et de gestion des données de test. Ce dernier point étant programmé début 2013.

A très court terme, je prévois d'écrire une feuille de route visant à déployer les outils Squash TM, Mantis et l'infocentre recette ainsi que les nouvelles pratiques de test au sein du pôle Études, puis au niveau de la DSI. Cette feuille de route sera mise en œuvre par chaque chef de projet et je les accompagnerai dans leur démarche à partir du mois de septembre 2012. Ce sera ainsi l'occasion d'éprouver de nouveau, tout ou partie de la solution dans des contextes différents.

De plus, je pense également faire monter en version le logiciel Squash TM. En effet, depuis sa mise en production, deux versions mineures (v.1.1 et v1.1.1) et une majeure (v1.2) sont disponibles en ligne. Il faudra également effectuer les éventuelles opérations de maintenance sur l'infocentre recette, qui je le rappelle, exploite les données de Squash TM et Mantis pour la réalisation du tableau de bord recette.

Enfin, le projet OSIDI est toujours d'actualité. La mise en production de nouvelles aides régionales fin 2012 va nécessiter une nouvelle étape de recette fonctionnelle, j'espère ainsi exploiter l'expérience acquise pour être plus efficace. En tout état de cause, le retour sur investissent du projet RFSI dépendra du déploiement réussi des outils et des pratiques, et des résultats obtenus par les équipes du pôle Études et de la DSI à l'avenir.

8 Annexes

8.1 Définitions possibles d'un système d'information

Un SI est un ensemble de composants (informations, logiciels, matériels, documentation) permettant de supporter les activités métier des individus d'une organisation. De ce point de vue, il peut être abordé selon quatre grandes dimensions : humaine, organisationnelle, technologique et financière [23] comme le montre le schéma ci-après.

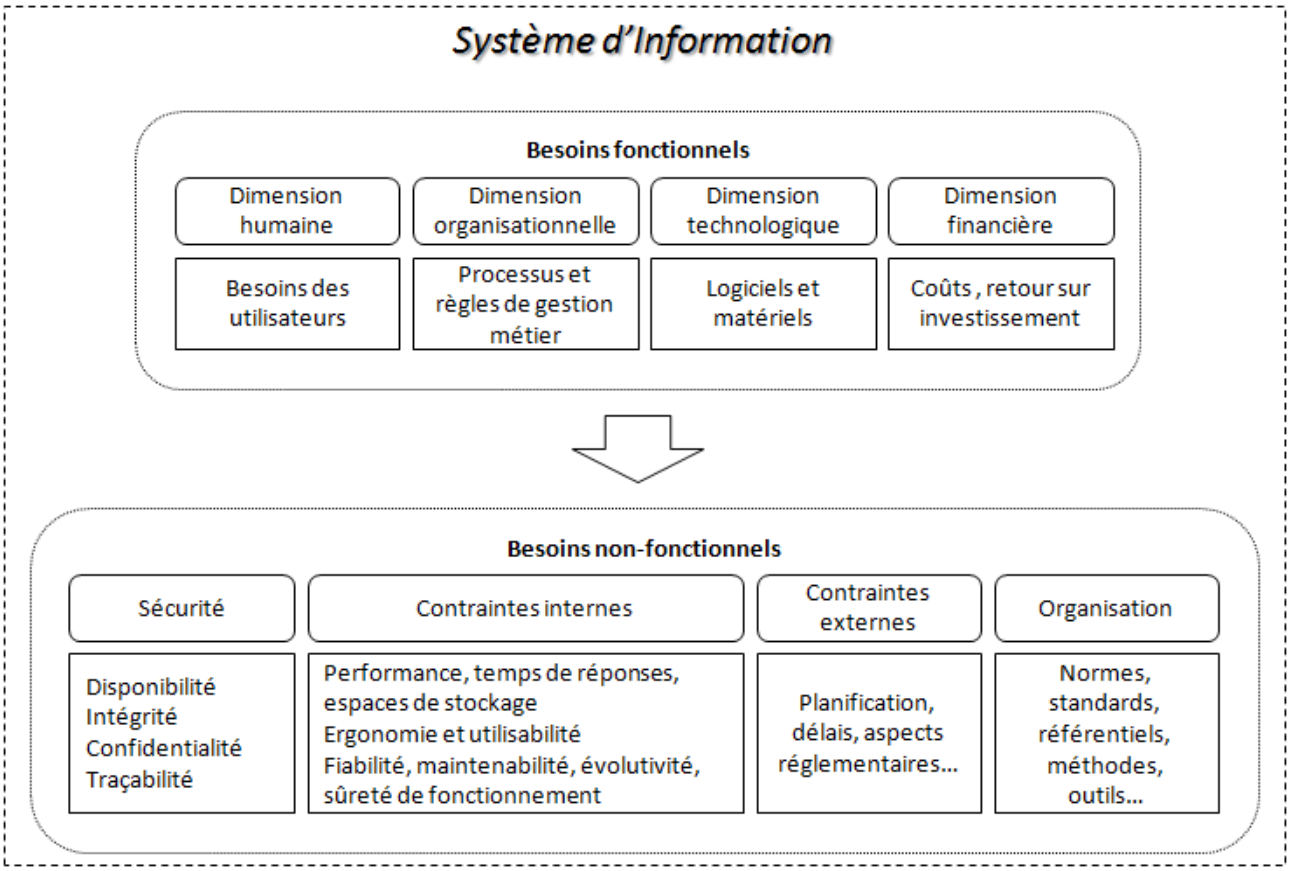


Illustration 31: Architecture fonctionnelle et non-fonctionnelle d'un SI

Le schéma précédent met en évidence la distinction opérée entre les besoins fonctionnels et non-fonctionnels. Les besoins fonctionnels sont exprimés par des utilisateurs et induits par les processus et règles de gestion métier de l'organisation. Ils sont supportés par des logiciels (applications métier...) et des matériels (serveur, réseau...) qui possèdent un coût et dont on espère un retour sur investissement à plus ou moins long terme. Les besoins non-fonctionnels représentent toutes les contraintes devant être respectées par le SI.

Une autre manière de définir le SI consiste à le décomposer en différentes couches communicantes allant de la vision métier de l'organisation jusqu'à l'implémentation technique des logiciels et matériels en passant par les fonctionnalités et services fournis par le SI ainsi que les applications métier mises en œuvre [29].

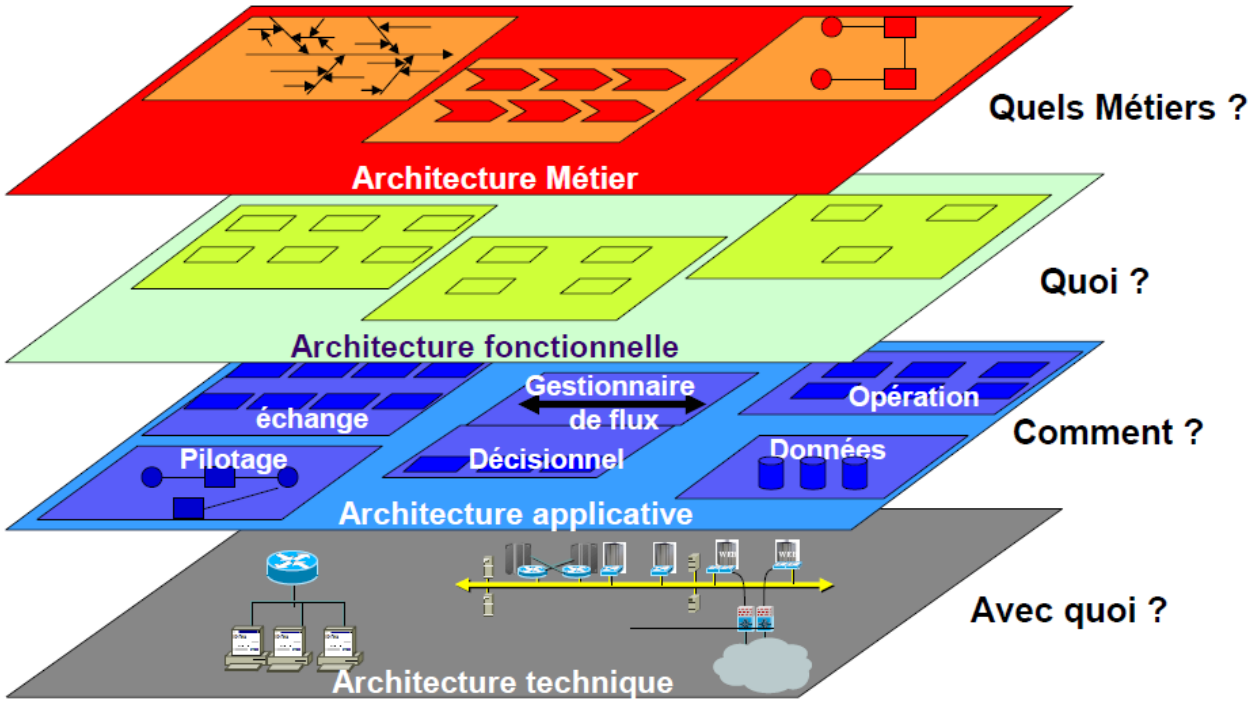


Illustration 32: Architecture urbanisée d'un SI

Cette vision urbanisée du SI représente l'ensemble des moyens (organisation, acteurs, procédures, systèmes informatiques) nécessaires au traitement et à l'exploitation des flux d'informations dans le cadre d'objectifs définis au niveau de la stratégie de l'organisation, des métiers, de la réglementation.

8.2 Partitionnement par classes d'équivalence et test aux limites

Par illustrer simplement ces techniques, nous pouvons imaginer le cas de test logique décrit succinctement dans le tableau ci-après.

Objectif du test	Valider la règle de gestion du solde d'une prestation.
Exigences couvertes par le test	Exigence #1 : règle de gestion du solde d'une prestation. Si le pourcentage de réalisation de la prestation (**paramètre « %Réalisé »**) est supérieur ou égale à 90% alors la prestation est soldée à 100%, sinon elle est soldée à hauteur du pourcentage réel de réalisation.
Résultat attendu	Le montant de la facture de la prestation est correct.

Ce cas de test logique fait intervenir le paramètre « %Réalisé » qui sera valorisé dans plusieurs cas de test physiques en appliquant les techniques partitionnement par classes d'équivalence et de test aux limites.

D'après l'exemple précédent, nous pouvons déduire de la description de l'exigence, l'existence de deux classes d'équivalence :

- classe d'équivalence #1 : [0% ;90%],
- classe d'équivalence #2 : [90% ; 100%].

L'application des techniques permet d'identifier les valeurs représentatives de chaque classe, c'est-à-dire trois extrêmes de classe et deux valeurs aux bornes. Il faut noter que les valeurs inférieures à 0% ou supérieures à 100% correspondent à des cas de test physiques non-passants qu'il faudra spécifier en conséquence.

Dans ces conditions, les valeurs pertinentes du paramètre « %Réalisé » sont pour le cas passant :

- cas de test physique #1 : %Réalisé = 0%,
- cas de test physique #2 : %Réalisé = 89,9%,
- cas de test physique #3 : %Réalisé = 90%,
- cas de test physique #4 : %Réalisé = 90,1%
- cas de test physique #5 : %Réalisé = 100%.

Nous constatons ainsi que cinq cas de test physiques passants suffisent à couvrir l'ensemble des valeurs possibles du paramètre « %Réalisé ».

8.3 Test par paires

Le pairwise testing ou test par paires est une technique qui permet de réduire le nombre de combinaisons de valeurs en entrée du logiciel à tester. Il repose sur le principe, tiré de l'expérience, qui stipule que la majorité des erreurs sont détectées par des combinaisons de deux valeurs de variables.

A titre d'exemple, imaginons une interface graphique proposant trois champs associés à des boutons radio (prenant la valeur 0 ou 1) qui déclenchent des comportements différents du logiciel en fonction de leur combinaison. Un utilisateur qui souhaiterait tester tous les cas de figure devrait réaliser au total 8 tests (2^3).

Par ailleurs, il y a 12 paires de valeurs possibles (3×2^2) qui sont données dans les tableaux ci-après. Les trois boutons radios sont nommés A, B et C.

A	B
0	0
0	1
1	0
1	1

A	C
0	0
0	1
1	0
1	1

B	C
0	0
0	1
1	0
1	1

Or, d'après ces tableaux, on constate que :

- la donnée de test (A=0, B=1, C=1) couvre 3 paires :
  - (A=0, B=1), (A=0, C=1) et (B=1, C=1),
- la donnée de test (A=0, B=0, C=0) couvre 3 paires :
  - (A=0, B=0), (A=0, C=0) et (B=0, C=0),
- la donnée de test (A=1, B=0, C=0) couvre 2 paires :
  - (A=1, B=0) et (A=1, C=0),
- la donnée de test (A=1, B=1, C=1) couvre 2 paires :
  - (A=1, B=1) et (A=1, C=1).

Il y a donc 2 paires qui ne sont pas couvertes : (B=1, C=0) et (B=0, C=1).

Par conséquent, il faudra 6 tests pour couvrir l'ensemble des 12 paires possibles pour obtenir un résultat équivalent au test exhaustif (8 cas soit 25% de moins). Bien entendu, cette technique est d'autant plus efficace qu'il y a de paramètres en entrée du logiciel.

Il faut enfin noter que des outils disponibles sur le marché permettent de faire ces calculs automatiquement (cf. http://www.pairwise.org/tools.asp).

8.4 Quelques logiciels utiles pour les tests

Le tableau ci-après donne la liste des logiciels de test les plus connus sur le marché en fonction du type de test supporté (source [13]).

Les produits de tests

	Borland	Compuware	HP Mercury	IBM Rational	Open Source
Analyse statique		DevPartner		Rational Application Developer	
Tests unitaires		DevPartner	HP Quality Center	Rational Purify, Rational Application Developer, Rational Test Real Time	Junit
Gestion des tests	SilkCentral Test Manager	QADirector, QACenter	TestDirector	Rational Test Manager, Rational ClearQuest Test Management	DejaGnu, TestLink
Couverture de code		DevPartner		Rational PurifyPlus, Rational Application Developer, Rational Test Real Time	
Tests fonctionnels	SilkTest	TestPartner	HP Quality Center	Rational Functional Tester, Rational Robot	FitNesse
Tests d'intrégration				Rational Application Developer, WebSphere Developer for zSeries	Cactus
Tests de non-régression	SilkTest		HP Quality Center	Rational Functional Tester	
Tests de performance	SilkPerformer	QALoad, DevPartner	HP Performance Center	Rational Performance Tester	OpenSTA, JMeter
Tests de charge	SilkPerformer	QACenter	HP Performance Center	Rational Performance Tester	OpenSTA, JMeter
Tests Web		TestPartner	HP Quality Center	Rational Application Scan, Rational Policy Tester	OpenSTA, Jmeter, Selenium
Tests processus métier			HP Business Process Testing	Tivoli Intelligent Orchestrator	
Gestion des anomalies	SilkCentral Test Manager	CA Software Change Manager	TestDirector	Rational ClearQuest	Bugzilla, TRAC, Scarab, Mantis
Gestion de la qualité	StarTeam+Silk	CARS		Rational ClearQuest, Rational Test Manager, Rational ClearQuest Test Management	
Tableau de bord		QAPortal	HP Quality Center Dashboard		
Portail		QAPortal			
Requirement Based Testing	oui				
Model Based Testing	oui, avec LEIRIOS Smart Testing	oui, avec LEIRIOS Smart Testing			
Risk Based Testing	oui	oui			
Test Driven Requirements					FitNesse
CMMI	oui		HP Quality Center - HP PPM	Plugin Rational Method Composer (CMMI 2 et 3)	
Méthodologie		CARS	BTO		

8.5 Un exemple de données de test

Le tableau ci-dessous précise les données de test à utiliser pour le cas de test « Effectuer une demande en ligne ».

Cas de test	Règle de gestion	Données de test (cas passant)	Nb tests nécessaires	Données de test (cas non-passant)	Nb tests nécessaires
Effectuer une demande en ligne	La fiche « Mes informations doit être complétée » au préalable pour accéder au formulaire	Fiche complétée	1	Fiche non complétée	1
	La date de saisie de la demande doit se situer entre la date de début et la date de fin de dépôt de l'aide	Date de demande = Date de début Date de demande = Date de début +1 jour Date de demande = Date de fin - 1 jour Date de demande = Date de fin	4	Date de demande = Date de début -1 jour Date de demande = Date de fin +1 jour	2
	Aucune autre demande « StudéO-Aide à l'achat d'un ordinateur portable » du même type ne doit avoir été effectuée sur la même période	Aucune demande en cours ou instruite sur la même période	1	2 cas : en cours sur la même période instruite sur la même période	2
	Saisie du code métier (Ordinateur portable)	Code métier correct	1	Code métier incorrect	1

Bibliographie

	Recette fonctionnelle des systèmes d'information
[1]	J. AKOKA et I. WATTIAU-Notes de cours : Ingénierie des systèmes d'information : Méthodes avancées, La recette fonctionnelle d'un Système d'Information–CNAM-2009
[2]	B. LEGEARD, F. BOUQUET et N. PICKAERT-Industrialiser le test fonctionnel, des exigences métier au référentiel de tests automatisés-Dunod-2009
[3]	C. LELOUP-Gouvernance des systèmes d'information, Recette d'une application : période de grand danger, revue de l'AFAI, vol. 83, pages 12 à 14-2006-http://www.afai.fr/public/doc/316.pdf
[4]	B. LEBLANC-Fiche de capitalisation Delf, Fiche la recette fonctionnelle-Delf-2007-http://www.delf.fr/IMG/pdf/recette_-la_recette_fonctionnelle.pdf
[5]	J. WATKINS-Test logiciel en pratique-Vuibert-2009
[6]	J. LAVISSE-Exposés Systèmes et Réseaux de l'école d'ingénieurs 2000, Les tests et les logiciels de gestion de tests-Université Paris-Est Marne-la-Vallée-2008
[7]	B. PASSION et O. PATRIS-Dossier GLG101 Test et validation du Logiciel, Panorama des outils de tests-CNAM-2007-http://home.nordnet.fr/~ericleleu/cours/glg101/Panorama_outils_tests.pdf
[8]	L. AUVIGNE-EZRAN-Dossier GLG101 Test et validation du Logiciel, Démarche de recette-CNAM-2007-http://home.nordnet.fr/~ericleleu/cours/glg101/Demarche.pdf
[9]	S. CALIMET, P. FIRMIN et E. LELEU-Dossier NFE209 Audit et gouvernance des Systèmes d'Information, Les tests : l'état de l'art, tests et validation du logiciel-CNAM-2009-http://home.nordnet.fr/~ericleleu/cours/nfe210/Tests_V10.00.pdf
[10]	Livre Blanc de BORLAND-Réussir l'automatisation du processus de test fonctionnel-2006-http://www.borland.com/resources/fr/pdf/white_papers/functional_testing_whitepaper.pdf
[11]	B. LEGEARD-Génération automatique de tests à partir de spécifications : Principes et applications industrielles-LIFC Université de Franche-Comté-2004-http://lifc.univ-fcomte.fr/home/afadl2004/textes/autres/TutorialLegeard.pdf

[12]	G. MANTEL et P. BOUILLER-Automatisation des recettes fonctionnelles : un levier pour la conduite du changement-Valtech Technology-2006-http://www.valtech.fr/etc/medialib/library/productivity/fr.Par.58248.File.tmp/IN04-TestsFonctionnels-Article.pdf
[13]	P. TRAN-Le test logiciel en voie d'industrialisation, revue l'Informaticien, vol. 57, pages 58 à 63-2008
[14]	Y. OKBA-Synthèse bibliographique : la recette fonctionnelle d'un système d'information-RCA-2011
	Ingénierie des exigences
[15]	J. AKOKA et I. WATTIAU-Notes de cours : Ingénierie des systèmes d'information : Méthodes avancées, Ingénierie des besoins–CNAM-2009
[16]	R. BUJOLD-Ingénierie des exigences, Une méthode simple et systématique, revue canadienne de l'IEEE, vol. 48, pages 6 à 10-2004
[17]	PH. DES MESNARDS-Réussir l'analyse des besoins-Eyrolles-2007
[18]	J. GABAY-Maîtrise d'ouvrage des projets informatiques, guide pour le chef de projet-Dunod-2009
[19]	Livre Blanc de BORLAND-Une définition et une gestion efficace des exigences améliorent les systèmes et la communication-2006-http://www.borland.com/fr/rc/requirements-definition-management/index.html
[20]	J. HEUMANN-The five levels of Requirements Management Maturity-IBM Rational Software-2003-http://www.ibm.com/developerworks/rational/library/content/RationalEdge/feb03/ManagementMaturity_TheRationalEdge_Feb2003.pdf
[21]	R. BUJOLD-Ingénierie des exigences, Le gratuiciel GenSpec, revue canadienne de l'IEEE, vol. 51, pages 13 à 16-2005
[22]	***Ingénierie des Système d'Information***
[23]	J. AKOKA et I. WATTIAU-Notes de cours : Ingénierie des systèmes d'information : Audit et Gouvernance, Systèmes d'Information–CNAM-2008

[24]	L. TOURNANT-Notes de cours : Ingénierie des systèmes d'information : Audit et Gouvernance, Management et gestion de projet–CNAM-2008
[25]	C. MORLEY-Management d'un Projet Système d'Information - Principes, techniques, mise en œuvre et outils-Dunod-2008
[26]	The Chaos Report-The Standish Group-éditions de 1994 et 2009-http://www.standishgroup.com/
[27]	Rapport de veille technologique, Les méthodes agiles de développement-Vivansa, pages 13 à 15-2010-http://www.noratek.net/wp-content/uploads/2010/06/13514-DLV-20209_RPT_Methodes_Agiles-1.00.pdf
[28]	Y. DROTHIER-Comprendre l'industrialisation informatique-JDN Solutions-2005-http://www.journaldunet.com/solutions/dossiers/pratique/industrialisation-informatique.shtml
[29]	J. AKOKA et I. WATTIAU-Notes de cours : Ingénierie des systèmes d'information : Audit et gouvernance, Urbanisation des Système d'Information–CNAM-2008
[30]	JF. PILLOU-Tout sur les Systèmes d'Information-Dunod-2006
[31]	J. AKOKA et I. WATTIAU-Notes de cours : Ingénierie des systèmes d'information : Méthodes avancées, Projet informatique : estimation de l'effort–CNAM-2009
[32]	J. AKOKA et I. WATTIAU-Notes de cours : Ingénierie des systèmes d'information : Méthodes avancées, La qualité des systèmes d'information–CNAM-2009
[33]	J. AKOKA et I. WATTIAU-Notes de cours : Ingénierie des systèmes d'information : Conduite du changement–CNAM-2008
[34]	B. BOEHM-Software Engineering Economic-Prentice Hall-1981
[35]	P. ROQUES-UML 2 par la pratique : Études de cas et exercices corrigés-Eyrolles-2009
	Normes et standards
[36]	J. AKOKA et I. WATTIAU-Notes de cours : Ingénierie des systèmes d'information : Audit et Gouvernance, Stratégie des systèmes d'information–CNAM-2008

[37]	Norme IEEE 830-Pratique recommandée par IEEE pour la préparation de spécifications d'exigences de logiciel-IEEE Computer Society-1993-http://www.cours.polymtl.ca/log3410/bibliographie/IEEE/Pratique_Recommandee_Par_IEEE_pour_la_Specification.pdf
[38]	Norme IEEE 1233-Guide de l'IEEE pour la spécification d'exigences de systèmes-IEEE Computer Society-1998-http://www.cours.polymtl.ca/log3410/bibliographie/IEEE/Guide_IEEE_Pour_la_Specification.pdf
[39]	Norme IEEE 610-IEEE Standard Glossary of Software Engineering Terminology-IEEE Computer Society-1990-http://www.idi.ntnu.no/grupper/su/publ/ese/ieee-se-glossary-610.12-1990.pdf
[40]	ISO 9126-Information technology. Software product evaluation : Quality characteristics and guidelines for their use-International Organization for Standardization-1994
[41]	Norme IEEE 1044-Classification for Software Anomalies-IEEE Computer Society-2009 -http://www.baskent.edu.tr/~zaktas/courses/Bil573/IEEE_Standards/1044_2009.pdf
[42]	Glossaire CFTL/ISTQB des termes utilisés en tests logiciels-CFTL/ISTQB-2007-http://www.gasq.org/boards/cftl/cms/files/Dokumente/Glossaire des tests de logiciel - 2 0 F ISTQB.pdf
[43]	Direction Générale de la Modernisation de l'État (DGME)-Référentiel Général d'Accessibilité pour les Administrations-2010-http://references.modernisation.gouv.fr/rgaa-accessibilite
[44]	Direction Générale de la Modernisation de l'État (DGME)-Référentiel Général d'Interopérabilité-2010-http://references.modernisation.gouv.fr/rgi-interoperabilite
[45]	Direction Générale de la Modernisation de l'État (DGME), Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI)-Référentiel Général de Sécurité-2010-http://references.modernisation.gouv.fr/rgs-securite
[46]	Direction Générale de la Modernisation de l'État (DGME)-Charte ergonomique-2010-http://references.modernisation.gouv.fr/charte-ergonomique
[47]	Norme IEEE 829-2008-Standard for Software and System Test Documentation-IEEE Computer Society-2008

[48]	Norme AFNOR Z 67-130-Système de traitement de l'information, Recommandation de Plan Qualité Logiciel-AFNOR-1987
[49]	Norme ISO 9241-110-Ergonomie de l'interaction homme-système-International Organization for Standardization-2006
[50]	Norme ISO 10006-Systèmes de management de la qualité-Lignes directrices pour le management de la qualité dans les projets-International Organization for Standardization-2003
[51]	R. DUBREUILLE-Notes de cours : Information et communication pour l'ingénieur– CNAM-2009
[52]	Norme AFNOR Z 67-130-Système de traitement de l'information, Recommandation de Plan Qualité Logiciel-AFNOR-1987
[53]	Norme AFNOR NF X50-150-Analyse de la valeur – Analyse fonctionnelle. Vocabulaire-AFNOR-1990
[54]	Norme IEEE 830-IEEE Recommended Practice for Software Requirements Specifications -IEEE Computer Society-1998
[55]	Décret n°2006-975 du 1 août 2006 portant code des marchés publics-http://www.legifrance.gouv.fr/affichTexte.do;jsessionid=E08B677B9CF6091995EE165D1865BB03.tpdjo09v_1?cidTexte=JORFTEXT000000264576&dateTexte=20120504
[56]	Y. PARE-Fondamental de gestion des marchés publics-ENACT de Nancy-2006

Table des matières

Index des illustrations

La recette fonctionnelle d'un Système d'Information : enjeux, méthodes, outils et exemple d'application

---

Résumé

La direction des systèmes d'information de la Région Champagne-Ardenne mène depuis fin 2010 un projet visant à industrialiser l'étape de recette fonctionnelle de ses projets informatiques.

La recette d'un système informatique peut être définie comme un ensemble d'actions et de tests permettant de vérifier la conformité d'un système informatique aux besoins fonctionnels, avec l'objectif de valider celui-ci. Les résultats des tests sont alors appréciés en fonction de critères d'acceptation aboutissant ou non à la validation du système informatique.

Dans ce contexte, l'objectif principal de ce projet est de définir un cadre de travail pour la maîtrise d'ouvrage permettant de vérifier, d'évaluer et de valider correctement un système informatique réalisé par une maîtrise d'œuvre.

Le présent document se propose donc d'expliciter le contexte et la problématique d'un tel projet au sein d'une administration publique comme la Région Champagne-Ardenne. Il s'agira ensuite de faire une synthèse de l'état de l'art en matière de tests appliqués à la recette fonctionnelle.

Enfin, les travaux réalisés permettant de définir et d'évaluer une solution (organisation, méthodes, techniques, outils…) pour faire face à cette problématique seront présentés et commentés en fin de document.

Mots clés : recette, test d'acceptation, qualification, validation, gestion de projet

---

Summary

The information system department of the "Région Champagne-Ardenne" has launched a project at the end of 2010 to industrialize the acceptance testing of its IT projects.

Acceptance testing is a formal testing with respect to user functional needs to determine whether or not a computer system satisfies the acceptance criteria and to enable an authorized entity to determine whether or not to accept the system.

In this context, the main objective of this project is to define a framework for the contracting owner to verify, assess and validate a computer system designed and produced by the project management.

The context of a public administration like the "Région Champagne-Ardenne" and the project issues are developed in this document. Then, this document will introduce the state of the art and the best particles on acceptance testing.

Finally, this document will present an overview of all the work that has been done during the project to define and to assess a solution (organization, methods, test design techniques, testing tools…) in order to cope this specific problem.

Keywords : acceptance testing, qualification, validation, project management