

UNIVERSITE DU QUÉBEC À MONTREAL

APPROCHE SYSTÉMIQUE DE LA CONSCIENCE DE LA SITUATION POUR
LA RÉDUCTION DES ERREURS DE DÉCISION EN ENVIRONNEMENT
COMPLEXE : CAS DES PROJETS INFORMATIQUES

THÈSE
PRESENTÉE
COMME EXIGENCE PARTIELLE
DU DOCTORAT EN INFORMATIQUE COGNITIVE

PAR
KOFFI AOKOU

SEPTEMBRE 2018

UNIVERSITÉ DU QUÉBEC À MONTRÉAL
Service des bibliothèques

Avertissement

La diffusion de cette thèse se fait dans le respect des droits de son auteur, qui a signé le formulaire *Autorisation de reproduire et de diffuser un travail de recherche de cycles supérieurs* (SDU-522 – Rév.07-2011). Cette autorisation stipule que «conformément à l'article 11 du Règlement no 8 des études de cycles supérieurs, [l'auteur] concède à l'Université du Québec à Montréal une licence non exclusive d'utilisation et de publication de la totalité ou d'une partie importante de [son] travail de recherche pour des fins pédagogiques et non commerciales. Plus précisément, [l'auteur] autorise l'Université du Québec à Montréal à reproduire, diffuser, prêter, distribuer ou vendre des copies de [son] travail de recherche à des fins non commerciales sur quelque support que ce soit, y compris l'Internet. Cette licence et cette autorisation n'entraînent pas une renonciation de [la] part [de l'auteur] à [ses] droits moraux ni à [ses] droits de propriété intellectuelle. Sauf entente contraire, [l'auteur] conserve la liberté de diffuser et de commercialiser ou non ce travail dont [il] possède un exemplaire.»

REMERCIEMENTS

Cette thèse n'aurait pu se concrétiser sans le concours de certaines personnes et organismes.

Pour leurs conseils et encadrements scientifiques, j'aimerais remercier mes directeurs de thèse, Albert Lejeune, professeur au département de management et technologie de l'École des Sciences de la Gestion de l'UQAM, et Daniel Lemire, professeur au département des Sciences et Technologie de la TELUQ. Tout au long de ma recherche, leurs suggestions et commentaires, toujours pertinents, m'ont permis de parfaire le contenu ainsi que la forme de ma thèse. Toute ma gratitude s'adresse aussi à Pierre Poirier, professeur au département de Philosophie de l'UQAM, qui m'a fait l'honneur de me conseiller dans la recherche de mon sujet de thèse.

Un grand merci à Mitchell Wasserman et Mark Adessky pour avoir autorisé et encouragé cette recherche au sein de leur entreprise.

Reconnaissance particulière à mes parents, lesquels ont toujours su me soutenir tout au long de mes études.

Merci à ma compagne et à mes enfants, qui ont été d'un soutien indéfectible tout au long de cette aventure.

DÉDICACE

À la mémoire de mon père, qui a toujours été fier de moi.

TABLE DES MATIÈRES

LISTE DES FIGURES.....	xi
LISTE DES TABELAUX.....	xv
LISTE DES ACRONYMES	xiv
RÉSUMÉ.....	xviii
INTRODUCTION.....	1
CHAPITRE I	
ETAT DE LA QUESTION ET PROBLEMATIQUE	9
1.1 État de la question	9
1.1.1 La conscience de la situation de l'individu	9
1.1.2 Conscience de la situation d'équipe	11
1.2 Définition de la problématique	12
1.2.1 L'analyse de la conscience de la situation d'équipe	12
1.2.2 Conciliation de deux domaines de recherches	14
1.3 Les propositions qui guident cette recherche	15
1.4 Les objectifs.....	16
1.5 Pertinence de l'étude sur la conscience de la situation.....	19
1.5.1 Pour le domaine de la gestion de projets informatique	19
1.5.2 Intérêt de l'étude pour les sciences cognitives	20
1.5.3 Intérêt de l'étude pour l'informatique	21
1.5.4 Intérêt pour la recherche sur la conscience de la situation	21
1.6 Méthodes de recherche utilisées	22
CHAPITRE II	
ÉTAT DE L'ART.....	25

2.1	Introduction	25
2.2	État de l'art de la gestion de projet informatique	26
2.2.1	Définitions de la gestion de projets	26
2.2.2	Un projet a trois dimensions structurelles	29
2.2.3	Projets informatiques et modèle mental partagé	31
2.2.4	Les tâches en gestion de projets informatiques	33
2.2.5	Projet informatique et méthodologies de développement	36
2.2.6	Concept de succès des projets informatiques	40
2.2.7	Risques et facteurs de risque des projets informatiques.....	41
2.2.8	Projet et acquisition de compétences non techniques	43
2.2.9	Conclusion de l'état de l'art de la gestion de projets informatiques	44
2.3	État de l'art sur la conscience de la situation.....	45
2.3.1	Définitions et domaines d'application des recherches sur la CS	45
2.3.2	La CS de l'individu est une fonction de ses modèles mentaux	47
2.3.3	La CS de l'individu : un état et un système en constante évolution.....	51
2.3.4	Vision écologique de la CS	52
2.3.5	Équipe et CS d'équipe	53
2.3.6	CS distribuée	57
2.3.7	Groupe ou équipe?	60
2.3.8	Équipes et macro cognition	63
2.3.9	Les erreurs de décision	65
2.3.10	Conclusion de l'état de l'art sur la CS.....	71
2.4	État de l'art sur la modélisation du social.....	73
2.4.1	Introduction	73

2.4.2	Historique de la modélisation des exigences.....	73
2.4.3	Les approches de la modélisation de l'entreprise.....	77
2.4.4	Conclusion de l'état de l'art sur la modélisation du social	103

CHAPITRE III

PROPOSITION D'UNE MÉTHODOLOGIE POUR L'ÉTUDE DE LA CONSCIENCE DE LA SITUATION D'ÉQUIPE		105
3.1	Introduction	105
3.2	Critères requis pour les méthodes d'analyse de la CS.....	105
3.3	Techniques d'analyse de la conscience de la situation.....	106
3.4	Conclusion sur les approches de mesure de la conscience de la situation .	116
3.5	Une approche de recherche-action	117
3.5.1	Choix et justification de l'approche proposée.....	117
3.5.2	Recherche-action et sciences du design	120
3.5.3	Recherche-action et étude de cas	123
3.5.4	Que devons-nous mesurer?	124
3.6	Activités de recherche	125
3.6.1	Analyse des besoins en CS.....	127
3.6.2	Analyse des ressources de CS	129
3.6.3	La validation de la solution proposée.....	131
3.7	Conclusion	132

CHAPITRE IV

ANALYSE ET MODÉLISATION DE LA CONSCIENCE DE LA SITUATION .		133
4.1	Introduction	133
4.2	Méthodes	133
4.2.1	Dispositif d'étude et collecte de données.....	133
4.2.2	Échantillonnage des participants	135

4.3	Dispositif expérimental	135
4.3.1	Observations.....	135
4.3.2	Les entrevues.....	137
4.3.3	Enquêtes contextuelles	138
4.3.4	Investigation des évènements récents	139
4.4	Les résultats	142
4.4.1.1	Le processus normal de production.....	142
4.4.1.2	Les équipes impliquées	142
4.4.2	Buts et fonctions des développeurs	164
4.4.3	Capture des exigences en CS	168
4.4.4	Capture de l'état des ressources supportant la CS.....	175
4.4.5	Les barrières et problèmes.....	180
4.4.6	Validation des données sur les besoins et les barrières à la CS	200
4.4.7	Discussion et interprétation des résultats.	201
4.4.7.1	Les besoins des équipes de développement en CS.....	201
4.4.7.2	Les besoins en ressources CS (personnel et technologie)	209
4.4.8	Analyse et modélisation des dépendances stratégiques	212
4.5	Conclusion.....	225
CHAPITRE V		
PROPOSITION DE SOLUTION D'AMÉLIORATION DE LA CS		227
5.1	Introduction	227
5.2	Solutions d'amélioration de la CS d'équipe	228
5.3	Les stratégies de fortification des relations entre les acteurs.....	230
5.3.1	Agir sur les dépendances pour une meilleure CS d'équipe.....	230
5.3.2	Atténuer les vulnérabilités.....	230

5.4	Le modèle stratégique de dépendance proposé	231
5.5	Modèle de raisonnement stratégique des DEV	235
5.6	Les changements dans le processus et dans les relations entre les acteurs. 236	
5.6.1	Décloisonnement des équipes pour de meilleurs modèles mentaux partagés 236	
5.6.2	Amélioration de la communication entre les équipes.	237
5.6.3	Ajustement des principes Agile au contexte de l'Entreprise.....	238
5.6.4	Un système d'évaluation du personnel basé sur des mesures objectives 240	
5.6.5	Clarification et redéfinition des rôles et responsabilités des équipes..	240
5.6.6	Un système d'information conforme à la méthodologie agile	241
5.7	Les recommandations pour une formation des équipes sur la CS	242
5.7.1	Modèles mentaux partagés et communication inter-équipes	242
5.7.2	Contextualisation des décisions	242
5.7.3	Meilleure communication entre les chefs d'équipes	243
5.7.4	Feedback.....	244
5.7.5	Formation sur la CS.....	245
5.8	Programme de formation sur la conscience de la situation d'équipe	246
5.8.1	Le contenu du cours	246
5.8.2	Le support de cours	249
5.8.3	Les stratégies d'enseignement.....	249
5.9	Conclusion	249
CHAPITRE VI		
ÉVALUATION DE LA SOLUTION PROPOSÉE		251
6.1	Introduction	251

6.2	Reformulation des objectifs et impacts de la formation	251
6.3	Méthode d'évaluation	252
6.3.1	Échantillonnage	252
6.3.2	Le processus d'évaluation.....	252
6.3.3	Comparaison des données pré et post formation.....	257
6.4	Résultat de l'évaluation	258
6.4.1	Valeur et utilité.....	258
6.4.2	Changement dans les comportements et attitudes.....	259
6.5	Changements dans le travail	265
6.6	Discussion sur l'évaluation de la formation	270
6.7	Conclusion	276
	CONCLUSION	277
	ANNEXE A	
	QUESTIONNAIRE CDM PROBES (O'Hare <i>et al.</i> , 2000).....	295
	ANNEXE B	
	QUESTIONNAIRE D'ÉVALUATION PRE SEMINAR / POST FORMATION ..	297
	ANNEXE C	
	SYLLABUS DU COURS SUR LA CS	303
	ANNEXE D	
	QUESTIONNAIRE DES ENQUETES CONTEXTUELLES	309
	BIBLIOGRAPHIE	311

LISTE DES FIGURES

Figure

2.1	Modèle de la CS en environnement de décision dynamique	50
2.2	Conceptualisation de la CS de l'individu (Salas et al., 1995).....	52
2.3	Conceptualisation de la CS d'équipe (Salas et al., 1995).....	54
2.4	Illustration de la CS distribuée (Salmon et al., 2008)	59
2.5	Le méta modèle de M*-OBJECT	79
2.6	Vue d'ensemble de l'architecture de la méthodologie M*-OBJECT	80
2.7	Méta modèle de la vue Téléologique de la modélisation de l'entreprise	82
2.8	Méta modèle de la vue « Sociale » de la modélisation de l'entreprise	83
2.9	Le méta modèle de la vue « Processus » de la modélisation de l'entreprise.....	84
2.10	Vue du modèle des activités de GBRAM	87
2.11	Meta modèle de KAOS (Respect-IT, 2007).....	92
4.1	Photo des installations de l'équipe BI	143
4.2	Représentation des équipes de développeurs chez l'entreprise.....	144
4.3	Flux général du travail dans le processus de production.....	153
4.4	Les processus du cycle de vie de l'application	154
4.5	Liste des tâches en exécution dans un cycle	159
4.6	Le processus de déploiement d'une application.....	161
4.7	Les équipements de communication	179
4.8	Vue de l'application de déploiement des « builds ».....	180
4.9	Le modèle de dépendance stratégique des DEV	223
5.1	Modèle de dépendance stratégique proposé.....	232
5.2	Modèle de raisonnement stratégique des DEV	235

6.1 Évaluation de l'ensemble de la formation par les participants.....	259
6.2 Effet perçu de la formation sur le comportement des participants.....	260
6.3 Résultats du test statistique non paramétrique de Wilcoxon.....	263
6.4 Comparaison des changements de comportement perçus	274
6.5 L'enthousiasme pour cette formation comparé à Team SA Training	275

LISTE DES TABELAUX

Tableau

2.1 Les définitions de la conscience de la situation	46
2.2 Variables qui influencent la performance de l'équipe	65
2.3 Classification des travaux sur la CS	72
3.1 Résumé des techniques de mesure de la conscience de la situation.....	107
3. 2 Comparaison entre recherche-action et sciences du design	122
4.1 Questionnaire des enquêtes contextuelles	138
4.2 Questionnaire d'investigation sur événements récents	139
4.3 Extrait de rapport d'incident	140
4.4 Rôles et attributions de l'équipe GESTIONNAIRE DE PRODUIT	144
4.5 Rôles et attributions de l'équipe BILLING.....	145
4.6 Rôles et attributions de l'équipe CLAIM.....	145
4.7 Rôles et attributions de l'équipe CARGO.....	146
4.8 Rôles et attributions de l'équipe BUSINESS INTELLIGENCE	146
4.9 Rôles et attributions de l'équipe DBA	147
4.10 Rôles et attributions de l'équipe INTEGRATION.....	147
4.11 Rôles et attributions de l'équipe GESTIONNAIRE DE PROJET	148
4.12 Rôles et attributions de l'équipe IT	148
4.13 Rôles et attributions de l'équipe GESTIONNAIRE DE CONFIGURATION,	149
4.14 Rôles et attributions de l'équipe GESTIONNAIRE DE COMPTE.....	149
4.15 Rôles et attributions de l'équipe GESTIONNAIRE DES OPERATIONS	150
4.16 Rôles et attributions de l'équipe ARCHITECTE	150
4.17 Rôles et attributions de l'équipe FRAMEWORK.....	151

4.18 Rôles et attributions du superviseur général	151
4.19 Liste des environnements d'application.....	152
4.20 Les objectifs des développeurs.....	166
4.21 Besoins en information des équipes de développeurs	169
4.22 Moyennes de l'importance et des fréquences des communications.....	175
4.23 Les barrières à la CS.....	181
4.24 Exemple de requêtes de maintenance de données.....	191
4.25 Tableau des dépendances entre les acteurs	216
 5.1 Les améliorations au système.....	 229
 6.1 Définition des « builds » de BILLING.....	 266
6.2 Définition des « builds » de CLAIM.....	267
6.3 Note sur la fusion des équipes CLAIM et BILLING	268
6.4 Exemple de requête de tâche à effectuer (PBI)	270

LISTE DES ACRONYMES

ARCH	Architecte
AT	Gestionnaire de compte
BI	Business Intelligence
CS	Conscience de la situation
DEV	Développeurs d'application
Dependum	L'objet de la dépendance entre 2 acteurs (dans une relation de dépendance)
Dependee	L'acteur dont on dépend (dans une relation de dépendance)
Depender	L'acteur qui dépend d'un autre (dans une relation de dépendance)
Grooming	Rencontre de travail des équipes en méthodologie agile/scrum. Rencontre d'identification de la nature du travail
INT	Intégration
OPS	Gestionnaire des opérations
PO	Gestionnaire de produit
PBI	Product back log
PM	Gestionnaire de projet
PMBOK	Project Management Body of Knowledge
QA	Gestionnaire de qualité
SMM	Modèles mentaux partagés (<i>shared mental models</i>)
Soft-goal	Exigence non-fonctionnelle (en ingénierie de logiciel)
Means/ends	Moyen/fins

RÉSUMÉ

Cette recherche porte sur l'applicabilité et l'importance dans une entreprise informatique, du concept de conscience de la situation; concept présenté depuis vingt-cinq ans comme un élément fondamental de l'adaptation cognitive des humains aux environnements complexes.

La recherche se donne trois objectifs : le premier est d'identifier les difficultés réelles reliées à la conscience de la situation dans le domaine de la gestion des projets informatiques; le second est d'identifier les besoins en conscience de la situation pour diverses équipes de projets informatiques, analyser comment les besoins en conscience de la situation sont actuellement rencontrés, et ainsi établir des concepts et des exigences de réorganisation et de formation pour améliorer la conscience de la situation des équipes informatiques; le troisième concerne l'applicabilité des méthodologies informatiques d'analyse des exigences, à l'étude de la conscience de la situation.

Cette recherche est menée dans une grande entreprise informatique où nous avons utilisé une méthodologie de recherche-action pour réaliser une enquête sur les facteurs reliés à la conscience de la situation auprès des équipes d'informaticiens. Le rôle de la conscience de la situation sur la performance des individus et des équipes a été étudié. Les exigences de la conscience de la situation en gestion de projet informatique sont identifiées, ainsi que les technologies et les ressources en personnel utilisées pour son amélioration.

Les obstacles et les problèmes de conscience de la situation à l'intérieur et entre les équipes de développeurs ont été relevés. L'application de la méthodologie d'analyse des exigences iStar, nous a permis de mettre en évidence les dépendances et interactions entre les acteurs du système étudié. Sur la base de notre analyse, des solutions de réorganisation de l'environnement social ainsi que des recommandations pour le développement d'un programme de formation pour améliorer la conscience de la situation au sein des équipes de développeurs sont présentés.

L'originalité de notre travail porte sur le fait qu'il s'inscrit dans les toutes premières études sur la conscience de la situation d'équipe et sa mesure au sein d'un environnement de gestion de projets informatiques. À notre connaissance, il s'agit aussi de la seule recherche menée à ce jour dans ce domaine, où le chercheur est intimement impliqué dans la vie quotidienne des sujets étudiés.

Les résultats montrent qu'une recherche-action apporte des modifications substantielles de l'environnement de travail et amène les acteurs à être des participants très actifs dans le développement de leur propre conscience de la situation.

Mot clé : conscience de la situation, modèles mentaux, modélisation du social, réingénierie des processus, modélisation d'entreprise.

Déclaration d'absence de conflit d'intérêt

Ce projet a été réalisé avec l'accord et la collaboration de la direction de la société Oceanwide Inc. à des fins de recherche académique. La société espère tirer avantage des recommandations qui en sortiront. Outre le temps de recherche qui nous a été alloué, ce travail n'a fait l'objet d'aucun financement, promesse de financement ou compensation de quelque nature que ce soit de la part de la société. En tant que chercheur, nous n'avons aucun conflit d'intérêt personnel ou incompatibilité avec les objectifs de la société Oceanwide Inc.

INTRODUCTION

0.1 Motivation

Les organisations s'engagent dans des projets de développement d'applications informatisées (projets informatiques) dans le but de réaliser des objectifs d'affaires et des profits. Cependant, la gestion de ces projets au sein d'une structure organisationnelle est une tâche risquée étant donné la nature complexe et dynamique de ces structures. On rapporte plusieurs projets informatiques – qu'ils soient réalisés à l'interne ou par de la sous-traitance - qui n'ont jamais atteint entièrement les objectifs qui leur sont assignés et sont donc considérés comme non réussis (Respect-IT, 2007).

Les projets informatiques échouent pour de multiples raisons dont une non négligeable est liée à l'organisation interne des équipes en charge, c'est-à-dire, la nature des relations qui lient les membres, et la manière dont les informations de décision sont communiquées, comprises et gérées par les membres.

Dans le domaine de la gestion de projet informatique, la performance d'un projet est intrinsèquement liée à la qualité et à l'efficacité des décisions que prennent les acteurs impliqués. Que ce soit individuellement ou en équipe, pour mener à bien les activités requises dans le cycle de vie d'une application, les acteurs sont constamment engagés dans des processus de prise de décision en se référant le plus possible à leurs connaissances et aux informations qui leurs sont accessibles dans leur environnement.

Pour être efficace, les équipes de projets informatiques ont besoin d'une bonne coordination, d'une vision commune, d'une adhésion de tous à l'objectif global du projet et la connaissance de l'état du système.

Dans les études disponibles sur la problématique de la gestion des projets informatiques, une attention insuffisante a été accordée à l'erreur humaine. En réalité, le taux d'échec

dû à des défaillances techniques du matériel est faible par rapport à des facteurs liés à l'erreur humaine. Geneca (2011), dans leur étude intitulée « *doomed right from the start?* » sur les causes d'échec des projets informatiques révèle que 75% des chefs de projets informatiques croient dès le départ que leur projet est voué à l'échec. Les causes sont en grande partie dues à l'inadéquation des objectifs d'affaires vis-à-vis des exigences du projet. Dans une étude similaire menée par The Standish Group (Caroll, 2013), 51% des répondants affirment que les causes d'échec des projets informatiques sont liées à des objectifs flous et au fait que les parties prenantes sont en déphasage avec les objectifs du projet. Ces genres d'erreur existent et peuvent être systématiquement abordées.

Dans le milieu de l'aviation, il existe des exemples où la combinaison de l'insuffisance de communication et les facteurs humains connexes ont été identifiés comme ayant une influence significative sur la performance de l'équipage et ont été des facteurs contributifs dans des accidents graves (Glavin, 2011 ; Molesworth et Estival, 2015). Des statistiques publiées par IATA (2014), l'agence internationale de l'aviation civile, sur les causes d'accidents d'avion survenus sur les lignes commerciales entre 2010 et 2014, rapportent que dans 84% des cas, les erreurs sont d'origine humaine et que 21% de tous les accidents analysés durant cette période sont dus à des erreurs d'opération.

Que ce soit dans le domaine de l'aéronautique ou celui de projet informatique, les erreurs non détectées au moment requis ont des impacts considérables et mesurables sur les résultats attendus. Elles ont des impacts directs sur les coûts d'opération des entreprises ainsi que sur leur viabilité.

En revenant plus spécifiquement au domaine de la gestion de projets informatiques et en examinant les erreurs humaines qui s'y produisent, plusieurs questions clés peuvent être identifiées :

1) La première concerne des lacunes dans la détection des indices critiques concernant l'état de l'application ou du système ou de ses composantes. Elles ne sont pas rares ces situations où l'application informatique cesse de fonctionner pour des erreurs associées à une mauvaise codification; à l'implantation erronée des règles de gestion; à la mauvaise validation de données, à des modules manquants ou de mauvaises versions des composantes qui auraient dû être détectés par l'équipe de développeurs. On rapporte des incidents dans lesquels des systèmes sont mis en production avec des modules non fonctionnels, non testés ou des mises à jour incomplètes. Bien que plusieurs facteurs (par exemple ambiguïté, insuffisance et conflictualité des informations disponibles) puissent contribuer à ce type d'erreur, dans tous ces cas, l'état du système n'a pas été détecté par les humains avant la remise du système en production.

2) En plus des difficultés de détection et d'observation des indices présents dans le système, l'interprétation de leur signification peut parfois être difficile à réaliser. La perception de l'information n'implique pas nécessairement que les indices sous-jacents sont compris et correctement interprétés. Pertet, et Narasimhan (2005) par exemple constatent dans leur étude sur les causes des erreurs dans les applications web, que 40% des erreurs sont d'origine humaines où les opérateurs pensent résoudre des problèmes identifiés mais qui sont en fait mal compris (mauvaise configuration, erreur de procédure). Même lorsque les symptômes peuvent être observés normalement, une tâche considérable reste à diagnostiquer correctement la véritable cause.

3) Dans l'arène de la gestion de projet informatique, le travail est divisé entre les membres de l'équipe ou entre de multiples équipes. Pour réaliser le but commun, chaque membre des équipes de développement doit accomplir son rôle tout en considérant la coordination, la communication et le partage d'information à l'intérieur de l'équipe.

La nécessaire coordination pour le partage d'information ajoute un niveau de complexité aux tâches de diagnostic et d'interprétation des signaux perçus. Dans un tel

environnement, il est facile que les tâches et les informations ne soient pas proprement relayées d'une personne à l'autre ou d'une équipe à l'autre.

La présence de plusieurs personnes renforce la nécessité d'une compréhension claire des responsabilités et une bonne communication entre les individus de manière à soutenir l'exécution des tâches communes et à éviter les tâches exécutées en double par différents opérateurs.

4) En plus de la nécessité d'une coordination intra-équipe, une autre tâche importante pour les équipes de développeurs est la coordination des activités et la fourniture d'informations entre les équipes, que ce soit dans le même lieu géographique ou dans des lieux géographiques distants.

5) La coordination des activités intra-équipe et celles entre les équipes nécessitent l'adoption d'outils de collaboration et des ressources matérielles appropriées. Les insuffisances de coordination ajoutent un niveau de complexité à la situation qui augmente ainsi la probabilité de tâches non terminées, des tâches accomplies incorrectement, des informations importantes non communiquées, des problèmes qui passent inaperçus et les responsabilités qui deviennent plus diffuses.

0.2 Conscience de la situation

Nous venons de révéler 5 types de problèmes dans lesquels s'inscrivent les opérateurs humains issus des équipes de gestion de projets informatiques, en interaction avec des processus dynamiques et complexes : 1) La difficulté de diagnostiquer l'état du système; 2) les problèmes de compréhension des signaux que révèle l'état du système; 3) les difficultés de coordination du travail entre les membres d'une équipe; 4) les difficultés de coordination entre diverses équipes affectées au même système; et 5) l'inadéquation des ressources de collaboration à la nature des tâches.

En réponse à ces complexités, le concept de conscience de la situation (CS), aussi connu sous la dénomination de « *Situation Awareness* » en anglais, est présenté depuis vingt-cinq ans comme un élément fondamental de l'adaptation cognitive des humains aux environnements dynamiques et complexes. La CS consisterait à « Rester vigilant sur ce qui se passe autour de soi dans un environnement dynamique et complexe » (Moray et Sheridan, 2005). Elle s'est avérée être importante pour une grande variété d'opération de systèmes, y compris celui du développement de système informatique. Elle représente cependant, l'un des aspects les plus difficiles à cerner de la performance humaine, particulièrement, dans la plupart des tâches complexes où la prise de décision dépend de la mesure avec laquelle les individus ont développé une bonne connaissance de la situation.

La CS est cruciale dans les domaines dans lesquels les flux d'information peuvent être très élevés et où une mauvaise décision peut rapidement mener à de sérieuses conséquences (par exemple, le pilotage d'un avion, le rôle de soldat, le rôle d'anesthésiste, le traitement des patients gravement malades ou blessés, etc...). C'est pour cette raison que ce concept continue de recevoir une quantité considérable d'attention de la part des chercheurs et notamment ceux de la communauté "Human factors and ergonomics" (Salmon *et al.*, 2006; Stanton *et al.*, 2006, etc.). En effet, la CS a été reconnue dans plusieurs études comme un élément fondamental, et critique de la prise de décision humaine dans une large gamme de systèmes complexes et dynamiques, y compris dans le domaine de l'aviation et contrôle du trafic aérien (Nullmeyer *et al.*, 2005; Stepniczka *et al.*, 2015, Endsley, 2012); les interventions d'urgence, le commandement et le suivi des opérations militaires (Kaber *et al* 2013); les interventions en champ pétrolier en haute mer, la gestion des centrales nucléaires (Flin et O'Connor 2001; Sandhåland *et al.*, 2015), et urgences médicales (Schulz *et al.*, 2013; O'Connor *et al.*, 2013).

Le manque ou l'insuffisance de CS a toujours été unanimement identifié comme l'un des principaux facteurs dans des incidents attribués à une erreur humaine (Nullmeyer *et al.*, 2005; Johannesen, 2012; Singh, 2012; Molesworth, 2015). Pourtant, développer et maintenir un haut niveau de CS impose des exigences cognitives pour les acteurs humains en termes de temps, d'attention et d'efforts. Heureusement, la charge cognitive associée à des niveaux élevés de CS peut être atténuée lorsqu'on fonctionne dans un environnement où les règles de fonctionnement, les procédures de travail, les outils de travail ou même la disposition physique de ces composantes, ou tout simplement le système, est orienté-CS (voir Endsley, 2012), et/ou les programmes de formation sont orientés-CS (Strater et Bolstad, 2009). Par conséquent, les tests de CS et leur évaluation jouent un rôle majeur pour comprendre et amener les systèmes à réaliser leurs objectifs. Dans les entreprises informatiques, les équipes de développeurs ont besoin de soutien et de formation pour déterminer l'état actuel du système¹, en plus de programmes de formation professionnelle orientés sur les compétences techniques.

Dans la littérature, il existe plusieurs définitions de la CS, reflétant des divergences importantes quant à sa nature et aux processus cognitifs impliqués. Le concept de CS a longtemps été débattu, et les chercheurs ont suggéré différentes approches pour son étude :

- Du point de vue de l'individu, la CS est abordée comme un processus cognitif qui s'opère dans l'esprit de l'individu (Sarter, et Woods, 1991; Vidulich, 1994; Klein, 1995) à l'intérieur d'un système.
- Du point de vue de l'équipe, la CS d'équipe implique des activités uniques, telles que la coordination et le partage de l'information qui permettent aux

¹Le terme « Système » englobe l'application informatique ainsi que les procédures, les pratiques organisées, destinées à assurer la production de l'application.

individus d'interagir et de bien performer dans un environnement dynamique (Schwartz, 1990).

- En revanche, du point de vue de l'écologie des systèmes, la CS est étudiée sous le vocable de « *conscience de la situation distribuée* » et est comprise comme un processus qui se produit à travers les interactions entre les individus et les outils qu'ils utilisent pour atteindre leurs objectifs (Stanton *et al.*, 2010).

La définition suivante proposée par Endsley transcende cette catégorisation individu / équipe / système. Endsley (1988) définit la CS comme « la détection des éléments de l'environnement dans un volume d'espace et de temps, la compréhension de leur signification, et la projection de leur statut dans un avenir proche » (p.97). Lorsque nous rapprochons cette définition au domaine du développement d'application, cela signifie être conscient de l'état du système sur lesquels l'on travaille. Dans le présent contexte, le terme « Système » englobe l'application informatique ainsi que les acteurs, les procédures, les pratiques organisées, destinées à assurer la production de l'application. Cette définition proposée par Endsley est celle qui cadre mieux avec nos objectifs de recherche. L'une des caractéristiques fondamentales de cette définition est sa nature normative.

Nous reviendrons plus en détail sur les différentes définitions de ces concepts dans le chapitre 2

0.3 Structure de la thèse

Le reste du document est articulé de la façon suivante :

Le chapitre 1 aborde l'état de la question de recherche, la problématique, les propositions de recherche, et les objectifs que nous souhaitons accomplir.

Le chapitre 2 présente l'état de l'art du domaine d'étude, de la CS et des méthodologies informatiques d'analyse du social.

- La première partie de ce chapitre présente une revue de littérature sur la gestion de projets, et la gestion de projets informatiques.
- La deuxième partie explore la CS de l'individu, la CS d'équipe et aborde la notion d'erreur de décision dans le contexte d'équipe.
- La troisième partie traite des méthodologies d'analyse des besoins telles qu'utilisées en ingénierie informatique, notamment, la méthodologie iStar, KAOS Téléologique, M* et GBRAM.

Le chapitre 3 traite de la méthodologie adoptée dans cette étude. Il aborde notamment les questions de mesure de la CS, et de la démarche générale de la recherche.

Les chapitres 4, 5 et 6 présentent les résultats et la validation de la solution proposée.

- Le chapitre 4 porte sur les résultats et les interprétations du premier travail de terrain.
- Le chapitre 5 propose des solutions d'amélioration de la CS d'équipe
- Le chapitre 6 porte sur l'application et l'évaluation de la principale contribution de cette thèse.

La conclusion générale présente l'originalité de notre contribution à la recherche sur la CS d'équipe et la modélisation du social. Il fait un rappel des objectifs de la recherche, de la démarche adoptée, les résultats obtenus, et donne des pistes de recherches futures.

CHAPITRE I

ETAT DE LA QUESTION ET PROBLEMATIQUE

1.1 État de la question

1.1.1 La conscience de la situation de l'individu

Le modèle de Endsley de la CS que nous adoptons dans cette recherche, illustre trois stades ou niveaux de formation de la CS: la perception, la compréhension, et la projection, tel qu'il ressort de la définition proposée plus haut.

Le niveau 1 dans la réalisation de la CS consiste à percevoir l'état, les attributs, et la dynamique des éléments pertinents dans l'environnement. Ainsi, le niveau 1 CS, le niveau le plus élémentaire de la CS, implique les processus de surveillance, de détection de repère, et la simple reconnaissance, qui conduisent à une prise de conscience de plusieurs éléments de la situation (des objets, des ressources matérielles et immatérielles, des événements, des personnes, des systèmes, des facteurs environnementaux) et leurs états courants (les lieux, les conditions, les modes, les actions).

Concrètement, la perception inclut les problèmes de performance des requêtes, les règles de gestion obsolètes, les modules mal codifiés, l'état de fonctionnement des systèmes, la vulnérabilité des équipes, la mauvaise qualité de l'information disponible.

L'étape suivante dans la formation de la CS implique une synthèse des observations décousues du niveau 1 CS à travers les processus de reconnaissance de patrons, l'interprétation et l'évaluation. Le niveau 2 CS nécessite d'intégrer cette information

pour comprendre comment il va avoir un impact sur les buts et objectifs établis. La CS de niveau 2 impliquerait la compréhension par les acteurs de la signification des états du système observé. Plus précisément ce qui comprendrait leur diagnostic des facteurs causaux associés à des symptômes observés. Un développeur d'application de niveau 1 CS sait peut-être qu'un sous-système particulier ne fonctionne pas correctement. Un développeur de niveau 2 CS comprend aussi ce qui ne va pas spécifiquement avec ce sous-système.

Le niveau 3 CS, la capacité de projeter l'état du système ou les actions des éléments de l'environnement dans un proche avenir, est considéré comme le plus haut niveau de CS dans les systèmes dynamiques. Le niveau 3 CS est réalisé grâce à la connaissance de l'état et la dynamique des éléments et la compréhension de la situation (niveaux 1 et 2 CS), puis l'extrapolation de ces informations en avant dans le temps afin de déterminer comment elles affecteront les états futurs de l'environnement opérationnel. Un développeur de niveau 3 CS serait en mesure de projeter l'effet qu'un défaut particulier pourrait avoir sur les performances futures de l'application.

Bien que la CS soit généralement présentée en termes de fonctionnement d'un système dynamique complexe, tel que celui du pilotage d'un aéronef ou celui des pompiers en service, le concept s'applique également au domaine du développement d'application en milieu d'entreprise. La complexité des systèmes d'entreprises qui abritent les équipes de développeurs et la nature distribuée des équipements et des composants du système constituent un « défi significatif » à la capacité des développeurs pour déterminer l'état du système (niveau 1 CS) au cours des activités de diagnostic ou de détection d'erreurs et de réparation. Parvenir à mettre ensemble des indices observés afin de former une bonne compréhension de la nature sous-jacente des dysfonctionnements (niveau 2 CS) est un « problème significatif » dans les activités de diagnostic. Dans le domaine de la gestion de projets informatiques, les développeurs peuvent souhaiter avoir la capacité de prévoir ce qui arrivera à la performance de leur

application avec (ou sans) certaines mesures prises ou selon certains ajustements d'équipement ou de fonctionnalités de l'application ou des procédures opérationnelles (niveau 3 CS). Cette tâche peut être encore plus difficile pour les développeurs s'ils reçoivent souvent peu ou pas de commentaires sur les effets de leurs actions. Ils peuvent aussi avoir des difficultés à développer un modèle mental approprié pour faire des prédictions précises, s'ils ne disposent pas d'une expérience adéquate dans leur domaine. Car la connaissance des événements du passé renforce la capacité des développeurs à faire des projections sur le futur. Cette capacité est particulièrement importante et requise pour effectuer des diagnostics efficaces.

1.1.2 Conscience de la situation d'équipe

Dans la gestion de projet informatique, comme dans beaucoup d'autres domaines, l'exigence de la CS devient importante et critique étant donnée la présence de plusieurs acteurs dans les équipes et la présence de multiples équipes.

Nous reviendrons plus en détail sur la notion d'équipe et de groupe dans le chapitre 2. Les principaux facteurs contribuant à la notion d'équipe sont des objectifs partagés, l'interdépendance des actions, et la division du travail entre les membres, en termes de responsabilités établies pour atteindre des objectifs. Sur cette base, les équipes auront une certaine division du travail, et les membres, une certaine indépendance dans leurs exigences de CS. Cependant, les membres auront également des objectifs communs et leurs actions seront interdépendantes, et donc elles auront forcément des CS partagées, du fait qu'ils partagent certaines informations du système. La CS d'équipe peut être définie comme « la mesure dans laquelle chaque membre de l'équipe possède la CS nécessaire pour ses responsabilités » (Endsley, 1989). Alors que la CS partagée est considérée comme « la mesure dans laquelle les membres de l'équipe possèdent la même CS sur les besoins des CS partagées » (Endsley et Jones, 2001, p. 48). Dans ce contexte, le concept de CS d'équipe incarne la nécessité pour chaque individu de l'équipe d'avoir les informations nécessaires pour son travail spécifique.

La CS de l'équipe peut être déterminée en examinant les buts, et les besoins en CS des membres de l'équipe. La bonne performance d'une équipe exige que non seulement chaque membre de l'équipe ait une bonne CS sur ses besoins individuels, mais aussi la même CS à travers les besoins partagés. Nous nous référons à ceci comme la « CS partagée ». La même analyse est applicable à la CS d'une équipe unique versus la CS de l'ensemble des équipes de l'entreprise.

La problématique de cette recherche couvre non seulement la question des défaillances du niveau de la conscience des acteurs individuels sur leur environnement mais englobe aussi l'identification des informations requises pour une CS et comment elles sont distribuées à travers les membres de l'équipe et entre les équipes.

1.2 Définition de la problématique

1.2.1 L'analyse de la conscience de la situation d'équipe

Mesurer la CS d'équipe en tant que concept en lui-même est aujourd'hui prématuré dans un projet de recherche sur la CS. Pour cela, il faut attendre qu'il y ait une meilleure compréhension de ce que représente ce concept. Pour l'heure, deux mesures importantes doivent être considérées en premier, pour étudier la CS d'équipe : 1) la CS de l'individu, et 2) les procédures de fonctionnement d'équipe que les membres de l'équipe utilisent pour construire et échanger des informations et améliorer la coordination entre eux. Il faudra par la suite mesurer aussi les modèles mentaux partagés – définis comme la compréhension et la représentation mentale organisées et partagées des connaissances de l'équipe à propos des éléments clés de l'environnement de l'équipe (Mohammed et Dumville, 2001, p. 90) - par les membres de l'équipe. En effet, plusieurs études ont émis l'hypothèse que les modèles mentaux partagés constituent un élément important de la CS d'équipe (Robertson et Endsley, 1994; DeFranco *et al.*, 2011; Burtscher et Manser, 2012).

La mesure de la CS de l'individu est une préoccupation importante pour la recherche dans le domaine des sciences cognitives et a déjà été traitée abondamment par un certain nombre de chercheurs (Fracker, 1988; Sarter, et Woods, 1991; Endsley, 1995a; Tenney *et al.*, 1995). Pour sa part, Fracker (1991) a présenté un rapport détaillé sur les différentes méthodes disponibles pour la mesure de la CS de l'individu (par exemple l'auto-évaluation, les protocoles verbaux, des mesures explicites et des mesures implicites). Chacune de ces méthodes a démontré des faiblesses (par exemple, elles sont intrusives ou incomplètes) et aucune n'a démontré des niveaux acceptables ni en termes de fiabilité ni en termes de validité. Ce constat constitue pour notre problématique un formidable défi. Nous présenterons les techniques de mesure de la CS, les mieux connues, dans notre chapitre 2 dédié à l'état de l'art sur les méthodes et outils de mesure de la CS.

Aussi difficile que peut être la mesure de la CS, celle des modèles mentaux représente un défi encore plus grand. Quelques rares techniques sont maintenant disponibles pour évaluer les modèles mentaux des membres d'une équipe (par exemple, *Pathfinder*; Ifenthaler 2014), mais elles ont besoin de tests additionnels et le développement pour être généralisées. La mesure des états cognitifs est si essentielle à la compréhension de la CS que des efforts devraient être faits pour développer de meilleurs outils de mesure. En attendant que ces outils soient disponibles, nous pouvons continuer d'avoir un aperçu sur la CS d'équipe en nous concentrant sur l'évaluation de la CS de l'individu et des processus d'équipe. En focalisant notre attention sur ces aspects plus comportementaux (et parfois intentionnels) de la performance humaine, il est possible de faire des inférences concernant la nature et la qualité de la CS l'équipe.

1.2.2 Conciliation de deux domaines de recherches

L'autre défi de cette problématique est d'arriver à concilier deux domaines de recherche apparemment distincts : l'un est l'étude de la CS sous l'angle purement cognitif, et l'autre est l'ingénierie des exigences en génie logiciel.

Cette recherche que nous proposons sur la CS en milieu d'entreprise, ambitionne entre autres, d'étudier les aspects comportementaux des individus et des équipes, les types de relations qu'entretiennent les équipes et les acteurs qui les composent au regard des décisions à prendre et des objectifs à réaliser. Un tel travail se présente après tout comme une analyse de l'environnement social de l'entreprise étudiée. En ingénierie informatique, elle représente l'étape préliminaire d'analyse des besoins. Elle met particulièrement en application les méthodologies d'analyse préliminaire qui imposent une démarche systématique et incrémentale de l'élaboration des exigences par un processus coopératif itératif d'analyse du problème, de documentation des observations résultantes dans une variété de formats de représentation et de vérification d'exactitude de la compréhension acquise.

Afin de concilier ces deux domaines, notre approche est basée sur la prémisse que le processus d'ingénierie des exigences se définit sur trois dimensions comme proposé par Pohl (2013) :

- **Les dimensions cognitives** : les questions du domaine cognitif concernent des différentes orientations des modèles en termes de compréhension du processus lui-même et de validation des exigences.
- **Les dimensions sociales** : dans le domaine social, il est tenu compte du processus social complexe dans lequel la communication et l'interaction de coopération entre les parties prenantes des exigences déterminent la qualité du produit final.

- **La représentation** : les questions de représentation sont diversement abordées. Elles vont d'une description informelle telle que les expressions en langage naturel aux langages plus formels de modélisation conceptuelle.

Le processus d'analyse cognitive ou celui de l'ingénierie des exigences représente une interaction continue entre les facteurs issus des trois dimensions précitées, générant des versions successives de ce qui a été traditionnellement appelé la « spécification des exigences » en ingénierie informatique

Concrètement, en appliquant la démarche méthodologique de l'ingénierie des exigences du génie logiciel, plus particulièrement celle proposée par la modélisation du social, nous souhaitons apporter un support informatique à l'étude de la CS d'équipe sur plusieurs aspects. Les modèles qui en résulteront seront utilisés comme moyen pour réfléchir sur des problèmes, trouver des solutions alternatives, parvenir à des accords et élaborer les besoins des parties prenantes d'une manière formelle. En outre, le paradigme d'agent et de but offre des avantages sur les autres paradigmes en ce sens que les concepts tels que rôle, but, interaction, et dépendance font partie du langage de tous les jours des acteurs du système. Dans une entreprise réelle, comme c'est le cas dans ce présent travail de recherche, il y a des rôles qui sont remplis par des vraies personnes tous les jours, y compris les parties prenantes qui sont engagées dans le processus d'ingénierie des exigences. Ces individus sont donc familiers avec ces concepts et sont confortables à les utiliser.

1.3 Les propositions qui guident cette recherche

Proposition 1 : La réduction des erreurs de décision dans un environnement opérationnel complexe, tel que celui du développement d'application informatique, exige des niveaux élevés de prise de conscience individuelle de la situation, mais aussi un niveau élevé de conscience coordonnée entre les membres de l'équipe. En d'autres

termes, les individus faisant partie des équipes dans ce contexte doivent développer une compréhension commune précise de la situation.

Proposition 2 : La redéfinition de la configuration sociale dans le but de l'amélioration de la CS d'équipe peut contribuer à la réduction des erreurs de décision et à de meilleures performances des équipes.

Proposition 3 : La formation sur la CS d'équipe peut aider les équipes à changer leurs comportements et améliorer leur compréhension des informations de leur environnement.

1.4 Les objectifs

Le but de notre thèse est de montrer l'intérêt de la prise en compte de la CS et par là, la modélisation du social, comme étape préliminaire de la conception de solutions visant à réduire les erreurs de décision dans les environnements complexes. Pour y parvenir, nous avons trois objectifs.

Objectif 1 : Dans les équipes de projets informatiques, comme dans des équipes de nombreux autres domaines, des erreurs se produisent souvent parce que l'information n'est pas transmise avec succès d'un individu à l'autre au sein d'une équipe ou d'une équipe à l'autre. Les échecs peuvent se produire au niveau 1 (information non transmise avec succès), niveau 2 (information non comprise correctement ou de la même manière), ou de niveau 3 (les implications futures de l'information transmise sur les événements futurs ne sont pas comprises). Les façons dont ces défaillances peuvent survenir et les facteurs qui contribuent à ces échecs n'ont pas encore été examinés du point de vue de la CS dans la gestion de projets informatiques tel que nous l'abordons dans notre étude.

Le premier objectif de cette recherche est donc de se pencher sur les difficultés reliées à la CS dans le domaine de la gestion de projets informatiques en milieu entreprise. Cette information serait utile pour proposer une configuration alternative du social, soutenir le développement d'équipes cohésives de développeurs et la promotion de la CS de l'équipe. Cela est essentiel pour le développement d'un environnement dans lequel les acteurs ont une compréhension commune de qui fait quoi, quand et pourquoi.

Les individus ont besoin de comprendre non seulement l'état du système sur lequel ils travaillent, mais aussi ce que les autres personnes ou les autres équipes font, car les deux facteurs contribuent tous autant au processus de prise de décision et ultimement à la performance.

Objectif 2 : À l'intérieur d'une équipe, il est important d'évaluer les compétences non-techniques des membres afin de leur fournir les compétences nécessaires pour fonctionner efficacement. Plus précisément, une formation peut être fournie pour aider les équipes dans la réalisation de leur CS. Ce qui pourra agir sur la réduction des erreurs de décision, car la CS constitue un facteur critique qui permet aux membres de l'équipe de mener à bien les tâches de développement d'application qui constituent leurs spécialités.

Quoique dans la littérature, il n'existe pas spécifiquement des exemples de programme de formation en CS destinés à des équipes de développeurs d'application, plusieurs programmes de formation ont été couronnés de succès dans le domaine de l'aviation et de la médecine. Gramopadhye *et al* (1993) a connu un succès en matière de formation en inspection visuelle auprès d'une équipe de maintenance d'aéronef. Taylor *et al* (1995) ont réussi à améliorer les objectifs de performance de haut niveau (par exemple, fiabilité et sécurité) dans l'aviation après avoir institué un programme de formation du personnel d'opération et de maintenance d'aéronefs aux méthodes de communication assertive et positive et à l'acquisition de compétence dans la coordination du travail en équipe. Endsley et Robertson (2000) ont obtenu de bons résultats avec leur programme

de formation sur l'amélioration de la CS à l'attention des équipes de maintenance d'avion. Enfin, un programme récent de O'Connor (2013) sur la formation de jeunes médecins sur le travail d'équipe, la communication et la CS, a connu des résultats concluants.

Bien que ces exemples de formation ont pu aider la CS, seul celui de Endsley traite directement de la CS comme un objectif primordial. Il apparaît aussi dans ces programmes que des facteurs additionnels doivent être pris en considération pour optimiser la CS d'équipe. Par exemple, des compétences sont nécessaires pour identifier les informations critiques et veiller à ce qu'elles passent bien entre les équipes (et les membres de l'équipe) et ensuite interprétées sur la base d'un protocole commun de fonctionnement entre les membres de l'équipe.

La présente étude cherche aussi à identifier les besoins en CS pour diverses équipes de développeurs, analyser comment les besoins en CS sont actuellement rencontrés dans un environnement de développement typique, et ainsi proposer des solutions alternatives, et établir des concepts et des exigences de formation pour améliorer la CS en équipes de développeurs.

Objectif 3 : Fournir un support informatique à l'étude de la CS. Il s'agit ici de formaliser l'étude du concept de la CS en utilisant des langages appropriés et des standards d'ingénierie des exigences du génie logiciel. A l'issue de cette étude, une spécification complète de la solution devra être produite et sera considérée comme le résultat le plus élémentaire dans le cycle de vie du système. Si cette spécification est exprimée seulement en langage naturelle, il peut avoir des ambiguïtés dans sa compréhension et cela peut conduire à des conceptions et des implantations divergentes de la solution. Pour éviter des interprétations divergentes des spécifications, il est parfois préférable d'utiliser un langage formel de représentation. De plus, un langage formel peut offrir des prises en charge d'un raisonnement.

La mise en œuvre de cet objectif de support informatique à l'étude de la CS d'équipe nous amènera à réaliser ce qui suit :

- Offrir une représentation formelle, en langage informatique, de la dynamique sociale dans laquelle s'opèrent les relations entre les acteurs;
- Améliorer la compréhension d'un système opaque complexe dans une spécification complète du système;
- Transformer les besoins opérationnels en des spécifications complètes du système, à travers un processus itératif de définition et de validation;
- Susciter des raisonnements à propos des connaissances du système à travers l'usage de la sémantique des langages de représentation formelle sur le système;
- Identifier et consolider les différentes visions d'amélioration du système, de toutes les parties prenantes, dans un format non-ambiguë, vérifiable, et consistant qui est manipulable et modifiable;
- Se servir des artefacts offerts par les outils d'analyse des exigences, comme un medium de communication entre les parties prenantes, pour régler les visions conflictuelles de la solution proposée;
- Transformer les connaissances informelles en des représentations formelles

1.5 Pertinence de l'étude sur la conscience de la situation

1.5.1 Pour le domaine de la gestion de projets informatique

L'analyse des études de l'erreur humaine notamment dans le domaine de l'aviation permet de constater que la grande majorité des erreurs sont dues non pas aux mauvaises décisions, mais plutôt à des problèmes avec la CS (Johannesen, 2012; Schulz *et al.*,

2013). Des études similaires peuvent être répliquées dans le domaine spécifique du développement d'application puisque la nature des deux environnements présente des similarités évidentes en termes de structure de gouvernance basée sur l'affectation des rôles aux acteurs, la présence de plusieurs équipes et la nature dynamique des relations entre elles, la distribution des équipes et leur caractéristique « naturaliste » (réalités d'un monde naturel par opposition aux configurations de laboratoire) (Zsombok et Klein, 2014). La présente étude vise spécifiquement à déterminer si la CS est pertinente dans le domaine de la gestion de projets informatiques et voir la manière dont elle sert à la prise de décision et à la performance dans cet environnement bien particulier.

En outre, étudier la CS au sein des équipes de développement d'application peut donner l'occasion de faire ressortir les défis et les difficultés organisationnelles qui y sont vécues par les intervenants et tenter de trouver une meilleure compréhension des causes.

Nous formulerons des recommandations spécifiques pour améliorer la CS par le biais de la mise en place d'une meilleure organisation interne et par le biais d'un programme de formation. De tels programmes devraient aussi être applicables à tous les acteurs de l'industrie du développement d'application.

1.5.2 Intérêt de l'étude pour les sciences cognitives

En étudiant les modèles de la prise de décision en environnement naturel, nous avons constaté que la classification ou la vision que l'on fait de la situation influence largement la prise de décision et la performance (Zsombok et Klein, 2014). L'examen des besoins en CS dans les équipes de développeurs d'application opérant dans un environnement naturel, peut fournir une perspective unique sur les défis et les problèmes autant sur le plan individuel que dans les équipes organisées. Les raisons de nombreuses erreurs et lacunes de performance en équipe ne sont pas apparentes à première vue. L'étude de cette composante de la cognition humaine qu'est la CS, et la

mise en place d'outils de mesure donneront une meilleure compréhension des causes de l'erreur humaine dans un environnement qualifié de complexe.

1.5.3 Intérêt de l'étude pour l'informatique

Comme nous l'avons relevé dans la section de la problématique, cette étude de la CS en milieu entreprise se présente après tout comme une analyse de l'environnement social ou comme l'étape préliminaire d'analyse des besoins, requise dans l'ingénierie des systèmes d'information d'aide à la décision, des systèmes multi-agents ou des systèmes collaboratifs. Nous avons choisi de mettre en application la méthodologie d'analyse des exigences i^* (nouvellement connue sous la dénomination de iStar) au cours de laquelle nous élaborerons le modèle social du système étudié en explorant les concepts d'intelligence artificielle tels que ceux d'acteur, de sociabilité, d'intentionnalité, de dépendance, de vulnérabilité et d'autonomie. De plus, la méthodologie iStar nous recommande d'adopter une démarche systématique et incrémentale de l'élaboration des exigences. Il en résulte une série de modèles proposés pour représenter le problème étudié.

1.5.4 Intérêt pour la recherche sur la conscience de la situation

- Questions de mesures de la CS

La théorie ne peut pas aller au-delà de l'étape conceptuelle sans le développement et les tests d'outils de mesure proposés. La mesure, en soi, contribuera à la construction et à la validation des modèles précis de la CS d'équipe. Sans indicateurs quantifiables de la CS d'équipe, il est difficile d'articuler ce qui constitue une bonne CS. Cette information est particulièrement importante pour fournir une rétroaction sur le rendement des acteurs pendant la formation. Enfin, la mesure est indispensable pour évaluer les approches pédagogiques de la formation. De bonnes mesures fourniront une indication de la mesure dans laquelle la formation est efficace.

- Questions de formation

Les recherches sur les stratégies de formation de l'équipe fourniront des informations utiles à deux niveaux. Tout d'abord, les résultats de la recherche sur les processus critiques et les comportements pour former peuvent être utilisés pour affiner les connaissances sur la CS d'équipe. Deuxièmement, étant donné que le but de la formation de l'équipe est d'améliorer les performances, les données de cette recherche permettront de mieux comprendre les effets de la CS d'équipe sur la performance de l'équipe.

1.6 Méthodes de recherche utilisées

Nous adoptons une approche systémique dans laquelle les équipes de l'entreprise étudiée sont considérées comme des acteurs (sociaux) qui dépendent les uns des autres pour des objectifs à atteindre, des tâches à accomplir et des ressources à fournir. La méthodologie adoptée devra nous permettre de comprendre la nature, le fonctionnement et le rationnel de ces relations, et aussi de proposer un modèle de dépendance stratégique qui décrit le réseau des interdépendances entre les acteurs en termes de besoin en CS. En outre, notre démarche méthodologique devra être le cadre d'une évaluation quantifiable de la CS.

Nous optons donc pour une méthode de recherche qui se veut qualitative puisqu'il s'agit avant tout d'explorer et de comprendre un aspect particulier de la cognition humaine dans un environnement unique (Blaikie, 2009), avec quelques considérations quantitatives puisque le modèle de CS que nous adoptons est normative et qu'il nous permet d'identifier les éléments mesurables de la CS (Endsley et Jones, 2001).

Une démarche holistique d'étude de la CS d'équipe : L'absence d'une méthode unique qui peut mesurer à la fois la CS individuel et la CS d'équipe géographiquement

distribué et en temps réel, justifie l'usage d'une approche de **Recherche-action** (Baskerville et Myers 2004) dans laquelle plusieurs méthodes de collecte de données et de mesures sont utilisées. Une telle approche garantit que les données CS peuvent être efficacement recoupées entre les mesures afin d'en assurer la fiabilité et la précision. Le cadre méthodologique que nous proposons, prévoit donc d'utiliser une batterie de méthodes de « Human Factor » pour obtenir de bonnes performances. Il comporte les méthodes suivantes :

- 1) Analyse de la documentation existante;
- 2) Observation directe des sujets pendant l'exécution de leur tâche;
- 3) Usage des questionnaires d'entrevues semi-structurées;
- 4) Usage de la méthode EAST (*Event Analysis of Systemic Teamwork*) proposée par Walker *et al* (2006), pour prendre des données de la structure hiérarchique des tâches et capturer les principales composantes du travail d'équipe pour le cycle de vie d'application.

Échantillonnage : A l'image d'autres études similaires sur la CS (Endsley, et Robertson 2000) nous ne pouvons pas étudier la CS de toutes les équipes des entreprises de développement d'application. Nous avons donc travaillé sur une seule entreprise suffisamment représentative de l'industrie. Les pratiques de cycle de vie de développement qui s'y déroulent prennent en compte les étapes standards que l'on pratique dans ce domaine d'affaire.

Étude étendue : La CS d'équipe n'est pas un état statique mais plutôt le résultat de plusieurs processus qui se mettent en branle (par exemple, identification de problème, partage d'information) au sein d'une équipe. Il est évident que le recours à une seule mesure ne serait pas adéquat pour déterminer la CS d'équipe sur un ensemble de tâches spécifiques. En conséquence, pour mesurer le processus CS aussi bien que la CS (le

produit), des mesures devraient être effectuées sur une série d'évènements clés et sur une longue période pendant que les équipes sont à l'œuvre dans leurs tâches.

Proposition d'une solution alternative : À travers la démarche de modélisation iStar, une méthode d'ingénierie des exigences, nous analysons les composantes sociales du système étudié. Ainsi, notre démarche s'appuie sur des règles méthodologiques de génie logiciel en plus des méthodes narratives d'études de la CS. Nous proposons des solutions d'amélioration de la CS d'équipe selon un formalisme méthodologique qui tient compte des standards du design préliminaire typique des projets informatiques notamment ceux en relation avec les systèmes d'aide à la décision et les systèmes collaboratifs.

Validation de la solution proposée : La validation de la solution proposée auprès des mêmes sujets étudiés, est nécessaire. Les mesures de *Valeur* et *d'utilité* subjectives de la solution ainsi que les mesures objectives des changements de comportement observés après le déploiement de la solution, permettront d'évaluer la validité et la pertinence de la solution proposée.

CHAPITRE II

ÉTAT DE L'ART

2.1 Introduction

À la lumière de nos objectifs, nous faisons des propositions de solutions qui tendent à modifier des procédures organisationnelles, des règles de fonctionnement et de gestion d'une entité corporative pour une meilleure productivité de ses équipes. L'introduction d'un nouveau système ou la modification d'un système existant occasionne une redistribution des tâches et des responsabilités entre les « agents du monde » - les humains, les matériels, les logiciels, les unités organisationnelles, etc. Avant de proposer de telles solutions d'amélioration, il est nécessaire de comprendre le fonctionnement du système existant. L'identification des besoins en CS telle que définie dans nos objectifs, s'inscrit dans cette logique.

Trois types d'analyses sont donc à explorer et à appliquer dans cette recherche. D'une part, il s'agit de l'exploration de notre domaine d'étude qui est celui de la gestion de projet. Suivra ensuite une analyse du social qui utilise des narrations en langage naturel pour décrire les buts et les propriétés des acteurs, la nature et les motivations des relations entre les acteurs, et les relations entre acteurs et buts. Cette analyse descriptive fait une présentation plus nuancée de la CS d'équipe. Enfin, une analyse basée sur des modèles pour faire un lien plus direct et avoir une traçabilité des modèles visuels vers le système proposé, permettant ainsi à l'analyse d'avoir plus d'impact sur une conception et une implantation technique éventuelle de la solution envisagée.

En plus de la revue de la littérature sur la gestion de projets informatiques en rapport avec la dynamique des équipes, l'état de l'art que nous proposons dans ce chapitre

présente des visions complémentaires de l'analyse de la CS et celle du social qui constituent la charpente de notre travail.

2.2 État de l'art de la gestion de projet informatique

Le premier thème que nous aborderons dans ce chapitre sur l'état de l'art est consacré à l'exploration de notre domaine d'étude, celui de la gestion de projets informatiques. Il est vu sous l'angle de l'organisation des acteurs, des équipes participantes, et des mécanismes d'échange d'informations et de connaissances ainsi que leurs impacts sur les résultats attendus.

2.2.1 Définitions de la gestion de projets

Dans sa définition la plus simple, un projet est un ensemble planifié de tâches inter - reliées à exécuter sur une période fixe et dans certaines limites de coûts et autres (BusinessDictionary.com, 2016). Cependant, quand on prête plus attention au concept de projet, on distinguera 2 grands courants de pensée : la pensée traditionnelle qui considère un projet comme une mission temporaire (Taylor, 2003; Rose, 2013), et celle qui la considère plutôt comme routinière et répétitive (Davies et Brady, 2000; Engwall, 2003; Cicmil, 2006). Le manuel de PMBOK (Rose, 2013) qui constitue le manuel de référence académique et professionnelle en matière de gestion de projet, définit un projet comme « une mission temporaire entreprise pour créer un produit ou un service unique ». Les projets de façon générale seraient des systèmes sociaux temporaires (plutôt que permanents), ou des systèmes de travail qui sont constitués par des équipes au sein ou à travers les organisations pour accomplir des tâches particulières dans des délais (Manning, 2008). Il est aussi un effort coordonné, utilisant une combinaison de ressources humaines, techniques, administratives et financières afin d'atteindre un objectif spécifique dans un délai déterminé (Taylor, 2003). Dans une organisation, les projets et les opérations diffèrent principalement en ce sens que les opérations sont

continuelles et répétitives alors que les projets sont temporaires et uniques (Rose, 2013). Les tâches accomplies dans le cadre du projet sont circonscrites au champ d'action du projet et dans une certaine mesure, elles sont non routinières d'un projet à l'autre (Söderlund, 2000). Selon cette vision, de façon générale, pour de nombreuses organisations, les projets sont un moyen de répondre à ces besoins qui ne peuvent être abordés dans les limites normales d'exploitation des organisations.

Il apparaît que cette théorie traditionnelle de la gestion de projet souffre toujours des rêves rationalistes du début du 20^{ème} siècle (Buchanan, 1991; Packendorff, 1995) qui sont basées sur une perception selon laquelle le projet serait comme un système d'activité distinct et gérable qui, une fois conçu avec les techniques de planification appropriées, peut être isolé de l'environnement et mis en œuvre. Pourtant, les projets sont rarement des expériences ponctuelles, en particulier dans les entreprises professionnelles (Cicmil, 2006). Les projets ont toujours une histoire et un avenir potentiel, et se déroulent dans des contextes organisationnels (Engwall, 2003, Davies et Brady, 2000), c'est-à-dire qu'ils sont intégrés dans d'autres environnements systémiques, souvent permanents et évolutifs. Les organisations comportent certaines propriétés structurelles auxquelles les équipes se réfèrent lorsqu'elles s'engagent dans des activités organisationnelles, y compris la gestion de projets informatiques. Les projets sont intégrés dans un réseau social complexe de personnes, de ressources, d'institutions, de conditions de marché, et les activités qui s'y déroulent sont toujours affectées par les caractéristiques de ce réseau (Davies et Brady, 2000). Les tâches en tant que constituantes des projets, contiennent souvent des éléments de routine (c'est-à-dire des éléments opérationnels) et familières qui permettent des « économies de répétition » et le développement des capacités du projet (Davies et Brady, 2000). Dans ce sens, Davies et Bray proposent de mettre en place des changements organisationnels, des routines et des processus d'apprentissage pour exécuter à moindre coût et plus efficacement des projets similaires.

Ces dernières définitions de projet s'appliquent à tout type de projet, y compris les projets du domaine de l'informatique. Cependant, la notion de « projet informatique » elle-même, quoique fermement établie dans le langage des affaires, semble difficile à définir. Elle englobe une multitude de domaines, et les définitions qui y sont proposées font le plus souvent référence à l'objet ou au résultat issu de l'exécution du projet. Dans ce sens, le dictionnaire en ligne thefreedictionary.com (Thefreedictionary.com, 2017) le définit comme « Le développement, l'installation et la mise en œuvre de systèmes informatiques et d'applications ». McLeod et Smith (1996), pour leur part, proposent une définition qui nous semble à la fois englober les notions de gestion de projets classiques et des considérations informatiques. Selon cette vision, la gestion de projet informatique serait « le processus de planification, d'organisation et de délimitation des responsabilités pour la réalisation des objectifs spécifiques des organisations en matière de technologie de l'information ». Un projet informatique serait donc, tout projet dont tout ou une partie des résultats inclue les technologies de l'information ou un artefact informatique quelconque. On distinguera alors plusieurs types de projets informatiques selon le résultat produit à l'issue de son exécution :

- Développement de système informatique : un système personnalisé est développé à partir de zéro et entretenu tout au long de sa durée de vie;
- Implémentation des packages : une application prédéveloppée est installée, éventuellement avec des modifications;
- End User Computing : où les utilisateurs cibles du système participent de manière significative à son développement. Ceci est commun dans le cas des systèmes d'aide à la décision ou les systèmes de modélisation;
- Projets de prototypage : Cela peut être utilisé lorsqu'une technologie inconnue doit être utilisée, ou lorsque les exigences ne sont pas claires;

- Projets d'architecture de systèmes ou plan directeur informatique : ils sont utilisés pour définir le plan de systèmes stratégiques pour une organisation. Un plan est dérivé de la stratégie d'affaires et comprend l'ensemble des systèmes qui seront mis en œuvre par l'organisation au cours du calendrier de planification, normalement cinq ans;
- Projets de sélection (de fournisseurs, d'équipements ou d'application) : les package d'application ou la technologie sont sélectionnés pour répondre aux besoins de l'entreprise;
- Projets de réingénierie d'affaires : l'on recherche de nouvelles façons de traiter les processus d'affaires afin d'améliorer l'efficacité;
- Projets de mise en œuvre technologique tels que l'installation d'un réseau ou d'un système de messagerie électronique.

2.2.2 Un projet a trois dimensions structurelles

Il apparait dans la littérature et dans la variété de définition du concept de projet, une concordance sur ses constitutions. Il s'y dégage donc une perspective de structuration autour de trois dimensions ou propriétés structurelles fondamentales qui sont le « temps », la « tâche » et les « ressources ». La dimension ressources évoquée ici, est le plus fréquemment reliée à l'équipe de projet (Jacobsson *et al* 2015) quoiqu'elle puisse comprendre aussi les ressources matérielles et financières.

La dimension Tâches

« La tâche » traite des activités du projet en fonction de ce qui doit être fait. Elle reflète l'idée que les projets conduisent à certains résultats, guidés par des « projections » des produits ou de services souhaités ou des états futurs. Les tâches se réfèrent à la fois aux objectifs généraux et aux différents jalons des objectifs du projet, qui sont liés à certains produits, services ou autres résultats du projet. Elles se réfèrent aussi aux sous-tâches

qui sont attribuées aux participants du projet dans la réalisation des objectifs du projet (Engwall, 2003).

La gestion des tâches peut aider les individus à atteindre des objectifs, ou des groupes d'individus à collaborer et partager des connaissances pour la réalisation des objectifs collectifs (Riss *et al.*, 2005).

La dimension Temps

La dimension « Temps » informe sur la rapidité de l'exécution, ainsi que quand l'exécution des tâches du projet doit être complétée (Rose, 2013). Cette dimension temporelle se réfère à la fois à la contrainte temporelle de l'ensemble du projet et aux délais consécutifs pendant le projet. Les temps sont également liés à des jalons qui marquent les situations dans lesquelles certaines tâches sont accomplies et qui sont essentielles pour qu'un projet se déroule. Les Temps sont des dispositifs de structuration temporelle importants (Orlikowski, 2002).

La dimension Équipe

La dimension « Équipe » peut se référer à l'ensemble de l'équipe de projet ou aux sous équipes. Les équipes de projet ne sont pas seulement constituées par des individus travaillant ensemble temporairement mais par des positions que les acteurs prennent (Bechky, 2006) et les pratiques relationnelles dans lesquelles ils sont engagés avec d'autres pendant le projet (Grabher, 2002). Les relations à l'intérieur des équipes et entre les équipes sont régies par des mécanismes de dépendance, de confiance et de contrôle liés aux tâches en cours (Cicmil, 2006).

La gestion d'équipe est la capacité d'un individu ou d'une organisation à administrer et à coordonner un groupe d'individus pour effectuer une tâche. La gestion d'équipe implique le travail d'équipe, la communication, l'établissement d'objectifs et les évaluations de performance. En outre, la gestion d'équipe est la capacité d'identifier les

problèmes et de résoudre les conflits au sein d'une équipe (BusinessDictionary.com, 2017).

2.2.3 Projets informatiques et modèle mental partagé

Cette vision qui présente les projets comme étant rarement des expériences uniques et ponctuelles rejoint celle de la théorie des modèles mentaux qui traite l'environnement organisationnel comme un ensemble de règles symboliques et normatives (Mohammed et Dumville, 2001). Les acteurs dans cette structure sont considérés comme des agents bien informés qui appliquent des règles et des ressources en interaction et, ce faisant, ont un impact sur le flux continu d'événements. Les acteurs au sein des équipes de projet naviguent dans leur environnement social et organisationnel en construisant des descriptions mentales du but et de la forme du système dans lequel évoluent les projets, ainsi que des explications sur le fonctionnement et les états du système. Ils construisent en interne, des descriptions ou des explications du fonctionnement et des états du système, ainsi que des prédictions des états futurs du système (Rouse et Morris, 1986). L'on se réfère à cette construction sous le terme de « modèle mental ». Les modèles mentaux traduisent la réalité en représentations internes, et ces traductions guident la façon dont les acteurs font face aux exigences posées par la réalité (Park et Gittleman, 1995, Rook, 2013). Les membres de l'équipe de projet ont leurs propres modèles mentaux qui représentent leur compréhension des buts, des objectifs et des caractéristiques de l'équipe, ainsi que les liens entre leur propre travail et leurs actions collectives (Marks *et al.*, 2002).

Sur le plan des acteurs donc, les attentes en matière de rôle et les modèles d'interaction ne sont pas entièrement spécifiques à un projet. Elles sont liées au contexte organisationnel même si elles doivent être réajustées dans le cas particulier de chaque projet (Bechky, 2006). Les acteurs impliqués dans les projets ont leurs expériences personnelles acquises au sein de l'organisation et celles-ci définissent leur

comportement et la manière dont ils agissent à l'intérieur de leurs équipes et la nature des relations qu'ils entretiennent avec les membres de l'équipe.

Le partage des modèles mentaux est essentiel au fonctionnement des équipes de projets informatiques. Un modèle mental partagé ou une compréhension partagée par l'équipe, peut être considéré comme existant lorsque les modèles mentaux individuels des différents membres de l'équipe sont semblables. Autrement dit, une compréhension partagée se développe progressivement grâce à l'échange d'informations, en fonction des membres de l'équipe qui analysent les conséquences possibles de leurs actions et modifient leurs comportements (Qureshi, *et al.*, 2006).

Des études (Smith-Jentsch *et al.*, 2005; DeFranco *et al.*, 2011; Jo, 2012; Maynard et Gilson, 2014; Garg et Camp, 2015) ont documenté les effets bénéfiques des modèles mentaux partagés sur la performance des équipes. La plupart s'accordent pour définir les modèles mentaux partagés comme, « la compréhension organisée et la représentation organisée des connaissances sur un élément de l'environnement de l'équipe ». Ils peuvent améliorer la coordination et l'efficacité du travail de l'équipe surtout lors de l'exécution des tâches complexes, imprédictibles, urgentes et/ou nouvelles. Pour leur part, Lim et Klein (2006) avancent que les équipes dont les membres structurent et organisent les connaissances reliées à leur équipe de manière similaire sont susceptibles de trouver qu'il est relativement facile de coordonner leurs activités. Ils sont susceptibles de se mettre plus facilement d'accord sur les priorités et les stratégies d'équipe, ce qui rend plus efficace l'exécution des tâches. La compréhension partagée aide les membres de l'équipe à formuler des explications et des attentes collectives sur les processus de travail de l'équipe et à faciliter la communication et la coordination des activités de l'équipe (Cannon-Bowers et Salas, 1997; Jeffery *et al.*, 2005).

Sur une note plus granulaire, la littérature sur les modèles mentaux d'équipe suggère que les membres d'une équipe ne possèdent pas seulement un modèle mental, mais de

multiples modèles mentaux (Klimoski et Mohammed, 1994). Cannon-Bowers, Salas et Converse (1992) ont proposé qu'une équipe est plus susceptible d'être efficace si les membres de l'équipe partagent quatre types de modèles mentaux :

- Le modèle d'*équipement* capte la compréhension partagée des membres de l'équipe, de la technologie et équipement avec lesquels ils exécutent leurs tâches d'équipe.
- Le modèle de *tâche* capte les perceptions des membres de l'équipe et la compréhension des procédures d'équipe, des stratégies, des contingences de tâches et des conditions environnementales.
- Le modèle d'*interaction* de l'équipe reflète la compréhension par les membres de l'équipe des responsabilités, des normes et des modèles d'interaction des membres de l'équipe.
- Et le modèle d'*équipe* résume la compréhension par les membres de l'équipe des connaissances, des compétences, des attitudes, des forces et des faiblesses des autres.

Chacun des quatre modèles mentaux peut avoir une influence sur la prédiction du rendement de l'équipe. Il y a une relation directe entre la similitude des modèles mentaux au niveau de l'équipe et la performance de l'équipe. Dans une étude en laboratoire, Smith-Jentsch *et al.*, (2005) ont montré que la similitude du modèle mental de *travail* (combinaison du modèle d'*équipement* et de *tâche*) était significativement liée aux processus d'équipe et au rendement de l'équipe.

2.2.4 Les tâches en gestion de projets informatiques

Dans les sections précédentes de cet état de l'art, nous avons vu que la gestion de projets informatiques comporte des tâches routinières et des tâches non routinières. Ces dernières sont dictées par l'ensemble des normes écrites et non écrites, les procédures

et les habitudes organisationnelles qui rythment la cadence des activités dans l'entreprise.

Lorsque nous explorons plus spécifiquement les questions liées à la nature du travail d'un développeur informatique, les réalités ne sont guère différentes. Le développement d'application va bien au-delà de la simple activité traditionnelle de conception qu'effectue un développeur des années 1970 où la méthodologie de développement dominante de l'époque était linéaire et rigide avec ses phases distinctes et isolées de conception, de programmation, de test, et d'implantation (Awad, 2005). C'était l'époque des approches dites « waterfall » où le développeur travaille en vase clos durant la phase de conception. Dans cette méthode, les tâches de développement sont majoritairement intellectuelles et comportent peu de routines.

Avec un modèle plus moderne de développement collaboratif et dynamique tel que l'approche « Agile », et particulièrement dans les entreprises informatiques de service logiciel, les défis sont majeurs et répondent à de nombreuses contraintes technologiques et environnementales (Zhang, 2010). Une entreprise spécialisée dans la fourniture de services logiciels (*software as service* ou *cloud computing* en anglais) fait le choix d'opérer dans « un modèle de service logiciel qui donne un accès réseau pratique et à la demande à un pool partagé de ressources informatiques configurables (par exemple, stockage, applications et services) pouvant être rapidement provisionnées et libérées à la satisfaction du client ». Dans un tel modèle, l'entreprise prend de nombreux engagements dont notamment ceux de a) *facilité d'accès*, b) de *disponibilité de service* et c) de *réduction de risque*.

La *facilité d'accès* et la *disponibilité* se résument à offrir à ses clients, les services hébergés dans le cloud, généralement basés sur le Web, facilement accessibles 24 heures sur 24 via une variété d'appareils avec des connexions Internet. La notion de *réduction de risques d'affaires et techniques* elle, se réfère au fait qu'en externalisant

l'infrastructure de service vers le cloud, un client transfère ses risques d'affaires au fournisseur, souvent mieux informés et mieux équipés pour gérer ces risques.

Dans ce modèle de service, le support du client en environnement de production est aussi important que la création ou la mise à jour de l'application elle-même. Pour ce faire, les problèmes détectés sont classifiés par niveau de sévérité (Patel, 2009; Metheny, 2017) et les développeurs sont mis à contribution pour la recherche de solution. Les problèmes de sévérité *critique* (défaillance complète d'une fonctionnalité) sont traités immédiatement sans délai. La résolution est de l'ordre d'heures ou de minutes, car les conséquences peuvent être très graves. Elles sont souvent monétaires, de réputation, mais peuvent aussi toucher des vies humaines. Les exemples sont nombreux : En septembre 2014, le quotidien LaPresse de Montréal au Canada, rapportait que vingt établissements de santé de la région de Montréal - dont l'Institut de cardiologie - ont subi une longue panne de leurs systèmes informatiques, empêchant certains médecins d'avoir accès aux dossiers médicaux de leurs patients ou encore à leurs radiographies (Lapresse.ca, Septembre 2014); en février 2017, une panne de logiciel de contrôle de radar à l'aéroport d'Amsterdam a paralysé le trafic aérien pour plusieurs heures (Independent.co.uk, Février 2017); en mars 2007, l'Agence du revenu du Canada a suspendu certaines applications informatiques liées à l'impôt sur le revenu des particuliers, en raison de problèmes dans l'entretien des logiciels (affaires.lapresse.ca, mars 2007). Dans tous ces cas d'urgence, les développeurs doivent intervenir dans des conditions stressantes qui requièrent une attention particulière pour ne pas empirer la situation.

Concrètement, en dehors du travail purement conceptuel et intellectuel généralement attribué au développeur, celui-ci se retrouve avec des attributions additionnelles très variées et qui restent constantes quel que soit le projet. Il s'agit entre autres de :

- Documentation
- Création de package de déploiement

- Déploiements de package
- Diagnostic en situation de crise en suivant des procédures établies
- Correction de bogues en situation de crise
- Provisionnement et configuration applicative du serveur
- Préparation des tests unitaires (scenario de test)
- Préparation des données de tests unitaires
- Maintenance des données
- Mise à jour des bases de codes
- Réalisation des estimations
- Documentation des investigations en suivant des gabarits précis
- Suivi des travaux auprès d'autres équipes
- Etc...

2.2.5 Projet informatique et méthodologies de développement

L'étude des modèles mentaux partagés nous informe de la nécessité de considérer les impacts de la configuration de l'environnement de travail sur les équipes. L'approche de développement de logiciel adoptée par une organisation a donc une grande influence sur la manière dont les projets informatiques y sont gérés.

Dans cette section, nous abordons une revue de la littérature de ces approches mais particulièrement celles des méthodes Agiles.

Diverses approches ont été proposées pour le développement de logiciels au fil des années (Riemenschneider, 2002; Awad, 2005). Peu ont survécu, mais les méthodologies dites « Agile » introduites dans les années 1990 persistent et sont devenues de plus en plus populaires. Leurs processus sont destinés à soutenir les itérations plutôt que les phases que proposent les méthodologies de développement traditionnelles. Au cœur de la méthodologie Agile se trouve une promotion active des interactions continues entre les développeurs et les clients (Schwaber, et Beedle, 2002).

Le terme Agile a été popularisé par le Manifeste Agile (Fowler et Highsmith, 2001) qui définit ses valeurs et ses principes pour le développement de logiciels dans le cadre duquel les exigences et les solutions évoluent grâce à l'effort collaboratif d'équipes multifonctionnelles auto-organisées. Agile préconise une planification adaptative, un développement évolutif, une livraison précoce et une amélioration continue, et il encourage une réponse rapide et souple au changement.

Les trois méthodologies Agile les plus utilisées sont le Scrum, le Kanban et le Scumban.

- **Scrum** est conçu pour des équipes de trois à neuf développeurs qui divisent leur travail en actions qui peuvent être terminées dans des cycles de durée fixes (appelés « Sprints »), suivent les progrès et participent à des réunions quotidiennes de 15 minutes et collaborent pour livrer des logiciels exploitables à chaque Sprint
- **Kanban** est une méthodologie de visualisation du flux de travail, qui permet d'équilibrer la demande avec la capacité disponible et de repérer les goulots d'étranglement. Les éléments de travail sont visualisés pour donner aux participants une vue des progrès et des processus, du début à la fin. Les membres de l'équipe tirent le travail au fur et à mesure que la capacité le permet, plutôt que de travailler dans un processus à la demande.
- **Scrumban** est une méthodologie Agile hybride de Scrum et Kanban et a été conçue à l'origine comme un moyen de passer de Scrum à Kanban. Aujourd'hui, Scrumban est un cadre de gestion qui émerge lorsque les équipes emploient Scrum comme méthode de travail mais utilisent la méthode Kanban comme une lentille pour visualiser, comprendre et améliorer continuellement leur fonctionnement. (Ladas, 2009).

Le Manifeste Agile pour le développement de logiciel repose sur douze principes: (Fowler et Highsmith, 2001)

- Satisfaction du client par la livraison précoce et continue de logiciels précieux
- Bienvenue aux exigences changeantes, même en retard de développement
- Un logiciel fonctionnel est livré fréquemment (des semaines plutôt que des mois)
- Une coopération étroite et quotidienne entre les parties prenantes et les développeurs
- Les projets sont construits autour d'individus motivés, à qui l'on doit faire confiance
- La conversation en tête-à-tête est la meilleure forme de communication (co-localisation)
- Un logiciel fonctionnel est la principale mesure du progrès
- Développement durable; les parties prenantes et les développeurs doivent être capables de maintenir un rythme constant
- Une attention continue à l'excellence technique et à la bonne conception
- La simplicité - l'art de maximiser le travail non réalisé - est essentielle
- Les meilleures architectures, exigences et conceptions émergent des équipes auto-organisatrices
- Régulièrement, l'équipe réfléchit à la manière de devenir plus efficace et ajuste ses pratiques en conséquence.

Une itération dans SCRUM est composée de cinq phases :

- Phase 1 : Répertorier les problèmes et améliorations au logiciel.
- Phase 2 : Concevoir la solution
- Phase 3 : Développer la solution
- Phase 4 : Tester la solution
- Phase 5 : Déployer la solution

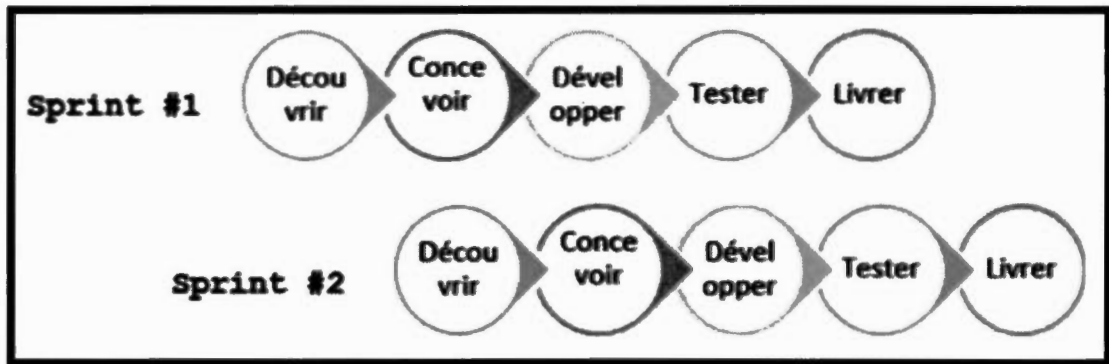


Figure 2.0 : Le cycle de vie du logiciel dans la méthodologie Agile

Les décisions en méthodologie Agile

Le Sprint comporte quatre activités dans lesquelles des décisions sont prises concernant le logiciel, la coordination et la performance de l'équipe. Les activités exécutées dans un cycle Sprint sont les suivantes :

- **Planification du Sprint** : C'est la rencontre qui marque le début de chaque Sprint. Il se compose d'un ensemble d'activités planifiées qui meubleront le travail de l'équipe durant le Sprint.
- **Exécution du Sprint** : C'est la période entre la fin de la rencontre de Planification et le début de la Révision du Sprint. C'est le moment où l'équipe travaille sur le produit réel pour atteindre l'objectif de Sprint. Le logiciel est développé et testé ici.
- **Révision du Sprint** : C'est généralement une rencontre qui implique l'équipe, et les parties prenantes. Elle se concentre sur le travail effectué par l'équipe lors du sprint, en comparant l'engagement au début du Sprint avec la livraison réelle à la fin du Sprint. Cette rencontre inclura généralement une démonstration du logiciel testé qui a été développé dans le sprint.

- **Rétrospective** : C'est l'occasion pour l'équipe de réfléchir à la façon dont elle travaille ensemble et de rechercher activement les domaines à améliorer.

Les décisions prises au cours de ces 4 activités sont de divers types et peuvent varier d'une organisation à l'autre. Elles sont opérationnelles, tactiques ou stratégiques (Drury *et al.*, 2012). De plus, l'adoption d'Agile peut modifier considérablement les descriptions de travail et les coutumes de l'équipe du projet. Par exemple, le chef de projet n'organise plus l'équipe, mais l'équipe s'organise et prend les décisions concernant ce qu'il faut faire (Schwaber et Beedle 2002). Au lieu de cela, le chef de projet travaille à supprimer les obstacles du processus, exécute et prend les décisions dans les Scrums quotidiens. Son rôle est maintenant celui d'un coach beaucoup plus que d'un manager de projet.

2.2.6 Concept de succès des projets informatiques

Le « succès » dans le domaine de la gestion de projet informatique est un concept ambigu. Sa définition varie selon la perception que l'on se fait de ce qui constitue ou non un projet réussi (Thomas et Fernández, 2008; Fayaz *et al.*, 2017). Certains auteurs parlent de réussite lorsque les conditions suivantes sont satisfaites : temps alloué, budget alloué, niveau de performance et spécification attendus, acceptation par les clients/usagers, apporte des changements de portée minimale ou mutuellement convenus, sans perturbation du flux de travail principal de l'organisation, sans changement à la culture d'entreprise (Kerzner, H., 2013). D'autres comme Baccarini font une distinction entre deux concepts de succès: succès de la gestion de projet et succès du produit (Baccarini, 1999). Le succès de la gestion de projet comporte trois composantes : (1) satisfaction des objectifs de temps, de coût et de qualité (input et output du projet) ; (2) qualité du processus de gestion de projet ; et (3) satisfaction des besoins des parties prenantes lorsqu'ils se rapportent au processus de gestion de projet. Le succès du produit comporte aussi trois composantes : (1) la réalisation des objectifs organisationnels stratégiques du gestionnaire du projet (objectif du projet); (2) la

satisfaction des besoins de l'utilisateur, et (3) la satisfaction des besoins des parties prenantes (lorsqu'ils sont liés au produit).

Quelle que soit la définition pour laquelle on opte pour évaluer le succès d'un projet, il convient de noter que la gestion de projets informatiques comporte de nombreux risques et le tableau des résultats révélés dans plusieurs études laisse entrevoir que les taux de succès de tels projets sont loin de répondre aux attentes de ses initiateurs (Kendra et Taplin, 2004, Carroll, 2013). Ces études rapportent que la plupart des projets informatiques ne parviennent pas à atteindre leurs objectifs. Elles notent des résultats où, environ seulement un tiers de tous les projets informatiques sont livrés à temps, en respectant le budget initial et avec les fonctionnalités requises. Les raisons évoquées, incluent entre autres, pauvre ou manque de communication entre les membres de l'équipe, et manque de coordination et de collaboration (Suraweera *et al.*, 2006, Daniels et LaMarsh, 2007).

Les problèmes avec les projets informatiques ne sont pas nouveaux. Des études plus anciennes avaient déjà relevé des résultats similaires et ont traité des difficultés que les projets informatiques ont à rencontrer les exigences de qualité (Kerzner, H., 1989), et les taux élevés des abandons (Ewusi-Mensah, K., 1997). Les projets informatiques sont abandonnés, généralement à des stades tardifs lorsque la plupart des dépenses ont déjà été engagées. Parmi ceux qui parviennent à fournir les résultats souhaités, certains le font avec beaucoup de retard sur le calendrier initial et en excédant le budget initial.

2.2.7 Risques et facteurs de risque des projets informatiques

Les équipes de gestion rencontrent plusieurs défis dans l'exécution des projets informatiques car ces derniers comportent des particularités que les autres projets n'ont pas. Dans la littérature sur la gestion de projets informatiques, les auteurs ont diversement abordé la question des risques à tel point qu'il nous est non convenable de donner une liste exhaustive ici. Nous retenons néanmoins une liste simplifiée des

risques qui nous semblent plus fréquemment évoqués. Cinq défis majeurs distinguent les projets informatiques des autres types de projets (Schmidt *et al.*, 2001, Tesch *et al.*, 2007)

- **Des risques uniques** : Les risques ne sont pas nécessairement plus importants avec les projets informatiques, mais ils sont souvent plus difficiles à identifier que dans d'autres types de projets. En plus, les résultats des projets informatiques ont souvent des impacts plus importants sur le succès d'autres projets dans l'organisation.

- **Des exigences de développement rapide** : en raison de la pression pour fournir rapidement une fonctionnalité nouvelle pour améliorer les opérations d'affaires, les gestionnaires de projet informatiques ont tendance à adopter une méthode de développement de logiciel qui privilégie la vitesse sur la qualité.

- **Changement technologiques fréquentes** : la rapidité des changements de technologique peut aussi constituer un défi pour les projets informatiques si les gestionnaires de projets informatiques sont amenés à changer d'outils en cours d'exécution de projet.

- **Multiples dépendances avec d'autres projets** : les projets informatiques sont intégrés aux processus d'affaires et ont souvent des liens avec d'autres projets informatiques. Ces liens et cette interdépendance peuvent engendrer des défis supplémentaires pour identifier toutes les exigences importantes du projet. Ils peuvent également conduire à des conflits de priorités et de conflits de ressources entre les projets.

- **Relation de compétence conflictuelle entre le chef de projet et les parties prenantes**. D'un côté, la complexité de la matière informatique échappe souvent aux parties prenantes des projets informatiques et de l'autre côté, les gestionnaires de projets informatiques ne maîtrisent pas toujours les règles d'affaires faisant partie des

objectifs de leurs projets. Ces écarts de compétence sont sources de conflit entre ces deux groupes d'intervenant.

2.2.8 Projet et acquisition de compétences non techniques

Le manuel de référence en matière de gestion de projet de PMBOK, fait la promotion des connaissances, des compétences et des outils techniques, et les présente comme meilleures pratiques dans la discipline, autant dans les programmes de formation scolaire que dans les milieux professionnels. Cependant, plusieurs autres études soulignent de plus en plus la nécessité d'un équilibre entre les compétences techniques et les connaissances et compétences non techniques ou humaines (Sampson, 2007; Pant, et Baroudi, 2008). Sampson B (2007, p.41) écrit : « Les compétences requises pour la gestion de projet sont maintenant souvent divisées en 50/50 en compétences traditionnelles, telles que la gestion et l'ordonnancement des risques, et les compétences orientées vers les humains, comme les relations interpersonnelles et la communication. »

Plus concrètement, de nombreuses approches de solutions sont suggérées pour aider les gestionnaires de projets à augmenter leur taux de succès. La liste des compétences et habiletés qui contribuent à la réussite du projet semble sans fin : connaissance des normes organisationnels, compréhension de la motivation des membres des équipes de projet, connaissance de la dynamique organisationnelle et du comportement de l'équipe (Brandel, 2006) ; capacité des gestionnaires à déléguer, à coordonner les idées d'individus divers, à être flexible et à faire face aux risques et au stress (DiVincenzo, 2006); être axé sur les résultats, avoir de solides compétences interpersonnelles, comprendre l'organisation, s'engager dans des valeurs de l'entreprise, exercer un solide style de leadership caractérisé par l'autorité, la responsabilité, l'adaptabilité, et s'engager à contrôler et à planifier (Kerzner, H. 2013); implication de la haute direction pour s'assurer que le projet ne manque pas de ressources et promouvoir l'esprit de changement à tous les niveaux, y compris sur eux-mêmes (Subba Rao, 2000); pouvoir

diriger efficacement les équipes (Sacco, 2006); et savoir planifier, négocier et rechercher des consensus (Black, 2006)

La liste des compétences et habiletés qui contribuent à la réussite des projets informatiques semble montrer que le succès de projet peut être atteint avec un ensemble de compétences techniques et d'excellentes compétences humaines et interpersonnelles. Certains auteurs préconisent que ces compétences humaines peuvent être enseignées et apprises plutôt que génétiques ou innées (Gillard, 2009, Pant, et Baroudi, 2008).

2.2.9 Conclusion de l'état de l'art de la gestion de projets informatiques

Sur le plan de la structuration, les spécifications des tâches, les contraintes de temps et les relations d'équipe sont des propriétés structurelles qui caractérisent conjointement les projets comme systèmes temporaires. Cependant, lorsqu'on va au-delà de la vision traditionnelle du projet, on se rend compte qu'il n'est pas aisé de dissocier les projets des activités opérationnelles des organisations.

Les projets sont rarement des expériences ponctuelles, c'est-à-dire qu'ils sont intégrés dans d'autres environnements systémiques, et les activités qui s'y déroulent en sont toujours affectées. Ce qui fait qu'au niveau des acteurs, les attentes en matière de rôle et les modèles d'interaction ne sont jamais entièrement spécifiques à un projet. Elles sont le produit des leçons apprises au sein de l'organisation.

Cinq défis majeurs distinguent les projets informatiques des autres types de projets : 1) des risques uniques ; 2) des exigences de développement rapide ; 3) des changements technologiques fréquents ; 4) de multiples dépendances avec d'autres projets ; 5) des relations de compétence conflictuelles entre les chefs de projet et les parties prenantes.

Pour réussir un projet, des compétences techniques et humaines sont nécessaires. Certains auteurs préconisent qu'au même titre que les compétences techniques, ces

compétences humaines peuvent être enseignées et apprises plutôt que innées ou génétiques.

2.3 État de l'art sur la conscience de la situation

2.3.1 Définitions et domaines d'application des recherches sur la CS

Dans la plupart des études existantes sur la CS, l'attention est surtout portée sur les processus qui mènent à une CS efficace au niveau de l'individu. Les travaux sur la CS d'équipe, la détermination de ce qui est nécessaire pour la CS d'équipe ou la CS partagée, les mécanismes d'amélioration de la CS d'équipe et ses impacts sur la performance sont plutôt rares. Et parmi ces études sur la CS d'équipe, presque tous sont dédiées à l'industrie de l'aviation; les urgences médicales, et les plates-formes pétrolières en haute mer. Cependant, quel que soit le domaine d'application des recherches sur la CS, les définitions proposées pour cette composante de la cognition humaine restent globalement concordantes (cf. le tableau 2.1). Dans cet état de l'art, nous revoyons quelques-uns de ces travaux sur la CS de l'individu, la CS d'équipe ainsi que la CS distribuée, et sur l'erreur en équipe.

En parcourant la multitude de définitions proposées pour le terme « conscience de la situation », on constate que ses premières principales composantes sont l'allocation de l'attention et la connaissance résultante (Fracker, 1989), ensuite viennent la perception, la compréhension, et la projection (Endsley, 1989). Bien que cette variété de concepts puisse suggérer qu'il y a peu de cohérence dans ce domaine conceptuel, une recherche au-delà des définitions proposées par un certain nombre de chercheurs (y compris Fracker, 1989, et Endsley, 1989) démontre des accords conceptuels sous-jacents qui fournissent un cadre pour la compréhension.

Tableau 2.1 Les définitions de la conscience de la situation

Cela signifie simplement que le pilote a une compréhension intégrée des facteurs qui contribueront à la sécurité du vol de l'avion dans des conditions normales ou non-normales. Plus cette connaissance est élevée, plus grand est le degré de conscience de la situation.	Regal <i>et al</i> (1988)
La détection des éléments de l'environnement dans un volume d'espace et de temps, la compréhension de leur signification, et la projection de leur statut dans un avenir proche	Endsley (1988)
La connaissance découlant de la focalisation de l'attention sur une zone d'intérêt à un certain niveau d'abstraction	Fracker (1989)
La connaissance et l'anticipation des événements, facteurs et variables pouvant affecter la conduite sûre et efficace de la mission	Taylor (1990)
Elle est basée sur l'intégration des connaissances résultant des évaluations récurrentes de la situation	Sarter, et Woods (1991)
La conscience adaptative tournée vers l'extérieur, générant les connaissances sur l'environnement dynamique de la tâche et le comportement nécessaire à la satisfaction d'un objectif externe	Smith et Hancock (1995)
En termes psychologiques, la conscience de la situation implique plus que la perception ou le « pattern matching »: il faut sans doute l'utilisation de toutes les fonctions cognitives supérieures qu'une personne peut apporter dans une tâche	Vidulich (1994)
Elle comprend des attentes, des indices saillants, des buts possibles et des actions associées	Klein (1995)
La prise d'information continue sur un environnement ou un système dynamique, l'intégration de ces informations aux connaissances antérieures pour créer une image mentale cohérente et l'utilisation de cette image pour la perception, l'anticipation et les réponses à des événements futurs	Wickens (1995)
La capacité à maintenir une perception précise de l'environnement, interne et externe à l'avion, l'identification de problèmes ou de problèmes potentiels, de besoins d'action, de déviations de la mission et la conscience des tâches exécutées	Prince et Salas (1997)
Le résultat d'un processus dynamique de perception et compréhension d'événements permettant l'anticipation de changements et le pronostic quant aux conséquences sur la réalisation de la tâche	Nofi (2003)

Il en ressort quatre facteurs clés :

- 1) Une fixation sur les processus cognitifs au moment de l'évaluation de la situation qui conduit à l'intégration et la compréhension de l'information,
- 2) Une composante temporelle dans laquelle les évènements du passé et ceux du présent sont utilisés pour faire des projections sur le futur (Endsley, 1989; Fracker, 1988; Sarter et woods, 1991) et
- 3) La distinction faite entre le processus d'évaluation de la situation et l'état (le produit ou l'objectif) de la CS (Fracker, 1988; Sarter et Woods, 1991; Tenney *et al.*, 1995; Nofi, 2003).
- 4) Une composante écologique qui place le sujet dans son environnement (Endsley, 1989; Smith et Hancock, 1995; Chalandon, 2007).

Ces différentes notions seront clarifiées dans les sections suivantes

2.3.2 La CS de l'individu est une fonction de ses modèles mentaux

Au niveau de l'individu, la CS implique d'être conscient de ce qui se passe dans le voisinage, afin de comprendre comment les informations, les événements, et ses propres actions auront un impact sur les buts et objectifs assignés, à la fois immédiatement et dans un avenir proche. Quelqu'un avec une « bonne » ou une « forte » CS a généralement un degré élevé de connaissance en ce qui concerne les entrées, les traitements et les sorties d'un système, c'est à dire une caractéristique presque innée de « sentir » les situations, les gens, et les événements qui se jouent en raison de variables que le sujet peut contrôler. Le manque ou la « mauvaise » CS a été identifié par plusieurs études comme l'un des principaux facteurs d'erreur et de mauvaises décisions attribuées à l'erreur humaine (Johannesen, 2012; Singh, 2012; Molesworth, 2015)

Selon Sarter et Woods (1991), l'évaluation adéquate des situations résultent des connaissances qui peuvent faire partie de la CS. Pew (1994) décrit le processus d'évaluation de la situation comme une sorte de charge de travail mental. Il a affirmé que les évaluations de la situation sont du plus grand intérêt dans la mesure des différences cognitives individuelles et dans le développement d'un programme de formation approprié, mais que les deux, CS et l'évaluation de la situation, doivent être comprises pour mener une bonne conception de système.

En décrivant la réalisation de la CS (ou le processus des évaluations de la situation), beaucoup conviennent que les individus perçoivent des informations critiques dans l'environnement à partir de l'exploration et de l'observation de l'environnement en ayant certaines attentes de ce qui sera perçu (Fracker, 1989; Neisser, 1976; Tenney *et al.*, 1995). Ces attentes sont fondées en partie sur les connaissances préexistantes des tâches de l'individu (Kass *et al.*, 1990; Tenney *et al.*, 1995). Ensuite, les individus intègrent dans leur mémoire de travail et comprennent des bribes d'information perçues (Endsley, 1995a). Cette intégration / compréhension est renforcée par les connaissances pertinentes, ou schémas, stockées dans la mémoire à long terme de l'individu (Endsley, 1990; Fracker, 1998; Sarter et Woods, 1991). Selon Endsley (1995), le schéma sélectionné est utilisé pour guider la compréhension, la projection dans l'avenir, et la sélection des actions nécessaires, de sorte que la CS à réaliser sera une fonction directe des modèles mentaux appropriés et la capacité de mémoire de travail.

Bien qu'il n'y ait pas une définition convenue des modèles mentaux (Rook, 2013), il ressort de la littérature sur la prise de décision en environnement dynamique, que les modèles mentaux sont une « représentation personnelle de notre monde ». Au niveau de l'individu, ils sont généralement reconnus pour servir trois fonctions clés: décrire, prédire et expliquer notre monde. Une supposition qui reste persistante tout au long de la dynamique du système est que les modèles mentaux sont créés, modifiés et utilisés pour supporter toutes les prises décision en environnement dynamique.

D'autres études, comme par exemple celles de Fracker (1988) et Zsombok *et al* (2014) trouvent que lorsque les connaissances préexistantes ne correspondent pas facilement à l'environnement, la mémoire de travail peut supporter des recherches supplémentaires d'informations dans l'environnement et si nécessaire construire un nouveau schéma à partir de ceux déjà disponibles à l'individu. Le modèle le plus populaire de cette approche de la CS est le modèle « *Situation awareness in dynamic decision making* » (cf. figure 2.1) proposé par Endsley (1995). Selon ce modèle, il y a un certain nombre de facteurs qui influencent le processus d'acquisition et de maintien de la CS. D'abord, les individus ont des habilités pour acquérir une CS dépendamment de leurs capacités cognitives, qui à leur tour peuvent être influencées par des capacités innées, l'expérience et la formation. De plus, les individus peuvent posséder certaines idées préconçues et des objectifs qui peuvent influencer leur perception et l'interprétation de leur environnement. En outre, Endsley affirme que la CS est aussi fonction de la conception (design, ergonomie) du système tant en termes de la mesure dans laquelle le système fournit les informations nécessaires, et le format dans lequel cette information est fournie. Enfin, d'autres caractéristiques de l'environnement de la tâche, comme le stress, la charge de travail, la complexité du système et autres, peuvent affecter également la CS de l'individu.

De ce modèle de Endsley, on peut déduire 3 caractéristiques importantes :

- Une différenciation entre CS en tant que « produit » et la CS comme un « processus »;
- Le « processus » d'acquisition et de maintien de la CS est cyclique et adaptatif;
- Il y a un certain nombre de facteurs à la fois internes et externes à l'individu qui affectent l'acquisition et le maintien de la CS.

Ce modèle de la CS en environnement dynamique a été testé dans plusieurs études, dans divers domaines d'application, notamment la médecine (Singh *et al.*, 2012) où les

auteurs ont étudié 254 cas de diagnostics de cancer de poumon et du cancer colorectal. Il s'est avéré qu'il y a des erreurs de diagnostic dans 32% des cas et que pour toutes ces erreurs, les praticiens ont manqué au moins un niveau de CS, soit, la perception de l'information, la compréhension de l'information ou la prévision d'événements futurs, et le choix d'une action appropriée induit par ces 3 niveaux.

Gillespie *et al* (2013) pour leur part, ont révisé le modèle de la CS en environnement de décision dynamique, pour son applicabilité chez les équipes de chirurgiens. Ils ont fait valoir l'importance de la communication et son exactitude à l'intérieur de l'équipe chirurgicale en salle opératoire et son effet sur la CS et la réduction des erreurs de décision. Récemment, pour valider le 1^{er} niveau de la CS tel que stipulé par le modèle de Endsley, Souza *et al* (2015) ont proposé une méthodologie pour l'évaluation de la qualité de l'information en étudiant des données sur les événements de cambriolage en milieu urbain.

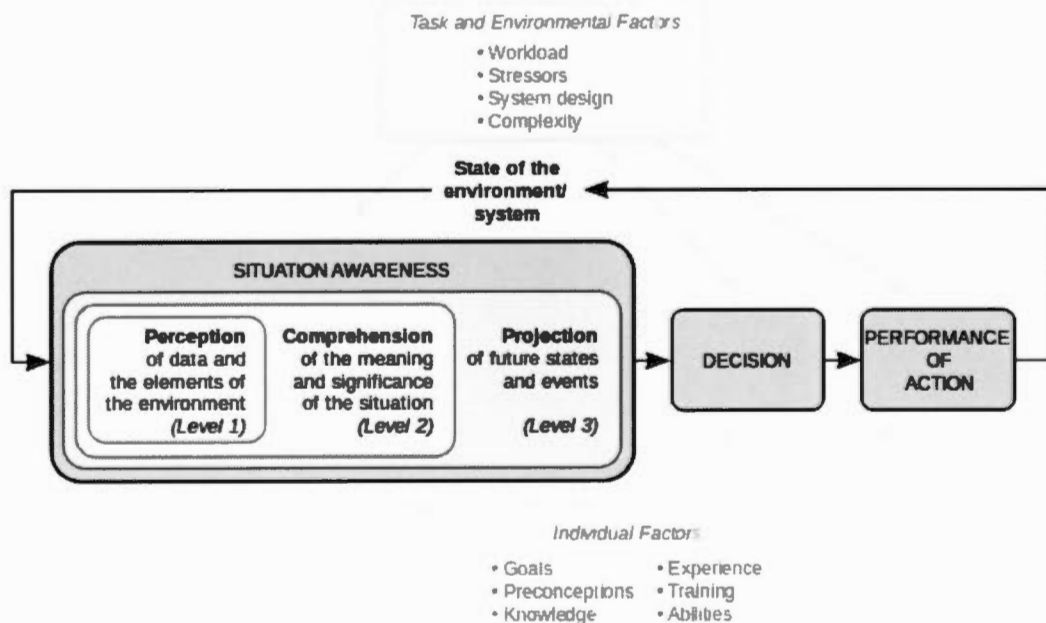


Figure 2. 1 Modèle de la CS en environnement de décision dynamique (Endsley, 1995a)

2.3.3 La CS de l'individu : un état et un système en constante évolution

Sarter et Woods (1991) ont souligné l'importance de l'aspect temporel de la CS et la nécessité qu'il puisse être continuellement mis à jour ou modifié à travers le processus d'intégration des évaluations continues de la situation. Cela implique la capacité de se souvenir des événements précédents. Ils ont souligné la complexité de la CS en déclarant qu'elle devrait être considérée comme un système ouvert en constante évolution qui peut être affecté par une variété de variables.

Les évaluations de la situation conduisent à des anticipations qui aident à guider le comportement d'exploration additionnelle (Tenney *et al.*, 1995), et les informations provenant de la CS sont utilisées pour guider les activités individuelles et prédire des événements futurs (Kass *et al.*, 1990). Ainsi, bien que la CS soit considérée comme un état, il convient de noter que la CS est dynamique et est continuellement modifiée et mise à jour au fil du temps.

La figure 2.2 présente une vue de haut niveau des principaux constituants de la CS de l'individu. Simplement dit, la CS se produit comme une conséquence d'une interaction des connaissances préexistantes et les attentes de l'individu; les informations disponibles dans l'environnement; et les compétences cognitives de traitement qui incluent l'allocation de l'attention, la perception, et la projection. Il en résulte une augmentation de la connaissance de l'individu, un changement dans les attentes, et un autre cycle d'extraction de l'information.

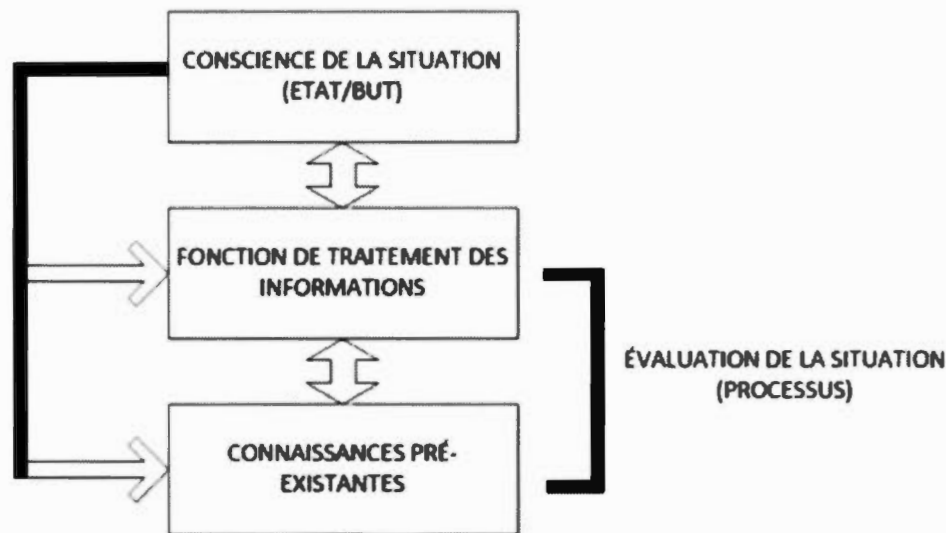


Figure 2. 2 Conceptualisation de la CS de l'individu (Salas *et al.*, 1995)

2.3.4 Vision écologique de la CS

Bien que l'interprétation de ce concept écologique place l'individu au centre de l'analyse, la problématique de la CS trouve ses bases dans la théorie de la compétence qui définit ce qui doit être connu pour être en mesure de résoudre un problème de traitement de l'information. Cette vision de la CS précise ce qui doit être connu pour résoudre une classe de problèmes posés lors de l'interaction entre l'individu et un environnement dynamique. Selon Smith et Hancock (1995), la CS doit être abordée en qualité de propriété de l'interaction individu/environnement, plutôt qu'en termes de processus cognitif qui s'opère dans l'esprit de l'individu. Elle serait « la conscience adaptative tournée vers l'extérieur, générant les connaissances sur l'environnement dynamique de la tâche et le comportement nécessaire à la satisfaction d'un objectif externe » (Smith et Hancock, 1995).

Chalandon (2007) pour sa part, présente le concept de la CS comme un système dual individu-tâche dans lequel l'individu se retrouve dans ce qui est conceptualisé comme le « sujet » et où la tâche serait l'« environnement ». Conséquemment, l'évaluation de

la situation amène à explorer deux points de vue ontologiquement différents mais complémentaires : d'une part, dans la vue du sujet, la CS est le produit d'une activité d'interprétation, et le cadre de représentation fonctionnelle définit cette vue comme une construction dynamique personnalisée. Dans cette vue du sujet, la CS est associée aux invariants internes du couplage sujet/environnement. D'autre part, la vue de l'environnement définit les interactions possibles en situation et les contraintes significatives associées à ces interactions. Cette vue associe la CS aux invariants externes du couplage sujet/environnement.

2.3.5 Équipe et CS d'équipe

La CS d'équipe représente beaucoup plus de complexité que la simple combinaison de la CS des membres isolés de l'équipe (Schwartz, 1990). La CS d'équipe implique des activités uniques, telles que la coordination et le partage de l'information.

L'étude de la CS d'équipe et sa compréhension sont importantes en sciences cognitives pour deux raisons. Premièrement, comme Cannon-Bowers et Salas (1990) l'ont souligné, la performance dans de nombreux systèmes complexes dépend des activités coordonnées d'une équipe de personnes. Aussi, comme indiqué précédemment, il est démontré dans le domaine de l'aviation que la CS joue un rôle crucial dans la performance des équipes, et il est prouvé que cela affectera d'autres types d'équipe (Cannon-Bowers *et al.*, 1995). Deuxièmement, bien que beaucoup de recherches aient été menées dans le domaine de la performance de l'équipe, il reste encore du travail à faire pour connaître la nature et les propriétés du travail d'équipe (Salas *et al.*, 2008). Étant donné que la CS joue un rôle essentiel dans la façon dont certaines équipes fonctionnent, un examen plus approfondi de la CS d'équipe pourrait s'avérer bénéfique pour améliorer le travail d'équipe. Par conséquent, dans la section suivante, nous passerons en revue les processus et les comportements par lesquels la CS peut être établie en équipe et fournirons un cadre pour la conceptualisation de la CS d'équipe. Nous aborderons par la suite la notion d'équipe et de travail d'équipe.

2.3.5.1 Vue d'ensemble de la CS de l'équipe

Dans la plupart des situations impliquant la prise de décision dynamique ou le contrôle d'une situation dynamique, la tâche est soit effectuée par l'individu au sein de l'équipe ou par l'équipe. Le domaine de gestion de projets informatiques est un exemple de système dynamique où la question de la CS peut être abordée au niveau de l'équipe plutôt qu'au niveau de l'individu. Comme dit précédemment, il y a eu très peu de documentation et très peu d'effort pour expliquer le processus ou l'état de la CS d'équipe. Ces études ont cherché à identifier les comportements critiques et les processus cognitifs qui contribuent au développement de la CS.

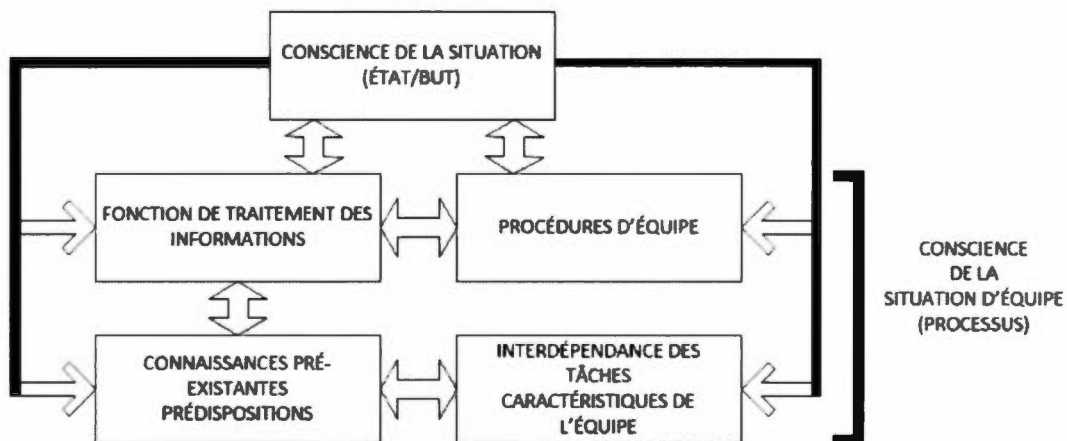


Figure 2. 3 Conceptualisation de la CS d'équipe (Salas *et al.*, 1995)

Bolman (1979) fait référence à la CS de groupe en tant que « théorie de la situation de l'équipage » (en anglais, *Crew's theory of the situation*), qui selon lui, est affectée par deux facteurs: la théorie de la pratique, qui est décrite comme le processus par lequel les membres individuels du groupe testent leurs théories de la situation en collectant et en partageant des informations; et la capacité des membres du groupe à mettre en commun leurs compétences pour adhérer et défendre une théorie particulière. Bolman

soutient donc que dès qu'une théorie de la situation est implantée dans un groupe, les membres du groupe partagent ladite théorie et s'en servent comme un cadre de référence commun pour leur tâche de groupe. Cependant, la théorie de la situation du groupe n'est pas statique. Quand les incohérences surviennent, les membres sont tenus de remettre en question ces incohérences et modifier leur théorie en conséquence.

Bolman (1979) prend en compte plusieurs comportements comme essentiels dans le développement, le maintien et la modification de la CS du groupe. Ces comportements comprennent le monitoring des informations spécifiques sur des positions; la confirmation et le recoupement des informations dans le groupe; la communication des informations de situation pertinentes aux autres membres; et les coordinations d'activités.

Wellens (1993) définit la CS d'équipe comme le « partage d'une vision commune entre deux ou plusieurs individus concernant des événements environnementaux actuels, leur signification, et leur statut futur ». Il a suggéré que la CS d'équipe pourrait être maximisée lorsque chaque membre opère une surveillance d'un segment différent de l'environnement avec suffisamment de chevauchement avec les autres membres pour assurer les possibilités de coordination. Wellens a également brièvement discuté de la notion de modèles mentaux partagés. Son utilisation du terme décrit deux aspects : l'un pour l'accord sur le fonctionnement du groupe et l'autre pour la compréhension commune d'un problème particulier auquel le groupe est confronté. Ce dernier aspect rejoint aussi l'usage qu'en a fait Orasanu (1990) dans sa conceptualisation du modèle de problème partagé, qui est développé par la communication des évaluations de la situation, la détermination du problème, et l'élaboration de plans pour gérer le problème. Bien que Wellens ait reconnu l'importance de la communication dans le maintien de la CS d'équipe, il a souligné que la réalisation de la communication requiert des efforts et de l'attention.

Endsley (1995) a soutenu que la CS d'équipe a deux composantes. D'une part, la CS exigée de chaque membre de l'équipe et d'autre part, le chevauchement dans la CS qui est nécessaire chez les membres de l'équipe, en particulier pour la coordination. Dans leurs études dans le domaine de l'aviation, Robertson et Endsley (1994) ont noté la relation entre la gestion des ressources de l'équipage et la CS de l'équipe (CRM, *Crew Resource Management*). Ils ont utilisé cette relation comme une base sur laquelle construire une explication de la façon dont la CS peut être étudiée et enseignée. À partir de la liste de contrôle de simulation d'opération de ligne aérienne qu'ils ont développé pour la mesure du CRM (Foushee et Helmreich, 1993), ils ont extrait les attitudes et comportements qui paraissent être affectés positivement par la formation dispensée sur le CRM et ont décrit comment ceux-ci pourraient affecter la CS. Robertson et Endsley ont établi ainsi une connexion entre les attitudes de l'équipage étudié et les comportements, puis entre attitudes et modèles mentaux partagés, et enfin à la CS de l'individu. Certains des comportements qu'ils ont identifiés incluent la communication, l'autocritique, la préparation et la planification, les séances d'information de l'équipage, et la répartition des tâches. Les acquis de cette recherche ont été répliqués plus tard par O'Connor (2013) dans son étude sur la formation des jeunes médecins sur les concepts de CS.

Application aux organisations

La capacité d'une organisation à être continuellement consciente d'elle-même et de son environnement est étudiée sous le vocable de conscience de la situation. Dans sa thèse de doctorat (McManus, 2008), McManus a fait une investigation sur 10 entreprises provenant de divers secteurs, tailles, et types d'industries dans le contexte néo-zélandais pour découvrir quels sont les problèmes communs qui favorisent ou créent des obstacles à une résilience accrue. Les entreprises étudiées proviennent des secteurs suivants : Fabricant / exportateur, autorité locale, entrepreneur privé, service public, éducation, grossiste, détaillant, producteur primaire, fournisseur de technologie. Dans

ses conclusions, il ressort que la résilience organisationnelle est entre autres définie en fonction de la prise de conscience globale de la situation. Cette autre étude récente de Demir *et al* (2017) démontre que dans les entreprises à commande non hiérarchique, une communication efficace d'équipe, élément fondamental de la coordination d'équipe, est essentielle à la fois pour une prise de conscience de la situation de l'équipe et pour la performance de l'équipe. Le terme « Entreprise non hiérarchiques » exclut bien sûr les organisations militaires, les casernes de pompiers et autres entreprises similaires où les tâches sont plutôt de nature opérationnelle. C'est dire que le concept de conscience de la situation fait son chemin vers les entreprises où la majorité des tâches ne sont pas de nature opérationnelle. En fait, l'application de ce concept tend à dépasser la sphère de la cognition de l'individu et englobe maintenant la cognition et la coordination de l'équipe. Ceci est étudié sous divers concepts cognitifs dont « Conscience de la situation d'équipe » (Demir *et al.*, 2017), « Connaissances contextuelles partagées » (Rognin, 2000), « Cognition d'équipe » (Espinosa *et al.*, 2004), « Sensibilisation à l'espace de travail partagé » (Pinelle *et al.*, 2003), etc... mais en fait, tous traitent de ce que nous avons qualifié dans notre thèse de conscience de la situation d'équipe.

Ainsi, pour supporter l'émergence de l'étude de la CS dans les organisations auprès des chercheurs, Kurapati *et al.*, 2012 proposent dans leur article « *A Theoretical Framework for Shared Situational Awareness in Sociotechnical Systems* », un cadre d'analyse pour une approche systémique de la gestion des changements dans les systèmes sociotechniques.

2.3.6 CS distribuée

Le modèle de la CS de l'individu et de l'équipe est un peu différent de celui de la CS distribuée (en anglais *Distributed Situation Awareness*), qui est présenté par Hutchins (1996) et Stanlon *et al* (2006, 2010). Stanton *et al* ont fait valoir que le concept de CS distribuée qui englobe non pas un individu, mais un système non statique complexe, y compris les sous-systèmes humains et non humains, et les interactions entre eux, peut

amener à une meilleure compréhension de la façon dont certaines performances sont fournies par une équipe. Conséquemment, dans un contexte d'équipe distribuée, le processus cognitif survient au niveau du système plutôt qu'au niveau de l'individu. Selon ce point de vue, les individus ainsi que les objets qu'ils utilisent sont considérés comme faisant partie d'un système cognitif où la connaissance est un produit construit à la suite de l'exécution des tâches coordonnées.

Selon Stanton *et al* (2010), la CS distribuée est une caractéristique clé des systèmes collaboratifs complexes qui peut être définie comme « la connaissance activée pour une tâche spécifique, dans un système, à un moment précis par des agents spécifiques » (p. 34). En considérant ce point de vue, on serait tenté de dire qu'il est important d'examiner les interactions entre les agents (acteurs humains et non-humains), y compris les interactions entre les individus et les interactions entre les individus et les outils, pour décrire comment l'information CS est distribuée et coordonnée au sein du système (Salmon *et al.*, 2009).

Fioratou *et al* (2010) ont révisé le concept de la CS distribuée pour son applicabilité chez les anesthésistes. Les auteurs ont fait valoir que l'anesthésiste, le chirurgien, le patient, et le suivi du patient sont des sous-systèmes qui interagissent de manière dynamique, en permanence, et à la fois implicitement et explicitement. Par exemple, en relevant l'état du patient à l'anesthésiste, l'appareil d'affichage de l'anesthésiste est un médiateur physique entre le patient et l'anesthésiste et peut être classé comme un dispositif de communication indirecte entre eux. Récemment, plusieurs autres études (Nazira *et al.*, 2014; Sandhåland *et al.*, 2015) s'accordent à dire qu'un système de communication efficace est un ingrédient fondamental dans la performance des systèmes. L'importance de la communication est capitale surtout lorsque l'incertitude s'introduit dans le système et que les agents humains sont contraints de fonctionner hors de leurs routines normales d'opération. Par exemple, Nazira *et al* (2014) rapportent dans leur étude sur la sécurité sur les plateformes pétrolières, que lorsque

les opérateurs postés sur le pont doivent rapporter les anomalies observées (à l'aide de leur sens), aux contrôleurs, la communication ne peut plus être simplement considérée comme un canal de livraison de messages. Les communications dans ces scénarios sont plutôt des analyses au cours desquelles les agents évaluent la gravité des situations afin d'aboutir à des décisions correctes et rapides. Il est donc essentiel que la bonne information soit transférée à la bonne personne dans l'équipe, au bon moment pour permettre à chaque individu d'atteindre et de maintenir la CS nécessaire à sa fonction dans le système (Stanton *et al.*, 2010)

La figure 2.4 donne une représentation de la CS distribuée telle que proposée par Salmon *et al* (2008)

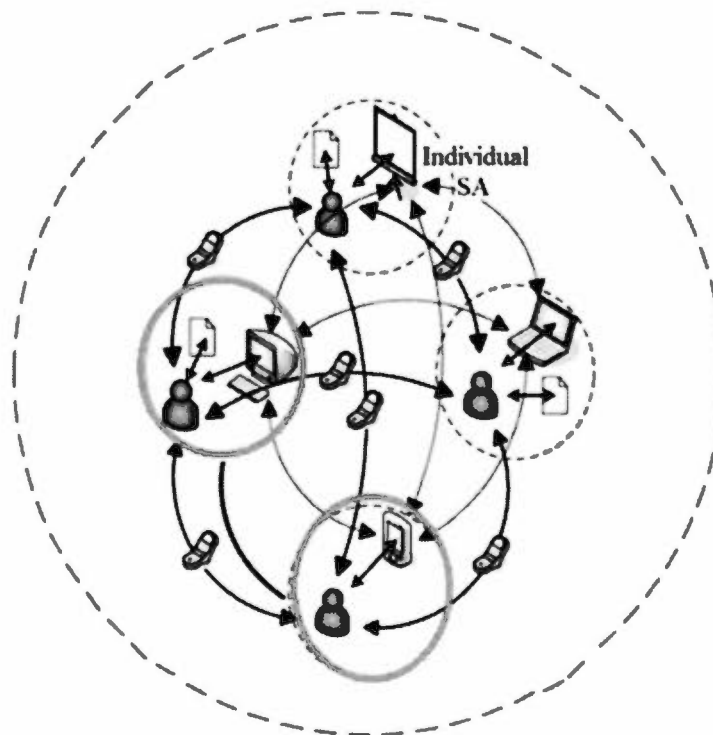


Figure 2.4 Illustration de la CS distribuée (Salmon et al., 2008)

2.3.7 Groupe ou équipe?

Il apparaît de ce qui précède que des chercheurs ont appliqué diversement l'étude de la CS au groupe et à l'équipe. Il convient cependant de s'attarder un peu sur ces notions afin de les situer par rapport à notre étude.

Johnson et Johnson (1991) définissent un groupe comme « deux ou plusieurs personnes dans une interaction en face-à-face, chacun conscient de son appartenance au groupe, chacun est au courant de l'appartenance des autres au groupe, et chacun a conscience de leur interdépendance positive dans leurs efforts pour atteindre des objectifs communs ». Une équipe est plus structurée qu'un groupe, en ce que chaque membre a un rôle et travaille de manière solidaire avec les autres vers un but commun significatif (Morgan *et al.*, 1986). Sundstrom *et al* (1990) pour leur part définissent une équipe de travail comme un petit groupe d'individus qui partagent les responsabilités de leur organisation; alors que pour Cohen et Bailey (1997), une équipe peut être considérée comme une collection d'individus qui sont interdépendants dans leurs tâches; qui partagent les responsabilités des résultats de leur travail; qui se voient et qui sont vus par d'autres comme une entité sociale à part entière intégrée dans un système social plus large (par exemple une organisation). La « Responsabilité » serait l'obligation encourue par les individus dans leurs rôles dans l'organisation pour effectuer efficacement les tâches (Kerzner, 2013). En analysant toutes ces définitions, on se rend compte que les équipes ont tendance à avoir une histoire et un avenir.

Dans la littérature de la psychologie organisationnelle, il y a eu des débats autour de la différence entre équipe et groupe. Le groupe a été utilisé comme descripteur beaucoup plus vaste que l'équipe. Les groupes sont souvent formés et dissous pour accomplir une tâche spécifique à court terme. Pour Cameron et Green (2015), un groupe est une collection d'individus qui tracent une frontière autour d'eux, ou peut-être que nous de l'extérieur traçons une frontière autour d'eux afin de les définir en tant que groupe.

Paris *et al* (2000) identifient d'autres caractéristiques qui différencient les équipes des petits groupes, entre autres, l'usage de multiples sources d'information, l'interdépendance des tâches, la coordination entre les membres, des objectifs communs et précieux, la spécialisation des membres dans des rôles et responsabilités, connaissances spécifiques sur des tâches, communication intensive, et des stratégies d'adaptation pour aider à répondre aux changements.

En regardant les équipes de développeurs informatiques en entreprise, nous devons réaliser qu'il y a des décisions de groupe et des décisions d'équipe même si l'essentiel du personnel constitue une équipe. Dans le cadre de la présente étude, il est juste de conclure à partir d'ici qu'une équipe est une sorte de groupe. Par conséquent, la plupart des principes du groupe s'appliquent aussi à l'équipe, et notre travail de recherche sur les groupes pourraient être également applicable dans le cadre de l'équipe. L'inverse n'est cependant pas vrai.

2.3.7.1 Les propriétés et performances du groupe

- Les propriétés sociales du groupe

Tout groupe de personnes travaillant ensemble sur une période de temps développent certaines propriétés sociales de base. Ce sont:

- **Cohésion** : Ceci est familièrement connu comme l'esprit d'équipe ou le moral. Il est considéré comme le processus psychologique qui transforme de nombreuses personnes dans un groupe (Vaughan et Hogg, 2005). Opérationnaliser le concept a été difficile, mais il semble être lié à l'attraction interpersonnelle, à l'augmentation de la conformité et à l'amélioration de la communication entre les membres.
- **Socialisation** : Ceci peut être définie comme la « relation dynamique entre le groupe et ses membres. Une relation qui décrit le passage de ses membres

dans le groupe en termes d'engagement et d'évolution des rôles ». (Vaughan et Hogg, 2005).

- **Normes** : Ce sont des croyances partagées sur le comportement approprié pour un membre du groupe. Ils fournissent un cadre de référence qui permet aux membres d'adopter un comportement approprié pour eux-mêmes, et d'évaluer le comportement des autres.

Ces mêmes propriétés doivent être trouvées dans une équipe de gestion de projets informatiques, car ils sont un agrégat de personnes qui travaillent ensemble quotidiennement aussi bien dans le même lieu physique qu'à distance. Par conséquent, ces propriétés du groupe seraient exposées et prises en compte dans notre étude.

- **Les propriétés structurelles du groupe**

Aux propriétés sociales que nous venons de voir, s'ajoutent deux propriétés structurelles qui sont couramment identifiées dans les groupes. Il s'agit du rôle et du réseau de communication

- **Rôles** : Ils représentent la division du travail et la fonction remplie par chaque membre du groupe.
- **Réseau de communication** : Cela peut être défini comme un ensemble de règles qui régissent la communication appropriée entre les membres du groupe.

Il semble que ces propriétés aussi soient compatibles avec un groupe de gestion de projets informatiques. Tout d'abord, dans une entreprise, les rôles sont explicitement mentionnés (par exemple, directeur du test de qualité, responsable analyste, architecte de base de données, architecte système), et aussi la présence d'un réseau de

communication qui est influencé par le système hiérarchique ou collaboratif des organisations.

2.3.7.2 Mémoire transactive : la mémoire du groupe

Un autre phénomène observé dans les groupes est la mémoire transactive. Un des avantages du travail en groupe est que les groupes ont tendance à retenir plus d'informations que le meilleur individu du groupe (Stephenson, 2015). Le terme « mémoire transactive » se réfère à une théorie développée par Wegner *et al* (1991) qui suggère que les groupes développent un système partagé de codage de connaissances, où chaque individu est seul responsable d'une partie de ce que le groupe a besoin de savoir. Chaque membre comprend qui est responsable et spécialiste de chaque domaine de la mémoire, et chacun sait comment accéder à l'expertise à travers des processus de communication. De cette façon, un système de mémoire transactive peut fournir aux membres du groupe de plus et meilleures connaissances que tout individu isolé. La théorie de la mémoire transactive suggère qu'un système de mémoire transactive est mise en évidence lorsque chaque membre de l'équipe possède une expertise unique, et que l'un se fie sur l'autre pour prendre la responsabilité d'un aspect spécifique de la tâche ou de la connaissance affairant à la tâche, et coordonne le traitement de l'information et les activités des tâches (Lewis 2003, 2011). Sans un système de mémoire transactive dans un groupe, il y a forte chance que les expertises des membres soient redondantes, incompatibles et difficile à intégrer. Les équipes avec un système de mémoire transactive moins développé font plus face à des informations discordantes et à des niveaux plus élevés de tension (Hood *et al.*, 2014)

2.3.8 Équipes et macro cognition

Dans la dernière décennie, une quantité importante de recherches sur l'équipe a mis l'accent sur le concept de modèles mentaux partagés (SMM). Les chercheurs s'accordent à dire que les équipes ont tendance à développer un modèle mental partagé

(Jonker *et al.*, 2011; Van den Bossche *et al.*, 2011; Gentner et Stevens, 2014). Ceux-ci peuvent essentiellement être décrits comme des structures de connaissances ou de représentations cognitives qui sont partagées entre les membres de l'équipe, et sont utilisées pour organiser l'information, comprendre les événements et faire des prévisions, et prendre des décisions (Rouse et Morris, 1986). Les modèles mentaux permettent une compréhension similaire de la situation entre les membres de l'équipe, et orientent la façon dont les membres de l'équipe vont interagir (Gentner et Stevens, 2014). Ils ont également été utilisés pour expliquer la performance coordonnée des équipes (Van den Bossche *et al.*, 2011), en particulier, il a été suggéré qu'un SMM peut conduire à l'utilisation d'une communication plus proactive et donc à des meilleures performances.

Jonker *et al* (2011) mettent en avant l'idée de « *team as system* ». Essentiellement, l'idée est d'amener les modèles mentaux partagés à un niveau plus loin. Une équipe peut être considérée comme un système. L'esprit de l'équipe serait de ce point analogue à l'esprit d'une personne. Les membres de l'équipe interagissent entre eux pour former un tout intégré. Ceci à son tour, implique les affirmations suivantes :

- Les processus cognitifs sont liés au comportement de l'équipe
- L'équipe a des ressources d'attention limitées
- Il peut arriver qu'on ne soit pas en mesure de récupérer certaines connaissances importantes de la mémoire de l'équipe.
- L'esprit d'équipe est affecté par la motivation de l'équipe
- Une équipe doit apprendre à coordonner ses actions
- Les équipes peuvent tirer des conclusions (par la reconnaissance de patterns dans les informations détenues par les différents membres de l'équipe)

Cette façon de voir l'équipe infère que la littérature relative à la CS de l'individu sera également applicable à la compréhension de la CS d'équipe.

Cooke *et al.*, 2013 pour leur part, vont dans une direction un peu différente. Selon leur théorie, la cognition de l'équipe se trouve dans les interactions entre les membres de l'équipe plutôt que dans les propriétés statiques de la structure de leurs connaissances partagées. Ils soutiennent que la cognition de l'équipe est une activité et devrait être mesurée et étudiée au niveau de l'équipe dans le contexte de l'environnement de l'équipe, où la tâche de l'équipe se déroule.

A la lumière des propriétés et fonctions que possèdent une équipe, il nous paraît logique que l'étude de la CS d'équipe en milieu entreprise et dans les équipes de gestionnaires de projets informatiques, s'attarde un peu plus sur ces concepts et voit dans quelles mesures les propriétés de l'équipe peuvent être pris en compte dans l'analyse des besoins en CS. Le tableau 2.2 présente un résumé des variables qui peuvent influencer la performance de l'équipe (Morgeson *et al* 1997; Meister 1985).

Tableau 2.2 Variables qui influencent la performance de l'équipe

Facteurs	Description
Facteurs contextuels	Variables qui appartiennent à l'environnement dans lequel s'effectuent les activités de l'équipe.
Facteurs structurels	Variables émanant principalement des sources externes à l'équipe, mais qui peuvent inclure certaines variables internes à l'équipe (par exemple l'organisation).
Facteurs de conception de l'équipe	Variables inhérentes à l'équipe elle-même : la façon dont elle est conçue.
Facteurs procéduraux	Variables inhérentes à l'équipe elle-même : la manière dont elle fonctionne.
Facteurs de contingence	Variables résultant des sources internes et externes à l'équipe

2.3.9 Les erreurs de décision

On ne peut pas aborder la question de la CS d'équipe sans parler des erreurs et des questions de la qualité des décisions. Par conséquent, dans notre recherche sur la CS dans une équipe de développeurs d'application informatique où les acteurs sont

constamment confrontés à des situations de décision, il est important de reconnaître ce facteur et tenter d'établir une compréhension de comment et pourquoi les erreurs de décision se produisent. On peut faire valoir que si nous pouvons développer une bonne compréhension de ce qui cause des erreurs de décision en environnement dynamique, et mettre en œuvre des aides et des interventions appropriées, nous devrions être en mesure de suggérer une solution qui réduirait de manière significative l'erreur liée à la prise de décision.

2.3.9.1 Explication des erreurs de décision

Klein (1998) définit de façon générale trois principales catégories d'erreur de décision. Deux d'entre elles sont associées à la cognition du décideur :

- Le manque d'expérience du décideur.
- Le manque d'information (ce qui influe sur le développement d'une bonne connaissance de la situation).

La troisième est un facteur externe

- La mauvaise simulation mentale (situation dans lesquelles les décideurs auraient remarqué les signes d'un problème, mais les auraient ignoré).

Il est probable cependant, qu'il existe une corrélation entre le contexte et la cognition, qui contribuerait à l'erreur.

2.3.9.2 Environnement, cognition et erreur

Au niveau de l'individu, le contexte entourant la prise de décision en environnement naturel est considéré comme très important. Lewandowsky et Kirsner (2000); Orasanu *et al* (2001) suggèrent qu'au lieu de se concentrer sur la cognition interne de l'individu, si l'attention est plutôt tournée vers la disponibilité de l'information, les objectifs

poursuivis et le niveau d'expérience, il devient plus facile de retirer le blâme sur le décideur et d'investiguer plutôt sur la qualité de la situation.

Selon cette vision, les erreurs de décision s'inscrivent dans différents types de schémas contextuels.

- **Contexte relationnel** : Il s'agit des réseaux relationnels et des processus de collaboration entre les individus. Il prend également en compte l'humeur et le moral créés par ces interactions. Il comprend particulièrement l'analyse de la communication implicite et explicite, et la compréhension partagée qui en résulte. C'est là que « Pressions organisationnelles et sociales » se définissant comme une catégorie prédictive d'erreur, cadrerait (Orasanu *et al.*, 2001).
- **Contexte physique** : Il s'agit de l'environnement physique dans lequel les décideurs se trouvent. Cet environnement peut avoir des éléments qui sont dynamiques, et d'autres qui sont constants. Par exemple, pour un développeur informatique, l'espace qu'il occupe demeure relativement constant, toutefois, l'équipement utilisé par le personnel avec qui il travaille et les clients qu'il reçoit sont dynamiques. L'environnement dynamique contient souvent des indices qui déclenchent le schéma d'expert et la prise de décision. Cet environnement dynamique créerait parfois des informations ambiguës qui alimenteraient la prise de décision. Ainsi, tel que mentionné par Orasanu *et al* (2001), l'ambiguïté résultant d'un environnement dynamique est prédictive d'erreur.
- **Contexte émotionnel** Orasanu *et al* (2001) : Il s'agit des émotions humaines enlacés dans le scénario qui pourrait affecter la perception, et / ou la performance. Cela pourrait se référer aux :
 - Stress et niveaux d'anxiété
 - Facteurs de personnalité (associés à des individus)

- Événements qui déclenchent l'émotion (comme une mauvaise nouvelle)
- **Contexte temporel** : Ceci fait référence à des trames de temps dans lesquels l'événement se trouve. La durée de la décision à prendre, ou le temps disponible pour que la décision soit prise entrerait dans cette catégorie. En termes de sous-composantes on aurait :
 - Heure de la journée.
 - Durée de l'événement

Des risques changeants au fil du temps (dynamique des risques; Orasanu *et al.*, 2001) seraient un artefact des contextes inter-reliés temporels et physiques.

Pour comprendre l'erreur due au manque de CS, à la fois, le contexte et la connaissance doivent être examinés; mais les frontières entre les individus et le contexte doivent être adoucies. La nécessité apparente pour mettre une telle limite est en partie un effet secondaire de la tentative de traiter l'individu comme une unité isolée de l'analyse cognitive, sans d'abord localiser l'individu dans un monde culturellement construit Hutchins (1996).

D'autres types d'erreur et préjugés qui seraient liés à l'environnement des organisations et que nous devons prendre en considération, sont ceux identifiés dans la recherche classique sur la décision par Kahneman et Tversky (1982). Il s'agit des heuristiques. Ces « raccourcis mentaux » qui produisent des réponses avec moins d'efforts cognitifs mais avec des risques d'erreurs. Nous ne les révisons pas en détail ici, mais certains de ces types d'erreur seront traités dans notre solution (formation sur la CS) au même titre que celles présentées dans la présente section.

2.3.9.3 Travail en équipe et erreurs de décision

Dans de nombreuses situations naturelles des organisations, une équipe de personnes est nécessaire pour aider à la prise de décision. Ainsi, une compréhension de l'erreur au niveau de l'équipe est également nécessaire. En termes d'erreur de décision en équipe, il y a des déclencheurs cognitifs individuels, ainsi que les causes organisationnelles et sociales. Par conséquent, toute décision prise dans un environnement d'entreprise doit être considérée à trois niveaux: cognitif (individuels), organisationnel (équipe / contexte) et social (équipe / contexte). Nous avons déjà abordé les erreurs liées aux individus. Ici, nous ne développerons désormais que les déclencheurs organisationnels et sociaux des erreurs de décision en équipe.

- Les déclencheurs organisationnels d'erreurs de décision

Il est intéressant de noter que de nombreuses erreurs de décision sont attribuées à des décisions de gestion de haut niveau (Sonesh *et al*, 2013). Il est donc évident que la prise de décision à un certain niveau dans l'entreprise soit examinée dans le contexte organisationnel dans lequel elle se produit.

Dans le domaine de la psychologie organisationnelle, la culture organisationnelle, la structure du groupe, la structure du pouvoir, la structure de rôle et la structure de communication ont été identifiés comme des facteurs organisationnels qui peuvent potentiellement influencer les résultats de performance (Buchanan et Huczynski, 1997).

- Les déclencheurs sociaux d'erreurs de décision

Les explications des influences sociales sur les décisions couvrent un grand domaine de la recherche, en particulier issu du domaine de la psychologie sociale de prise de décision. Jones et Roelofsma (2000) suggèrent que certains préjugés sociaux importants à considérer dans la prise de décision en équipe sont : l'effet de faux

consensus, la pensée unique, la polarisation de groupe, et la propagation de l'engagement du groupe.

Autant que les préjugés sociaux qui peuvent influencer sur l'issue de la décision, il existe plusieurs autres processus de groupe documentés qui sont considérés comme susceptibles d'avoir un impact sur la qualité de la décision de l'équipe. Un exemple de ceci est le processus connu sous le nom « brainstorming ». La génération d'idées nouvelles est importante dans un contexte d'entreprise mais en dépit des preuves que des réflexions en groupe génèrent plus d'idées que les membres pris individuellement, il n'a pas été constaté que ces derniers sont plus créatifs dans le groupe (Mullen *et al* 1991; Thompson et Wilson, 2015). En fait, il a été plutôt établi que les groupes où les individus créent leurs propres idées sans interaction, sont plus créatifs que ceux qui travaillent en tant que groupe (Thompson et Wilson, 2015).

Les raisons pour lesquelles cela pourrait être le cas sont notamment :

- Appréhension d'évaluation : certains membres du groupe qui tiennent à donner une bonne impression d'eux présentent un degré d'auto censure;
- Paresse sociale : des membres présentant une perte de motivation parce que la tâche est de nature collective;
- Blocage de la production : les individus doivent composer avec d'autres qui génèrent des idées en même temps qu'eux. Ils sont alors souvent étouffés sur le plan créatif.

Selon Kalargiros et Manning (2015), les limitations de la méthode brainstorming seraient plutôt associées à des différences culturelles entre les membres qui composent le groupe.

2.3.10 Conclusion de l'état de l'art sur la CS

Malgré la confusion apparente qui se dégage de la multitude de définitions proposées pour décrire la CS, une analyse plus élaborée révèle que toutes les recherches semblent se mettre d'accord sur quatre facteurs clés pour sa conceptualisation : 1) une fixation sur les processus cognitifs de l'individu au moment de l'évaluation de la situation, qui conduit à l'intégration et à la compréhension de l'information; 2) une composante temporelle dans laquelle les événements du passé et ceux du présent sont utilisés pour faire des projections sur le futur; 3) une distinction faite entre le processus d'évaluation de la situation et l'état de la CS; et 4) une composante écologique qui place le sujet dans son environnement.

Parmi les différentes approches qui ont été proposées pour l'étude du concept de CS, nous avons retenu la classification suivante : du point de vue de l'individu, la CS est abordée comme un processus cognitif qui s'opère dans l'esprit de l'individu; du point de vue de l'équipe, la CS d'équipe implique des activités telles que la coordination et le partage de l'information; en revanche, du point de vue de l'écologie des systèmes, la CS est comprise comme un processus qui se produit à travers les interactions entre les individus et les outils qu'ils utilisent pour atteindre leurs objectifs.

Dans les limites de cette classification, quel que soit la vision dans laquelle l'on aborde l'étude de la CS, il semble que ce qui importe ultimement, c'est la compréhension et l'amélioration de la performance humaine à l'intérieur d'un système réel, naturel, dynamique et complexe. La performance est considérée comme un objectif externe vers lequel mène la CS.

Le tableau 2.3 présente les travaux majeurs qu'il y a eu dans l'étude de la conscience de la situation, ses variantes dénominations et ses domaines d'application.

Tableau 2.3 Classification des travaux sur la CS

Terminologies	Classification	Exemple de domaine d'application
<i>Conscience de la situation</i> Chalandon, (2007); Endsley,(1995); Smith et Hancock, (1995)	Individu	Urgence (casernes pompiers, militaire, aviation)
<i>Conscience de la situation distribuée</i> Stanlon <i>et al</i> (2006, 2010); Fioratou <i>et al</i> (2010)	Individu Équipe Artefacts technologiques	Urgence (casernes pompiers, militaire, aviation, navigation maritime)
<i>Cognition distribuée / Système cognitif commun</i> Hutchins, et Klausen, (1996)	Individu Équipe Artefacts technologiques	Aviation
<i>Théorie de la situation de l'équipage</i> Bolman (1979)	Équipe	Urgence (militaire, aviation)
<i>Sensibilisation à l'espace de travail partagé</i> Pinelle <i>et al.</i> , 2003	Équipe	Organisations sociotechniques
<i>Connaissances contextuelles partagées</i> Rognin, (2000)	Équipe	Organisations sociotechniques
<i>Cognition d'équipe</i> Espinosa <i>et al.</i> , (2004)	Équipe	Organisations sociotechniques
<i>Conscience de la situation d'équipe</i> Demir <i>et al.</i> , (2017)	Équipe	Organisations sociotechniques

2.4 État de l'art sur la modélisation du social

2.4.1 Introduction

De récents travaux en ingénierie des exigences ont adopté une perspective orientée agent. La notion d'agent ou d'acteur (parties prenantes) dans l'ingénierie des exigences, cependant, concerne « les acteurs dans le monde » (Yu, 2011), sur lesquels la plupart des analystes n'ont aucun contrôle. Le but d'introduire une abstraction d'acteur (ou de but, ou toute autre abstraction) pour la modélisation des exigences est de soutenir l'acquisition, l'exploration et l'analyse des solutions alternatives possibles des « systèmes dans le monde » de la façon dont ils se rapportent les uns aux autres, et de découvrir les avantages et les inconvénients des solutions alternatives. Un rôle majeur de l'ingénieur des exigences est d'aider les acteurs à exprimer leurs besoins et leurs préoccupations, à explorer des alternatives, et comprendre les implications. Les caractéristiques telles que l'intentionnalité, l'autonomie et la socialité y ont des connotations bien particulières et constituent des attributs d'acteur dans le monde.

La littérature fait état de l'usage de méthodologies de modélisation du social. Nous présenterons certaines de ces méthodologies et mettrons en évidence leurs avantages et limites par rapport aux objectifs de notre recherche. Nous nous attarderons particulièrement sur la méthodologie iStar qui semble mettre l'emphasis sur certaines propriétés clés de l'acteur, notamment l'intentionnalité, la dépendance et la vulnérabilité. Mais avant, il importe de faire un rappel historique de ce qu'est la modélisation des exigences.

2.4.2 Historique de la modélisation des exigences

Au cours des deux dernières décennies il y a eu de grandes avancées dans l'ingénierie des systèmes, notamment avec le développement des abstractions puissantes et naturelles de haut niveau avec lesquelles l'on peut modéliser des systèmes complexes. Des abstractions de procédures, de types de données, d'objet et d'agent en sont des

exemples. Dans l'ingénierie des exigences, la nécessité de modéliser le monde est reconnue depuis longtemps, du fait que les exigences concernent la définition des relations entre une solution informatique et son environnement. Son objectif est de comprendre les besoins et les contraintes des parties prenantes.

Les principales activités dans l'ingénierie des exigences incluent l'analyse de la structure de l'organisation et de domaine, les investigations d'acquisition, la négociation, la spécification, la communication et la documentation. Les techniques de modélisation et d'analyse sont conçues pour aider l'analyste dans ces tâches.

Les techniques d'analyse structurée ont popularisé l'utilisation d'approches systématiques pour exprimer et analyser des exigences. Elles sont axées sur la modélisation des activités et des données (entrées, sorties, et les flux de contrôle entre activités, entité-relation), et leur décomposition hiérarchique (Ross, 1978). Les diagrammes de flux de données que ces techniques recommandent, incluent notamment le stockage de l'information, ainsi que des sources externes, délimitant ainsi les interfaces d'un système avec son environnement (DeMarco, 2002). L'attention de l'analyste est mise sur les fonctionnalités du système et sur la solution technique à réaliser.

Plus tard, les techniques d'analyse ont intégré des descriptions complexes en des ensembles structurés de diagrammes basés sur un petit nombre de concepts ontologiques, permettant ainsi une analyse de base. Par exemple, on pourrait vérifier l'exhaustivité et la cohérence par l'appariement des flux d'entrée et de sortie. Par la suite, ces tâches ont été soutenues par des ateliers logiciels (outils *CASE*), bien que ce soutien reste très limité de par le degré de formalité existant dans les modèles. Ces techniques continuent d'être largement utilisées de nos jours.

Des concepts de but et d'objectif ont commencé par être pris en charge dans certaines méthodologies pour gérer les connaissances dans les systèmes. Par exemple, RML

(Borgida *et al.*, 1985), au lieu de se concentrer intensément sur les modèles de conception de systèmes complexes, regarde les buts et les objectifs d'un projet. RML utilise ensuite des modèles pour décomposer ces objectifs dans les exigences qui sont facilement compréhensibles par les parties prenantes et les développeurs.

Les analystes commencent par comprendre que la définition des exigences n'est pas un simple processus de transfert de connaissances dans lequel les ingénieurs d'exigence suscitent et documentent des connaissances actuelles du client (Grady, 2010). Au contraire, il est un processus cognitif dans lequel les parties prenantes trouvent de façon collaborative, ce qui doit être fait par la compréhension des problèmes et des domaines, en apprenant d'autres parties prenantes dans un rapport de négociation et de discussion des points de vue différents. Pour renforcer l'analyse, diverses approches de formalisation ont été introduites, accompagnées d'améliorations ontologiques appropriées. Ainsi, en plus des concepts fonctionnels d'activités et entités, on verra des abstractions qui permettent de « capturer plus de connaissance du monde », comme ceux de la logique du premier ordre RML. D'autres logiques, modales (pour décrire des informations sur le temps), déontiques (pour exprimer les permissions et obligations), linéaires ou sous-structurelles (pour les ressources et leurs utilisations) ont également été introduites pour la modélisation des exigences (Nuseibeh, et Easterbrook, 2000). Beaucoup de ces fonctions ont par la suite trouvé leur chemin vers la modélisation orientée objet, par exemple UML (Rumbaugh *et al.*, 2004) qui offre des ontologies statiques et dynamiques regroupées dans une unité de comportement. Les ontologies statiques servent à souligner la structure statique du système en utilisant des objets, des attributs, des contraintes, des opérations et des relations. Les ontologies dynamiques pour leur part soulignent le comportement dynamique du système en montrant les collaborations entre des objets et des changements aux états internes des objets. Cependant, l'analyse effectuée avec ces modèles continuent d'être sur le comportement et les interactions. Il n'y a pas de concepts ou des considérations d'alternatives stratégiques intentionnelles (Yu, 2002).

L'approche composite de conception (Feather, 1991) considère des systèmes multi-agents dans lesquels les agents peuvent jouer un ou plusieurs rôles et interagir les uns avec les autres afin d'échanger des connaissances et coordonner leurs activités, de manière à réaliser des objectifs ou des exigences fonctionnelles à l'échelle du système. Cette approche identifie d'abord la nécessité de considérer les systèmes et leurs environnements en termes d'agents qui ont le choix. Les décisions et les actions d'un agent peuvent imposer des limites à la liberté d'action d'autres agents (Yu, 2002).

La modélisation dirigée par le but a été incorporée dans un certain nombre de framework d'analyse des exigences (Wooldridge, 2000; Yu et Mylopoulos, 1994). Ils utilisent des concepts abstraits pour décrire le système organisationnel, et des concepts concrets pour bâtir l'application. Ils fournissent des mécanismes d'acquisition graduelle des exigences (par exemple, Potts *et al.*, 1994). Ils soutiennent l'utilisation répétée des questions « Pourquoi ? Comment? et Pourquoi pas? » dans les constructions hiérarchiques moyens/fins, pour comprendre les motivations, les intentions et les raisons (Yu, 2011). Ils révèlent des conflits et aident à identifier des solutions potentielles (Robinson et Volkov, 1998).

L'introduction du concept du but dans l'ontologie des modèles d'exigences représente un changement important. Auparavant, le monde à modéliser était composée d'entités et d'activités et de leurs variantes. Les nouvelles ontologies ont attribué des buts aux acteurs. En d'autres termes, pour faire l'ingénierie des exigences, il ne suffit pas de s'occuper des aspects statiques et dynamiques, on doit aussi reconnaître l'intentionnalité dans le monde (Yu, 2011).

L'étape logique suivante pour l'analyse des exigences est de partir de l'ingénierie des exigences dirigée par les buts vers une ingénierie à part entière orientée vers l'acteur, pour marquer une reconnaissance du social ainsi que l'intentionnalité. La nécessité de cette étape est évidente si l'on prend en compte la nature changeante des systèmes et

de leurs environnements. Dans un monde naturaliste, il est plus difficile de rencontrer des systèmes qui ne composent pas avec des acteurs.

2.4.3 Les approches de la modélisation de l'entreprise

Parmi les travaux les plus significatifs en matière d'ingénierie des exigences dirigée par les buts ou par l'agent, nous en avons pu répertorier une dizaine, mais dans cette liste, les méthodologies suivantes cadrent beaucoup plus avec notre domaine d'étude. Il s'agit notamment de : M*-OBJECT (Berio *et al.*, 1995); l'approche téléologique (Loucopoulos et Kavakli, 1995); GBRAM (Antón, 1996); KAOS (Dardenne *et al.*, 1993); ISTAR (Yu, 1994) et Actor-Network Theory (Latour, 2005)

2.4.3.1 M*-OBJECT

M*-OBJECT est une méthodologie pour l'analyse, la conception et l'implantation des systèmes d'information développés dans les environnements CIM. Elle est basée sur une approche orientée objet et elle couvre l'analyse approfondie des aspects statiques et dynamiques des informations.

M*-OBJECT comprend trois phases principales: l'analyse de l'organisation, l'analyse conceptuelle, et l'analyse de l'implantation.

- La phase d'analyse de l'organisation produit un ensemble de besoins structurels.
- La phase d'analyse conceptuelle fournit une spécification formelle exécutable des fonctionnalités statiques et dynamiques de l'entreprise. L'information statique se réfère à la structure des données et des contraintes d'intégrité que les données doivent satisfaire. L'information dynamique se réfère au comportement de l'organisation, c'est à dire les opérations qui doivent être effectuées sur les données et les relations de causalité qui existent entre elles.

- La phase d'analyse de l'implantation fournit la description des bases de données nécessaires à l'implantation du système d'information du système CIM.

A chaque phase, un formalisme dédié et un ensemble de directives sont fournis. Ils peuvent être informatisés.

La phase d'analyse de l'organisation est celle qui nous concerne le plus dans cette recherche puisque c'est durant cette phase que l'on produit un modèle d'organisation basé sur les concepts d'agent, d'évènement, de fonction (processus et activité) et de composant. C'est également au cours de cette phase que l'on analyse la situation actuelle de l'organisation pour définir une nouvelle structure, plus productive ou plus efficace de l'organisation du système de production en cours d'analyse. La sortie dans cette phase est une description de l'environnement du système, c'est à dire le cadre dans lequel la conception et la mise en œuvre du système d'information auront lieu, et la spécification des exigences du système, c'est à dire une description de haut niveau, axée sur l'utilisateur, des composants de l'objet entreprise qui sera automatisé. L'idée de base de la phase d'analyse de l'organisation de M*-OBJECT, est que le monde réel peut être décrit en utilisant les concepts suivants: agent, composants, événements, temps, et, selon le niveau d'abstraction, fonction, processus et activité. Plus précisément, la tâche d'analyse est réalisée en utilisant le méta-modèle de M*-OBJECT décrit à la figure 2.5.

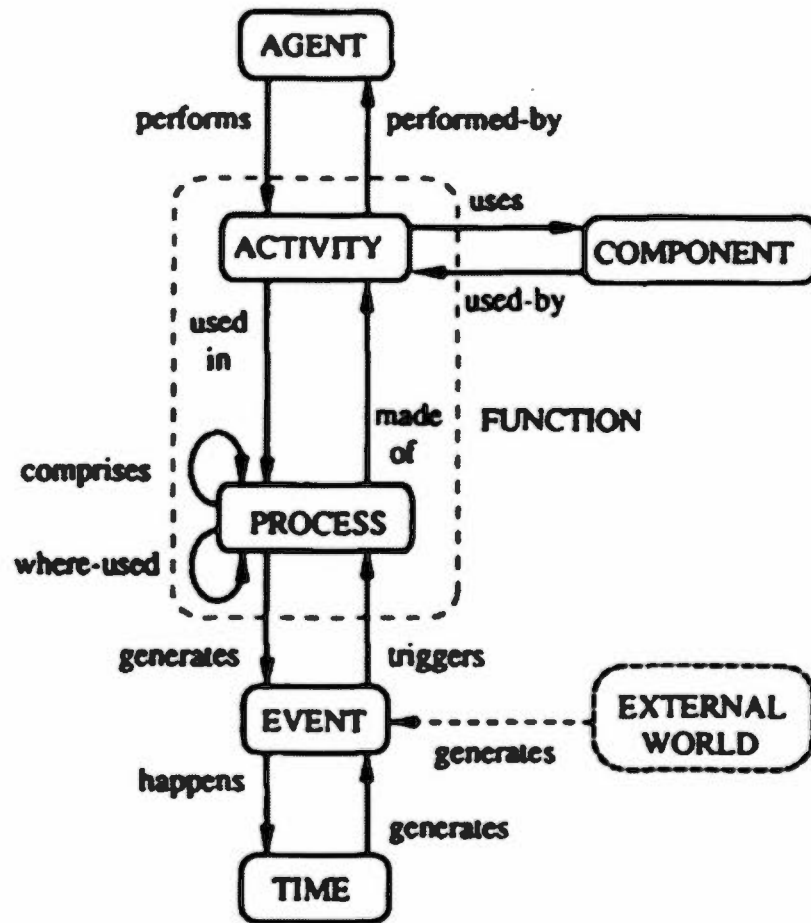


Figure 2.5 Le méta modèle de M*-OBJECT

Le modèle d'organisation est utilisé pour décrire les différentes unités fonctionnelles de l'entreprise objet à trois niveaux de détail (environnement, unités d'organisation, et centres de travail) différents.

L'architecture d'ensemble de la méthodologie de M*-OBJECT est illustrée dans la figure 2.6.

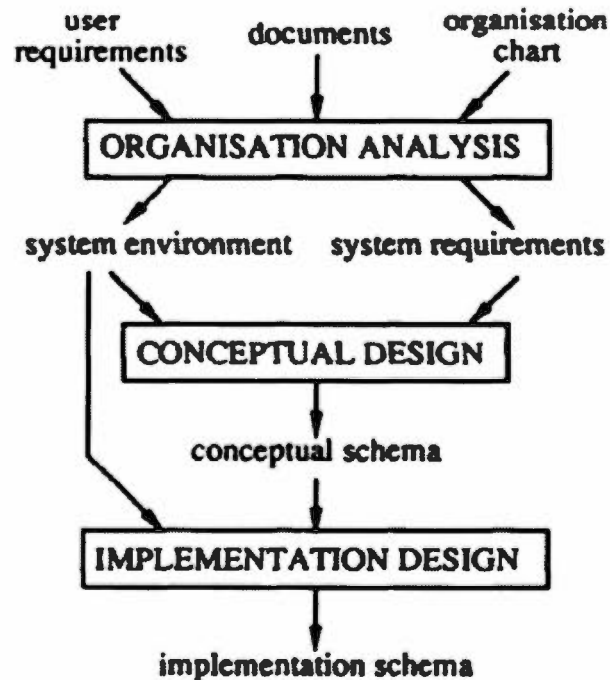


Figure 2.6 Vue d'ensemble de l'architecture de la méthodologie M*-OBJECT

En termes d'avantages, M*-OBJECT est soutenue par un ensemble intégré de modèles qui permettent la description de l'entreprise à trois niveaux différents, qui correspondent aux différentes phases de la méthodologie: 1) niveau gestion ou niveau organisation pour lequel une représentation claire et précise du comportement de l'entreprise est requise, 2) le niveau conceptuel ou au niveau du système qui fournit une vision commune, un pont entre les gestionnaires et les ingénieurs d'une part et des spécialistes en informatique d'autre part, et 3) le niveau de mise en œuvre ou de développement d'applications qui ne s'adresse qu'aux spécialistes en informatique et programmes d'application.

En termes de limite, la méthodologie M*-OBJECT n'aborde pas explicitement les questions de relation de dépendance entre les acteurs du système. Elle n'offre pas la possibilité de capturer les raisons qui existent derrière la tâche affectée aux acteurs.

2.4.3.2 L'approche téléologique pour la modélisation d'affaires

Proposé par Loucopoulos et Kavakli (1995), cette approche est fondée sur la modélisation explicite des objectifs de l'entreprise, les rôles sociaux et les opérations du point de vue téléologique. L'un de ses principes premiers est qu'un modèle d'organisation est pertinent s'il permet de fournir des explications sur le comportement de l'entreprise. Cette approche établit l'analyse des buts et l'analyse des dépendances organisationnelles comme la première étape vers une compréhension en profondeur de l'entreprise. Elle offre la possibilité de capturer les raisons qui existent derrière la tâche de l'entreprise et aussi explique comment une certaine activité est affectée à un acteur spécifique de l'organisation.

L'approche téléologique est composée de cinq éléments de base: les objectifs, les rôles, les acteurs, les processus et les ressources. Les objectifs sont au cœur du processus de modélisation, car ils fournissent des explications claires sur la configuration actuelle et future de l'entreprise. Le concept de l'acteur comprend les personnes, le système logiciel, etc. Les processus sont les mécanismes qui permettent les changements d'états dans le système organisationnel. Enfin, les ressources sont les moyens informationnels ou physiques qui sont produites à la suite des processus d'affaires.

L'approche téléologique comprend trois vues complémentaires pour représenter un modèle organisationnel: la vue téléologique, la vue sociale, et la vue de processus. Chaque vue est décrite ci-dessous:

La vue téléologique: les objectifs des parties prenantes sont représentés dans cette vue. Le concept d'objectif est le point focal de cette vue. Elle implique des intentions et représente aussi des solutions aux problèmes de l'entreprise. Les contraintes, qui sont

les objectifs opérationnels, sont formulées en termes de propriétés et d'actions. Les objectifs sont décomposables en des sous-objectifs. Le méta modèle de cette vue est illustré dans la figure 2.7.

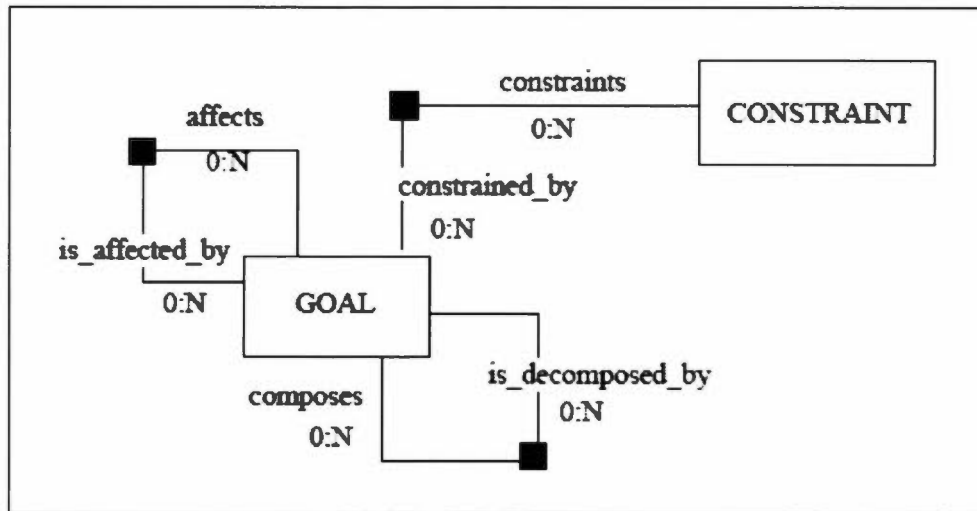


Figure 2.7 Méta modèle de la vue Téléologique de la modélisation de l'entreprise

Vue sociale: les acteurs organisationnels et leurs interactions sont détaillés dans cette vue. L'acteur est un facteur clé de ce modèle puisque l'acteur est considéré comme l'entité responsable de l'exécution des activités de l'organisation. Les rôles sont un ensemble de processus qui sont affectés à un agent spécifique. Cette assignation dépend de leurs objectifs et leurs capacités. Un acteur peut jouer plusieurs rôles en même temps. Le méta modèle de cette vue est illustré dans la figure 2.8.

Vue du processus : Elle fournit une vue d'ensemble du processus en cours dans l'entreprise. Elle comprend les vues fonctionnelle et comportementale. Elle illustre les processus et les ressources (données, produits, etc.), qui sont pertinents à l'exécution des éléments du processus. La vue des processus permet la représentation des déclencheurs (*triggers*) qui correspondent à des changements dans l'entreprise. Les événements représentent les dépendances dynamiques entre les processus et peuvent

être générés par des procédés ou par des conditions temporelles. La figure 2.9 illustre le Meta modèle de la vue du processus

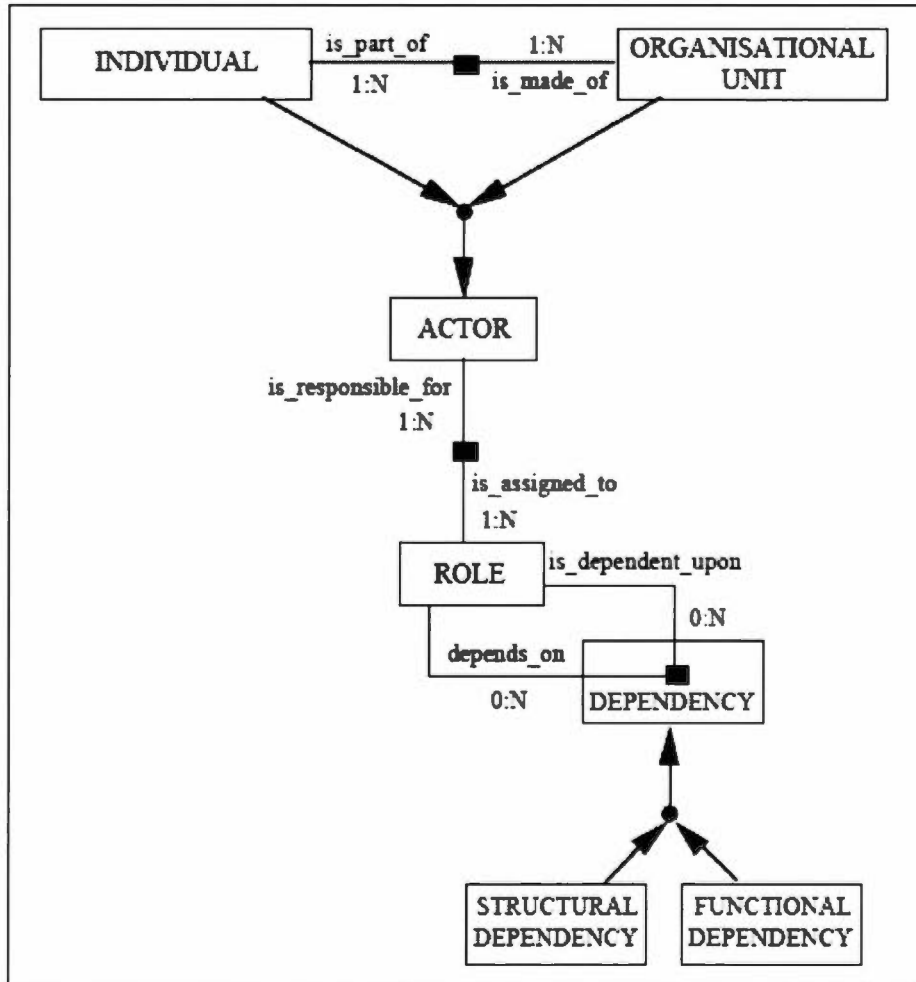


Figure 2.8 Méta modèle de la vue « Sociale » de la modélisation de l'entreprise

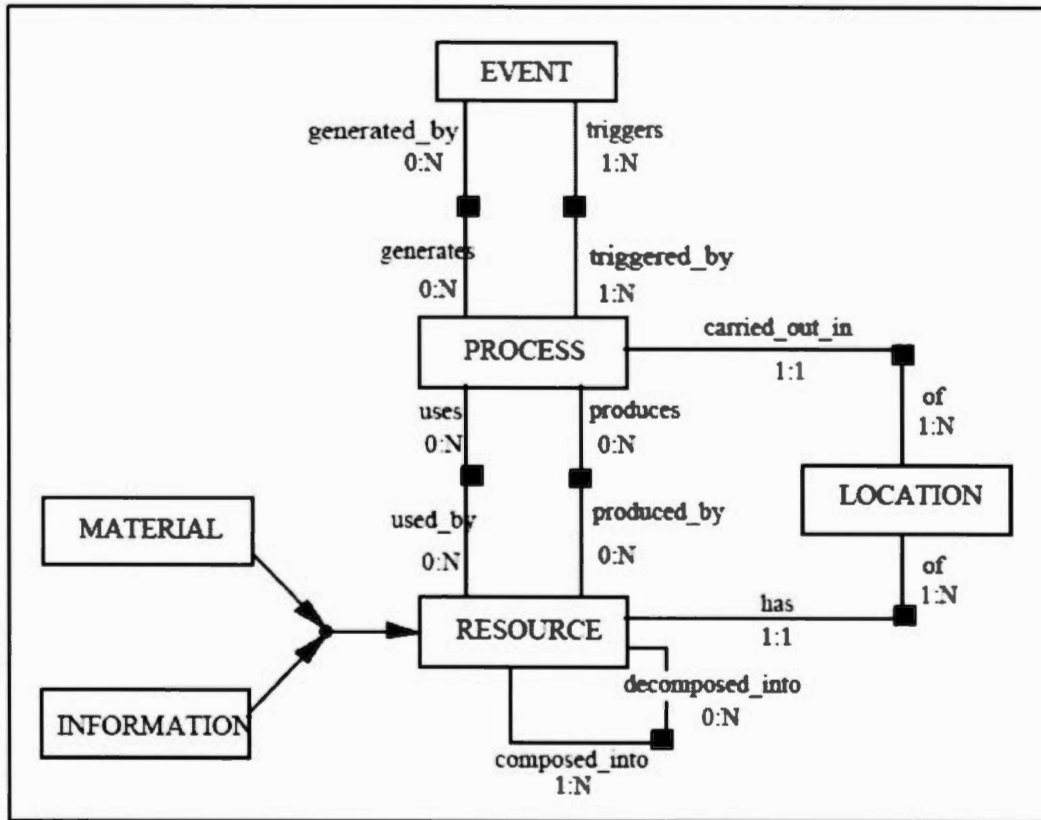


Figure 2.9 Le méta modèle de la vue « Processus » de la modélisation de l'entreprise

Les avantages de cette méthodologie sont les suivantes:

- Les représentations du modèle téléologique peuvent être très utiles pour construire une première série d'exigences soit pour le modèle d'affaires ou pour le système de logiciel.
- La démarche proposée donne une notation graphique bien définie pour chaque facette d'affaire.
- La technique nous permet de définir des dépendances fonctionnelles et structurelles. Cette caractéristique est utile pour déterminer le moment où les

tâches d'un certain acteur influent sur l'exécution des tâches des autres acteurs organisationnels.

De l'autre côté, les principaux enjeux de cette proposition sont les suivants:

- Deux types d'analyses doivent être effectués. Le premier est la détermination des objectifs de haut niveau de l'entreprise et leur raffinement jusqu'à ce que les activités opérationnelles soient analysées (analyse normative). La deuxième analyse porte sur les détails des opérations des processus d'affaires actuels (analyse descriptive). Cependant, aucun détail n'est donné quant à la conciliation des deux spécifications lorsqu'il n'y a pas correspondance précise entre elles.
- Il n'y a qu'une brève explication sur la décomposition des objectifs. Aucun détail n'est donné sur les objectifs contradictoires ou redondants. En outre, il n'y a aucune description formelle des objectifs suscités, ce qui rend difficile la validation du modèle de but.
- Seule une brève explication de la traçabilité entre les différentes vues de la méthodologie est donnée.
- Il n'y a pas une association explicite entre le modèle de but et le modèle de processus. Cela rend difficile l'identification des processus qui apportent un soutien à un objectif d'entreprise spécifique.
- L'explication complète du modèle d'affaires implique l'analyse des trois modèles. Par conséquent, il n'est pas possible d'avoir une représentation globale unique du processus d'affaires actuel, qui peut être très utile pour la réingénierie des processus.

2.4.3.3 L'approche GBRAM pour l'analyse des exigences

L'approche de modélisation du social GBRAM (Goal-Based Requirements Analysis Method) est proposée par Antón (1996). Dans GBRAM, les objectifs sont utilisés comme les mécanismes appropriés pour identifier et justifier les exigences d'un système de logiciel selon le modèle d'affaire. Les objectifs sont obtenus à partir de la description des processus fonctionnels actuels et également d'après les descriptions des rôles et responsabilités des parties prenantes. Il s'agit ici d'une démarche bottom-up qui permet de recueillir les exigences.

GBRAM est composé de deux processus principaux: L'analyse des objectifs, et le raffinement des objectifs. La figure 2.10 montre les activités dans lesquelles un analyste est intimement impliqué dans l'application du GBRAM. La sortie de GBRAM est toujours un document sur les exigences de la solution (logiciels).

Analyse des objectifs : L'analyse des objectifs est le processus de découverte de la documentation associée à l'entreprise afin d'identifier et de clarifier les objectifs de l'entreprise. Cette approche met l'accent sur la description des objectifs abstraits et le comportement spécifique que les intervenants attendent du système. Les principales étapes de l'analyse des objectifs sont les suivants: a) l'exploration des activités pour chercher des informations pertinentes; b) l'identification des objectifs et des acteurs responsables; et c) l'organisation des activités associées à des objectifs selon les dépendances entre objectifs.

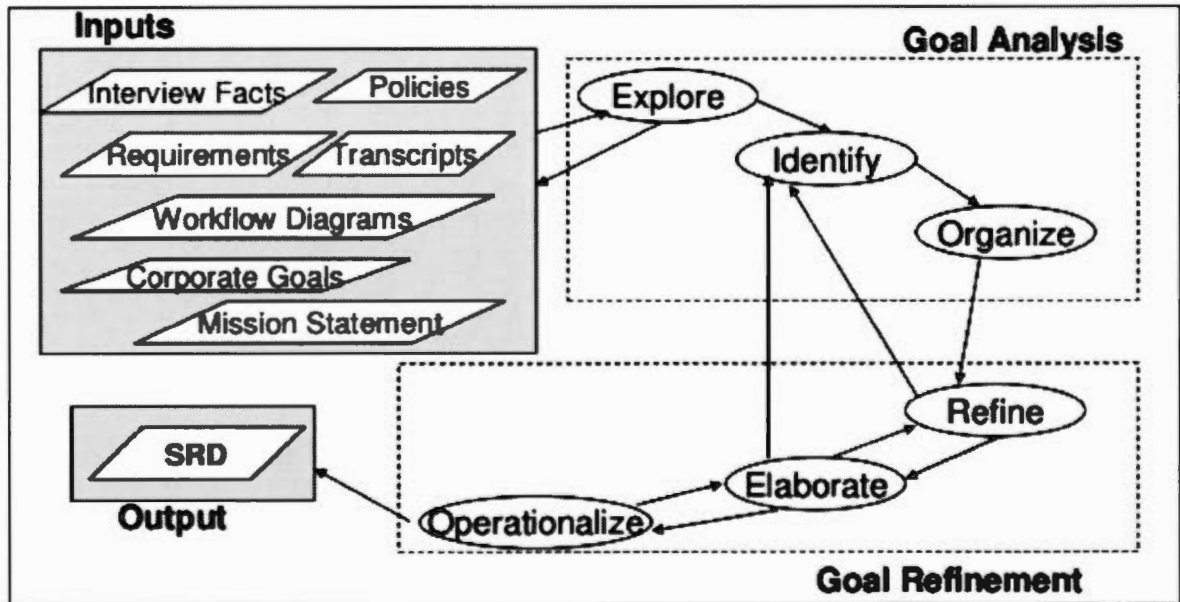


Figure 2.10 Vue du modèle des activités de GBRAM

Raffinement des objectifs : Le but de cette phase est l'identification des objectifs de haut niveau de l'entreprise jusqu'à ce que le niveau des objectifs opérationnels soit atteint. Pour ce faire, deux types de mesures doivent être effectués: a) **Élaboration:** cette étape permet l'identification des obstacles à la réalisation des objectifs et permet également l'identification des restrictions afin de faire une description opérationnelle des objectifs suscités; b) **Raffinement:** implique la détermination des pré et post conditions nécessaires à l'obtention des objectifs opérationnels.

Dans la méthodologie GBRAM, les objectifs sont classés en deux types: les objectifs de maintenance (maintenance goals) et les objectifs de réussite (achievement goals). Les objectifs de maintenance reflètent les objectifs les plus abstraits de l'entreprise. Les objectifs de réussite décrivent les mesures qui prescrivent le comportement actuel des processus d'affaires.

Les avantages de cette méthode sont les suivantes:

- L'une des principales contributions de GBRAM est la définition d'une démarche claire pour susciter les objectifs abstraits afin de définir un ensemble d'objectifs opérationnels qui mèneront les exigences relatives au système d'information. Cette approche permet de définir les raisons de l'existence d'activités d'affaires.
- GBRAM offre des mécanismes appropriés pour détecter les objectifs redondants et consolider les objectifs synonymes. Les restrictions de buts sont utilisées comme mécanismes « de finition ». Cela permet à l'analyste de déterminer quand le processus de raffinement des objectifs doit cesser.
- Cette méthodologie considère la définition de pré et post-conditions nécessaires à l'accomplissement des objectifs.

En termes de limites :

- GBRAM n'offre que la possibilité de description en langage naturelle, des objectifs suscités. Il n'y a donc pas de formalisme ou de notation graphique pour représenter les objectifs suscités. Il en est ainsi pour les objectifs proposés.
- Le processus de modélisation de GBRAM se termine lorsque les objectifs opérationnels ont été suscités. Par conséquent, cette technique n'offre pas de mécanismes pour définir un modèle d'entreprise qui associe explicitement le modèle de processus d'affaires avec le modèle de but suscité.
- La méthodologie GBRAM n'aborde pas explicitement les questions de relation de dépendance entre les acteurs du système. Elle n'offre pas la possibilité de capturer les raisons qui existent derrière la tâche affectée aux acteurs.

2.4.3.4 KAOS: Ingénierie des exigences basée sur les buts

KAOS (Knowledge Acquisition in Automated Specification) est une méthode d'acquisition des exigences proposée par Dardenne *et al* (1993). KAOS couvre l'identification des besoins de l'entreprise, des exigences, des agents responsables et si nécessaire, des comportements dont ils ont besoin d'adopter afin de satisfaire les exigences et les restrictions du système. La méthodologie repose sur la construction d'un modèle de gestion des exigences.

La particularité de KAOS est qu'il est en mesure de mettre en œuvre un raisonnement basé sur les buts. Un but définit un objectif auquel le système doit répondre, généralement grâce à la coopération de plusieurs agents tels que les dispositifs ou les humains. KAOS fait une distinction entre les objectifs et les propriétés du domaine qui sont des énoncés descriptifs de l'environnement tels que les normes, les politiques organisationnelles, etc.

La méthodologie se concentre d'abord sur les objectifs d'affaire, puis spécifie les exigences qui amènent à ces objectifs d'affaires. Elle offre à l'analyste des possibilités d'aligner les exigences sur les buts et les objectifs d'affaires. Cet alignement augmente les chances pour le nouveau système d'apporter une valeur ajoutée à l'entreprise.

Dans KAOS, les objectifs globaux sont des objectifs non-opérationnels à atteindre par une entreprise. Cela signifie que l'objectif ne peut être formulé en termes d'objets et actions disponibles pour un agent spécifique dans le système. Les objectifs globaux sont réduits à l'aide des raffinements AND / OR jusqu'à ce qu'ils puissent être assignés comme des responsabilités à des agents. Ceux-ci deviennent des exigences pour le système à construire, ou des hypothèses sur les agents existants dans l'environnement. Un raffinement AND implique que la conjonction des sous-buts est une condition suffisante pour réaliser le but parent. Un raffinement OR associe un but à des sous-buts alternatifs pour lesquels l'accomplissement du but de plus haut niveau exige

l'accomplissement d'au moins un de ses sous-buts. KAOS suit une stratégie « top-down » pour le processus d'acquisition des exigences à partir des objectifs abstraits et les affine itérativement jusqu'à ce que le niveau opérationnel soit atteint.

Le modèle des exigences de KAOS se compose de cinq sous-modèles liés par des règles de cohérence inter-modèles.

Le modèle central est le **modèle de buts** qui décrit les buts du système et de son environnement. Ce modèle est organisé dans une hiérarchie obtenue grâce au raffinement de buts de plus haut niveau (les buts stratégiques) vers des buts de bas niveau (les buts opérationnels ou exigences).

Le modèle objet définit les objets (agents, entité, etc...) du domaine. Il permet de décrire le vocabulaire du domaine. Il est représenté par un diagramme de classes UML.

Le modèle des responsabilités prend en charge l'assignation des objectifs aux agents de façon réalisable.

Le modèle des opérations résume tous les comportements que les agents doivent avoir afin de remplir leurs exigences. Ces comportements sont exprimés en termes d'opérations effectuées par les agents sur les objets décrits dans le modèle d'objet. Ils peuvent créer des objets, provoquer des transitions d'état d'objets ou déclencher d'autres opérations par le biais d'émission et de réception d'événements

Le modèle des comportements résume tous les comportements que les agents doivent accomplir pour satisfaire les exigences. Ces comportements sont exprimés sous la forme d'opérations exécutées par les agents responsables.

KAOS offre un ensemble de patrons de raffinement qui décomposent les buts. Ces patrons de raffinement sont regroupés selon le comportement de différents types de buts, et servent à guider l'analyste dans le processus d'acquisition. Les patrons sont

classés en cinq modèles bien établis: *Achieve*, *Maintain*, *Cease*, *Avoid* et *Optimize*. Ces modèles ont un impact direct sur les comportements possibles du système. Les aspects concernant la génération de comportements sont analysés avec les buts *Achieve* et *Cease*. Les buts *Maintain* et ceux *Avoid* sont utilisés pour restreindre les comportements, et enfin, les buts *Optimize* concernent les comparaisons de comportement.

Dans cette proposition, les actions sont des relations mathématiques à travers l'ensemble des objets du système. L'application des actions définit les transitions d'état. En outre, la spécification des actions doit comprendre les préconditions, déclencheurs et postconditions. La paire pré-condition-post-conditions capture les états de transitions produits par les actions de l'application.

Les restrictions dans KAOS sont les objectifs opérationnels à atteindre par l'entreprise. Les restrictions doivent être formulées en termes d'objets et d'actions disponibles pour un agent spécifique dans le système. Les contraintes sont assurées en limitant les actions et les objets existants (par le biais du renforcement des conditions, des invariants, etc.) ou par l'introduction de nouvelles actions et objets.

Dans KAOS, les agents, les événements, les entités et les relations sont joints dans la catégorie des objets concernés.

La Figure 2.11 résume l'ensemble des concepts et notations qu'offre la méthodologie KAOS. Elle donne un aperçu des quatre premiers sous-modèles de KAOS ainsi que des principaux éléments qui les constituent

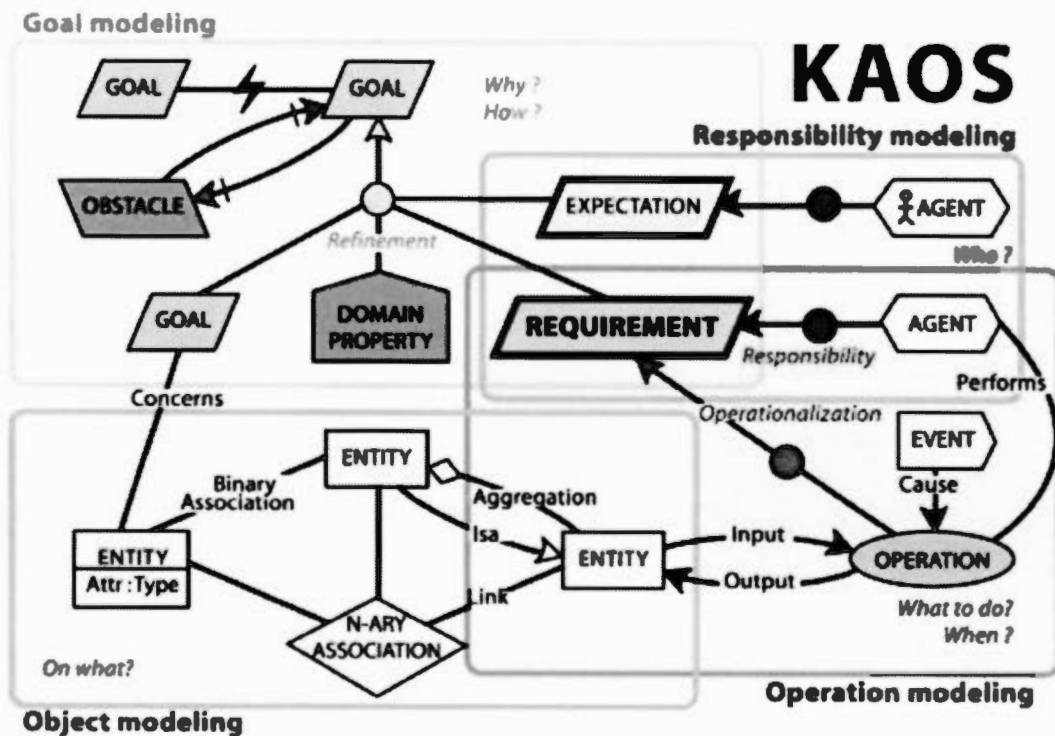


Figure 2.11 Meta modèle de KAOS (Respect-IT, 2007).

KAOS fournit une méthode qui permet d'inférer des objectifs de haut niveau sur les restrictions opérationnelles. Les principales étapes de l'acquisition des besoins sont les suivantes: identification des objectifs à partir des documents initiaux; formulation des objectifs et identification des objets; retrouver de nouveaux objectifs grâce à des questions POURQUOI; retrouver de nouveaux objectifs grâce à des questions COMMENT; inférer les interfaces d'agent; identification des opérations; opérationnaliser les objectifs; anticiper les obstacles; gérer les conflits.

KAOS met l'accent sur du formel et offre un ensemble de primitives de modélisation qui permet de comprendre les objectifs et de spécifier formellement les exigences d'un système d'information. Ses concepts sont présentés dans un graphe où les nœuds

représentent une abstraction (but, action, restriction, et objet) et où les arcs capturent les liens sémantiques entre ces abstractions.

Les avantages de KAOS sont les suivantes:

- Un traitement exhaustif des objectifs d'affaires qui en offre une classification selon différents niveaux d'abstraction. Cette orientation permet aux analystes de construire des modèles de buts très complets et précis.
- Une méthode spécifique pour chaque étape de la modélisation est présentée. Ceci constitue l'un des principaux avantages de cette technique puisque les directives précises sont fournies pour construire les diagrammes de modélisation.
- Tous les éléments du méta-modèle de KAOS ont une formalisation correspondant dans la logique temporelle. Cette approche formelle permet à l'analyste d'effectuer un raisonnement automatique sur la spécification de but pour détecter les incohérences et les redondances.

De l'autre côté, les principales limites de cette méthode sont les suivantes:

- Il n'y a aucune notation graphique précise pour supporter la méthode d'analyse d'objectif de KAOS
- Il n'y a pas une vue unique des actions impliquées dans les processus d'affaires.
- La méthode KAOS arrête à la phase d'exigences et ne traite pas les autres niveaux de développement de logiciels; ce qui oblige les analystes à utiliser une autre méthode pour le développement de leurs systèmes.
- Il n'est pas possible d'analyser la relation complexe qui existe entre les acteurs organisationnels.

- KAOS ne considère pas la définition d'un modèle d'affaires explicite qui reflète le contexte organisationnel. Il ne considère que l'acquisition des besoins jusqu'à ce que le niveau opérationnel soit atteint. Dans ce contexte, la technique ne prévoit pas de mécanisme bien définis pour la réingénierie de l'entreprise.
- Une autre limitation de KAOS est l'absence de capacités d'évaluation d'inférence. La réalisation de sous objectifs ne signifie pas la réalisation de leurs objectifs parents dans tous les cas.

Les applications de KAOS

La méthodologie KAOS est utilisée pour découvrir les buts et leurs classifications dans plusieurs domaines d'application notamment, le traitement des langages naturels (Casagrande, 2014; Lezcano *et al.*, 2015); dans l'énergie (Yip *et al.*, 2015) et dans la modélisation des collisions de voiture (Cailliau *et al.*, 2013)

Yip *et al* (2015) ont utilisé KAOS pour développer une arborescence de buts pour structurer et expliquer les types de dépendances associées aux besoins énergétiques des États-Unis. À partir des objectifs identifiés, ils ont formulé des déclarations d'exigence primitives pour le secteur de l'énergie des États-Unis, en tirant parti des normes de gestion de qualité provenant des exigences ISO 9001. Arenas *et al* (2015) ont montré comment les concepts d'ingénierie des exigences et des techniques d'analyse de KAOS (objectifs de raffinement, raffinement obstacle) axés sur les buts peuvent fortement et systématiquement soutenir le processus de production des *planifications de continuité d'affaire* de haute qualité.

2.4.3.5 Actor-Network Theory (ANT)

L'idée centrale de la théorie ANT est d'étudier et de théoriser sur la façon dont les réseaux se déroulent, de retracer les associations qui existent, comment elles se déplacent, comment les acteurs sont inscrits dans un réseau, comment certaines parties d'un réseau forment un réseau entier et comment les réseaux atteignent une stabilité temporaire (ou à l'inverse, pourquoi certaines nouvelles connexions peuvent former des réseaux instables) (Doolin et Lowe 2002, McLean et Hassard 2004)

La théorie ANT a été conçue par Latour, Callon et Law entre 1978 et 1982 (Law, 2009). Elle a d'abord été développée pour la sociologie, l'anthropologie et les études scientifiques et technologiques, mais a été progressivement adoptée par d'autres domaines, y compris les systèmes d'information. Malgré son influence grandissante dans la compréhension des humains et leurs interactions avec des objets inanimés, ANT reste encore profondément contestée.

L'une des caractéristiques principales de ANT est l'accent mis sur les entités inanimées (idées, objets, etc...) et leurs effets sur les processus sociaux. Ainsi, l'abstraction d'acteur définit inclusivement tout acteur indépendamment de son statut d'humain ou non humain. Un acteur peut cependant seulement agir en combinaison avec d'autres acteurs et dans des constellations qui donnent à l'acteur la possibilité d'agir - c'est parce que la réalité est supposée être activement jouée par divers acteurs à un moment et à un lieu particuliers (McLean et Hassard 2004).

Au-delà du concept central de réseau, ANT offre des caractéristiques intéressantes pour l'étude de la CS en ce sens qu'elle suppose que si un acteur, indépendamment de sa position, est supprimé ou ajouté au réseau, comme c'est le cas lorsqu'une nouvelle technologie ou un employé est introduit dans une équipe, le fonctionnement de l'ensemble du réseau sera affecté (Doolin et Lowe, 2002). Cependant, les réseaux

évoluent constamment vue que la réalité sociale et organisationnelle est à la fois complexe et fluide (Law, 2009).

La théorie ANT est bâtie sur les concepts fondamentaux suivants :

- **Réseau**

ANT utilise la notion de « réseau » d'une manière qui est fondamentalement différente de son utilisation standard en sociologie, puisqu'il ne s'agit pas principalement de cartographier les interactions entre les individus. Plutôt, il s'agit de cartographier comment les acteurs définissent et distribuent des rôles, et mobilisent ou inventent les autres pour jouer ces rôles.

- **Acteur**

Les acteurs sont les processus, les idées, les objets et les humains qui composent le réseau. Les acteurs doivent être considérés à part entière et ne peuvent jamais être que des intermédiaires et tout simplement transmettre l'effet d'une entité à une autre sans la transformer. Ce concept de diversité d'acteurs rejoint celui de iStar où les buts et « Soft-goals » peuvent être attribués non seulement aux acteurs humains, mais aussi aux acteurs non humains (systèmes, machines, etc.) conçus par les humains. Par contre dans iStar, l'on fait une nette distinction entre les deux types d'acteurs.

- **Irréductibilité**

Tout phénomène, humain et non humain, se considère comme une entité ontologiquement valable. Rien ne doit être rejeté comme une simple façade pour une essence plus profonde et rien ne peut être réduit à autre chose. Cette idée met fin à la dichotomie moderne de l'esprit humain subjectif et à la réalité extérieure objective.

- **Traduction**

Tout acteur peut être allié à tout autre acteur. L'instanciation d'une relation entre deux acteurs commence par la compréhension entre eux. La traduction concerne un acteur qui se rend compréhensible en définissant son propre sens en fonction du cadre de référence de l'autre acteur.

La « traduction » est un concept fondamental de la théorie ANT. Elle désigne la dynamique par laquelle un acteur recrute les autres dans son projet. C'est un processus continu et « jamais un accomplissement complet » (Law, 2009).

Par la traduction, tous les acteurs conviennent que le réseau vaut la peine d'être construit et défendu. Il se préoccupe de la façon dont les acteurs et les organisations se mobilisent, se juxtaposent et tiennent ensemble les morceaux dont ils sont composés. Il définit comment les acteurs sont parfois capables d'empêcher ces morceaux de suivre leurs propres inclinations et de se séparer de cette composition. Lorsque le processus de traduction est bien exécuté, les acteurs se métamorphosent d'un ensemble hétérogène de morceaux, vers des acteurs « ponctués ». Ce concept de « ponctuation » rejoint en plusieurs points celui du modèle mental partagé présenté dans ce chapitre. Il est question ici des mécanismes mis en place pour l'atteinte progressive d'une compréhension partagée au sein des membres du réseau (équipe) grâce à l'échange d'informations, en considérant que les membres de l'équipe modifient continuellement leurs comportements en analysant les conséquences possibles de leurs actions. Le concept de ponctuation est le même que celui de « boîte noire » que nous expliquons un peu plus loin.

- **Alliances**

Les alliances sont ce qui apparaît entre les acteurs lorsque la traduction est réussie. Tout acteur donné est seulement aussi fort que ses alliances avec d'autres acteurs. Chaque acteur œuvre pour renforcer son influence en maintenant ses alliances.

- **Coût et risque d'alliances**

La traduction, et l'alliance qui en résulte sont coûteuses. Comme toute négociation stratégique, la traduction est également risquée. Chaque acteur a son propre ensemble de termes et conditions pour une alliance. Les acteurs peuvent résister à l'alliance ou entrer dans une alliance de mauvaise foi (Ramiller et Wagner, 2009). Il y a toujours un compromis et donc le besoin de négociation. Il y a une similitude entre ce concept de risque et celui de dépendance stratégique de la méthodologie d'ingénierie des exigences iStar (Yu, 2011) que nous présentons dans ce chapitre.

- **Durabilité**

La réalité d'un acteur n'est pas cachée mais est clairement exposée dans ses alliances avec d'autres acteurs. Si une alliance ne change que légèrement (et les alliances changent constamment), l'acteur peut être considéré comme différent.

- **Boîtes noires**

Une boîte noire est un réseau bien établi d'acteurs alliés qui est si fort que l'assemblage est compté comme un seul acteur.

- **Action à distance**

L'action à distance montre comment un acteur peut agir sur un autre qui est loin de lui-même (physiquement ou conceptuellement). Pour qu'un acteur puisse agir à distance, il doit rendre les autres acteurs mobiles.

ANT et la modélisation dans le domaine des systèmes d'information

On retrouve dans la littérature des systèmes d'information quelques tentatives de représentations graphiques en application des concepts de ANT. Cependant, tous ne couvrent que partiellement les concepts de ANT. Parmi celles-ci, on peut citer :

- L'Analyse dynamique des réseaux d'acteurs (Bots *et al.*, 1999): Elle est proposée pour analyser les perceptions des acteurs dans un réseau corporatif autour de la notion de procédures organisationnelles. Cette méthodologie permet aux analystes de « rechercher les éléments de perception qui sont au cœur d'un problème » (Bots *et al.*, 1999, p.2) et d'analyser la logique d'un acteur donné ainsi que celle du réseau en complétude ou en antagonisme à la prospection des acteurs.

Dans l'analyse dynamique des réseaux d'acteurs, le concept d'acteur se réfère à des entités humaines, et le concept de facteurs se réfère à des entités non humaines. Cette distinction entre les 2 types d'acteurs fait entorse au principe d'irréductibilité de ANT.

- La deuxième étude a utilisé la théorie ANT pour modéliser les communautés en ligne (Potts, 2008). Cette étude fait une distinction nette entre les acteurs humains qui au cœur du modèle et les non-humains qui sont secondaires.

2.4.3.6 Analyse des buts et agents intentionnels avec iStar (intentionnalité distribuée)

La méthodologie iStar (Yu, 1994) est utilisée pour modéliser et analyser les relations entre les acteurs stratégiques dans un réseau social, tels que les organismes humains et d'autres formes de structures sociales. Les acteurs sont des unités semi-autonomes dont les comportements ne sont pas entièrement contrôlables ou prévisibles, mais sont régis par les relations sociales. iStar considère que les acteurs dépendent les uns des autres pour les objectifs à atteindre, les tâches à exécuter, et les ressources à fournir. Par sa dépendance sur quelqu'un d'autre, un acteur peut atteindre des buts qui seraient autrement irréalisables.

Le cadre d'analyse de modélisation iStar était une tentative d'introduire certains aspects de la modélisation sociale et du raisonnement dans les méthodes d'ingénierie des systèmes d'information, en particulier au niveau des exigences. Contrairement aux

méthodes d'analyse traditionnels qui s'efforcent d'éviter le côté humain des systèmes, iStar reconnaît la primauté des acteurs sociaux. Les acteurs sont considérés comme étant intentionnels, c'est à dire, ils ont des objectifs, des croyances, des capacités et des engagements. L'analyse s'attarde sur la façon dont les objectifs des différents acteurs sont atteints en considération de certaines configurations des relations entre les acteurs humains et du système, et quels types de reconfigurations de ces relations peuvent aider les acteurs à promouvoir leurs intérêts stratégiques.

Le cadre d'analyse de modélisation iStar se compose de deux types de modèles : le modèle de dépendance stratégique (SD) et le modèle de raisonnement stratégique (SR).

- **Le modèle de dépendance stratégique de iStar**

Le modèle de dépendance stratégique (SD) se présente comme un ensemble de nœuds et de liens où chaque nœud représente un acteur et chaque lien entre deux acteurs indique qu'un acteur dépend de l'autre pour quelque chose afin que le premier puisse atteindre un certain objectif. Il aide à identifier les intervenants, analyser les opportunités et les vulnérabilités, et reconnaître divers mécanismes pour atténuer la vulnérabilité. Le modèle SD est utilisé pour exprimer le réseau des relations stratégiques entre les acteurs intentionnels. Il représente les dépendances stratégiques entre les acteurs, mais ne représente pas le rationnel interne derrière ces dépendances.

Le modèle SD vise à capter la structure intentionnelle des processus. Il est une caractérisation de niveau supérieur d'un processus, car il capture ce qui compte pour les acteurs, tout en laissant le non-essentiel. Le modèle de dépendance stratégique (SD) fournit une description intentionnelle d'un processus en termes de réseau de relations de dépendance entre les acteurs. Il fournit une image des agents en modélisant explicitement leurs relations externes intentionnelles les uns avec les autres. Les objectifs et les désirs internes des acteurs ne sont pas explicitement modélisés.

- **Le modèle de raisonnement stratégique de iStar**

Le modèle de raisonnement stratégique (SR) fournit une description intentionnelle des processus en termes d'éléments de processus et de leur justification. Alors que le modèle stratégique de dépendance (SD) maintient un niveau d'abstraction en modélisant seulement les relations entre les acteurs externes, le modèle SR renonce à cette abstraction afin de permettre une compréhension plus profonde du raisonnement des acteurs stratégiques sur les processus qui doivent être explicitement exprimés.

Le modèle SR décrit les relations intentionnelles qui sont « internes » aux acteurs, tels que des relations *moyens/fins* qui relient les éléments de processus, fournit une représentation explicite de « pourquoi » et « comment », et des alternatives.

L'utilisation de connaissances représentées par et dans ces modèles, facilite la génération et l'exploration systématique des solutions alternatives.

Le modèle SR est un graphe avec plusieurs types de nœuds et de liens qui travaillent ensemble pour fournir une structure de représentation qui exprime le raisonnement derrière les processus.

- **Les concepts clés de iStar**

Autonomie des acteurs

Dans iStar, l'acteur est la charpente de la modélisation. C'est une abstraction qui est utilisée pour se référer à une entité active qui est capable d'une action indépendante. Les acteurs peuvent être des humains, matériels et logiciels, ou des combinaisons de ceux-ci. Les acteurs sont considérés comme intrinsèquement autonome - leurs comportements ne sont ni entièrement contrôlables, ni entièrement connaissables. Cette notion d'autonomie est distincte de celui de l'intelligence artificielle, où elle se réfère à des capacités avancées à réaliser par la conception et par une mise en œuvre

technologique. Dans la modélisation sociale, l'autonomie de l'acteur est considérée comme une caractéristique des phénomènes sociaux dans le monde réel avec laquelle les concepteurs de système doivent composer.

L'intentionnalité

Pour expliquer et caractériser leur comportement, nous attribuons la motivation et l'intention à des acteurs. En modélisant ce que les acteurs ont l'intention de réaliser, nous obtenons une caractérisation de niveau supérieure sans préciser leur comportement exact.

Dans la modélisation iStar, nous nous concentrons sur les propriétés et les relations intentionnelles plutôt que le comportement réel. En ne décrivant pas directement le comportement, une description intentionnelle offre un moyen de caractériser les acteurs qui respecte le principe de l'autonomie. La modélisation intentionnelle fournit une expressivité plus riche qui est approprié de la conception sociale du monde. En attribuant l'intentionnalité, nous pouvons exprimer pourquoi un acteur engage certaines actions, ou préfère une solution plutôt qu'une autre. Une ontologie intentionnelle permet d'opérer des analyses sur les relations *means/ends* et celle de l'espace des solutions alternatives pour chaque acteur.

Sociabilité

Les phénomènes sociaux sont sans doute infiniment riches. Le traitement que l'on peut en faire à travers un modèle est nécessairement limité. Dans iStar, l'aspect social se limite sur le fait que « *le bien-être d'un acteur dépend d'autres acteurs* ». Les acteurs dépendent les uns des autres pour les objectifs à atteindre, les tâches à accomplir, et les ressources à fournir. En dépendant d'autres acteurs, l'acteur « *Depender* » profite des occasions qui sont disponibles via l'acteur « *Dependee* ». Ces dépendances sont donc stratégiques pour les acteurs concernés, car ils peuvent être bénéfiques ou nuisibles à

leur bien-être. Les acteurs choisiraient quelles dépendances avoir en fonction de leur jugement sur les gains et pertes potentielles de leur part.

2.4.4 Conclusion de l'état de l'art sur la modélisation du social

À la vue de ce qui précède, il ressort que dans l'histoire de l'évolution des techniques et méthodes de modélisation des exigences, une étape importante a été franchie : la modélisation du monde est devenue aussi importante que la modélisation de la machine. L'ingénierie des exigences doit donc désormais être définie par rapport à l'environnement de l'entité étudiée.

Parmi les modèles et langages de représentation des buts et des acteurs qui existent, la méthodologie iStar apporte des contributions significatives à la vision et à la démarche d'analyse du monde, qui cadre bien avec les objectifs de la présente recherche. iStar nous permet de mener le processus d'acquisition des exigences à un niveau d'abstraction qui est très proche des utilisateurs finaux. Elle inclut notamment des mécanismes pour la prise en compte des buts et besoins des agents, et la proposition de solutions alternatives. Dans ce sens, elle concorde avec l'esprit de la recherche-action que nous menons. Pour sa part, ANT, même si plusieurs de ses concepts d'analyse, notamment ceux d'acteur, de risques, et de traduction ont des similarités avec ceux de iStar, il reste qu'elle n'exploite pas les notions d'intentionnalité, de buts et les mécanismes de dépendance de façon aussi élaborée que le fait iStar. Aussi, il n'offre pas d'outil de représentation graphique de ces concepts de manière à susciter les échanges entre les analystes et les parties prenantes.

CHAPITRE III

PROPOSITION D'UNE MÉTHODOLOGIE POUR L'ÉTUDE DE LA CONSCIENCE DE LA SITUATION D'ÉQUIPE

3.1 Introduction

Ce chapitre traite essentiellement de la méthodologie de recherche adoptée dans ce travail. Il est composé de deux parties. La première partie présente les principales méthodes couramment utilisées dans les études sur la CS de l'individu ou la CS d'équipe. Une brève description de chacune des différentes catégories et les méthodes examinées au sein de chaque catégorie est faite dans la section 3.3. La deuxième partie du chapitre porte sur une proposition d'une approche pour l'étude de la CS qui prend en compte les avantages de l'ensemble des méthodes présentées dans la première partie.

3.2 Critères requis pour les méthodes d'analyse de la CS

Le choix des méthodes que nous présentons est basé sur des critères suivants :

- La disponibilité des techniques de collecte d'information qu'elles utilisent ainsi que leur applicabilité à une équipe;
- Le domaine d'application;
- La fiabilité;
- La validité;
- Les avantages et limites.

Les critères de mesure simultanée de la CS de l'individu à plusieurs endroits en même temps et ceux de la mesure en temps réel ont une moindre importance dans ce présent travail.

3.3 Techniques d'analyse de la conscience de la situation

Le tableau 3.1 présente un résumé des techniques de mesure de la CS. Une brève description de chacune des différentes catégories et les techniques examinées au sein de chaque catégorie sont présentées dans cette section. Pour réaliser ce travail, nous avons utilisé une présentation exhaustive et beaucoup plus élaborée des techniques de mesure de la CS proposée par Salmon *et al* (2009).

- **Les techniques « *SA requirements analysis* »**

Cette technique proposée par Endsley (1993) stipule qu'une analyse des besoins en CS représente la première étape de toute évaluation de la CS en environnement complexe. L'analyse est menée afin de déterminer en quoi consiste effectivement la CS d'un opérateur affecté à une tâche ou à un environnement en cours d'analyse. Dans cette technique, on décrit une procédure générique pour la conduite d'une analyse des besoins en CS qui implique l'utilisation d'un entretien non structuré administré par un expert en la matière; des analyses des tâches orientées-but et des questionnaires afin de déterminer les exigences pertinentes en CS. Le résultat de l'analyse des besoins de CS est ensuite utilisé lors de l'élaboration de la technique d'évaluation de la CS, afin de déterminer quels éléments composent la CS de l'opérateur et ainsi savoir quoi mesurer.

Cette technique est générique et peut être aussi bien utilisée dans les études de la CS de l'individu que celles de la CS d'équipe. Cependant, elle ne peut être effectuée que par un expert du domaine. Le fait que la présence de ce dernier soit requise peut occasionner des coûts d'opération importants surtout si l'étude devrait s'échelonner sur une longue période.

Tableau 3.1 Résumé des techniques de mesure de la conscience de la situation (Adapté de Salmon *et al.*, 2009)

Techniques d'analyse de la CS	Catégorie / Domaine	Type d'application	Expert du domaine requis?	Outils requis	Méthode validée?
CARS McGuinness et Foy (2002)	Self rating / Militaire (opération d'infanterie)	Individu	Non	Stylo et papier	Oui
MARS Matthews et Beal (2002)	Self rating / Militaire (opération d'infanterie)	Individu	Non	Stylo et papier	Oui
SABARS Matthews et Beal (2002)	Observer rating / Militaire (opération d'infanterie)	Individu	Oui	Stylo et papier	Oui
SAGAT Endsley (1995b)	Freeze on-line probe / Aviation (militaire)	Individu	Non	PC simulateur	Oui
SPAM Durso <i>et al.</i> , (1998)	Real-time probe / Contrôle du trafic aérien	Individu	Oui	Simulateur PC Téléphone	Oui
SA requirements analysis Endsley (1993)	SA requirements analysis / Générique	Individu / Équipe	Oui	Équipement Vidéo Audio	
Performance Measures	Performance Measures / Générique	Individu	Non	PC	Non
Eye tracker	Process Indices / Générique	Individu	Non	Dispositifs de suivi oculaire -PC	Non

- **Les techniques « *Freeze probe* »**

Les techniques *Freeze probe* font recours à l'administration en ligne (pendant l'exécution de la tâche) de requêtes liées à la CS au cours d'une simulation de tâche. Pendant la simulation, l'on procède au gel de la tâche en cours d'analyse. Typiquement, une tâche est choisie au hasard, gelée et un ensemble de requêtes CS concernant la situation actuelle est administré à l'acteur. Ce dernier est tenu de répondre à chaque requête sur la base de sa CS au point de gel. Au cours de ces gels, tous les dispositifs tels qu'écran et fenêtres opérateurs sont généralement masqués. Un ordinateur est utilisé pour sélectionner et gérer les requêtes et également enregistrer les réponses. Le principal avantage de cette famille de technique de gel est leur caractère direct. Cependant, les techniques *Freeze probe* sont critiquées pour leur intrusion dans l'exécution des tâches primaires, et aussi, elles ne peuvent s'appliquer que là où il y a une simulation de la tâche en cours d'analyse.

La technique *situation awareness global assessment* (SAGAT) est l'une des plus populaires des techniques *Freeze probes*. Elle a été développée pour évaluer la CS des pilotes dans les trois niveaux de CS proposées dans le modèle à trois niveaux de Endsley (Endsley, 1995a) que nous avons présenté dans le chapitre 1. SAGAT comprend un ensemble de requêtes conçu pour évaluer la CS du participant aux niveaux a) perception des éléments, b) compréhension de leur signification, et c) projection dans le futur.

SALSA (Hauss et Eyferth 2003) est une autre technique de *Freeze probe* qui a été développée spécifiquement pour une utilisation dans le contrôle de la circulation aérienne. Les requêtes de SALSA sont basées sur quinze aspects de vol d'avion, tels que le niveau de vol, la vitesse au sol, le cap, la tendance verticale, les conflits et le type de conflit.

Il y a deux principaux avantages associés aux approches *Freeze probe*. Tout d'abord, elles offrent une mesure directe de la CS de l'opérateur, ce qui élimine les divers problèmes liés à la collecte de données de CS post-épreuve et subjectives (voir description des techniques d'auto-évaluation dans le présent chapitre). Deuxièmement, l'approche SAGAT est plus utilisée et mieux validée parmi les mesures de la CS disponibles. Elle a constamment démontré fiabilité et validité dans un certain nombre de domaines. Selon Jones et Kaber (2004), de nombreuses études ont été réalisées pour évaluer la validité de SAGAT et la preuve suggère que la méthode est une mesure de CS valable. Endsley (2000) rapporte qu'il est prouvé que la technique SAGAT a un degré élevé de validité et de fiabilité pour mesurer la CS de l'individu.

- **Les techniques « *Real-time probe* »**

Une approche alternative à l'utilisation des techniques *Freeze probe* très intrusives, est l'utilisation des techniques *Real-time probe*. Les *Real-time probe* impliquent l'administration en ligne (pendant l'exécution de la tâche) de requêtes liées à la CS, mais sans le gel de la tâche en cours d'analyse. Le recours à un expert du domaine pour l'administration de la requête est un prérequis pour cette technique. En fait, les experts en la matière développent des requêtes soit avant ou pendant l'exécution des tâches et les administrent, sans gel, aux points pertinents lors de l'exécution de la tâche. Le contenu des réponses et le temps de réponse sont considérés comme une mesure de la CS du participant. Jones et Endsley (2000) rapportent que quand il n'y a pas de simulation du système en cours d'analyse et quand la tâche ne peut pas être gelée, les techniques *Real-time probe* peuvent représenter une option viable pour mesurer la CS. On fait valoir que le principal avantage associé à des techniques *Real-time probe* est la réduction des intrusions. Toutefois, les requêtes des techniques *Real-time probe* peuvent également servir à diriger l'attention du participant vers les éléments de l'environnement, résultant à introduire des données biaisées. La réduction alléguée de l'intrusion est également discutable, puisque le fait de recevoir des requêtes et y

répondre impose aussi pour le participant un degré d'intrusion sur sa tâche principale. Les *techniques* Real-time souffrent également d'un certain nombre d'autres défauts majeurs. En raison de la nature dynamique et imprévisible des tâches complexes, les requêtes CS seraient probablement générées en temps réel, et non avant l'exécution de la tâche. La génération de requêtes en temps réel constituerait un grand fardeau sur les experts du domaine, et peut s'avérer trop difficile pour les environnements complexes. En outre, lors de l'utilisation d'une approche *Real-time* dans des environnements complexes, nombreux experts du domaine seraient nécessaires en raison de la quantité de personnel impliqué. En conséquence, les coûts d'opération de cette technique peuvent particulièrement gripper dans les cas de la mesure de la CS d'équipe.

La méthode *Situation present assessment method* (SPAM) (Durso *et al.*, 1998) est une technique d'usage en temps réel développée pour une utilisation dans l'évaluation de la CS des contrôleurs de la circulation aérienne. La technique implique l'utilisation de requêtes en ligne en temps réel pour évaluer la CS des opérateurs de tours de contrôle. Par téléphone, l'analyste explore la CS de l'opérateur en utilisant des requêtes CS associées à la tâche accomplie en se basant sur les informations pertinentes recueillies dans l'environnement (par exemple, lequel des deux avions A ou B a la plus haute altitude?). Le temps de réponse pour la requête (pour les réponses qui sont correctes) est considéré comme un indicateur de la CS de l'opérateur. En outre, le temps nécessaire pour répondre au téléphone agit comme un indicateur de charge de travail.

- **Les techniques « *Self-rating* »**

Les techniques *Self-rating* sont utilisées pour obtenir une évaluation subjective de la CS des participants. Typiquement administré après les épreuves, le *Self-rating* amène les participants à fournir eux-mêmes une évaluation subjective de leur CS tel qu'ils le perçoivent via une échelle de notation de la CS.

La technique *The situation awareness rating technique* (SART) (Taylor, 1990) est un exemple de technique d'évaluation subjective qui a été initialement développée pour l'évaluation de la CS des pilotes. SART utilise les dix dimensions suivantes pour mesurer la CS des opérateurs : la familiarité de la situation, l'attention, la quantité de l'information, la qualité de l'information, l'instabilité de la situation, la complexité de la situation, la variabilité de la situation, l'excitation, et la capacité mentale. SART est administrée après l'épreuve et implique la notation de la part du participant sur chaque dimension sur une échelle de notation de sept points (1 = faible, 7 = haut) afin d'obtenir une mesure subjective de la CS. Les dix dimensions de SART peuvent aussi être condensées en un raccourci de 3 dimensions (3-D) SART, dans lequel les participants sont amenés à juger leur demande d'attention, leur fourniture d'attention et leur compréhension.

Autres variantes des techniques d'évaluation subjective :

- La technique *Situation awareness rating scales* (SARS) (Waag et Houck, 1994) développée pour le domaine de l'aviation militaire.
- La technique *Crew awareness rating scale* (CARS) (McGuinness et Foy, 2000) utilisée pour évaluer la CS et la charge de travail des commandants des centres de commandement et de contrôle (McGuinness et Ebbage, 2002). CARS comprend deux ensembles distincts de questions basés sur le modèle à trois niveaux de la CS (Endsley, 1995a). CARS est administrée après l'épreuve et implique que les participants procèdent à une notation de chaque catégorie sur une échelle de 1 (idéal) à 4 (pire) (McGuinness et Ebbage, 2002).
- La technique *Mission awareness rating scale* (MARS) (Matthews et Beal, 2002) est une version étendue de CARS, conçue spécifiquement pour une utilisation dans l'évaluation de la CS dans des exercices militaires. La technique MARS a été

développée pour une utilisation en milieu de terrain « monde réel », plutôt que dans les simulations d'exercices militaires.

L'utilisation des techniques d'auto-évaluation pour mesurer la CS dans des environnements complexes est attrayante pour un certain nombre de raisons :

- Ces techniques d'auto-évaluation sont non-intrusives pour l'exécution des tâches, elles sont administrées après les épreuves.
- Elles sont très rapides et faciles à utiliser et ne nécessitent que très peu de formation.
- En raison de leur nature simpliste, elles génèrent très peu de coût.
- Les auto-évaluations de la CS peuvent être obtenues auprès des différents membres de l'équipe (Endsley *et al.*, 2000) et constituent un potentiel dans l'évaluation de la CS de l'équipe.
- La majorité des techniques d'auto-évaluation ne nécessite qu'un stylo et du papier comme outils de travail, où les participants évaluent leur propre CS à la fin de la tâche en cours d'analyse. Il n'y a pas d'obligation de recours à des simulateurs coûteux, à des experts du domaine, ou à un long processus de formation, ce qui réduit considérablement le coût de la procédure.

La simplicité et le faible coût des techniques d'auto-évaluation expliquent en partie, la généralisation de leur utilisation. Cependant, malgré les avantages encourageants de ces techniques, leur utilisation dans la mesure de la CS dans des environnements complexes est discutable en raison d'un certain nombre de défauts qu'elles contiennent. Même si la majorité des techniques d'auto-évaluation est générique et peut être appliquée dans de nombreux environnements, une approche spécifique pour la mesure de la CS de l'équipe reste encore à trouver.

- **Les techniques « *Observer rating* »**

Les techniques *Observer rating* sont les plus couramment utilisées pour évaluer la CS lors des tâches effectuées « sur le terrain » surtout dans le domaine militaire. L'application de ces techniques requiert généralement un expert du domaine. Celui-ci observe les participants accomplir les tâches en cours d'analyse, puis fournit une évaluation de leur CS. Les évaluations de la CS sont basées sur les observations des comportements reliés à la CS, exposés par les participants au cours de l'exécution d'une tâche.

The situation awareness behavioural rating scale (SABARS) est une technique de *Observer rating* qui a été utilisée pour évaluer la CS du personnel d'infanterie dans les exercices de formation sur le terrain (Graham et Matthews, 2000; Matthews et Beal 2002). La technique consiste en l'observation des participants pendant l'exécution de la tâche par des experts du domaine. Les participants sont notés sur 28 comportements observables liés à leur CS. Une échelle de cinq points de classement (1 = très mauvais, 5 = très bon) et une catégorie supplémentaire « non applicable » sont utilisées. Les 28 éléments de mesure du comportement sont conçus spécifiquement pour évaluer la CS des chefs de peloton (Graham et Matthews, 2000).

Les principaux avantages associés à l'utilisation de ces techniques sont leur nature non-intrusive et leur capacité à être appliquées « sur le terrain ». Cependant, la mesure dans laquelle les observateurs peuvent évaluer précisément la CS du participant est discutable, et aussi plusieurs experts du domaine sont obligatoires dans certains cas. En analysant attentivement cette catégorie de technique, on s'aperçoit qu'elle comporte d'autres défauts cruciaux qui peuvent restreindre leur utilisation. Le principal inconvénient associé à l'*Observer rating* concerne la validité conceptuelle de la mesure. La mesure dans laquelle les observateurs peuvent évaluer avec précision la construction interne de la CS reste très incertain (Endsley, 1995b). Alors qu'il existe des comportements observables qui peuvent indiquer certaines choses concernant la

CS du participant, le niveau interne réel de cette CS ne peut être mesuré avec précision par la seule observation. Ces techniques peuvent également être sujettes à des biais, en ce sens qu'elles peuvent susciter des modifications de comportement chez les participants puisque ces derniers sont conscients de la présence d'un observateur. En présence d'un observateur, le participant peut par exemple adopter une attitude plus réglementaire, et par conséquent, les données obtenues sont sujettes à des biais. Enfin, dans certains domaines, l'application des techniques *Observer rating* nécessite l'accès répété à plusieurs experts du domaine sur une longue durée de temps. Cela peut se révéler problématique, voire impossible.

- **Les techniques « *Performance measures* »**

L'utilisation de *Performance measures* pour évaluer la CS consiste à mesurer les aspects pertinents de la performance des participants pendant la tâche en cours d'analyse. En fonction de la tâche, certains aspects de la performance sont enregistrés afin de déterminer une mesure indirecte de la CS. Par exemple dans une évaluation de la CS des conducteurs d'automobile, Gugerty (2011) mesure la détection des dangers, la détection de freinage, et l'évitement des collisions lors d'une tâche de conduite simulée. Les mesures de performance sont simples à obtenir et sont non-intrusives car elles sont générées lors de l'écoulement naturel de la tâche. Cependant, elles génèrent un certain nombre de problèmes concernant la relation entre la CS et la performance. Par exemple, un participant expert peut atteindre la performance même lorsque la mesure de sa CS n'est pas acceptable. De même, un participant novice peut posséder un niveau supérieur de CS mais toujours atteindre une performance inférieure en raison d'autres facteurs tels que l'inexpérience.

L'erreur que peut faire l'analyste est d'ignorer l'importance de l'expérience des participants et conclure qu'une bonne performance est nécessairement fonction d'un niveau élevé de CS et vice versa. La nature instable des relations entre la performance et la CS réduit l'utilisabilité de cette technique comme indicateur de la CS du

participant. Cependant, le fait que cette technique soit non couteuse et simple à utiliser, la rend toujours recommandable dans certaines situations si elle est utilisée conjointement avec d'autres techniques.

- **Les techniques « *Process indices* »**

Les techniques *Process indices* peuvent aussi être utilisées pour mesurer la CS. Elle consiste à enregistrer le processus que le participant utilise dans le but de développer une CS durant l'exécution de la tâche sous analyse. Un exemple de l'utilisation de *Process indices* pour évaluer de la CS est la mesure du mouvement des yeux du participant lors de l'exécution des tâches (Smolensky, 1993). Des dispositifs de mesures du mouvement des yeux peuvent être utilisés pour mesurer les fixations du participant, ce qui peut ensuite être utilisé pour déterminer comment le participant exploite son attention au cours de la tâche en cours d'analyse. Il y a un certain nombre d'inconvénients liés à l'utilisation d'un dispositif de suivi oculaire, y compris leur nature indirecte (comment sait-on que le participant voit effectivement ce qu'il regardait), le phénomène de '*look-but-failed-to-see*' (Brown 2001). Il y a aussi une incapacité à utiliser cette technique en dehors des paramètres de laboratoire. L'autre limite est l'opérationnalisation parallèle de protocole de transcription et d'analyse verbale (APV) qui génère une transcription écrite du comportement de l'opérateur pendant qu'il accomplit la tâche. Malgré le fait que le APV est un moyen qui permet d'atteindre les aspects cognitifs d'un comportement complexe, il instruit l'opérateur à « penser à haute voix » alors qu'il effectue la tâche en cours d'analyse. Enfin, cette technique comporte une procédure d'analyse des données qui est généralement très longue et qui requiert beaucoup de patience de la part de l'analyste.

3.4 Conclusion sur les approches de mesure de la conscience de la situation

Dans leur format actuel, les méthodes que nous venons de voir, et comme plusieurs autres que nous n'avons pas pu présenter dans cette recherche, sont insuffisantes pour la mesure de la CS dans des environnements complexes. L'examen de ces méthodes met en évidence un manque d'approches spécifiques de mesure de la CS d'équipe ou la CS partagée. Ces techniques sont plutôt conçues pour évaluer la CS de l'individu. En outre, l'état de l'art a révélé que ces différentes approches de mesure de la CS sont assaillies par des défauts distincts susceptibles d'entraver la qualité des données CS obtenues. La seule exception à cette conclusion pourrait être le « *SA requirements analysis* » qui requiert une procédure d'analyse des besoins en CS qui serait nécessaire avant toute autre forme d'analyse CS.

Les Techniques de gel « *freeze probe* » sont intrusives et ne peuvent pas être appliquées dans un contexte réel et en temps réel. Ces techniques d'enquête sont difficiles à appliquer et peuvent altérer la performance de la tâche du participant. Pour leur part, les techniques d'auto-évaluation souffrent d'une multitude de problèmes liés à la collecte des données de la CS après les épreuves (corrélation avec le rendement, l'incapacité des participants à évaluer les périodes de faible CS, etc.). La question de validité des techniques d'« *observer rating* » qui peuvent introduire des biais dans les données collectées pose également un problème de validité. Enfin, le rendement des « *Process indices* » est discutable. En outre, les techniques de validation de ces méthodes restent un problème. Mis à part SAGAT et SART, il existe des preuves de validation limitée associées aux techniques de mesure de la CS existantes.

3.5 Une approche de recherche-action

3.5.1 Choix et justification de l'approche proposée

L'étude de la CS dans les environnements complexes pose un défi considérable mais passionnant. La présente étude est un pas de plus dans l'investigation du concept de la CS d'équipe et de la CS partagée, mais cette investigation nécessite à son tour la mise à disposition des procédures de mesure fiables et valides. Comme nous le mentionnions dans la section 3.4 de ce chapitre, il apparaît que dans leur forme actuelle, les méthodes existantes pour la mesure de la CS sont insuffisantes à cette fin, et une nouvelle approche est nécessaire, mais ceci nécessite un travail plus approfondi qui sort du cadre de cette thèse. L'autre solution serait de combiner les techniques de mesure actuelles de la CS, les plus appropriées au contexte de notre étude pour former une « boîte à outils » de mesures de la CS.

Lors de la mesure de la CS, la fiabilité et la validité sont les critères à considérer. La fiabilité se réfère au degré dans laquelle la technique va générer les mêmes données dans les mêmes conditions, encore et encore et la validité se réfère à la mesure dans laquelle la technique sert effectivement à mesurer la CS, et non pas un autre processus psychologique. Dans ce sens, Endsley (1995b) rapporte que lors de l'examen des techniques de mesure de la CS, il est nécessaire d'établir que la technique 1) mesure la CS et ne mesure pas d'autres processus ou facteurs; 2) possède le niveau requis de sensibilité, c'est-à-dire la technique peut détecter avec précision des changements dans la CS, causés par de nouvelles technologies et des programmes; et 3) n'altère pas la CS au cours de la procédure de mesure.

En conséquence, nous avons opté pour une approche de recherche-action. Elle nous permet de faire usage de multiples techniques (par exemple des observations, des questionnaires d'auto-évaluation, des enquêtes contextuelles, des mesures subjectives), car elle nous semble répondre à nos objectifs de recherche. Ce choix est fait sur la base

de deux facteurs principaux. Tout d'abord, l'absence d'une technique unique qui peut faire face à la fois à la CS de l'individu et à la CS d'équipe géographiquement distribué et en temps réel. Deuxièmement, avec une approche multi techniques, il y a une forte chance de garantir que les données CS peuvent être efficacement recoupées entre les mesures afin d'en assurer la fiabilité et la précision.

L'idée d'utiliser plusieurs techniques pour obtenir des performances plus efficaces n'est pas nouvelle. La recherche-action prône une démarche holistique de résolution de problèmes, plutôt qu'une méthode unique de collecte et d'analyse de données. Elle permet d'utiliser plusieurs outils de recherche différents tout au long du projet de recherche (Dewey, 2007). Elle est souvent recommandée dans les études relevant du domaine de développement organisationnel, sociotechnique, et de la gestion (Brydon-Miller, 2003). La recherche-action, est une méthodologie de recherche scientifique et pratique permettant d'appréhender des situations sociotechniques rencontrées quotidiennement (Liu, 1997). La collaboration entre acteurs et chercheurs qui en résulte permet de traiter des problèmes posés et de faire progresser la connaissance scientifique. « Elle apparaît à la fois comme un mode d'actions innovant et comme une démarche de recherche fondamentale arrivée à maturité » (Crézé, 2006, p.12)

De nombreuses autres définitions existent pour la recherche-action. Selon Bradbury-Huang (2010), elle est une orientation vers la création de connaissances qui se pose dans un contexte de pratique dans des organisations, et exige que le chercheur travaille avec les praticiens. Elle implique de participer activement à une situation de changement, tout en effectuant simultanément des recherches. Contrairement à la recherche classique, son but n'est pas principalement ou uniquement de comprendre les arrangements existants, mais aussi d'amener vers le changement souhaité. Ce processus est considéré comme moyen de générer des connaissances et de responsabiliser les praticiens.

Denscombe (2014), pour sa part, estime que la recherche-action est un processus progressif de résolution de problèmes, mené par des chercheurs travaillant en équipe avec des acteurs. Elle permet d'améliorer la façon dont les acteurs praticiens traitent et résolvent les problèmes immédiats.

Enfin, selon (Wicks, 2009), en collaborant avec les chercheurs dans une recherche-action, les praticiens améliorent leurs stratégies, les pratiques et connaissances sur les environnements dans lesquels ils pratiquent.

Une caractéristique intéressante de la recherche-action est qu'elle utilise extensivement les actions de formation comme un des moyens privilégiés de partage de connaissance entre le chercheur et les acteurs bénéficiaires de la recherche (Dewey, 2007).

Généralement la conduite d'une recherche-action se déroule en cinq étapes cycliques. (Baskerville, et Wood-Harper, 1998; Mertler, 2008; Coghlan et Brannick, 2014). Il peut cependant avoir des variations et il est correct d'inclure ou d'omettre certaines étapes selon les besoins (Baskerville, et Wood-Harper, 1998).

Les cinq phases sont les suivantes :

- 1. Diagnostic (Identifier ou définir un problème)
 - o La recherche-action découle d'un problème, d'un dilemme ou d'une ambiguïté dans la situation dans laquelle les pratiquants se retrouvent
- 2. Planification de l'action (Considérer des alternatives pour résoudre un problème)
 - o Le résultat le plus important de la phase de planification est un plan détaillé de l'action que le chercheur a l'intention de prendre ou le changement qu'il a l'intention de faire. Quelles sont les modifications proposées au système? Comment le chercheur envisage-t-il de mettre en œuvre ses stratégies?
- 3. Application de l'action (Sélection d'un plan d'action)

- C'est la mise en exécution du plan. Il peut y avoir des écarts mineurs par rapport au plan initial.
- 4. Évaluation (Étudier les conséquences de l'action)
 - L'observation, et le suivi permettent d'évaluer l'effet de l'action ou de l'intervention et donc l'efficacité de la modification proposée.
- 5. Spécification de l'apprentissage (Identification des résultats généraux)
 - La réflexion régulière au sein de l'équipe du projet fait partie intégrante d'un projet de recherche-action. Les innovations peuvent être ajustées au fur et à mesure que l'activité se déroule et les participants sont concertés régulièrement.

En termes de démarche de collecte de données, dans une recherche-action, les investigations et collectes de données se déroulent sur le lieu de travail et de ce fait, aucun effort n'est fait pour « contrôler » le contexte de la recherche ou concevoir des scénarios, ou procéder à des « expériences » de laboratoire (Brydon-Miller, 2003).

3.5.2 Recherche-action et sciences du design

De par ses caractéristiques fondamentales, la recherche-action se rapproche des sciences du design. En effet, comme dans le cas des sciences du design, dans ses origines, l'essence de la recherche-action était un simple processus en deux étapes :

- Tout d'abord, **le stade de diagnostic** implique une analyse collaborative de la situation sociale par le chercheur et les sujets de recherche. Les théories sont formulées concernant la nature du domaine de recherche.
- Deuxièmement, **le stade thérapeutique** implique des expériences de changement collaboratif. Dans cette étape, des changements sont introduits et les effets sont étudiés (Denscombe, 2014)

En effet, dans leur démarche, les deux approches ont beaucoup de similarités. Cependant, dans les études récentes (Crowston et Prestopnik 2013, Miah *et al.*, 2014), la méthodologie des sciences du design est plus utilisée et recommandée dans les recherches sur l'amélioration des artefacts technologiques alors que la recherche-action correspond mieux aux questions d'amélioration des processus et relations touchant la psychologie cognitive. Le fait que nous travaillons sur la conscience de la situation et exprimons une « quête de compréhension et d'amélioration de la performance humaine », nous fait croire que notre travail se définit mieux dans un contexte de recherche-action.

De plus, l'usage d'une approche de recherche-action nous offre la possibilité de faire recours à plusieurs méthodes de collecte de données (par exemple des observations, des questionnaires d'auto-évaluation, des enquêtes contextuelles, des mesures subjectives). Ce qui nous garantit que les données collectées peuvent être efficacement recoupées afin d'en assurer la fiabilité et la précision.

Lorsqu'appliquée au domaine des systèmes d'information et projets informatiques, la description faite de la recherche-action par Baskerville et Myers (2004), nous relate ses 4 grandes caractéristiques :

- La recherche-action vise à mieux comprendre une approche immédiate concernant une situation sociale, en mettant l'accent sur la nature complexe et multivariée de ce paramètre social dans le domaine des systèmes d'information
- La recherche-action aide simultanément la résolution pratique de problèmes et élargit les connaissances scientifiques.
- La recherche-action est réalisée de manière collaborative et améliore les compétences des acteurs respectifs.

- La recherche-action est principalement applicable pour la compréhension des processus de changement dans les systèmes sociaux

Le tableau 3.2 compare les caractéristiques de la méthodologie de la recherche-action à celle des sciences du design (Järvinen, 2007).

Tableau 3. 2 Comparaison entre recherche-action et sciences du design

Recherche-action	Sciences du design
1 : La recherche-action est une méthodologie idiographique qui implique le travail sur un processus unique	1 : Les sciences du design sont une méthodologie idiographique qui implique le travail sur un processus unique
2 : La recherche-action met l'accent sur l'aspect utilité du futur système du point de vue des clients du système.	2 : Les produits des sciences du design sont évalués selon des critères de valeur ou d'utilité
3 : La recherche-action produit des connaissances pour guider la pratique en modification	3 : Les sciences du design produisent des connaissances de conception (concepts, constructions, modèles et méthodes).
4 : La recherche-action signifie à la fois la prise de mesures et l'évaluation.	4 : La construction et l'évaluation sont les deux principales activités de la science du design
5 : La recherche-action est menée en collaboration entre le chercheur-action et le client du système	5 : La recherche en sciences du design est initiée par le chercheur intéressé à développer des règles technologiques pour un certain type de problème. Chaque cas particulier est principalement axé sur la résolution du problème local en étroite collaboration avec les populations locales

6 : La recherche-action modifie une réalité donnée ou développe un nouveau système.	6 : La science du design résout les problèmes de construction (production de nouvelles innovations) et les problèmes d'amélioration (amélioration de la performance des entités existantes).
7 : Le chercheur intervient dans le contexte du problème.	7 (5) : La science du design est initiée par le chercheur intéressé à développer des règles technologiques pour un certain type de problème. Chaque cas particulier est principalement axé sur la résolution du problème local en étroite collaboration avec les populations locales
8 : La connaissance est générée, utilisée, testée et modifiée au cours du projet de recherche-action	8 : Les connaissances sont générées, utilisées et évaluées par l'action du design.

3.5.3 Recherche-action et étude de cas

Alors que la recherche-action est un processus actif de résolution progressive de problèmes conçu pour améliorer les stratégies, les pratiques ou un environnement de travail, l'étude de cas est plutôt un examen approfondi d'un seul ou d'un nombre très limité de cas, d'individu ou d'événement. C'est un processus plus intellectuel, où le chercheur examine des exemples à apprendre. La recherche-action nous permet de trouver de meilleures façons de faire les choses grâce à la résolution active de problèmes. L'étude de cas fournit un moyen systématique d'examiner les événements dans leur contexte réel, de recueillir et d'analyser les données et de rendre compte des résultats. Cela nous permet donc de comprendre pourquoi les choses se sont passées comme elles l'ont fait et de proposer des changements basés sur des exemples passés, plutôt que sur nos propres actions.

Le processus d'une étude de cas est généralement le suivant :

- Identifier et définir les questions de recherche;
- Sélectionner des cas et des techniques de décision pour la collecte et l'analyse de données;
- Collecter de données sur le terrain;
- Évaluer et analyser les données;
- Préparer le rapport

Les études de cas impliquent à la fois des données quantitatives et qualitatives et permettent aux chercheurs de voir au-delà des résultats statistiques. Cependant, les études de cas sont également critiquées car le fait qu'elles n'étudient qu'un nombre limité de cas ne permet pas d'établir facilement la généralité ou la fiabilité des résultats.

3.5.4 Que devons-nous mesurer?

Mesurer la CS revient à mesurer 1) la CS individuelle, 2) les procédures de fonctionnement d'équipe que les membres de l'équipe utilisent pour construire et échanger des informations et améliorer la coordination entre eux, 3) faudra par la suite mesurer les modèles mentaux partagés par les membres de l'équipe. À partir de l'analyse de ces aspects de la performance humaine, nous pouvons faire des inférences concernant la nature et la qualité de la CS d'équipe.

Cannon-Bowers *et al* (1989) ont identifié, dans le domaine de l'aviation, des indicateurs de la CS d'équipe dans les communications de l'équipage dans un aéronef. Nous estimons que ces indicateurs sont applicables à d'autres types d'équipe et bien évidemment aux équipes de développeurs d'application. Ils comprennent :

- l'identification d'un problème réel ou potentiel;
- la reconnaissance de la nécessité d'agir;

- la tentative de déterminer les causes de l'information divergente;
- la fourniture des informations à un autre membre de l'équipe avant qu'il ne le demande; et
- la démonstration d'une prise de conscience de l'état de sa propre tâche et ses propres performances.

De nombreux autres indicateurs ont aussi été suggérés comme étant importants pour la CS d'équipe. Ce sont :

- la confirmation et recoupement d'informations dans l'équipe;
- la coordination des activités (Schwartz, 1990);
- le partage de l'information (Wellens, 1993);
- la planification des tâches;
- la répartition des tâches; et
- la conduite des briefings avant les tâches (Robertson et Endsley, 1994).

Nous considérons ces indicateurs comme des primitifs de la mesure de la performance des équipes. Il nous faudra de toute façon procéder à une analyse plus spécifique des besoins en CS au sein des équipes étudiées. Cette démarche est détaillée dans la section 3.6.

3.6 Activités de recherche

Nous mènerons trois importantes activités de recherche :

- la détermination des besoins et des ressources pour la CS dans les équipes de développeurs;
- l'évaluation des besoins de formation pour améliorer la CS d'équipe;
- la validation de la solution proposée

L'objectif de l'analyse des besoins est de déterminer exactement ce que la CS signifie dans le milieu de la gestion de projets informatiques : Qu'est-ce qui doit être perçu par chaque personne dans son travail, qu'est-ce qui doit être compris et quelles projections doivent être faites? En nous basant donc sur une compréhension des besoins réels pour la CS dans diverses parties de l'organisation étudiée, les recommandations peuvent être faites pour améliorer les technologies ou facteurs organisationnels disponibles pour soutenir la CS, et pour créer un programme de formation qui améliore la CS dans les équipes de développeurs. Du fait que la CS soit essentielle pour la prise de décision et les performances dans de nombreux domaines, le développement de méthodes appropriées pour améliorer la CS dans un domaine donné est subordonné à une telle analyse.

Comme il ne peut être possible dans le cadre de cette seule recherche, d'examiner les pratiques de tous les prestataires de solutions informatiques, l'étude sera menée dans une entreprise spécifique de développement d'applications informatiques. Elle servira d'un site représentatif d'un environnement de gestion de projets informatiques. Spécifiquement, nous analyserons les opérations du cycle de vie de plusieurs applications en cours de développement à l'interne. Le cycle de vie inclut les étapes d'analyses, de conception, de développement, de contrôle de qualité, de déploiement et de support, et implique l'intervention de plusieurs équipes et ressources matérielles et technologiques.

Le sujet de cette étude est une grande entreprise informatique située au Canada, leader dans la fourniture de solutions informatiques de gestion des assurances maritimes. La méthode de développement de logiciel adoptée dans cette entreprise est le SCRUM, une méthode Agile (Schwaber, et Beedle, 2002) reconnue pour avoir une approche distinctive pour créer une conscience conjointe au sein de l'équipe. Nous croyons que les opérations liées au cycle de vie du développement d'application qui y ont cours sont assez représentatives des types d'activités réalisées dans le développement de logiciel,

même s'il est évident que des variations peuvent exister en raison de différences entre les compagnies informatiques ou les pratiques de gestion à divers lieux de production.

Cette compagnie, comme beaucoup dans l'industrie, a subi un certain nombre de réorganisations dans les précédents cinq ans, dont l'implantation de nouvelles méthodes de travail et la création de nouveaux départements. Bien que ces récents changements aient affecté de façon significative la manière de faire et les procédures internes, nous ne jugeons pas qu'ils puissent avoir un impact significatif sur les principaux objectifs de notre étude - déterminer l'applicabilité de la CS et de la CS d'équipe au développement d'application, et la conception de stratégies d'intervention pour l'améliorer.

En termes de séquence, la recherche sera menée d'abord en identifiant des besoins en CS et les ressources utilisées pour soutenir ces exigences dans cet échantillon représentatif d'entreprise et d'environnement de développement informatique que nous avons sélectionné. Des concepts pour la formation de CS d'équipe seront élaborés sur la base des résultats de l'analyse. Une méthodologie d'analyse conceptuelle de CS d'équipe sera appliquée pour conduire l'analyse. Cette méthode se compose de deux parties: une analyse des exigences CS et une analyse des ressources CS.

3.6.1 Analyse des besoins en CS

La première étape consistera à déterminer des exigences spécifiques des individus en matière de CS dans le domaine du développement. Cela sera résolu par une analyse des tâches orientées vers un but qui devra évaluer 1) les objectifs et sous-objectifs associés aux équipes de développeurs, 2) les exigences de décision associées à ces objectifs, et 3) les exigences en CS nécessaires pour traiter les décisions à tous les trois niveaux - détection, compréhension, et projection. Ce type d'analyse a été menée avec succès pour les pilotes dans plusieurs catégories d'aéronefs (Endsley, 1993), le contrôle du trafic aérien (Endsley et Rodgers, 1994; Endsley et Jones, 1995), dans la maintenance

des installations aéronautiques (Endsley, 1994; Endsley et Robertson, 2000) et dans l'anesthésie médicale (Schulz, 2013).

Dans l'analyse des tâches orientées-but, les objectifs abstraits de haut niveau sont analysés et décomposés en sous-objectifs spécifiques. Les buts les plus spécifiques sont ensuite décomposés en des décisions que les acteurs doivent prendre et les informations nécessaires pour atteindre les objectifs. Cette approche de décomposition des objectifs est illustrée dans la figure 3.2.

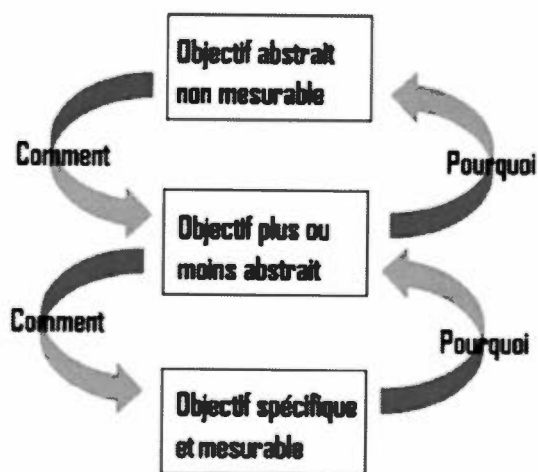


Figure 3.1 Processus de décomposition des objectifs

Les analyses seront orientées sur les activités reliées au cycle de vie de développement d'une application. Elles consisteront aux opérations suivantes :

- **Entrevues** : Il s'agira ici d'entrevues semi-structurées qui devront être menées auprès des différentes équipes impliquées dans la gestion des projets informatiques. Les participants proviendront aussi bien des locaux de Montréal que ceux des sites distants.

L'instrument questionnaire est sélectionné ici dans le but de collecter et de valider des données d'observation. Il nous permettra aussi d'avoir de meilleures

compréhensions du processus étudié. Le questionnaire est adapté du *Critical decision method probes* proposée par O'Hare *et al* (2000). Il servira à collecter des données supplémentaires sur les buts et objectifs; les situations d'erreur et les événements non usuels ayant survenus dans la vie du système étudié et la manière dont ils ont été gérés. Le recours à des techniques « *Critical decision method* » est nécessaire ici pour produire des représentations du système étudié; c'est à dire le réseau social de dépendance et la description des rôles.

- **Observations** : Il s'agit d'observer les équipes et les individus accomplir leurs activités quotidiennes de développement d'application, et principalement de participer aux rencontres de travail entre les équipes et aux seins de chaque équipe;
- Examen et analyse de la documentation de travail disponible;
- Examen et analyse des comptes rendus de réunion ainsi que d'autres outils de communications.

3.6.2 Analyse des ressources de CS

La deuxième partie de l'analyse de la CS d'équipe sera consacrée à l'identification des ressources CS utilisées dans l'environnement de développement pour atteindre les exigences CS répertoriées dans l'analyse des tâches orientées-but. Deux grandes catégories de ressources seront envisagées: 1) autres membres du personnel utilisés en tant que sources d'information et 2) les technologies utilisées comme sources d'information.

Pour fournir une évaluation des ressources CS du personnel, nous entreprendrons les actions suivantes :

- **Enquête contextuelle** : Une analyse des communications entre les équipes et les individus sera menée à l'aide d'une approche « d'enquête contextuelle ». L'approche d'enquête contextuelle met l'accent sur la compréhension et la description des

modes de communication au sein et entre les équipes en relation avec leurs objectifs de rendement et de travail.

Formellement définie, une enquête contextuelle est une « méthode d'entrevue semi-structurée qui permet d'obtenir des informations sur le contexte de travail, où on administre une série de questions aux acteurs, puis ils sont observés et interrogés pendant qu'ils travaillent dans leurs propres environnements » (Holtzblatt, 2014). Le facteur clé de différenciation entre l'enquête contextuelle et d'autres méthodes de recherche sur l'acteur est que l'enquête contextuelle se produit dans le contexte. Il ne s'agit pas simplement d'une interview ordinaire, et il n'est pas non plus question d'une simple observation. Il consiste à observer les gens accomplir leurs tâches et de les faire parler de ce qu'ils font pendant qu'ils le font. La variante de l'« enquête contextuelle » que nous utilisons dans cette recherche sera constituée d'entretiens semi-structurés dans lesquels on demandera à chaque personne de décrire ses fonctions et objectifs principaux et les équipes ou individus qui servent de ressources pour répondre à ces objectifs. Il en ressortira une cartographie montrant les communications et les interactions entre les membres de l'équipe et parmi les équipes. Chaque interviewé devra donner une estimation de la fréquence globale de communication avec chaque unité identifiée et l'importance de la communication pour atteindre leurs objectifs. Enfin, il leur sera demandé d'identifier les obstacles à la communication efficace et à la performance dans le cadre de leur travail. Les enquêtes contextuelles seront menées simultanément avec les entrevues des exigences CS.

- **Investigation des événements récents :** Il s'agira d'identifier des événements clés qui ont eu lieu dans un passé récent de la vie des équipes ou du système. À partir d'un questionnaire comme celui proposé par O'Hare *et al* (2000), nous allons procéder à une évaluation des événements, des équipes et des individus impliqués. Ce sera l'application d'une technique d'interview qui amènera les individus à

évaluer la nature de leur implication, leurs attentes dans l'issue de l'évènement. Cette technique d'interview est concordante avec la méthode « *The Critical Decision Method* » de (Klein, 1989). Selon Klein (1989), « *The Critical Decision Method* » est une stratégie d'entrevue rétrospective qui applique un ensemble de sondages cognitifs à des incidents non routiniers réels qui requièrent un jugement d'expert ou une prise de décision » (Klein, 1989, p.464). L'interview sera organisée en quatre étapes: rappel de l'incident initial et information sur l'incident; identification des points de décision dans l'incident; sondage sur les points de décision; vérification des informations recueillies. Le questionnaire proposé par O'Hare *et al* (2000) pour ce type d'entrevue est présenté dans l'annexe A (les adaptations sont indiquées en gras). Notamment, dans sa section « **Conscience de la situation** », nous y avons inclus des questions sur le recours à d'autres équipes dans la collecte des informations de décision.

En outre, les technologies utilisées pour l'obtention de chaque exigence dans le système actuel seront documentées.

Sur la base de cette analyse, nous ferons une évaluation de la mesure dans laquelle le système actuel soutient la CS dans des équipes de développeurs, les compétences et les aptitudes qui sont nécessaires pour obtenir une bonne CS dans cet environnement. Cette évaluation sera utilisée pour identifier des recommandations de conception de systèmes et concepts de formation pour améliorer la CS dans les équipes de développeurs. Elle servira aussi à vérifier l'applicabilité de la méthodologie d'ingénierie des exigences et de modélisation du social iStar.

3.6.3 La validation de la solution proposée

La validation de la solution proposée auprès des mêmes sujets étudiés est nécessaire. Les mesures de *Valeur* et *d'utilité* subjectives de la solution ainsi que les mesures

objectives des changements de comportement observés après le déploiement de la solution, permettront d'évaluer la validité et la pertinence de la solution proposée.

3.7 Conclusion

Il existe une multitude de méthodes d'analyse et d'évaluation de la CS. Prises individuellement, aucune ne peut prétendre faire des mesures convaincantes qui correspondraient aux attentes de notre recherche. Tout d'abord, les techniques de mesure de la CS que nous avons examiné, sont orientées sur l'évaluation du processus cognitif qui se déroule à l'intérieur de l'individu. De plus, elles sont assaillies par des défauts distincts susceptibles d'entraver la qualité des données CS obtenues suite à leurs applications. À part la méthode « *SA requirements analysis* », les autres méthodes ne sont presque pas applicables en dehors des dispositifs de laboratoire, c'est-à-dire, qu'elles ne sont pas conçues pour être expérimentées dans un environnement réel et complexe. Elles n'offrent pas de réelles solutions pour la mesure de la CS d'équipe.

Pour ces raisons, il nous a paru nécessaire de faire recours à une approche qui regroupe plusieurs méthodes existantes, et les plus appropriées au contexte de notre étude, afin de former une « boîte à outils » de mesures de la CS. Nous avons donc proposé une approche multi techniques, plus efficace dans le contexte de cette étude car elle est capable de mesurer à la fois la CS de l'individu et la CS d'équipe. En outre, elle offre une forte chance de garantir que les données CS puissent être efficacement recoupées entre les mesures afin d'en assurer la fiabilité et la précision.

Telle que présentée ci-dessus, l'approche de la recherche-action, nous apparaît comme une solution au défi considérable que représente l'usage des outils de mesures actuels de la CS de l'individu et la CS d'équipe dans les environnements complexes.

CHAPITRE IV

ANALYSE ET MODÉLISATION DE LA CONSCIENCE DE LA SITUATION

4.1 Introduction

Ce chapitre présente les démarches de la réalisation du travail de terrain, les résultats obtenus, leurs représentations formelles ainsi que leurs interprétations.

Une hiérarchie des buts dans le milieu de développement informatique a été développée pour plusieurs catégories de rôles au sein des équipes de développeurs et pour plusieurs équipes qui travaillent en étroite collaboration avec les développeurs pour atteindre les objectifs de l'entreprise. Ces objectifs ont été utilisés pour dresser une liste des exigences en informations pour chaque groupe. Ensuite, les ressources en personnel utilisées pour répondre à ces exigences en information sont déterminées. Les ressources technologiques utilisées pour répondre aux exigences en information ont également été répertoriées. Les obstacles et les problèmes pour atteindre les objectifs ont été identifiés au cours des entretiens. Dans ce chapitre, nous ne présentons que les résultats pour les équipes de développeurs conformément aux objectifs de notre étude. Mais avant d'élaborer sur les résultats, nous allons présenter l'application que nous avons faite de la méthodologie d'étude annoncée au chapitre 3.

4.2 Méthodes

4.2.1 Dispositif d'étude et collecte de données

La compagnie qui fait l'objet de cette étude a plus de 200 employés et intervient dans le domaine des solutions informatiques depuis 17 ans. Elle dispose d'une quarantaine

de clients répartis dans divers pays. Dans le reste de cette thèse, nous nous référerons à cette compagnie comme « L'entreprise » ou au terme équivalent anglais « The company » lorsque nécessaire. Comme il est de coutume dans l'approche de la recherche-action, nous avons accepté la responsabilité de garder la confidentialité des données, des noms des clients et partenaires, des noms des produits, services et marque de commerce, et des noms des employées de l'entreprise dans notre document (Kemmis, *et al.*, 2013).

Les données que nous présentons dans ces résultats ont été collectées durant une étude naturaliste du personnel des équipes de développement d'application à l'entreprise. Il n'y a pas eu de scénarios de laboratoire ni de simulation. Les équipes de développeurs (DEV) qui sont concernées englobent les unités de programmation, d'analyse (BA), de documentation (DOC), de bases de données (DBA), de business intelligence (BI), de test de qualité (QA), et d'intégration (INT). Les autres équipes étudiées sont celles qui travaillent en étroite collaboration avec les équipes de développeurs; il s'agit notamment des équipes d'architecte (ARCH), de gestion de projet (PM), de direction de produit (PO), de framework (FMK), et de gestion de configuration (CM). Globalement, la mission de toutes ces équipes y compris celle des développeurs consiste à produire des solutions logicielles qui sont fournies aux utilisateurs de l'entreprise. Toutes les équipes étudiées sont physiquement logées dans un même lieu, soit au siège social de l'entreprise situé dans la ville de Montréal au Canada. Parmi le personnel rencontré, certains travaillent sur des sites distants, les autres travaillent au siège.

Les opérations de cycle de vie de développement d'application étudiées sont formulées, planifiées, et exécutées sous la supervision des managers localisés au siège social de l'entreprise et sans la coopération d'aucune autre entité externe.

La compréhension des procédures de fonctionnement interne et les processus de production de l'entreprise n'a pas nécessité une formation particulière de notre part,

puisque nous collaborons avec les équipes de développeurs depuis 14 ans au moment de cette étude.

4.2.2 Échantillonnage des participants

Les analyses sont réalisées sur des données collectées auprès de 23 individus dans 8 équipes différentes.

Les employés participants possèdent des expériences et qualifications diverses. Ils ont entre 1 et 15 années d'expérience dans leurs domaines respectifs dans l'entreprise.

Parmi le personnel rencontré, 20 participants travaillent dans les bureaux de Montréal, 1 travaille à distance dans la même ville, 1 à Edmonton au Canada, 1 autre au Pérou et 1 travaille en Angleterre.

Les participants rencontrés n'avaient jamais participé à des études similaires réalisées à l'interne ou par des consultants externes sur cette même entreprise.

L'Age des participants membres des équipes étudiées varie entre 25 et 50 ans.

4.3 Dispositif expérimental

4.3.1 Observations

La collecte de données s'est échelonnée sur une période de 16 semaines. Nous avons procédé à de nombreuses observations et pris des notes dans un cahier de recherche. Ces observations ont essentiellement servi à mesurer le déroulement du processus de développement, l'analyse des procédures de travail, rôles et tâches des équipes impliquées et l'analyse de la compréhension que les participants ont de leur propre rôle et du rôle de leur équipe respective. Nous avons réalisé les activités suivantes :

- observation des membres des équipes de développeurs lors de l'exécution de leurs tâches individuelles de développement d'application et leurs interactions avec les membres de leurs équipes;
- participation aux réunions d'équipe de toutes les 5 équipes de développeurs (voir. figure 4.2);
- participation aux réunions conjointes entre équipes de développeurs et autres équipes;
- participation aux réunions de planning au cours desquelles se décide la liste des items à travailler (PBI) dans le cycle de développement;
- examen et analyse de la documentation de travail disponible sur le site web Intranet de l'entreprise. Il s'agit des documents d'analyse de système et des documents de procédure;
- examen et analyse des comptes rendus de réunion ainsi que d'autres outils de communication utilisés par les équipes de développeurs et les équipes de soutien.

Les informations recueillies lors des observations sont diverses et incluent les suivantes :

- rôles de chaque participant;
- tâches effectuées par les membres de l'équipe;
- dépendance des tâches;
- fréquence de communication avec les membres de l'équipe et autres équipes;
- utilisation des ressources technologiques;
- agenda des rencontres de travail;
- objectif des réunions;
- décisions prises durant les réunions;
- application des décisions des réunions antérieures;
- mode de résolution de problème en équipe;
- nature des informations échangées;
- problèmes de procédures identifiés par les membres ;

- problèmes de procédure observés;
- les procédures d'estimations des tâches.

4.3.2 Les entrevues

Pour compléter et valider les données que nous avons collectées lors de nos multiples observations et avoir une meilleure compréhension des processus, nous avons réalisé une quarantaine d'entrevues semi-structurées d'une durée de 30 à 45 minutes chacune. Au cours de ces entrevues, nous avons rencontré 14 participants dont 2 chefs d'équipe de développeurs, 1 architecte de système, 2 analystes, 1 gestionnaire de projet, 1 chef de produit et 7 développeurs. Les participants ont été rencontrés individuellement. Certains participants ont été rencontrés à plus d'une reprise. Nous avons inscrit sur papier les réponses recueillies lors de ces séances d'entrevu.

Les entrevues ont servi à collecter les informations de l'environnement de travail, à mesurer plus spécifiquement les éléments de CS (au niveau de la perception, de la compréhension et de la projection) que nous avons recueillis lors des exercices d'observation antérieurs. Les questionnaires utilisés ont de ce fait été légèrement adaptés pour chaque participant et équipe selon son rôle dans le processus. Globalement, le questionnaire est basé sur la méthode *Critical decision method* présentée dans le chapitre 3.

Nous avons collecté les données sur la vision du participant concernant :

- les tâches effectuées;
- les buts et objectifs de son équipe;
- la nature des relations et le mode de communication entre les membres de son équipe et entre son équipe et les autres équipes;
- le processus de production;
- les obstacles à une bonne circulation de l'information;
- les obstacles à l'atteinte de ses propres objectifs et ceux de son équipe;

- les suggestions d'amélioration du processus;
- les situations d'erreur vécues; et
- les événements non usuels ayant survenus dans la vie du système étudié et la manière dont ils ont été gérés.

4.3.3 Enquêtes contextuelles

Autant que pour les entrevues semi-structurées, les enquêtes contextuelles ont servi à collecter des informations de l'environnement plutôt que des tâches spécifiques des acteurs. Nous avons réalisé des enquêtes contextuelles auprès de 4 membres des équipes de développeurs dont 1 de l'équipe de FMK, 1 de BI, 1 de DBA et 1 de PM. Cela nous a permis d'avoir une meilleure compréhension des modes de communication au sein des équipes de DEV et entre les équipes en relation avec les DEV. Les enquêtes contextuelles ont été menées simultanément avec les entrevues des exigences CS.

Les questions principales qui ont meublé les entrevues semi-structurées sont présentées dans le questionnaire du tableau 4.1. Le questionnaire complet est présenté dans l'annexe D.

Tableau 4.1 Questionnaire des enquêtes contextuelles

SECTION 1

1. Comment définissez-vous le rôle et les objectifs de votre équipe.
2. Quelles équipes dépendent du travail de votre équipe?
3. Pour quelle information ou pour quel service?
4. Comment communiquez-vous avec cette équipe (processus, outils) ?
5. Rencontrez-vous des problèmes dans l'obtention de l'information ou des services de cette équipe?
6. De quelles équipes dépendez-vous pour accomplir votre travail?

Pour quelles informations ou quels services?

7. Comment communiquez-vous avec cette équipe (processus, outils) ?
8. Rencontrez-vous des problèmes à faire parvenir des informations ou vos services à cette équipe?
9. Pensez-vous que le processus de travail actuel fonctionne bien pour votre équipe?

4.3.4 Investigation des événements récents

Les entrevues ont permis d'identifier quelques événements marquants survenus dans le passé récent de l'entreprise en rapport avec le développement du système. Le questionnaire du tableau 4.2 nous a permis d'identifier ces événements, de comprendre les circonstances de leur déclenchement et quelles ont été les réactions et solutions observées de la part des membres des équipes DEV impliqués.

Tableau 4.2 Questionnaire d'investigation sur événements récents

1. Rappelez-vous d'un événement où vous ou votre équipe avez résolu un problème urgent.
2. Avez-vous eu toutes les informations dont vous aviez besoin pour prendre des décisions?
3. À aucun moment avez-vous eu des doutes sur la qualité, la fiabilité ou la pertinence de l'information disponible?
4. Quelle était l'information la plus importante dont vous vous êtes servi pour prendre votre décision?
5. Quelle était la source de cette information?
6. À aucun moment avez-vous eu des doutes sur la pertinence de votre décision ?
7. Vous est-il venu à l'idée des cas précédents dans lesquels des décisions semblables ont été prises?

8. Vous est-il venu à l'idée des cas précédents dans lesquels une décision différente a été prise?

Le tableau 4.3 montre les détails d'un exemple d'incident réel survenu en environnement de production.

Tableau 4.3 Extrait de rapport d'incident

(Le nom du client y est masqué pour des raisons de confidentialité)

[Application] Incident Report October 28, 2014 Incident Report Summary On Thursday, October 23 2014 at 10:07 AM EDT, a data maintenance script was applied to the [CLIENT] database which caused an incorrect policy status to be displayed on 423 policy transactions. Incident Report Details & Resolution: A planned data maintenance script was applied to the [CLIENT] production database as part of the Liability product implementation. Subsequent testing indicated that the script had updated an incorrect set of policy transaction records. The [Company] Production Operations team took immediate action to prevent further data inconsistency by preventing users from accessing the [CLIENT] website, while simultaneously initiating a database restore to a version prior to the application of the data maintenance script. The investigation revealed that a previous version of the planned [CLIENT] data maintenance script had been mistakenly checked into [The Company]'s source control system rather than the version used in pre-production testing. This incorrect version was then applied to the [CLIENT] production environment.

Tableau 4.3 (Suite)

To avoid this type of error from reoccurring, [The Company]'s data maintenance process has been amended to include an additional checkpoint prior to the application of data maintenance scripts to production. In addition, [The Company] will be implementing the data maintenance rollback framework used on the Cargo platform which allows for rapid corrections to be made after data maintenance script execution.

[The Company]'s [CLIENT] Account Manager worked closely with the [CLIENT]'s underwriters to review user activity between 10:07 AM EDT and 11:14 AM EDT and determined that no policies were bound during this window. There were two endorsements and one quoted renewal identified that needed to be manually re-entered. Subsequently, they have been re-keyed into the system.

Outage Duration Summary

Outage Window: Thursday, October 23 2014, 10:07 AM EDT to 11:37 AM EDT

Total Outage Duration: 90 minutes

Please do not hesitate to contact your Account Representative or Customer Support at support@[The Company].com if you have any questions.

Incident Timeline

Monday, October 27 2014:

- | | | |
|---|---|--------------|
| 6 | Confirmation the two endorsements and one quoted renewal were manually re-entered | 10:49 AM EDT |
| 5 | [CLIENT] website was returned to normal operation | 11:37 AM EDT |

4.4 Les résultats

4.4.1.1 Le processus normal de production

4.4.1.2 Les équipes impliquées

L'étude a révélé la présence de plusieurs équipes qui dépendent les unes des autres pour l'atteinte des objectifs de l'entreprise. La liste des équipes dénombrées ainsi que leurs rôles, sont présentés dans les tableaux 4.4 à 4.18. Les 5 équipes ou unités DEV que nous avons dénombrées fonctionnent de façon autonome. Chaque unité a un superviseur (Lead) et des membres. Chaque unité est en charge du développement et du maintien d'une partie de l'application (sous-application). Les équipes fonctionnent de façon cloisonnée et ont peu de rencontres formelles ensemble.

La figure 4.1 est une photo montrant les installations de l'équipe BI. L'ergonomie fait entrevoir une facilité de communication entre les membres pour le partage d'information de situation et des connaissances.



Figure 4.1 Photo des installations de l'équipe BI

Photo prise en Juillet 2015 au siège de l'entreprise à Montréal, Canada.

La structure schématique des équipes de développement est présentée dans la figure 4.2. Le schéma montre des équipes cloisonnées.

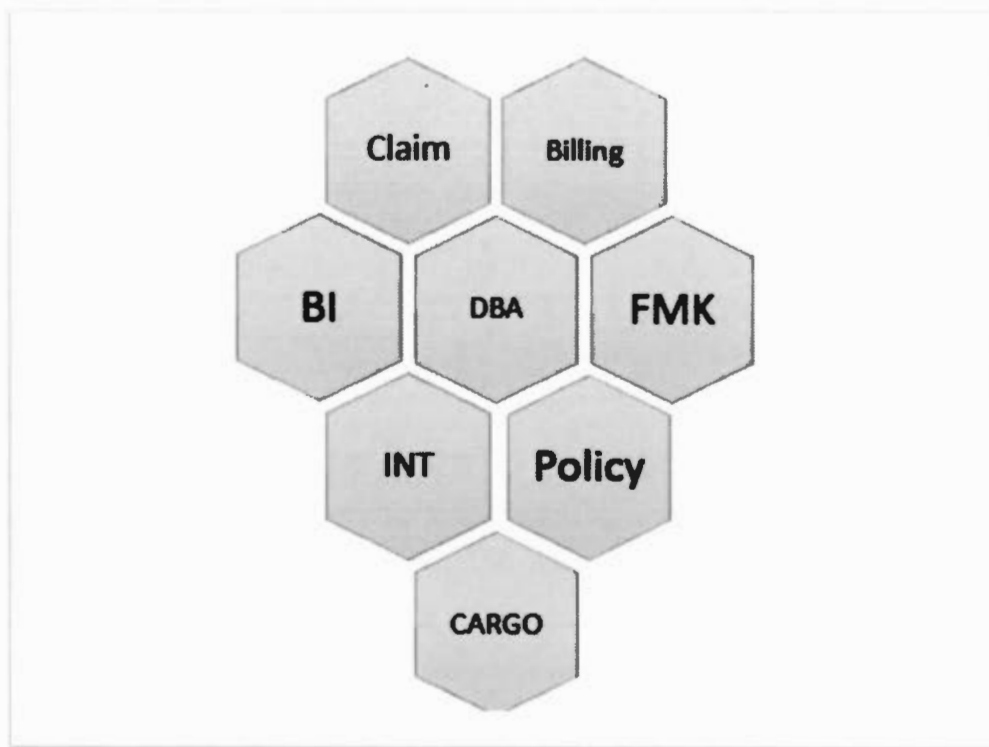


Figure 4.2 Représentation des équipes de développeurs chez l'entreprise

Tableau 4.4 Rôles et attributions de l'équipe GESTIONNAIRE DE PRODUIT

Nom de l'équipe	GESTIONNAIRE DE PRODUIT (PO)
Description	Acteur clé du cycle de vie du produit. Cette équipe comporte 4 membres.
Responsabilités	- Proposer une vision des applications à bâtir. - Communiquer sa vision du produit au client - Communiquer sa vision du produit aux développeurs - Prioriser l'exécution des travaux.
Contraintes	- Efficace, collaborative, produit livré à temps, produit fonctionnel

Tableau 4.5 Rôles et attributions de l'équipe BILLING

Nom de l'équipe	BILLING
Description	Unité de l'équipe de développeurs qui développe la sous-application « BILLING ». Elle comporte 4 membres
Responsabilités	- Développe les fonctionnalités de l'application BILLING - Développe des Web Services pour l'échange de données entre l'application BILLING et les autres applications - Met à jour la liste des versions de l'application BILLING - Fait la démonstration des composantes développées - Développe des procédures d'automatisation des tests de qualité - Effectue des tests de qualité de l'application BILLING - Documente l'application BILLING - Effectue la maintenance des données de l'application BILLING - Communique le statut de l'application BILLING aux parties prenantes.
Contraintes	- Efficace, collaborative, travail de qualité

Tableau 4.6 Rôles et attributions de l'équipe CLAIM

Nom de l'équipe	CLAIM
Description	Unité de l'équipe de développeurs qui développe la sous-application CLAIM. Elle comporte 4 membres
Responsabilités	- Développe les fonctionnalités de l'application CLAIM - Développe des Web Services pour l'échange de données - Met à jour la liste des versions de l'application CLAIM - Fait la démonstration des composantes développées - Développe des procédures d'automatisation des tests de qualité - Effectue des tests de qualité de l'application CLAIM - Documente l'application CLAIM - Communique le statut de l'application CLAIM aux parties prenantes.

Tableau 4.7 Rôles et attributions de l'équipe CARGO

Nom de l'équipe	CARGO
Description	Unité de l'équipe de développeurs qui développe la sous-application CARGO. Elle comporte 8 membres
Responsabilités	- Développe les fonctionnalités de l'application CARGO - Met à jour la liste des versions de l'application CARGO - Fait la démonstration des composantes développées - Développe des procédures d'automatisation des tests de qualité - Effectue des tests de qualité de l'application CARGO - Documente l'application CARGO - Communique le statut de l'application CARGO aux parties prenantes.
Contraintes	- Efficace, collaborative, travail de qualité

Tableau 4.8 Rôles et attributions de l'équipe BUSINESS INTELLIGENCE

Nom de l'équipe	BUSINESS INTELLIGENCE (BI)
Description	Unité de l'équipe de développeurs qui développe l'application BI (Business intelligence). Elle comporte 4 membres
Responsabilités	- Développe les fonctionnalités de l'application BI - Crée des rapports de gestion pour les utilisateurs - Met à jour la liste des versions de l'application BI - Fait la démonstration des composantes développées - Développe des procédures d'automatisation des tests de qualité - Documente l'application BI - Effectue la maintenance des données de l'application BI - Communique le statut de l'application BI aux parties prenantes. - Diagnostic les erreurs d'application en tout environnement
Contraintes	- Efficace, collaborative, application et rapport de qualité, diagnostic rapide, estimés réalistes.

Tableau 4.9 Rôles et attributions de l'équipe DBA

Nom de l'équipe	DBA
Description	Unité de l'équipe de développeurs qui est en charge de toutes les bases de données des applications du système. Toutes les unités de DEV utilisent les travaux des DBA. Elle comporte 3 membres.
Responsabilités	- Propose une architecture de stockage des données, - Développe des routines de maintenance des données - Automatise les scripts de maintenance de données - Révise les codes de maintenance de données développés par les autres équipes - Documente la base de données - Fait l'extraction des données pour le besoin des clients. - Développe des standards internes de gestion de base de données - Investigue les problèmes de requêtes dans tous les environnements
Contraintes	- Efficace, collaborative, codes de qualité qui n'occasionnent pas d'erreur majeure en environnement de production, diagnostic rapide, des estimés réalistes

Tableau 4.10 Rôles et attributions de l'équipe INTEGRATION

Nom de l'équipe	INTEGRATION (INT)
Description	Unité de l'équipe de DEV en charge de l'intégration de toutes les applications avec les systèmes clients Elle comporte 5 membres.
Responsabilités	- Propose une architecture d'intégration des données entre les applications du système. - Développe des routines d'intégration des applications avec les clients - Documente les procédures d'intégration - Exploite les Web Services proposés par les autres équipes
Contraintes	- Efficace, collaborative, travail de qualité, estimés réalistes

Tableau 4.11 Rôles et attributions de l'équipe GESTIONNAIRE DE PROJET

Nom de l'équipe	GESTIONNAIRE DE PROJET (PM)
Description	Unité de support aux équipes de DEV. Le PM est considéré comme le « propriétaire » du processus de travail dans l'équipe. Elle comporte 5 membres.
Responsabilités	- Veille à ce que les équipes DEV mettent en pratique les processus de développement et les valeurs d'équipe. - Veille à la cohésion à l'intérieur des équipes DEV - Fait le pont entre les DEV et les autres équipes dans la communication des travaux dépendants - Assure la bonne répartition du travail à l'intérieur des équipes DEV.
Contraintes	- Efficace, collaborative

Tableau 4.12 Rôles et attributions de l'équipe IT

Nom de l'équipe	IT
Description	Équipe technique. Unité de support aux équipes de DEV.
Responsabilités	- Assure le bon fonctionnement de l'infrastructure - Assure le bon fonctionnement des logiciels de travail - Assure la sécurité des équipements - Assure la sécurité des applications

Tableau 4.13 Rôles et attributions de l'équipe GESTIONNAIRE DE CONFIGURATION,

Nom de l'équipe	GESTIONNAIRE DE CONFIGURATION(CM)
Description	Spécialiste de configuration. Cette équipe est en charge du déploiement des codes d'un environnement à l'autre, et de la gestion des versions.
Responsabilités	- Développe des outils de déploiement d'application - Déploie les applications d'un environnement à l'autre
Contraintes	- Travail de qualité

Tableau 4.14 Rôles et attributions de l'équipe GESTIONNAIRE DE COMPTE

Nom de l'équipe	GESTIONNAIRE DE COMPTE (AT)
Description	Gestionnaire de compte. Leur rôle est de convenir de l'affectation des priorités aux travaux.
Responsabilités	- Gérer les affectations des priorités aux travaux de maintenance de données - Assurer de la qualité du produit fini avant livraison au client - Fait la démonstration des applications aux clients - Prépare les exigences de haut niveau aux équipes de développeurs. - Fournit des rapports ad hoc aux clients - Intermédiaire entre les développeurs et le client
Contraintes	- Efficace, collaborative, travail de qualité

Tableau 4.15 Rôles et attributions de l'équipe GESTIONNAIRE DES OPERATIONS

Nom de l'équipe	GESTIONNAIRE DES OPERATIONS (OPS)
Description	Elle veille sur le bon fonctionnement des applications en production. Elle comporte 2 membres.
Responsabilités	- Veille au bon fonctionnement des applications en environnement de production - Développe des outils de monitoring - Reçoit les alarmes et notifications - Propose des solutions de premier niveau aux problèmes de production (performance de l'infrastructure, performance des applications, performance de la base données)
Contraintes	- Efficace, collaborative, travail de qualité

Tableau 4.16 Rôles et attributions de l'équipe ARCHITECTE

Nom de l'équipe	ARCHITECTE (ARCH)
Description	Architectes systèmes. Ils sont en charge de l'orientation technique des applications. Elle comporte 5 membres.
Responsabilités	- Définit l'orientation technique des applications - Propose les standards internes de codification - Veille à l'application des standards internes de bonnes pratiques de codification
Contraintes	- Efficace, collaborative, travail de qualité

Tableau 4.17 Rôles et attributions de l'équipe FRAMEWORK

Nom de l'équipe	FRAMEWORK (FMK)
Description	Équipe de Framework. Elle comporte 6 membres. Toutes les unités de DEV utilisent les travaux de cette équipe.
Responsabilités	- Créer les composantes structurelles de l'application (les menus de navigation, grilles, dynamisme des pages, contrôle d'interface : éditeur de texte, sélectionneur de date, validateur d'entrée numérique - Définir les standards applicables aux composantes structurelles
Contraintes	- Efficace, collaborative, travail de qualité, les composantes en mis en production doivent être compatible avec toutes les versions des browsers.

Tableau 4.18 Rôles et attributions du superviseur général

Nom de l'équipe	SUPERVISEUR (SUP)
Description	Le superviseur général agit comme personne-ressource pour toutes les équipes de DEV et des AT. Cette équipe comprend un membre
Responsabilité	- Motive les équipes; - Coordonne les activités des DEV dans les situations d'urgences; - Communique l'état des travaux urgents, aux AT et PO; - Assure la qualité des applications; - Décide de l'orientation technologique de l'entreprise.
Contrainte	- Information précise

Tableau 4.19 Liste des environnements d'application

DEV	Environnement de développement
QA	Environnement de test de qualité
UAT	Environnement de test client. Il s'agit de l'environnement mis en place pour les clients pour effectuer et valider leur système avant la mise en production.
PROD	C'est l'environnement de production. Les applications développées par les équipes DEV sont livrées aux clients dans cet environnement. Les utilisateurs des applications connectent sur le système pour l'exploiter via Internet.
PROD-SUPPORT	Environnement de support à la production. Une réplique de PROD pour effectuer des diagnostics et supporter le client. Le client n'a pas accès à cet environnement.

4.4.1.3 Le flux général du travail

Comme illustré dans la figure 4.3, le flux général de travail dans lequel sont impliqués les développeurs est articulé autour de trois grands groupes d'acteurs :

- Le **client** formule ses besoins au gestionnaire de comptes. Il reçoit une application utilisable proposée par les développeurs
- Le **gestionnaire de comptes** joue l'intermédiaire entre le client et les développeurs. Il reçoit les requêtes du client et fournit les exigences au développeur.

- Le **développeur** reçoit les exigences du gestionnaire des comptes et produit une solution livrable au client.
- Le **développeur** est en relation de dépendance stratégique et réciproque avec les membres des autres équipes de l'entreprise.

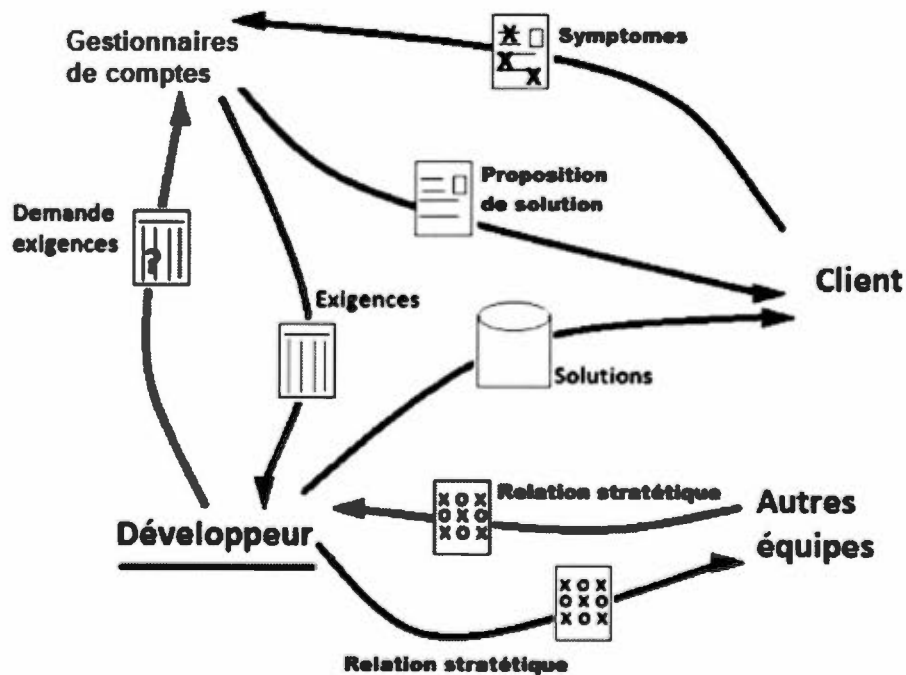


Figure 4. 3 Flux général du travail dans le processus de production

4.4.1.4 Le processus de production

Le figure 4.4 donne une vue schématique du processus normal de production suivi par toutes les équipes de développement. Le cycle de vie du développement pratiqué dans l'entreprise consiste en une série d'activités exécutées soit en équipe soit individuellement par chaque participant afin de réaliser un produit livrable au client. Le livrable est soit un système complet ou une amélioration à un système déjà existant en production.

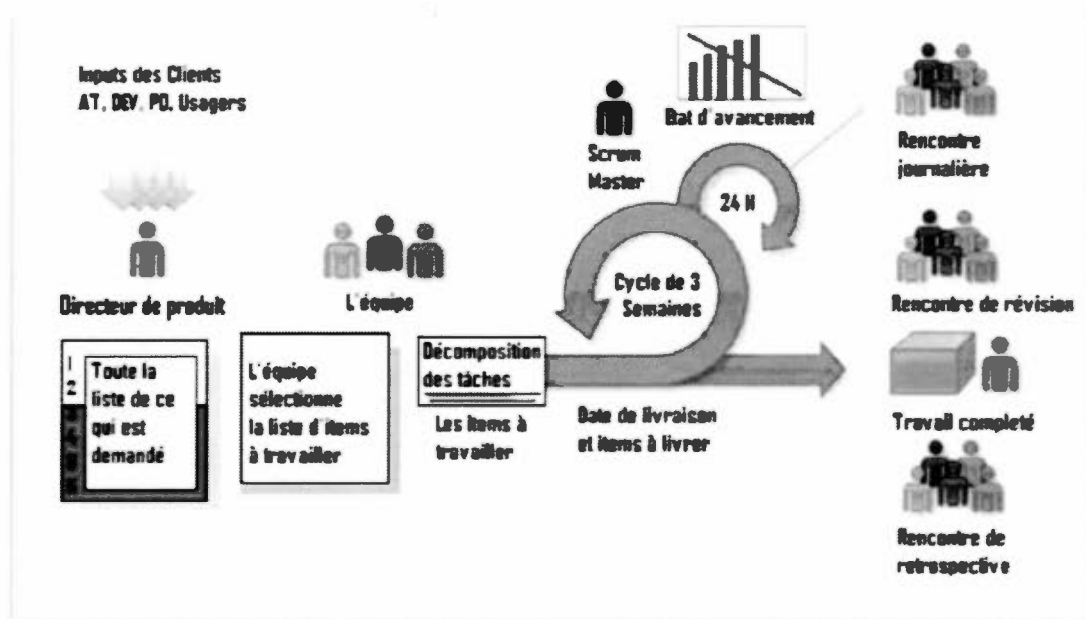


Figure 4. 4 Les processus du cycle de vie de l'application

Nous résumons ci-dessous, les activités qui ont cours dans le cycle et la séquence dans laquelle elles sont exécutées. Globalement les activités du cycle sont structurées de la façon suivante :

- Initiation
- Triage (*Pre-grooming*)
- Identification de la nature du travail (*Grooming*)
- Planification
- Développement
 - Programmation
 - Revue de code
 - Test
 - Contrôle de version

- Déploiement
- Révision
- Rétrospective

Activité 1 : Initiation

- Le PO rassemble et prépare la liste des PBI (Product back log). Les PBI sont les propositions d'amélioration faites par le PO et les clients, ou des bogues identifiés dans l'application.

Activité 2 : Triage

- **But** : Clarifier et donner des directives d'implantation aux DEV afin d'éviter des pertes de temps lors de l'activité du grooming
- **Quand** : Avant l'activité de grooming.
- **Inputs** : Le PO envoie par email la liste des PBI (Product back log), essentiellement des bogues et améliorations, aux architectes, aux chefs d'équipe DEV et aux gestionnaires de projets.
- **Actions** :
 - Les architectes approuvent les PBIs pour le *grooming*, donnent des recommandations et des suggestions d'implantation de la solution. Les architectes travaillent avec les DEV leads pour découper en plusieurs étapes les PBI jugés volumineux.
 - Si un PBI est complexe, les architectes convoquent une session de travail spéciale pour en discuter avant la session de grooming.

Activité 3 : Identification

- **But** : Discussion et compréhension des PBI par les équipes d'exécution des PBI (DEV, PM, PO)
- **Actions** : Les PO et l'équipe révisent tous les PBI, clarifient leur compréhension, donnent des estimés de haut niveau, divisent les PBI trop grand et précisent les critères d'acceptation de la solution. La liste définitive des PBI à travailler est déterminée.
- **Quand** : A travers le cycle
- **Inputs** : des PBI et des bogues révisés par les architectes.
- **Outputs** :
 - Estimés de haut niveau;
 - Description détaillée des PBI et des critères d'acceptation;
 - Les questions;
 - Les estimations de niveau de risque

Activité 4 : Rencontres quotidiennes

- **But** : Permettre aux membres de l'équipe DEV et PM d'avoir une connaissance de l'évolution des activités en cours.
- **Actions** : Au cours de cette activité, les DEV doivent répondre aux questions suivantes :
 - Qu'avez-vous réalisé hier?
 - Qu'allez-vous faire aujourd'hui?
 - Rencontrez-vous un blocage dans l'exécution de votre tâche?
 - Avez-vous besoin du travail d'un membre d'une autre équipe pour continuer le vôtre?

- **Entrées :** Les mises à jour des membres de l'équipe DEV sur l'avancement des tâches.
- **Sorties:**
 - Quels PBI seront travaillés aujourd'hui;
 - Les travaux dépendants d'autres équipes (les *Impediments*)

Activité 5 : Planification du cycle

- **But**
 - Réviser les PBI et Bug ciblés pour le cycle afin de permettre à l'équipe DEV d'avoir une meilleure compréhension de l'envergure de chaque item de travail;
 - Procéder aux assignations et estimés en termes d'heures;
 - S'assurer que l'équipe n'est ni sous-utilisée ni surchargée.
- **Quand :** Au premier jour du Cycle
- **Action :**

Les PBI et bogues sont révisés par tous les membres de l'équipe DEV et PO selon les priorités définies précédemment par les PO. Les items inclus dans la liste en dernière minute et qui n'ont pas fait l'objet de grooming, doivent être analysés avant d'être considérés dans la planification (les estimés peuvent être révisés lorsqu'il y a changement dans l'envergure du PBI et si le PBI a fait l'objet de grooming pour plus de 60 jours avant la tenue de la planification);

- Définition de la disponibilité et de la capacité de l'équipe de DEV : nombre d'heure par jour, jour de vacances;
- Clarification de l'envergure du PBI;
- Engagement de l'équipe à travailler sur les PBI et bogues;

- Si nécessaire, l'équipe peut suggérer de nouveaux PBI complémentaires pour les cycles à venir ;
 - Révision de la capacité de l'équipe;
 - Allocation du temps pour les items non planifiés ou inconnus dans le cycle en cours;
 - Chaque item doit être codé, vérifié, testé, documenté par différents membres de l'équipe.
- **Sorties :**
 - Décomposition des PBI en des tâches exécutables par 1 ou plusieurs DEV
 - Assignment des tâches à des membres de l'équipe DEV
 - Révision de la capacité (nombre d'heures) de DEV
 - Création des PBI pour d'autres équipes si nécessaire pour l'accomplissement des tâches de DEV.
 - Définition des critères de complétude des PBI et bogues.
 - Les DEV s'engage formellement sur les PBI et bogues à livrer en fin de cycle.

Bridge Team Sprint 64		
June 3 - June 22 2 work days remaining		
Backlog	Board	Capacity
+ - Create query Column options ✉		
Title	State	Assigned To
Dev - Upgrade Workflow package	In Progress	Michel
[Claims 82228] - As Mark I want to export my Claim's product...	New	David
QA - Testing by claims team	To Do	
[Claims 91373] Minor UI Adjustments for Claims	Approved	David
Dev - Coding	To Do	
QA - Testing by claims team	To Do	
[Claims 84235] As Mark, I want to see Dashboard Security Rig...	Committed	David
QA - Testing by claims team	Done	
Awaiting response from Claims	To Do	
Sprint Planning, Review & Retrospective	Committed	David
Doc - Planning	Done	Jason
Doc - Review & Retrospective	To Do	Jason
BA - Review & Retrospective	To Do	
QA Planning	Done	
QA - Review & Retrospective	To Do	
Dev - Review & Retrospective	To Do	
Dev Planning	Done	
Print button greyed out in IE11/IE8 and Chrome	Done	Steven
Coding 2.3.1	Done	George
Code review	Done	Jack
Rresearch	Done	George
QA - Testing Chrome (2.3 /2.4)	Done	Allan

Figure 4.5 Liste des tâches en exécution dans un cycle

Les exigences durant le cycle

Sur toute la durée du cycle, se déroulent de façon itérative, les activités de codification, de revue de code, de test de qualité, de documentation, de création des « builds » et de déploiement.

Codification : Cette activité est généralement exécutée par un seul développeur qui en est responsable. Il s'agit du travail sur une tâche associée à un PBI.

Revue de code : La revue de code est une activité menée par les chefs d'équipe mais elle peut être exécutée aussi par tout développeur expérimenté. Cependant, un développeur ne peut pas réviser son propre code. La revue de code assure la qualité des codes, permet de découvrir et de résorber des problèmes qui auraient échappés au développeur.

Test de qualité : Il s'agit du contrôle qui est fait pour s'assurer que le produit développé répond aux attentes du client. Le rôle du testeur est de découvrir des problèmes dans le logiciel. Il ne les corrige pas.

Documentation : Il s'agit ici de la mise à jour incrémentale du manuel d'utilisateur. Il est maintenu à mesure que l'application évolue.

Contrôle de version et déploiement : Les codes modifiés par les programmeurs sont enregistrés dans la base de codes. Ils composent les packages qui sont déployés d'un environnement à l'autre. Cette opération se déroule tout au long du cycle de vie de l'application. Le processus de déploiement est illustré dans la figure 4.6. Sur la figure, les environnements d'application ainsi que les séquences de déploiement sont indiqués dans la colonne de gauche, et les actions dans la colonne de droite.

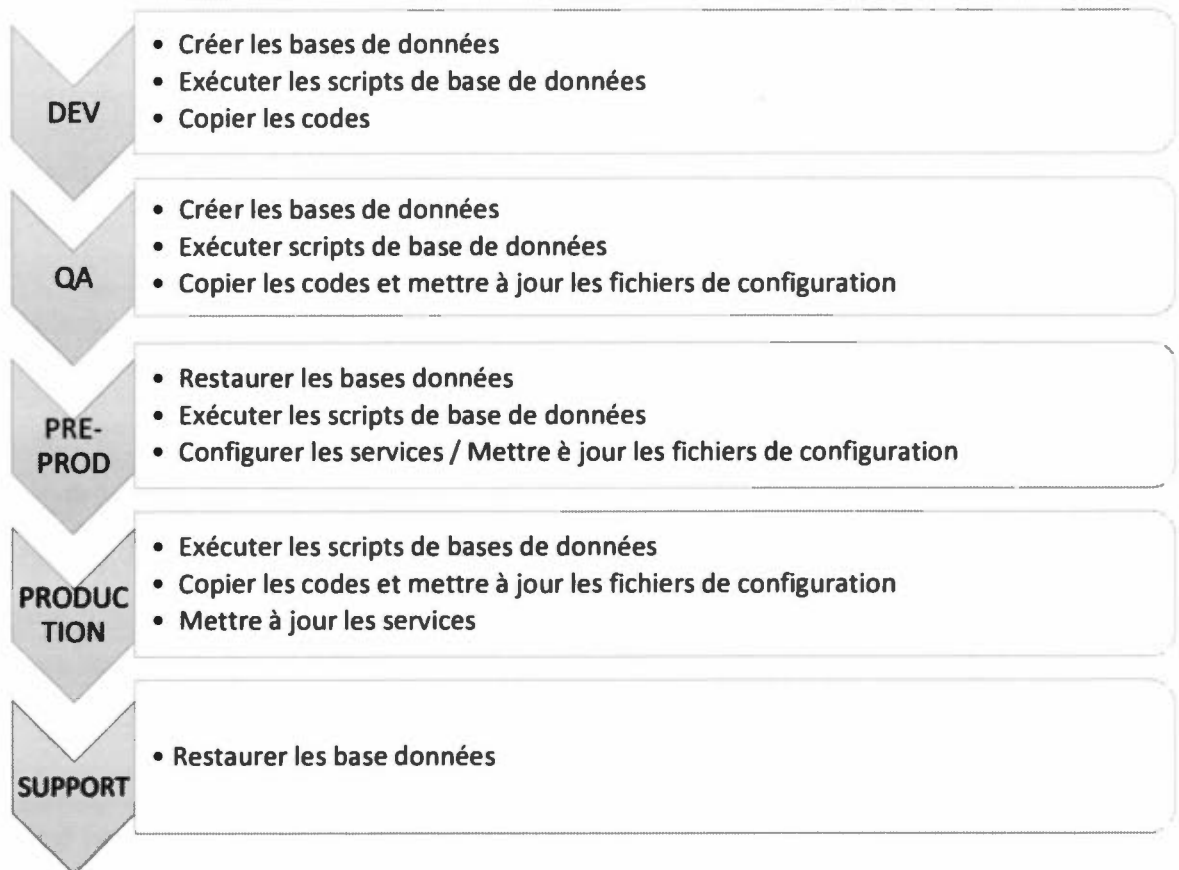


Figure 4.6 Le processus de déploiement d'une application

Autres exigences durant le déroulement du cycle

- **Changement à l'envergure du travail**

S'il y a des changements à l'envergure du travail sur un PBI, un nouveau PBI doit être créé. Le PO et l'équipe font une révision des exigences du PBI initiale et révisent l'impact du changement d'envergure sur des fonctionnalités existantes ou en cours de développement. Après révision de l'impact, le PO détermine si le travail sur le PBI doit continuer ou si le PBI tout entier doit être renvoyé à un cycle de vie ultérieur

- **Les urgences de support**

Lorsqu'il y a des items qui nécessitent un support urgent, ils sont priorisés par le PO, par rapport aux items déjà planifiés dans le cycle. Si l'item de support est important, il est inséré, analysé, priorisé, et décomposé en tâches et assigné aux DEV. Il est inclus dans le tableau des tâches du cycle en cours. S'il le faut, un autre item moins important est retiré du tableau et reporté.

- **Tâches de conception**

Une tâche de conception peut être nécessaire pour tout item où l'effort d'exécution est de deux jours et plus ou sur demande du chef d'équipe de DEV ou des architectes. Pour ce faire, des rencontres de travail de conception doivent être organisées entre les DEV, et au cas échéant, avec les architectes et membres d'autres équipes, selon les dépendances du travail sur d'autres équipes.

- **Tâches d'investigation**

L'exécution de certaines tâches nécessite une investigation initiale. Le temps passé sur l'investigation est dissocié du temps de développement.

Activité 6 : Révision du cycle

- **But :** Montrer le "delta" du produit aux parties prenantes et recevoir leur feedback.
- **Quand :** Le dernier jour du cycle

- **Actions :**

- Les items complétés dans le cycle et qui rencontrent les exigences de complétudes.
- Toute information importante.
- Les erreurs et leçons apprises.
- Démonstration des réalisations à toute l'équipe incluant les PO, et les AT.

- **Sorties :**

- Nouveau PBI et bogues pour des améliorations identifiées par les parties prenantes.

Activité 7 : Rétrospective du cycle

- **But :** Identifier les pistes d'amélioration et réviser le degré d'application des propositions antérieures.
- **Quand :** La toute dernière rencontre du cycle.
- **Actions :**
 - « Que devons-nous faire différemment? » Les PM gardent une liste des propositions d'amélioration et décisions issues de l'activité de rétrospection.
 - Qu'est-ce qui a bien fonctionné durant le cycle?
 - Qu'est-ce qui a moins bien fonctionné?

- **Sorties :**

- Prioriser les actions à prendre dans le future pour un meilleur rendement de l'équipe.
- Les leçons apprises.
- Les PBI non complétés à reporter sur le prochain cycle.

4.4.2 Buts et fonctions des développeurs

Nous avons pu élaborer une hiérarchie des buts et objectifs pour plusieurs catégories de rôles au sein des l'équipes de développeurs, et pour plusieurs équipes de soutien aux développeurs. Le travail d'identification des objectifs a été fait en 2 étapes : la création de la liste des besoins et l'épuration de la liste.

4.4.2.1 Génération d'une liste initiale d'objectifs

L'identification des objectifs a nécessité une analyse des sources d'information issues du processus de production, des documents de procédures existants, de l'organigramme du fonctionnement interne des équipes, et des interviews. Les documents de procédure ont servi à décrire les mandats de haut niveau qui gouvernent le système et l'organisation tout entier. L'étude des processus de production décrit le processus dans laquelle sont engagées les équipes de développeurs. Les interviews ont permis de collecter des informations sur les détails du fonctionnement des équipes à travers les anecdotes, les vécus personnels, les scénarios d'utilisation du système, les stratégies personnelles mise en place par chaque acteur, la nature des relations interpersonnelles, les opinions, les analyses et interprétations du fonctionnement du processus de la part de ceux qui animent le système.

Nous avons donc identifié dans un premier temps, plusieurs objectifs. Pour ce faire chaque énoncé est analysé en nous demandant « quel objectif il explique ou quel objectif il soutient? » et/ou « pour quel objectif il constitue un obstacle? ». Formuler les objectifs de cette façon a rendu facile le raisonnement sur les préconditions et post conditions des opérations du système.

4.4.2.2 Épuration de la liste des objectifs

Nous avons par la suite procédé à une épuration de la liste des objectifs générés à la première étape. Pour ce faire, nous avons 1) éliminé les doublons et 2) consolidé les objectifs synonymes. Par exemple, l'objectif « Placarder les problèmes » et « Reporter les problèmes dans un autre cycle » sont des synonymes. Dans ce cas, nous avons éliminé « Placarder les problèmes » de la liste finale. Les objectifs poursuivis par toutes ces équipes sont représentés dans le tableau 4.20.

Une analyse des objectifs des équipes de développeurs a révélé qu'ils semblent être orientés vers 4 rôles :

- la sécurité de l'application;
- l'assurance de la qualité de l'application;
- la performance de l'application;
- la qualité des données traitées; et
- la livraison des composantes de l'application à temps.

En général, les objectifs des superviseurs de haut niveau et des chefs d'équipe sont très semblables à ceux des développeurs. Ils assument pratiquement les mêmes sous-objectifs que les développeurs et ils viennent en support aux développeurs dans la résolution des problèmes de développement lorsque ces derniers sont dans le besoin.

Les superviseurs, cependant, ont également des sous-objectifs supplémentaires associés à la gestion des équipes de développeurs (attribution et priorisation des tâches, révision des travaux, évaluation de l'état des composantes développées, coordination interne de leur équipe et coordination avec d'autres équipes).

Tableau 4.20 Les objectifs des développeurs

1.0 Assurer la sécurité de l'application
1.1 Livrer une composante sans compromettre la sécurité de l'application
1.1.1 Trouver les problèmes potentiels
1.1.2 Régler les problèmes
1.1.3 Faire la maintenance des données
1.1.3.1 Déterminer la disponibilité des développeurs
1.1.3.2 Reporter les problèmes
1.1.4 Faire un travail de qualité.
1.2 Faire les mises à jour des logiciels de développement
2.0. Livrer une application de bonne qualité
2.1. Clarifier les exigences
3.0. Livrer une application performante
4.0. Livrer l'application à temps aux clients
4.1. Prioriser les tâches
4.2. Planifier les livrables
4.3 Faire des estimés réalistes

À l'intérieur des équipes de développeurs, nous avons observé des dépendances entre les membres selon la nature de leurs rôles dans l'équipe. En voici quelques exemples :

- les programmeurs sont dépendants des DBA pour les bases de données;
- les programmeurs sont dépendants des analystes pour la documentation des exigences;
- les BI sont dépendants des DBA pour la structure de la base de données;
- les analystes sont dépendants des DBA pour la structure de la base de données;
- les QA sont dépendants des programmeurs pour la mise à jour de la base de codes.

La révision des objectifs de chacun des autres départements impliqués dans le développement des applications a révélé d'importantes interdépendances entre les équipes. Par exemple :

- les DEV sont dépendants des ARCH pour des spécifications d'implémentation techniques;
- les DEV sont dépendants des PO pour des items à travailler;
- les DEV sont dépendants de AT pour des spécifications et des clarifications de travail;
- les CM sont dépendants des DEV pour les items à déployer et pour la mise à jour des bases de codes;
- les DEV sont dépendants des CM pour le déploiement des items à tester;
- les CM sont dépendants des DEV pour les « builds » à déployer;
- les DEV sont dépendants de l'équipe FMK pour des contrôles d'interface et leurs mises à jour;

- les INT sont dépendantes des DEV pour savoir comment consommer les Web services;
- les AT sont dépendants des DEV pour la documentation du guide de l'utilisateur;
- les DEV dépendent des IT pour des machines et des mises à jour de logiciel de travail;
- les AT ont besoin des DBA pour leur donner des rapports et des instructions SQL Ad hoc leur permettant d'interroger les bases de données;
- les DEV sont dépendants d'autres DEV pour les questions de compatibilité des versions des sous-applications.

Quoique ces interdépendances ne nous surprennent pas, elles soulignent la nécessité d'un bon transfert d'information entre les équipes. En examinant en détail les exigences en information de chaque équipe, il devrait être possible d'acquérir une compréhension des types d'information requis par chaque équipe et chaque unité et les moyens par lesquels le transfert de cette information peut être amélioré. L'interprétation de ces dépendances est présentée dans la section 4.9 de ce chapitre.

4.4.3 Capture des exigences en CS

À partir des entrevues et des observations effectuées, l'analyse nous a permis d'élaborer une décomposition des exigences en informations pour chaque objectif et sous-objectif identifiés pour chaque équipe ou unité de développeurs (tableau 4.21). Ces exigences en information sont identifiées au niveau de la perception, de la compréhension et de la projection.

En examinant le tableau 4.21, quelques observations doivent être faites :

(1) À tout moment, plus d'un objectif ou sous-objectif peuvent être opérationnels, bien que ces objectifs ne soient pas du même niveau de priorité. Les exigences CS dans le tableau 4.21 ne supposent aucune hiérarchisation entre elles, ou que chaque sous-objectif dans un objectif sera toujours présent.

(2) L'analyse cherche à définir ce que les DEV aimeraient idéalement savoir en termes d'informations pour atteindre chaque objectif. Il est de fait qu'ils doivent souvent fonctionner sur la base d'informations incomplètes et que certaines informations souhaitées peuvent ne pas être facilement disponibles dans le système tel qu'il fonctionne d'aujourd'hui.

(3) Les connaissances statiques, telles que les procédures ou les règles d'exécution des tâches sont en dehors des limites de cette analyse. L'analyse identifie d'abord des informations dynamiques sur la situation (informations qui changent d'une situation à l'autre) et qui affectent la manière dont les DEV accomplissent leurs tâches.

Cette analyse a été effectuée pour chacune des équipes de développeurs (l'équipe de BILLING, POLICY, CLAIM, etc.).

Tableau 4.21 Besoins en information des équipes de développeurs

Rôle des développeurs

1.0 Assurer la sécurité de l'application

1.1 Livrer une fonctionnalité sans compromettre la sécurité de l'application

1.1.1 Trouver les problèmes potentiels

- Compatibilité des composantes due aux mises jour d'une partie de l'application
- Obsolescence des composantes due à la réécriture de nouvelles composantes
- Problèmes à reporter

- Nouveaux problèmes
- Problèmes des utilisateurs
- Composantes mises en production avant test
- Déploiement de mauvaise version des composantes
- Fonctionnement général des serveurs d'application
- Fonctionnement général des serveurs de données
- Version des logiciels de travail
- Mise à jour du système d'exploitation des serveurs

1.1.2 Régler les problèmes

- Régler problème ou remettre à plus tard?
- Évaluation de l'impact de la solution
- Temps requis pour régler le problème
- Temps requis pour obtenir le détail des exigences et critères d'acceptation.
- Combien de temps cet item peut rester sans solution
- Capacité de l'équipe
- Les exigences sont –elles clairement exprimées?
- Qui doit clarifier les exigences?
- Qui est disponible pour clarifier les exigences?
- Quand est-ce que les exigences seront disponibles?

1.1.2.1 Régler les problèmes tout de suite ou les reporter

- ◆ Impact potentiel du problème sur la sécurité de l'application
- ◆ Temps nécessaire pour régler le problème
- ◆ Temps nécessaire pour recevoir les licences requises
- ◆ Durée dans laquelle le problème peut demeurer sans solution
- ◆ Le serveur sur lequel déployer la composante

Tableau 4.21(Suite)

- Les exigences sont –elles clairement exprimées?
 - Qui doit clarifier les exigences?
 - Qui est disponible pour clarifier les exigences?
 - Quand est-ce que les exigences seront disponibles?
- 1.1.2.1 Régler les problèmes tout de suite ou les reporter
- ◆ Impact potentiel du problème sur la sécurité de l'application
 - ◆ Temps nécessaire pour régler le problème
 - ◆ Temps nécessaire pour recevoir les licences requises
 - ◆ Durée dans laquelle le problème peut demeurer sans solution
 - ◆ Le serveur sur lequel déployer la composante
 - ◆ Disponibilité des développeurs
 - ◆ Charge de travail du jour
 - ◆ Impact de la solution sur le temps
 - ◆ Catégorie de report (pour la prochaine version ou pour plus tard)
 - ◆ Comment régler le problème?
 - ◆ Impact de la solution sur la sécurité
 - ◆ Impact de la solution sur les autres tâches
 - ◆ Niveau de risque associé au travail
 - ◆ Méthodes possibles
 - ◆ Sources possibles du problème
 - ◆ Historique des ratés de modification de la composante
 - ◆ Travail restant à faire
- 1.1.3 Faire de la maintenance
- 1.1.3.1 Déterminer la disponibilité des développeurs
- ◆ Charge de travail

Tableau 4.21(Suite)

1.1.3.2 Déterminer la disponibilité et la clarté des exigences

- ◆ Qui produit les exigences?
- ◆ Quand seront-elles prêtes?

1.1.3.2 Reporter les problèmes

- ◆ Type de problème
- ◆ Besoin d'un spike (investigation préliminaire)?
- ◆ Dans quel cycle?

1.1.4 Faire un travail de qualité.

- Investigation préliminaire avant la tâche?
- Besoin d'un spike?
- Check-in correct des codes
- Check-in dans quel répertoire? / Support ou branche primaire?
- Tâche exécutée correctement
- Les étapes à suivre
- Étapes complétées correctement?
- Localisation du code dans le répertoire des codes (TFS)
- Travail révisé par un pair?
- Travail testé?
- Analyse d'impact
- Qui fait quoi?
- Guide de l'utilisateur à jour?
- Régulation de l'industrie?

Tableau 4.21(Suite)

1.2 Déploiement

- Statut du package à déployer
- Numéro de « build »
- Fréquence des « builds »
- Date des « builds »
- Date des versions
- Fréquence des versions
- Date du déploiement
- Durée de l'exécution
- Nombre d'items compris à déployer
- Tous les items sont-ils approuvés
- Accord du client obtenu?
- Personnel sur le site le jour du déploiement
- Personnel de support en attente d'appel
- Plan de rollback

2.0 Livrer l'application à temps

- Date de livraison

2.1 Prioriser les tâches

- Meilleur ordre de réalisation?
- Interdépendance/séquencage des tâches
- Tâches à reporter
- Attribuer des tâches aux développeurs
- Compétence requises de la part du développeur

Tableau 4.21(Suite)

3.0 Performance de l'application

3.1 Fournir une application performante

- Trouver problèmes de performance potentiels
 - Rapporter des problèmes
 - Quels outils utiliser sans affecter l'application
 - A quel moment effectuer le test de performance?
 - Rapport de OPS
- Nouveau problème de performance
 - Problème de base de données ou de codes
 - Vitesse d'exécution de la requête
 - Moment d'exécution de la requête
 - Charge de l'application au moment de la requête
 - Jour de la semaine
 - Régularité du problème
 - Fréquence du problème
 - Expérience du développeur sur problème similaire

3.2. Régler problèmes de performance

- Compétence requises de la part du développeur
- Identifier la fonctionnalité non performante
- Scénario d'exécution
- Métriques
- Travail révisé par un pair?
- Travail testé?
- Analyse d'impact
- Taille des données de test

Elle montre les facteurs qui sont importants pour la qualité de l'information en équipes de développeurs, pas seulement en termes d'information de base nécessaire, mais également la façon dont cette information est utilisée, interprétée et analysée pour prendre des décisions et effectuer les tâches requises. Toute tentative visant à développer des technologies ou d'autres modifications permettant de supporter les développeurs doit directement agir sur l'exactitude de l'information ou permettre une facilité d'atteindre chacune de ces exigences.

4.4.4 Capture de l'état des ressources supportant la CS

4.4.4.1 Les ressources CS en personnel

Tableau 4.22 Moyennes de l'importance et des fréquences des communications

Ressource CS : personnel	Moyenne de l'importance	Moyenne des fréquences (%)
Superviseurs	3,2	14,25
FMK	2,5	1,50
ARCH	2,0	2,30
PO	1,0	5,00
PM	1,1	34,45
IT	3,0	<1,0
Autres DEV	1,0	26,20
AT	1,5	5,60
CM	3,0	10,14
INT	4,0	<1,00

La méthodologie d'enquête contextuelle nous a permis d'évaluer les ressources en personnel et les ressources technologiques utilisées dans l'organisation pour répondre aux exigences CS identifiées dans le tableau 4.21. Les estimations ont été faites par chaque individu interviewé sur la fréquence globale de communication qu'il effectue avec chacun des membres des autres équipes DEV et des membres du personnel des équipes de soutien. Chaque individu a assigné un nombre à chaque équipe avec laquelle

il interagit, pour refléter la fréquence globale de communication de chaque échange (sur un total de 100%).

Pour chacune de ces ressources CS, l'importance de la communication a également été évaluée sur une échelle de quatre points, où 1 représente une ressource très importante pour atteindre les objectifs de performance de l'équipe et 4 représente une ressource d'importance relativement faible. Les estimations de **moyenne** et de **l'importance** de fréquence de communication ont été déterminées pour chacune des unités de DEV.

Le tableau 4.22 présente les ressources CS en personnel, la moyenne de l'importance et la moyenne des fréquences pour les interfaces de communication du point de vue de DEV.

- Plusieurs personnels et les unités de travail ont été indiqués comme des ressources CS très importantes nécessaires pour accomplir le travail de DEV. Ce sont les autres DEV, les AT, les PO, suivis de près par les CM et les PM.
- La fréquence la plus élevée signalée de communication était avec les PM (34,45%), suivis par les autres DEV dont la plupart sont de la même équipe. Ensuite, vient les superviseurs, puis les CM (10,14%).

L'analyse des relations fait constater que les DEV ont une grande dépendance et vulnérabilité par rapport aux AT. Quelle que soit la bonne volonté des DEV, si les AT ne fonctionnent pas bien, leur travail sera affecté et le produit final ne sera pas performant.

Dans l'ensemble, une grande relation d'interdépendance a été trouvée entre les équipes et unités incluses dans cette étude. Chaque type de travail consiste à interagir avec entre 3 à 5 différentes équipes et unités pour obtenir (ou fournir) les informations nécessaires pour une bonne CS et un bon rendement au travail. En général, deux ou trois de ces interactions étaient considérés comme très importantes et constituaient la majorité des

interactions de chaque équipe ou unité. Cependant, pour la plupart des équipes, il y avait aussi de nombreux groupes avec qui les interactions sont seulement occasionnelles.

Les questions spécifiques concernant ces interactions qui peuvent avoir un impact significatif sur la CS ont été élucidées lors de l'étude. L'analyse et l'interprétation approfondie des relations de dépendance et des vulnérabilités des DEV sont présentées dans la section 4.4.7 (Discussion et interprétation des résultats).

4.4.4.2 Les ressources CS en technologies

En plus de déterminer les ressources en personnel utilisées pour la réalisation de la CS, les technologies utilisées au sein des unités de DEV ont également été examinées. Les principales technologies utilisées pour la transmission d'informations comprennent les systèmes suivants :

- Un système d'information pour la gestion des tâches et des activités de développement.

Ce système commercial présenté à la Figure 4.5, est doté d'une base de données centralisée et partagée par toutes les équipes. Il sert à répertorier les tâches, leur distribution aux membres de DEV et leur statut de progression. Chaque membre d'équipe peut donc se référer à cette ressource technologique pour avoir de l'information sur ce sur quoi travaille d'autres DEV et aussi sur le progrès de l'ensemble de l'équipe.

- Un système de monitoring de la performance des applications

Le système commercial **AddDynamic** est utilisé pour partager les informations sur les performances des différentes sections de l'application sur différents serveurs. Ces informations sont nécessaires pour les DEV car ces derniers s'y réfèrent pour faire des

tests de performance. Les résultats obtenus lors de ces tests sont partagés avec les « leads » développeurs de toutes les équipes de DEV. Le logiciel détecte et signale également les failles de sécurité éventuelles dans les applications en production. **AppDynamic** fonctionne en continu et enregistre les données de performance par date, par application, par section et pour chaque environnement. L'historique des statistiques générées peut être consulté au besoin. La figure 4.7 montre les équipements de communication donnant des informations en temps réel sur la performance du matériel et des composantes de l'application en environnement de production. Ces moniteurs sont disposés dans une allée centrale des bureaux où tous les développeurs peuvent les consulter.

- Un système de communication instantanée

Les développeurs disposent aussi de l'application de communication instantanée, Skype. Cette dernière est largement utilisée par le personnel situé au siège de l'entreprise et par celui qui est localisé à distance. Elle leur permet de travailler sur un même sujet, de communiquer l'état d'avancement des travaux, et de participer aux réunions de travail.

- Un système de messagerie

Le travail est également supporté par un système téléphonique; un système de courrier électronique géré par Microsoft Exchange et par la documentation technique qui peut être trouvée dans un format électronique sur l'application *Microsoft SharePoint*.



Figure 4.7 Les équipements de communication

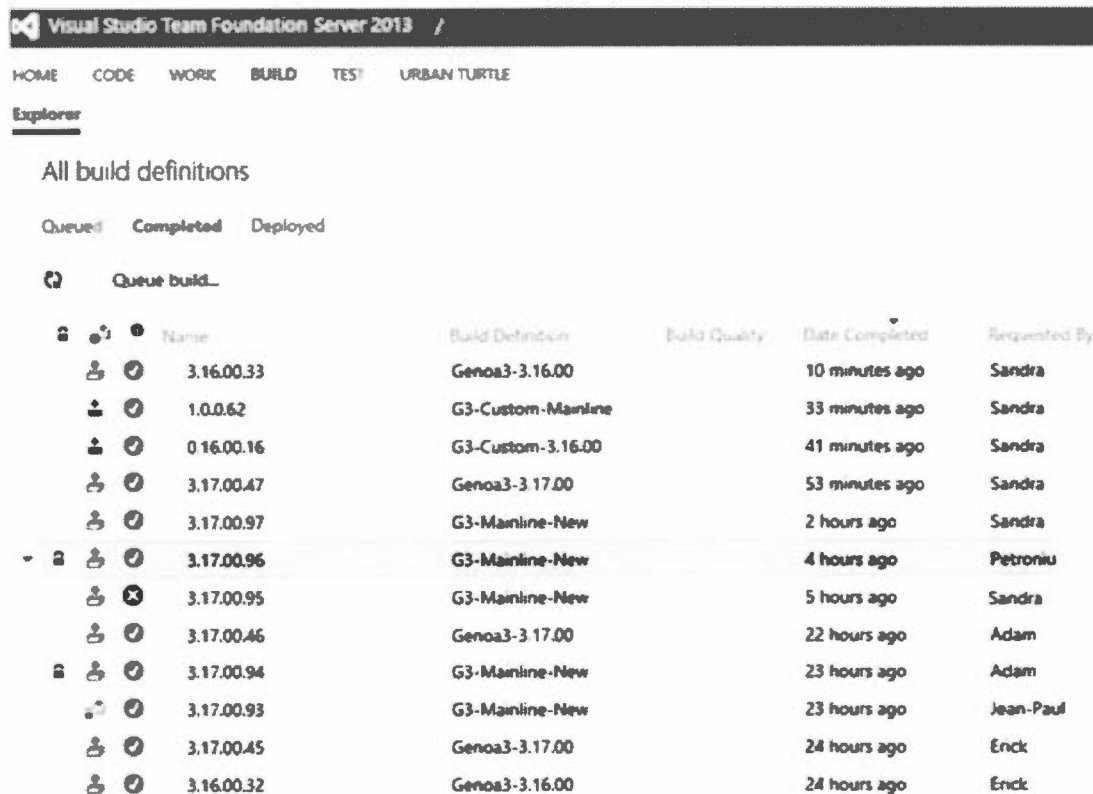
Photo prise en Juillet 2015 au siège de l'Entreprise à Montréal, Canada.

- Un système de gestion des déploiements

Enfin, l'application **Release Management** de Microsoft Visual Studio est utilisée pour gérer les déploiements des codes de l'environnement de test jusqu'en production. Release Management fournit une solution de déploiement continu qui rend les cycles de publication reproductibles, visibles et efficaces grâce à l'automatisation des déploiements. La figure 4.8 montre une vue de l'application de déploiement des « builds ».

Au moment de cette étude, nous n'avons pas relevé de problèmes majeurs de CS liés à l'utilisation de ces outils technologiques de productivité et de collaboration. Les rares

problèmes identifiés sont répertoriés dans la section 4.4.5 dédiée aux barrières et problèmes. Les recommandations impliquant ces technologies sont abordées plus tard dans le chapitre 5.



Name	Build Definition	Build Quality	Date Completed	Requested By
3.16.00.33	Genoa3-3.16.00		10 minutes ago	Sandra
1.0.0.62	G3-Custom-Mainline		33 minutes ago	Sandra
0.16.00.16	G3-Custom-3.16.00		41 minutes ago	Sandra
3.17.00.47	Genoa3-3.17.00		53 minutes ago	Sandra
3.17.00.97	G3-Mainline-New		2 hours ago	Sandra
3.17.00.96	G3-Mainline-New		4 hours ago	Petroniu
3.17.00.95	G3-Mainline-New		5 hours ago	Sandra
3.17.00.46	Genoa3-3.17.00		22 hours ago	Adam
3.17.00.94	G3-Mainline-New		23 hours ago	Adam
3.17.00.93	G3-Mainline-New		23 hours ago	Jean-Paul
3.17.00.45	Genoa3-3.17.00		24 hours ago	Enck
3.16.00.32	Genoa3-3.16.00		24 hours ago	Enck

Figure 4.8 Vue de l'application de déploiement des « builds »

4.4.5 Les barrières et problèmes

Au cours des entrevues, nous avons identifié de nombreux facteurs qui créent ou qui sont susceptibles de créer des obstacles à une communication efficace, et à une bonne performance. Ces facteurs sont des problèmes qui ralentissent ou freinent la performance et que les équipes de DEV doivent surmonter régulièrement afin d'atteindre leurs objectifs. Ils englobent des questions techniques, de ressources humaines, et organisationnelles. En général, la plupart des participants à cette étude

pensent que le système fonctionne bien. Cependant, presque tous pouvaient nommer quelques domaines où une amélioration serait possible. Les barrières et problèmes répertoriés sont énumérés dans le tableau 4.23.

Tableau 4.23 Les barrières à la CS

Barrières	Fréquence (sur 24 répondants)
Manque ou insuffisance de détails dans les travaux	15
Besoin d'avoir toutes les informations sur les exigences avant de commencer le travail	11
Indisponibilité de bonnes données pour compléter les tests	11
Des travaux de dernière minute	11
Des changements d'exigence en cours de cycle	11
Cloisonnement des équipes de DEV et travaux redondants	11
Incompréhension entre les équipes/confusion de rôle	11
Taux de rotation élevé du personnel	10
Longues attentes pour régler des problèmes simples	9
Non-respect des dates de livraison	8
Questionnement du niveau d'expérience du personnel	8
Des problèmes de performance de l'application non communiqués	7
Des projets en vase clos	7
Des interruptions fréquentes	6
Priorisation conflictuelle des travaux	6
Perte de temps pour corriger les données des clients	6
Manque de suivi pour certains sous-systèmes déployés en production	5
Des changements à l'implémentation technique ne sont pas communiqués	5
Les AT ne vérifient pas les items avant la mise en production.	5
Des interruptions de démonstration causées par des déploiements inattendus	5
Communication hâtive au client	5
Des changements dans l'industrie non communiqués aux analystes et aux QA	5

Tableau 4.23 (suite)

Des codes sans description envoyés en production pour exécution	5
Information insuffisante sur le statut des applications des unités de DEV	5
Communication inefficace à l'intérieur des équipes DEV	5
Management émotionnel	5
Des codes non testés	5
Les bogues non documentés	5
Mauvais usage de l'application	5
Des délais dans l'obtention des informations	4
Des déviations de solution	4
Cloisonnement des équipes de DEV et travaux redondants	3
Besoin de plus de rencontres entre les différentes équipes de DEV	3
Besoin de plus de formation sur les autres applications	2

- Insuffisance de détails dans les PBI

L'obstacle le plus fréquemment mentionné était celui du manque ou d'insuffisance de détails dans les PBI sur lesquels les DEV travaillent. Il s'agit ici des informations de spécifications nécessaires pour comprendre ce que demande le client, les envergures du travail à faire et les critères de test et validation utilisés par les QA pour déterminer si le travail qui est fait est acceptable ou non.

Selon la procédure de travail utilisée, l'on évite de faire en avance des travaux d'architecture non nécessaire. On se contente donc de faire des analyses conceptuelles et des architectures de façon évolutive et itérative. Dans un sens, cette façon de faire prévient le gaspillage de temps en empêchant de réaliser des travaux qui ne vont probablement jamais servir. On se concentre sur le strict nécessaire. Cependant, de l'autre côté, cette approche comporte des faiblesses. Les spécifications du travail à faire ne sont presque jamais claires dès le début du cycle malgré les bonnes intentions. Ceci a des impacts considérables sur l'avancement des travaux de développement et génère quelquefois des frustrations, puisque les DEV se retrouvent souvent bloqués dans leurs

travaux. Quelques fois, ils ne peuvent ni commencer leur travail, ni le terminer tant que les clarifications ne sont pas reçues des analystes (pour ce qui est des règles de gestion) et des architectes (pour ce qui concerne les exigences d'implantation technique). Cette situation génère une grande quantité de tâches non terminées en fin de cycle et oblige à les reporter pour être travaillées dans les cycles futurs.

- Indisponibilité des bonnes données pour compléter les tests

Pour effectuer des tests sur les livrables, les unités QA ont parfois besoin des données représentatives des différents scénarios pouvant valider les critères d'acceptation du travail. Souvent, ces données doivent être fournies par les AT qui doivent soit les générer eux-mêmes soit les recevoir des clients. Les DEV se plaignent que les données sont mises à leur disposition avec beaucoup de retard, ce qui retarde la livraison de leur travail. Ils souhaitent que les AT soient plus proactives et préparent les données au tout début du cycle, ce qui leur permettrait de concevoir rapidement les scénarios et plan de test.

- Incompréhension entre les équipes

Le personnel de DEV a également exprimé un certain degré de frustration en ce qui concerne d'autres équipes. Beaucoup ont estimé que d'autres groupes ne comprenaient pas vraiment ce qu'ils font. Par exemple, les DEV n'aiment que les PO leur disent comment faire leur travail, alors qu'ils sont ceux qui font face à la réalité technique du développement d'application. D'autre part, le personnel PO se sent incompris car ils avaient déjà travaillé dans le DEV pour la plupart. Ils pensent que ce sont eux qui ont les meilleures informations pour déterminer l'impact d'un problème donné sur le système (par exemple, les séquences d'accomplissement des tâches). Ils ne pensent pas que les DEV ont une vue globale de la situation.

La réalité en est qu'aucun groupe n'a une image complète des informations dont l'autre groupe dispose. Ces remarques sont aussi applicables à l'équipe de DEV vis-à-vis de

l'équipe de FMK pour l'utilisation des composantes produites par l'équipe FMK. Les DEV se plaignent que l'équipe de FMK leur dicte les composantes à utiliser dans leur solution. Ils pensent que lorsque les membres de l'équipe FMK veulent leur dicter ce qui va dans leur application, ils doivent d'abord s'assurer de bien comprendre le fonctionnement des applications et les besoins techniques des DEV.

Parfois les AT se substituent aux DEV en faisant des estimés des travaux de développement et en les communiquant directement aux clients avant l'avis des DEV. Ces derniers se plaignent que les AT ne comprennent pas l'importance de leur travail et qu'ils donnent des estimés irréalistes aux clients.

- Taux de rotation élevé du personnel

L'apparente instabilité de l'organisation était également une préoccupation importante. Durant la période de cette étude, il y avait de nombreuses réorganisations : acquisition de la compagnie par un compétiteur; des départs volontaires d'employés; des relocalisations de certaines personnes; la réduction volontaire des horaires de travail, et un grand rythme de recrutement pour remplacer les employés partis et pour répondre aux exigences d' enrôlement de nouveaux clients.

Cette situation a un impact sur le processus de production et sur la circulation de l'information puisque de nouveaux employés sont constamment injectés dans le système et que ces derniers ont besoin de temps pour comprendre le fonctionnement interne de l'entreprise.

- Longues attentes pour régler des problèmes simples

Les DEV ont exprimé le désir d'être en mesure de résoudre les problèmes localement dans leur équipe si seulement ils avaient les informations dont ils ont besoin et les autorisations pour les faire. Par exemple les DEV veulent être en mesure de déployer leurs codes sur l'environnement de test pour accélérer le processus sans attendre la file

d'attente des CM, ou être en mesure de mettre à jour un navigateur internet sans attendre l'équipe de IT. Ces requêtes apparemment simples prennent beaucoup de temps pour être traitées par les équipes qui en sont responsables. Cela occasionne des retards dans la livraison des tâches et des tâches qui sont reportées et repoussées d'un cycle à un autre.

En général, les DEV et les FMK trouvent que le processus de traitement de leur requête par les IT et les CM semble laborieux et n'est pas à leur avantage. Par exemple, la livraison d'un serveur de test par les IT, planifiée pour une semaine peut prendre plusieurs semaines sans raison apparente.

- Des problèmes de performance de l'application non communiqués

Les AT reçoivent de temps en temps des appels des clients sur des problèmes mineurs de l'application. Il semble que les AT aient leur liste de problèmes connus et qu'ils ont des réponses toutes faites aux clients. Ces problèmes de performance finissent par avoir des impacts non seulement sur la fonctionnalité demandée par le client mais peut affecter au même moment la performance de tout le système car la plupart de ces problèmes ne sont pas communiqués aux PO pour être priorisés et travaillés.

- Non-respect des dates de livraison

Il arrive que le développeur en charge d'un item particulier ne soit pas disponible en plein milieu du cycle. Le problème est que les items non travaillés sont simplement reportés pour le cycle suivant sans tenir compte des priorités ni des dates de livraison prévues initialement. Les développeurs savent globalement lorsque le cycle courant se termine mais ne savent pas quand est-ce qu'un item particulier dans le cycle doit être livré au client. L'information sur la flexibilité ou non des dates de livraison n'est pas systématiquement communiquée aux PM, ni aux DEV lead. Les items non travaillés sont simplement reportés ou sont travaillés précipitamment à la fin du cycle, occasionnant des retards sur d'autres items.

Le problème du non-respect des dates de livraison est aussi dû à l'injection d'un grand nombre d'items de dernière minute dans le cycle. Le travail sur ces items qualifiés d'urgents affecte toute la programmation des livraisons.

- Des travaux de dernières minutes et des interruptions fréquentes

En cours du cycle, les AT introduisent des requêtes de dernières minutes qui souvent, sont classifiées d'urgentes. Elles viennent des clients ou des AT qui oublient de les introduire en début du cycle. Pour gagner du temps, parfois, ces requêtes passent directement aux DEV sans l'avis des PO ni des architectes. De leur part, les DEV doivent interrompre des travaux en cours pour s'occuper des urgences formulées par les AT. Ceci constitue un bruit considérable dans le processus de travail. Il crée beaucoup de frustrations chez les DEV et les PO et occasionne des retards dans la livraison des tâches déjà planifiées.

Du point de vue des DEV, les priorités doivent venir d'une seule équipe, soit les PO. Les DEV trouvent cela confondant lorsque les AT interagissent directement avec eux pour changer les priorités à l'avantage de leurs seuls clients.

- Des changements d'exigence en cours de cycle

En cours de cycle, certains changements au PBI sont demandés par les AT. Il arrive que ces demandes soient directement renseignés dans les descriptions des PBI et remises au DEV pour codification. Parfois pour éviter que les PBI soient repoussés sur des cycles futurs, les AT procèdent au changement directement sans en informer les PO, ni les analystes. Le risque associé à cette pratique est que le PBI initial peut rapidement devenir très « gros » et prendre plus de temps à travailler. Cela a un impact sur les dates de livraisons des autres PBI déjà planifiés. Le système informatique de gestion des PBI offre des possibilités d'envoi de message d'alerte au PO en cas de mise à jour d'un PBI mais ces alertes sont souvent ignorées par ceux qui les reçoivent. Les

DEV ont exprimé le besoin que les AT révisent davantage les exigences en début de cycle lors des rencontres préparatoires avec les PO.

- Des activités et projets en vase clos

Certains projets sont développés en vase clos sans que les membres des équipes DEV soient mis au courant de ce qui se passe.

Des employés DEV sont engagés et il leur est assigné des projets particuliers sur lesquels ils travaillent durant de longues périodes sans que d'autres membres du personnel sachent exactement sur quoi ils travaillent. Cette situation n'est pas très appréciée par les DEV car les dits projets finissent par être intégrés dans l'application et ils doivent composer avec. Les DEV ont suggéré d'être mis au courant de tous les projets de développement qui se passent à l'interne afin d'en mesurer les impacts éventuels sur leur travail.

- Information insuffisante sur les applications livrées aux clients

Deux types de problèmes sont à signaler ici :

- Les DEV sont parfois surpris d'être appelés à intervenir d'urgence sur des problèmes causés en production par des tâches sur lesquelles ils ont eu à travailler précédemment. Ce qui surprend c'est qu'ils n'étaient pas au courant que leur travail a déjà été promu en production. Même si les DEV sont au courant des « builds » qui sont en production, ils en ignorent parfois le contenu exact. Ils aimeraient au moins connaître le contenu des « builds » et la date de mise en production. Cela leur permet de se préparer à une éventuelle intervention au cas où leur code serait source d'instabilité dans l'application.

- Les AT aussi veulent connaître le contenu des « builds » et les impacts potentiels sur l'ensemble de l'application, y compris chaque client. Ils n'ont pas besoin des termes trop techniques. Ce qui les intéresse, c'est de savoir si leur client est affecté ou non par la modification / amélioration apportée à l'application.
- Cloisonnement des équipes de DEV et travaux redondants

Le cloisonnement des équipes est tributaire de la manière dont les unités de développement sont organisées. Chaque unité DEV est distincte et autonome comme nous l'avons présenté plus haut. Un tel fonctionnement cause problème dans l'ensemble du fonctionnement des DEV. Nous avons plusieurs exemples sous cette rubrique :

- Un problème particulier sur la sécurité des différents modules de l'application est signalé au PO par l'équipe IT en charge des questions de sécurité. Chaque unité de DEV s'attelle sur une solution à son niveau dans le module dont il a la charge. A la fin, tous travaillent pour régler le même problème avec des ressources et des approches de solution différentes. Les DEV « lead » pensent que c'est une utilisation non optimale des ressources DEV.
- Le personnel QA dédié à l'équipe CLAIM travaille sur une technique particulière pour améliorer ses opérations de test d'application. Son homonyme dédié à l'équipe de BILLING travaille sur une technique similaire. Même si les 2 testeurs QA socialisent ensemble et se rencontrent quotidiennement, il n'y a aucune disposition dans le processus normal de production qui peut les amener à partager leurs connaissances et savoir-faire respectifs. Ce même problème se pose aussi au niveau du personnel CM

dédié au déploiement des « builds ». Ils travaillent chacun sur des solutions similaires sans jamais le savoir.

- Du côté des AT, ils demandent des fonctionnalités pour leur client. Puisque le système permet de telle configuration, des fonctionnalités sont implantées pour un client spécifique et ne le sont pas pour d'autres. L'on peut comprendre s'il est question de règle de gestion mais en fait il y a aussi des contrôles d'interface comme les *Advance lookup* et des *Date picker* qui peuvent être différents d'un client à l'autre
- Perte de temps pour corriger les données des clients

Les DEV se plaignent que les AT leur remettent des données erronées reçues des clients. Dans les équipes internes, il y a présomption que les données que les clients envoient aux DEV sont de bonne qualité. Lorsque les clients envoient des données à importer dans la base de données de l'application, parfois elles ne respectent pas les règles d'intégrité et les règles de gestion. Lorsque ces données sont remises aux AT, ces derniers les transmettent aux DEV sans validation préalable et sans connaître ni questionner la qualité. Les clients eux-mêmes ne testent nécessairement pas la qualité de leurs données avant de les envoyer à l'entreprise, ou ne comprennent pas très bien comment fonctionne l'application. D'un côté, les DEV comptent sur les clients et sur les AT pour recevoir des données de qualité. De l'autre, les clients comptent sur les DEV pour épurer les données avant la mise en production. À la fin, personne ne se sent responsable de la qualité des données. Les données sont ensuite intégrées dans la base de données telles que reçues et occasionnent beaucoup de problèmes d'inconsistance et rendent des fonctionnalités instables ou carrément inutilisables. Les DEV passent ensuite beaucoup de temps à nettoyer les données après les avoir mises en ligne.

- Des codes non testés

Il arrive quelques fois que des changements effectués aux codes par les DEV soient inclus dans les « builds » sans être testés par l'équipe de QA, ou des codes qui sont promus dans des environnements par erreur alors qu'ils ne le devraient pas. Ces codes finissent éventuellement en environnement UAT et Production sans jamais être vérifiés, ni contrôlés par qui que ce soit, ni par le client. Il y a forte probabilité que de tels codes introduisent des instabilités dans l'application. Lorsque découverts et rapportés par des clients ou par les AT, ces cas sont traités en urgence pour minimiser les impacts.

Il arrive aussi que des codes intentionnellement inclus dans un « build » pour un client particulier soient promus en production sans être testés par tous les AT. Généralement seul le AT responsable du client procède au test, croyant que l'impact du changement est circonscrit à son seul client. Mais en réalité, cette sous-estimation des implications du changement introduit par le « build », finit par affecter d'autres clients et éventuellement toute l'application.

- Les bogues non documentés

Les DEV se plaignent que les bogues ne sont pas suffisamment documentés. Les causes des bogues sont clairement identifiées dans les descriptions mais les informations sur le contexte d'utilisation ne sont pas suffisamment renseignées. Il s'agit ici de cas d'informations insuffisantes qui devrait permettre aux DEV de procéder aux modifications sans devoir à courir après les AT pour des compléments. Les DEV ont des difficultés à estimer avec exactitude le temps de travail nécessaire pour un item s'il n'est pas suffisamment documenté. La présence de ces types d'items dans la liste des attributions affecte les cédules de livraison.

Aussi, certains PBI, surtout les requêtes de maintenance de données, ne contiennent aucune information de contexte. Pour la plupart, ils ne sont que des directives

demandant à exécuter une tâche bien spécifique. Les DEV n'ont aucune information sur les raisons des demandes et comment les utilisateurs utilisent les fonctionnalités. Le tableau 4.24 montre un exemple typique de demande de maintenance de données envoyée par les AT aux DEV. Cette requête ne contient aucune information de contexte.

Tableau 4.24 Exemple de requêtes de maintenance de données

>There are 8 policies in the system that currently have incorrect. MiscCharge.ChargeDates. Below, you will see the MiscChargeIDs affected by this issue. Please update the MiscCharge.ChargeDate for the identified MiscChargeIDs.		
MiscChargeID	Current MiscCharge.ChargeDate	New MiscCharge.ChargeDate
826051	2011-11-01 00:00:00.000	2012-01-31 00:00:00.000
830383	2011-01-10 00:00:00.000	2011-02-09 00:00:00.000
833805	2011-01-14 00:00:00.000	2011-02-13 00:00:00.000
834181	2010-07-15 00:00:00.000	2011-09-15 00:00:00.000
834709	2011-04-16 00:00:00.000	2011-05-15 00:00:00.000
834716	2010-06-16 00:00:00.000	2010-07-15 00:00:00.000
835408	2010-12-18 00:00:00.000	2011-01-17 00:00:00.000
836430	2009-11-20 00:00:00.000	2010-11-20 00:00:00.000

- Mauvais usage de l'application

Pour contourner certains problèmes de l'application, les AT s'entendent directement avec les clients pour implanter des solutions rapides. Ces solutions vont parfois à l'encontre de l'usage normal de l'application et ne sont pas validées par les architectes, ni par les DEV « leads ». Souvent, ni les problèmes originaux, ni les solutions alternatives proposées ne sont pas communiqués aux PO pour être travaillés. Le seul

objectif de ces raccourcis de solution n'est que de sauter les étapes et éviter de mettre le client en fil d'attente. Si nous prenons un exemple concret de catégorisation d'entité fonctionnelle, au lieu de créer des sous-types pour identifier une entité fonctionnelle dans l'application, les AT conseillent aux clients d'utiliser d'autres attributs pour créer des intervalles de valeurs numériques qui serviront d'attribut de catégorisation. L'introduction d'une telle règle fait que les valeurs comprises par exemple, entre 1 et 10.000 sont affectées aux entités de sous-type 1 et les valeurs de 10.000 et + pour des entités de sous-type 2. La réalité est que de telles nouvelles règles ne sont connues que par les AT. Pour les mêmes problèmes, il y a autant de solutions que de AT. La conséquence est que des rapports de gestions sont affectés et ne reflètent plus les exigences. Lorsque de tels problèmes sont découverts, ils occasionnent beaucoup de temps de maintenance des données et affectent la qualité des données et du système.

- Des délais dans l'obtention des informations

Les développeurs passent beaucoup de temps à attendre le travail des autres. Il peut s'agir de simples demandes de clarifications nécessaires pour continuer leurs items. Lorsque les clarifications n'arrivent pas à temps, les travaux des DEV sont retardés et sont reportés. Ces reports peuvent parfois s'échelonner sur plusieurs cycles. Par exemple, dans l'équipe BI, on attend des clarifications des règles de gestion pour continuer le travail de modélisation de données. L'information doit venir des PO, et les PO doivent la recevoir des AT et des clients. Ces derniers ne sont pas nécessairement disponibles en tout temps car ils ont d'autres priorités. Tout ceci crée des délais indésirables qui se répercutent sur les délais de livraison. La cadence des activités se trouve donc grandement affectée.

Parfois, les développeurs laissent tomber la tâche en question et passent à autre chose. Dès fois ils cherchent eux-mêmes les informations par d'autres sources comme des anciennes règles de gestion, ou auprès des développeurs seniors, mais ne peuvent pas toujours les implanter sans les valider auprès des parties prenantes, PO et AT.

- Des déviations de solution

Il n'y a pas de règles assez claires quand un problème d'application peut être résolu par une mise à jour de version ou simplement par un travail de maintenance des données. La substitution des mises à jour de version par des scripts de maintenance est un problème qu'on retrouve un peu dans toutes les unités de développement. Lorsque le risque d'implantation d'une solution dans l'application est élevé ou lorsqu'elles manquent de temps pour implanter une solution complète et solide, les équipes de DEV préfèrent corriger le problème par un script de maintenance. Ces « scripts » déployés en production s'exécutent automatiquement selon une cédule. De cette façon, les conséquences des défaillances de l'application sont détectées et corrigées en arrière-plan. Même si cette approche de solution permet de sauver du temps, elle ne fait que reporter la vraie solution à plus tard. Lors de cette étude, nous avons dénombré des dizaines de scripts de ce genre encore actifs en production. Le risque est que, s'ils ne sont pas pris en charge convenablement dans le processus, ils sont oubliés et continuent d'opérer avec des règles de gestion obsolètes.

- Manque de suivi pour certains sous-systèmes déployés en production

Normalement les sous-systèmes fonctionnant en environnement de production bénéficient d'une attention particulière lorsqu'il y a problème les concernant. Cependant, nous avons observé qu'il y a un certain nombre de sous-systèmes qui manquent de support ou pour lesquelles il est difficile de trouver un développeur pour intervenir lorsqu'il y a un problème. Il s'agit généralement de projets déployés il y a plusieurs mois ou plusieurs années et dont les développeurs en charge sont aujourd'hui affectés à autres tâches ou ont simplement quitté l'entreprise. Lorsqu'il y a problème avec de tels sous-système, les équipes de support passent énormément de temps à localiser l'équipe qui a initialement travaillé sur le projet. L'équipe de support peut passer plusieurs jours ou même des semaines de recherche pour se rendre compte finalement que personne de ceux qui ont travaillé sur le projet n'est disponible. Ensuite

vient l'étape d'affectation d'un développeur sur la solution. Ceci peut également prendre plusieurs jours. Cette situation est source de nombreuses frustrations dans l'équipe CM, affecte la performance de toutes les équipes et la qualité du service rendu au client.

En raison du taux de rotation du personnel très élevé dans l'équipe CM, certains sous-systèmes en production manquent de support. Lorsqu'il y a problème avec ces sous-systèmes les membres de l'équipe de support qui sont supposés intervenir en première ligne n'ont pas toute l'information en leur main pour faire un diagnostic correct. Parfois, ils n'ont pas de système d'information interne pouvant les aider dans leur investigation pour comprendre les causes. Cette situation a un impact direct sur les DEV, puisque dans ces cas, ces derniers sont encore appelés à poser le diagnostic de premier niveau et à décider si une solution immédiate est nécessaire ou non.

- Des changements à l'implémentation technique ne sont pas communiqués

Il arrive qu'en cours de cycle, les architectes apportent des modifications conceptuelles à la manière dont une solution particulière doit être implémentée. Si ces modifications ne sont pas communiquées à temps aux DEV, ceux-ci effectuent leurs travaux avec les spécifications initiales. Lorsqu'ils sont mis au courant de ces nouvelles propositions des architectes, la solution initiale a déjà été implantée. Dans certains cas, la solution est déjà livrée dans l'environnement QA pour être testée. Cette situation dérange les DEV qui se plaignent de devoir retravailler leur solution après sa promotion sur QA. Ils préfèrent que les architectes fassent parvenir leurs suggestions à temps au tout début du cycle afin de réduire le travail en double.

- Les DEV se plaignent que les AT ne vérifient pas les items avant la mise en production.

Les items sont travaillés par des DEV et déployés en UAT, puis en production pour usage des clients sans qu'ils ne soient vérifiés par les AT. Une fois mis en production,

des erreurs sont identifiées et rapportées aux DEV par l'équipe de support. Les DEV se plaignent que les erreurs leurs soient rapportées de cette façon. Ces erreurs auraient dû être détectées bien avant, en environnement UAT ou QA si tous les chargés de comptes avaient pris le temps de tester les fonctionnalités. Quoique formellement indiqué dans le processus, les AT ne se sentent pas concernés de faire des tests de vérification si l'item n'est pas particulièrement développé pour leur client. Cependant, ils ignorent qu'un changement survenu dans une partie de l'application pour un client en particulier peut avoir des impacts indésirables sur toute l'application.

- Des changements dans l'industrie non communiqués aux analystes et aux QA

De temps en temps, il y a des changements dans les réglementations de l'industrie de l'assurance qui doivent être pris en compte par les équipes de développeurs de logiciel œuvrant dans ce secteur. Cependant, il arrive que les informations sur ces changements ne parviennent pas à toutes les équipes de l'entreprise au même moment et cela affecte le bon déroulement des travaux. Par exemple, le test « Compliance scan » qui donne aux systèmes informatiques la capacité de détecter les clients mis en quarantaine pour diverses raisons, telles que terrorisme ou blanchiment d'argent, ne s'effectue pas avec de bonnes données. Au moment de cette étude, certains membres de QA détenaient encore une liste non à jour de ces entités alors que les DEV en ont une autre. Les QA effectuent des tests avec cette liste et rejettent les travaux des DEV sur cette base. Les DEV passent ensuite beaucoup de temps à investiguer les problèmes reportés pour finalement se rendre compte de la divergence dans les données. Il s'agit ici d'une situation d'ambiguïté de l'information qui fait perdre du temps aux développeurs. Les DEV suggèrent que les QA et les AT s'informent davantage sur ce qui se passe dans l'industrie en termes de régulation et de rendre ces informations disponibles à tous et en même temps.

- Communication hâtive au client

Il arrive que l'ajout de nouvelles fonctionnalités soit communiqué par erreur aux clients par les AT, bien avant que le déploiement de production n'ait lieu, et avant que le manuel d'utilisateur ne soit complété. Bien que ce fait ne soit pas directement lié à la CS des équipes de développeurs, il constitue tout de même une situation embarrassante qui perturbe le travail des DEV dans une certaine mesure. Normalement les communications sur les nouveautés dans l'application devraient être synchronisées avec la sortie du guide de l'utilisateur et le déploiement effectif en production.

- Des interruptions de démonstration causées par des déploiements inattendus.

Les AT se plaignent que les DEV initient des demandes de déploiement exécutées par les CM en plein milieu d'une démonstration des fonctionnalités aux clients.

Lorsque les AT procèdent aux démonstrations de l'application, il est important pour eux que la séance ne soit pas perturbée par des mises à jour de l'application. Malheureusement, des mises à jour surviennent pendant les démonstrations et causent un malaise pour les AT devant les clients. Les AT estiment que les DEV sont au courant de leur cédule de démonstration et qu'ils ne devraient pas planifier des déploiements dans ces périodes. Pour leur part, les DEV pensent que les AT ne doivent pas planifier leur démonstration pendant qu'ils sont actifs dans le cycle ou pendant que l'application est en reconstruction. Ils estiment que la communication sur les dates de démonstration n'est pas suffisamment claire. Parfois, la perturbation signalée par les AT peut provenir d'une autre sous-application connectée, qui injecte des données ou qui est appelée par la première.

Les DEV relèvent que les avertissements de démonstration ne leur parviennent pas correctement ou ne les parviennent pas du tout. Ils veulent être au courant s'il y a des changements dans les dates de démonstration ou des changements concernant les fonctionnalités à démontrer. Toujours est-il que cette situation occasionne des

frustrations de part et d'autre et ne donne pas une bonne image de la compagnie devant les clients.

- Les OPS se plaignent des codes sans description que les DEV leur envoient pour exécution

De temps en temps pour régler certains problèmes urgents en production ou en environnement UAT, les DEV développent des solutions rapides et demandent à l'équipe OPS de procéder soit à l'exécution immédiate ou la programmation de l'exécution. Cette façon de procéder ne convient pas aux OPS car souvent ces codes développés hâtivement ne contiennent aucune description et sortent ainsi du processus règlementaire de planning-développement-test. Ces genres de solution n'ont pas non plus d'historique de test. Lorsque ces solutions créent d'autres problèmes en production, les OPS sont les premiers à recevoir des plaintes des clients. Ils aimeraient avoir suffisamment d'information sur ce que fait exactement ces codes spéciaux et comment faire des diagnostics en cas de problème. Puisque de toute façon ces types de solution Ad hoc ne peuvent pas être évités, les DEV doit prendre l'habitude de les documenter suffisamment afin de ne pas créer des frustrations au personnel de l'équipe OPS.

- Information insuffisante sur le statut des applications des unités de DEV

Les équipes de DEV et les AT ont manifesté l'urgence de recevoir les informations sur les dernières versions des sous-systèmes produits par toutes les unités de DEV. Ils aimeraient connaître l'état exact des applications et comment interagir avec elles. Un changement dans une application peut affecter d'autres applications puisque toutes les applications sur lesquelles travaillent les unités DEV sont connectées entre elles par le biais de web services. Par exemple, les membres de l'unité de CLAIM se plaignent que ceux de l'unité de POLICY font des changements dans leur application sans les tenir au courant. Ces changements affectent leur travail et ils aimeraient savoir d'avance ce

que les autres prévoient inclure dans leur cycle et qui éventuellement pourrait affecter le leur. Aussi, lorsque les AT font leurs tests d'acceptation, ils aimeraient savoir quelle version d'application est compatible avec laquelle.

Nous avons remarqué qu'il y a une grande interdépendance entre les unités de DEV au sein de la grande équipe des développeurs pour les échanges de données entre leurs sous-applications. Cette dépendance semble être mal évaluée par chacune des équipes et constitue actuellement un problème majeur qu'elles tentent de résoudre.

- Priorisation conflictuelle

Les PO attribuent des priorités à des PBI qui sont inclus dans la liste à travailler dans le cycle. Cette priorisation vient normalement des PO et des AT. Il arrive que les priorisations conçues par les AT et celles conçues par les OP soient en conflit. Lorsque les AT se rendent compte que leurs items ne sont pas travaillés, ils font pression pour que leurs items soient inclus dans la liste. Ceci dérange les DEV qui doivent arrêter ce qu'ils font pour s'occuper d'autres choses. Lorsque les DEV passent beaucoup de temps à investiguer un problème, ils ne sont pas contents d'interrompre leurs travaux pour passer à autre chose. Ils estiment qu'ils perdent du temps lorsque de pareils conflits surviennent. Ces conflits de priorisation sont également sources de report. Les AT de leur part, se plaignent que leurs items ne sont pas inclus dans les priorisations alors que cela le devrait. Ils pensent que les PO ne respectent pas leur priorité.

- Communication inefficace à l'intérieur des équipes DEV

A l'intérieur de la même équipe, les « leads » procèdent parfois à des nettoyages des bases de données afin de récupérer de l'espace disque, ou ils font des changements de configuration sur des serveurs de DEV, ou font des modifications de ressources partagées et oublient d'en informer les membres de leurs propres équipes. Même si en général de tels changements causent peu de problèmes à l'application, ils sont néanmoins source de perturbations et occasionnent des pertes de temps.

- Communication inefficace entre les PM et les DEV

Certains chefs d'équipe de DEV ont révélé qu'ils ne sont pas émis au courant par les PM, des cédules des déploiements en environnement de production. Cette opération est souvent planifiée et exécutée entre les PM, les AT et l'équipe OPS et en totale opacité pour les DEV. Les DEV souhaitent savoir lequel de leur item travaillé est promu en production afin de se préparer à intervenir au cas où la fonctionnalité développée ne fonctionne pas tel que prévu.

- Le niveau d'expérience du personnel

Le faible niveau d'expérience de certains membres du personnel a été décrit comme un problème. Cela s'explique par le fort taux de rotation du personnel en raison de nombreux recrutements et départs. Les individus les mieux formés ne pouvaient pas être tout le temps disponible pour travailler avec leurs collègues pour répondre aux attentes des clients. Avec la pression que l'entreprise reçoit des clients concernant les délais de livraison, cette situation a des impacts sur la performance et sur l'atteinte des objectifs de l'entreprise.

- Question de management

Les questions d'organisation ont également été mentionnées. En particulier, il semble que la haute direction doit quelques fois intervenir auprès des gestionnaires de projet avant qu'ils ne traitent certaines requêtes qui autrement resteraient sans réponses bien au-delà des délais prévus par le processus. Cette façon de faire, fait croire à un système orienté sur un mode de gestion émotive. Lorsque la direction intervient, les requêtes sont exécutées. Lorsqu'elle n'intervient pas, les choses trainent sans raison apparente. Le personnel de DEV a exprimé le désir d'une meilleure organisation dans laquelle le processus est équitable pour tous. En attendant un tel ajustement, ils se contentent de se référer à la direction générale pour que leurs requêtes soient déplacées vers le haut de la liste d'attente et pour être traitées rapidement.

- Perte de temps dans la saisie des fiches de temps.

Sur le plan technologique, les DEV paraissent bien outillés et ont révélé très peu de problèmes mis à part celui des saisies redondantes des données sur les heures travaillées dans le système de gestion des temps et dans le système de gestion des PBI. Les DEV estiment qu'ils perdent beaucoup de temps dans ces saisies multiples. L'intégration de ces systèmes commerciaux leur sauverait du temps.

- Des décisions prises en équipe mais non exécutées

Lors des rencontres d'évaluation de Sprint, et de rétrospectives, plusieurs décisions sont prises par les participants mais la plupart de ces décisions n'est pas exécutée car personne ne se sent vraiment responsable de son application. Chacun préfère se vaquer à ses occupations du moment. Certains développeurs estiment perdre du temps dans ces rencontres.

4.4.6 Validation des données sur les besoins et les barrières à la CS

La liste des problèmes répertoriés dans le tableau 4.23 (Barrières à la performance), exprime à la fois le comportement et les intentions des acteurs du système. La version originale de la liste a été présentée aux parties prenantes pour discussion et validation. Chaque partie a pu exprimer ses préoccupations par rapport à l'appréciation que les autres ont fait des relations qu'elles entretiennent. Nous avons pris en compte tous les commentaires et observations dans les versions consécutives du tableau. La version que nous présentons ici est celle qui a été validée par tous.

4.4.7 Discussion et interprétation des résultats.

4.4.7.1 Les besoins des équipes de développement en CS

Sur la base de l'analyse des besoins et des ressources CS pour chaque unité DEV, les obstacles et les problèmes exprimés, plusieurs observations et interprétations peuvent être faites sur la CS d'équipe dans le domaine de développement d'application.

Globalement, il ressort qu'il y a effectivement des problèmes pour la CS d'équipe puisque nous avons relevé la présence d'un certain nombre de lacunes dans le fonctionnement des équipes autant dans les comportements individuels des membres que dans celui de l'équipe. Ces lacunes peuvent être le résultat d'objectifs incompatibles; de confusion de rôle; de chevauchement de rôle; de manque d'informations nécessaires de la part d'un ou de plusieurs parties; de manque de compréhension de l'information exacte dont l'autre groupe a besoin, ou des interprétations différentes de l'information qui est transmise d'un groupe à un autre. Ces lacunes sont sources de situations d'erreur qui entravent l'atteinte des objectifs de toute l'entreprise. Tout comportement ou problème qui empêcherait l'entreprise d'atteindre les objectifs qu'elle s'est fixés constitue une situation d'erreur et agit sur la performance.

Les développeurs font face à plusieurs défis pour répondre à leurs exigences CS qui sont aussi liés à la CS de l'équipe.

- Confusion de rôles et d'objectifs.

Selon la définition de leur rôle et attributions, les AT n'ont pas les compétences nécessaires pour évaluer l'impact d'un bug particulier sur l'application. Cela ne fait pas partie de leur attribution. Ils doivent tout communiquer aux PO qui se chargeront de le faire évaluer par les architectes et les « leads ». Cette situation affecte également la CS

des DEV, puisque les évaluations faites par les AT sont souvent incorrectes et occasionnent des accumulations de travail en fin de cycle du côté des DEV.

- Manque d'informations ou informations inexactes nécessaires au travail des équipes

Tout d'abord, les développeurs passent beaucoup de leur temps et ressources à déterminer s'ils ont les bonnes spécifications ou quand et comment ils vont obtenir les bonnes spécifications. Un écart considérable existe entre les développeurs et les AT qui, parfois peuvent fournir de mauvaises spécifications, des spécifications inexactes ou des spécifications en retard. Parfois, les spécifications ne sont simplement pas suffisamment documentées au moment du démarrage des travaux par les DEV pour toutes sortes de raisons. Ces situations augmentent à la fois la probabilité d'erreur (l'implémentation des mauvaises règles de gestion) et peuvent conduire à des déficiences considérables, aux pertes de temps et des retards de livraison.

Lorsque les exigences ne sont pas suffisamment renseignées ou non disponibles, les développeurs doivent fréquemment impliquer leurs « leads » et les PO. Cela nécessite parfois l'implication de plusieurs autres équipes et du personnel, qui tous doivent être mis au courant de la situation pour prendre une bonne décision. Ce processus - qui sort du cycle normal- prend du temps et peut être sujet à des erreurs de mauvaise communication, conduisant à des problèmes de CS.

- Problèmes d'interprétations des signaux et de projection de l'état du système

Les règlements ne sont pas très clairs concernant les critères précisant quand un item doit être codé dans l'application ou être substitué par un script de maintenance. Les développeurs ont une discrétion de faire une codification ou introduire un script de maintenance. Même s'il peut être à la fois plus sûr et plus rentable de promouvoir ce type d'action dans certaines circonstances (par exemple, si les impacts d'une codification sont grands sur un autre sous-système ou manque de temps), cette pratique

peut facilement mettre à mal le système car elle est source d'erreur, puisque les règles de gestion qui sont implantées dans des scripts sont souvent obsolètes et difficile à maintenir.

Un meilleur partage de l'information est nécessaire à l'égard de ces questions afin que les développeurs, les AT et les PO puissent évaluer les impacts opérationnels et prendre des décisions qui sont en ligne avec les priorités et les réalités organisationnelles définies.

Par exemple, la connaissance au tout début du cycle, de l'impact d'un item, peut aider à dégager les ressources adéquates pour le travail ou simplement le prioriser pour le futur (Niveau 3 CS). Aussi, le client devrait en être informé à temps pour éviter les travaux de dernières minutes.

- Le report des travaux

Le système ne semble pas être actuellement optimisé pour éviter le report. Cependant, le processus de report pose un problème important. Les développeurs peuvent passer une quantité considérable de temps à investiguer une composante, se rapprocher de la solution pour arriver à un diagnostic, seulement pour trouver qu'ils ne peuvent pas implanter la solution en raison d'une indisponibilité de temps, des risques trop grands, des contraintes d'ordre technique ou architectural, ou des changements dans les spécifications. Ce processus est jugé assez inefficace et très frustrant car il crée des situations d'incomplétude de la résolution des problèmes pour lesquelles les DEV sont formés pour résoudre.

Le report est également lié à l'insertion des items de dernières minutes qui crée des interruptions dans le travail des DEV en modifiant leur priorité. Les développeurs deviennent très découragés quand ils ne sont pas autorisés à régler les choses sur lesquelles ils ont commencé à travailler. C'est aussi un gaspillage de temps et de

ressources humaines d'avoir à faire des « rollback » de code et à reporter le PBI de sorte qu'ils puissent être travaillés plus tard par la même équipe ou par une autre équipe.

Une compréhension commune des compromis des coûts et avantages dans la résolution des choses dans le cycle courant versus des résolutions plus tard permettrait aux intervenants d'avoir une meilleure compréhension des situations qu'ils rencontrent et de prendre de meilleures décisions.

- Transfert de connaissance / modèle mental

Le taux élevé de rotation du personnel est à l'origine de plusieurs problèmes de coordination et de retard de livraison. Cette situation a particulièrement un impact sur un des objectifs clé des DEV, celui de la livraison des applications dans les délais.

Les changements fréquents de responsabilité et l'affectation de nouveaux employés à des postes clés ne favorisent pas la continuité dans la fourniture des services et ressources de qualité nécessaires aux DEV pour accomplir leurs tâches. Cette situation est aggravée par le fait que dans certains cas, les départements de support au DEV n'assurent pas un bon transfert de connaissance entre les nouveaux et les anciens employés dans leurs équipes.

La capacité de projection de l'état du système peut aider les responsables d'équipe à anticiper les choses et à mettre en place un bon programme de formation pour les nouveaux employés. Cela permettra d'acquérir un bon modèle mental du processus d'ensemble et les projets en cours d'exécution.

- Besoins de communication inter-équipes

Bien qu'ils fournissent des informations sur les progrès de leurs propres tâches vers le sommet de façon continue, les PO et les PM ne fournissent souvent que peu ou presque pas d'informations vers les développeurs sur les autres équipes. Les PO ne réalisent pas

que les développeurs aient vraiment besoin d'informations sur ce que font les autres équipes. Par exemple, l'équipe de CLAIM n'est pas nécessairement au courant du progrès de l'équipe de BILLING alors qu'il semble avoir une grande dépendance entre ces 2 équipes. Ils ne sont pas au courant des activités en cours dans d'autres équipes qui peuvent avoir un effet sur leurs propres tâches (ou vice-versa). Sans cette connaissance, cependant, les développeurs n'ont aucun moyen de travailler efficacement sur les sous-systèmes de partage de données entre les applications. Dans certains cas, des tâches peuvent affecter les activités d'autres développeurs des autres équipes de manière qu'il génère des travaux en double ou occasionne des retards de livraison, sauf si les deux parties prennent des mesures de précaution. Ainsi, un manque d'information à jour sur ce que font les autres équipes DEV et sur les progrès réalisés contribue aux lacunes de la CS au sein des équipes de développeurs.

La satisfaction des besoins de communication inter-équipes peut résoudre les problèmes de cloisonnement des équipes que nous avons révélés précédemment. Ce problème génère beaucoup d'apprentissage en double. Les mêmes erreurs sont répétées dans plusieurs équipes et les connaissances et informations ne sont pas bien partagées entre équipes homonymes. Par exemple, le QA de l'équipe BI et celui de BILLING utilisent parfois des approches similaires et règlent certains problèmes de procédures ou d'outils de travail mais ne disposent pas de procédure formelle de partage de connaissances.

- Besoins de formation sur les aspects cognitifs

Le personnel a les qualifications nécessaires pour les tâches qui leur sont confiées. Ils bénéficient également des formations techniques à l'interne. Un programme spécial de formation technique est conçu à l'intention de chaque groupe d'employé. Il est communément admis auprès de la direction de l'entreprise qu'un niveau élevé de compétence technique prédit la capacité du personnel à produire des applications dans les délais, des applications performantes et sécurisées. Dans l'entreprise, il ressort que

la formation des compétences techniques est souvent considérée comme la stratégie la plus importante pour réduire les événements indésirables dans le processus de production.

A la suite des investigations de différents incidents qui sont survenus en environnement de production au cours des 12 derniers mois, nous avons constaté que le personnel impliqué était des seniors. Que ce soit les développeurs, les gestionnaires de compte, les gestionnaires de projet ou les spécialistes de configuration, tous avaient une expérience de haut niveau. De toute façon, seule cette catégorie d'employé est autorisée à déployer des applications en environnement de production.

Cependant, il apparaît que la formation sur les meilleurs outils et pratiques de développement ne visent qu'à produire des professionnels compétents dans leurs milieux respectifs. A ce jour, l'examen que nous avons fait des programmes d'enseignement fait croire qu'aucun effort n'y est mis pour y intégrer les facteurs cognitifs. Mais comme il ressort de l'analyse des incidents en environnement de production, les problèmes répertoriés sont en fait plus susceptibles de résulter d'erreurs de perception que d'incompétence technique. Les incidents sont souvent causés par des erreurs de jugement, de négligence, de compréhension incomplète de la situation, d'échec de vigilance et de perceptions erronées même si les compétences techniques des employés impliqués sont d'un niveau élevé.

Le programme de formation à l'endroit des employés doit inclure les aspects non techniques, notamment, du contenu qui peut aider le personnel à mieux comprendre et interpréter les signaux de leur environnement de travail.

- Inadéquation de la méthodologie de travail à la complexité du travail à charge

Au chapitre 2, nous avons présenté les lignes du manifeste Agile, ces principes de base qui constituent le socle même de la méthodologie Agile adoptée par l'Entreprise. Au regard des problèmes et obstacles à la CS soulevés dans ce chapitre, il nous apparaît

que l'application stricte de ces principes a un lien avec l'état actuel du fonctionnement des équipes. Nous avons relevé les concordances suivantes entre les problèmes soulevés et les principes Agile :

- Alors que plusieurs méthodologies traditionnelles recommandent d'impliquer le client de préférence dans la phase d'analyse et de conception, Agile va plus loin et encourage de travailler avec le client dans tout le cycle de vie de l'application : c'est le principe de « *Coopération étroite et quotidienne entre les parties prenantes et les développeurs* ». Dans l'Entreprise, les clients sont représentés par les clients internes dont le rôle officiel est d'aider les développeurs à affiner les besoins tout au long du projet par la communication. Les implications de la mise en œuvre d'une telle règle sont des interruptions fréquentes des développeurs au moment où ceux-ci ont besoin de concentration pour effectuer leur travail. Les conséquences liées aux interruptions ont déjà été relevés dans ce chapitre.
- L'esprit de documentation est presque absent de la gestion de projet en Agile. L'on encourage le développement rapide du logiciel au détriment de la documentation. Dans l'Entreprise, la documentation existante dépasse à peine ce qui est enregistré dans la description des tâches. Pour des équipes où l'on gère une centaine d'histoires travaillées par itération, la quantité d'information générée peut croître très vite. En plus, si on considère que pour chaque histoire travaillée, il peut y avoir plusieurs tâches associées, la recherche et le recoupage des règles d'affaires par histoire deviennent très laborieux. Cette situation constitue une barrière importante à la CS et au développement d'un model mental partagé dans ce sens que les employés novices ne peuvent que se rabattre sur la mémoire des plus anciens pour apprendre sur les pratiques de l'Entreprise.

- Le problème d'insuffisance de documentation est renforcé par un autre principe très utilisé dans l'entreprise celui de « *La conversation en face à face est la meilleure forme de communication* »
- Tel que recommandé par Agile, les logiciels sont livrés à une fréquence de 2 à 4 semaines. Ceci fait qu'à plusieurs occasions les logiciels sont livrés hâtivement sans avoir été suffisamment testés. Malgré les tests effectués à l'interne, quelques imperfections subsistent et ne sont découverts que par les clients en environnement de production. De toute évidence, la cadence de livraison des différentes versions des logiciels est clairement trop rapide quand on considère la grande taille et du logiciel et la quantité des développeurs impliqués dans sa mise en œuvre.
- Nous avons souligné le problème de report des travaux. Les développeurs passent beaucoup de temps sur des projets et qui finalement ne sont jamais déployés en production pour diverses raisons. Lorsqu'on est finalement prêt pour les déployer, soit les règles ont déjà changé ou les employés qui les avaient développés ont changé de rôle. Dans ces cas, il n'y a pas d'autres solutions que de reprendre le travail en entier. Cette tendance à livrer le logiciel rapidement et à autoriser des changements même en développement tardif, fait que certains travaux sont entamés avec très peu de connaissances sur le design et les estimations sont souvent en deçà de la réalité. Comme conséquence, les projets prévus pour un Sprint peuvent parfois s'échelonner sur deux ou trois, voire plus de Sprints, pour finalement être repoussés par d'autres projets jugés plus importants.
- L'autogestion des équipes fait que s'installe une tendance à l'isolement technique des équipes. Nous avons évoqué l'exemple de différentes équipes QA qui ont mis en place des routines similaires au lieu de se concerter pour un travail collaboratif. Des redondances de ce genre occasionnent des pertes de

temps pour la compagnie. La compagnie devrait inscrire une résolution dans ses décisions stratégiques à ce propos.

La méthodologie de développement Agile a fait ses preuves dans les entreprises qui l'ont adoptée. Globalement, les expériences répertoriées (Drury *et al.*, 2012, Wang *et al.*, 2012) montrent qu'elle permet un développement rapide des logiciels, augmente la satisfaction des clients, et redynamise les équipes. Cependant, il faut souligner qu'à mesure que la complexité des opérations s'intensifie, il apparaît des fissures dans l'application de ses principes fondamentaux. Des plans stables, des estimations exactes, et des prédictions sont souvent difficiles à obtenir; la productivité des équipes et la confiance en elles risquent d'être faibles. À partir de ce moment, un ajustement de ses principes doit s'imposer. De toute évidence, l'alignement strict sur les principes Agile/Scrum ne semble pas être la meilleure approche pour des équipes travaillant en environnement complexe, ayant un développement de produits avec des caractéristiques dynamique, non déterministe et non linéaire. Le strict alignement sur ses principes dans ces conditions, contribue à la persistance des obstacles à la CS.

4.4.7.2 Les besoins en ressources CS (personnel et technologie)

En examinant les ressources CS utilisées par chaque groupe, plusieurs commentaires généraux peuvent également être faits.

Il semble que les DEV interagissent souvent avec des membres des équipes de soutien sur place. Un peu plus haut dans l'organisation, les chefs d'équipe sont beaucoup plus susceptibles d'interagir avec d'autres groupes (tels que les CM, OPS, PO, PM) et avec des développeurs qui travaillent hors site. Il y a un besoin que ces groupes de développeurs comprennent les autres groupes avec lesquels ils interagissent. Par exemple, la compréhension du processus de déploiement des CM, la disponibilité des développeurs distants peut être très important en permettant au personnel DEV de développer une bonne compréhension de l'impact des décisions ou de comprendre pourquoi d'autres équipes font certaines déclarations.

En examinant les besoins et les ressources en CS des chefs d'équipe, des PM et PO, il est évident qu'ils jouent en grande partie le rôle des coordonnateurs et peuvent devenir des intermédiaires d'information. En plus des fonctions administratives, ils deviennent impliqués lorsque des problèmes surgissent et que leur assistance soit nécessaire pour fournir un soutien ou pour interagir avec d'autres équipes. Ce rôle est très critique dans le processus de réalisation d'une bonne CS au niveau de l'équipe.

En cas de problème majeur, quand les « leads » et les PM sont impliqués, ils doivent obtenir une quantité considérable d'informations de la situation des DEV sur le statut du PBI, des circonstances du problème entourant le PBI, les personnes impliquées, les solutions obtenues et les projections auprès du personnel sur place ou distant. Ce processus est long et implique de nombreux intervenants et parmi ceux-ci, certains n'ont pas une bonne compréhension de la situation. Il est donc possible que des informations importantes ne soient pas identifiées parce qu'elles tombent à travers les mailles du filet. Si les DEV « leads » et les PM ne possèdent pas une compréhension complète de toutes les informations pertinentes, même s'ils ont toute l'information, ils ne peuvent pas les transmettre correctement ou pas du tout à la bonne personne. Ils peuvent donc manquer de prendre la bonne décision dans ces circonstances.

En outre, les « leads » et les PM sont fréquemment responsables de la transmission d'informations de retour vers les DEV. Si seulement ils transmettent des informations au sujet de ce que les DEV devraient faire (la décision), mais pas en ce qui concerne la raison pour laquelle la décision a été prise, cela peut conduire à la fois à un manque de compréhension par les DEV et peut faire que les DEV n'aient pas la possibilité de contribuer spontanément et volontairement à des informations qu'ils jugent pertinentes pour la décision prise. Les « leads » et les PM forment un maillon essentiel dans la chaîne CS entre les différentes équipes et ont donc besoin d'avoir une compréhension complète des informations que les autres ont vraiment besoin et de la façon d'obtenir toutes les informations dont ils ont besoin eux-mêmes.

En examinant les besoins spécifiques en ressource des DEV en relation avec les AT, Il ressort que les DEV utilisent les AT pour des clarifications additionnelles sur les items à travailler. L'assistance des AT est sollicitée pour clarifier des règles de gestion ou pour comprendre comment l'application est utilisée par le client. Les informations que les DEV reçoivent des AT les aident à compléter un travail particulier dans le cycle en cours, et à mieux comprendre les implications de l'ensemble de leur travail sur l'application et l'atteinte de leurs propres objectifs.

Les DEV ont un grand besoin des Architectes et de leurs « leads » pour recevoir de meilleures recommandations sur les impacts des changements qu'ils (les DEV) apportent à l'application. Les architectes sous-estiment parfois les impacts de certains changements qui causent des erreurs. Les DEV pensent que les Architectes devraient jouer un peu plus un rôle de personne ressource en réalisant des analyses d'impact sur chaque changement majeur. Beaucoup d'erreurs peuvent être évitées si l'analyse d'impact est mieux organisée.

Il existe également des problèmes à suivre le statut des PBI (dans les cas de défauts dans l'application) quand il y a de nombreuses personnes qui ont accès au PBI pour y relater les résultats de leurs investigations. Tout le monde ne laisse pas de commentaires suffisamment explicites sur leurs trouvailles et recommandations. Le plus grand besoin CS pour les DEV est de déterminer des méthodes pour s'assurer qu'ils disposent d'informations correctes sur les PBI afin d'avoir de meilleures capacités d'estimation et ainsi projeter l'état du système (Niveau 3 CS). Après tout, les DEV ainsi que les PO travaillent sur des projections en faisant des planifications sur la base des exigences. Leur capacité à projeter l'état du système pourrait probablement être améliorée grâce à une meilleure rétroaction du système (un diagnostic clair sur ce qui n'a pas bien fonctionné en production).

Les DEV ont montré une satisfaction par rapport à la manière dont leur « leads » et superviseurs abordent les problèmes et la manière dont ils jouent leur rôle de personnes

ressources, cependant, ils aimeraient qu'ils soient plus proactifs car ils ont beaucoup de connaissances techniques et beaucoup d'expériences dans leur domaine respectif dans l'entreprise. Le défi est de combiner ces deux sources d'information plus efficacement pour parvenir à des diagnostics et des solutions appropriées. Cet écart entre ceux qui ont des informations de situation et ceux qui ont la meilleure connaissance technique peut être réduit avec une meilleure compréhension entre les deux groupes ou un meilleur usage des technologies qui aident à l'échange d'informations entre les deux groupes.

Chaque site de DEV (local ou distant) est en mesure de partager des informations pertinentes à travers l'utilisation des technologies de messagerie instantanées et des courriers électroniques. Par exemple, les problèmes détectés avec certains PBI ou fonctionnalités de l'application sont communiqués en temps réel à travers la messagerie où les destinataires sont organisés en groupe d'intérêt. Cette organisation permet à l'ensemble des équipes ainsi que les équipes dépendantes d'atteindre le niveau de connaissance le plus élevé possible. Le seul problème est que certaines discussions se passent sans que les personnes clés soient impliquées. La compréhension des rôles de chacun, de ce que fait exactement chaque équipe ou même chaque personne dans les équipes aidera tous les employés à rentabiliser leurs communications. Cela permettra à chaque intervenant de développer une compréhension de quelles informations ont besoin les autres groupes et la façon méthodique de faire circuler les informations nécessaires sur leurs propres situations.

4.4.8 Analyse et modélisation des dépendances stratégiques

Comme nous le mentionnions dans la section 4.4, les équipes de DEV sont impliquées dans des relations de dépendance avec les autres équipes de l'entreprise. Ces relations sont stratégiques car elles permettent à chaque partie de se rapprocher de la réalisation de ses propres objectifs. En nous référant à la notation proposée par la méthodologie

iStar que nous adoptons dans ce travail, ces relations peuvent être formulées formellement dans un modèle de dépendance stratégique (SD).

Nous représentons donc les liens de dépendance stratégique que nous avons répertoriés entre les DEV et les équipes de support. Dans cette caractérisation de niveau supérieur du processus étudié, nous capturons ce qui compte pour les acteurs, tout en laissant le non-essentiel.

Dans les relations de dépendance formulées ci-dessous, il apparaît que même si les DEV ont des habilités à accomplir leurs propres objectifs, ils deviennent rapidement vulnérables si l'autre partie n'est pas en mesure ou n'a pas l'intention de délivrer l'objectif ou la ressource ou la tâche, puisque dans une relation de dépendance, le *dependee* (celui dont on dépend) a toujours le choix d'exécuter ou non le *dependum* (le l'objet de la dépendance).

La structure d'ensemble du modèle SD qui est générée de ces multiples relations de dépendance est présentée dans la figure 4.22. Le modèle de dépendance stratégique se présente comme un ensemble de nœuds et de liens où chaque nœud représente un acteur et chaque lien entre deux acteurs indique qu'un acteur dépend de l'autre pour quelque chose afin que le premier puisse atteindre un certain objectif. Les concepts d'acteur et de dépendance se définissent comme suit :

- **Acteur** : Un acteur est modélisé ici comme un rôle ou un « agent dans le monde » et non comme un agent logiciel. Il s'agit d'une abstraction du comportement de l'agent et de son rôle dans le contexte spécifique étudié. Il va de soi que l'abstraction de l'agent dans d'autres contextes dans la même entreprise peut être différente. Graphiquement, les acteurs sont représentés par des cercles.
- **Dépendance** : Une dépendance représente un « accord » entre deux acteurs, où l'un dépend de l'autre pour respecter l'accord. L'accord peut être un objectif à satisfaire, une tâche à exécuter (sans connaître la motivation) ou une ressource à livrer.

Graphiquement, les objectifs sont représentés par un ovale, les ressources par un rectangle et les tâches par un hexagone.

Un *dependor*, en dépendant de quelqu'un d'autre (le *dependee*) pour quelque chose (*dependum*), peut accomplir un but ou un objectif qu'il ne serait pas capable (ou aussi bien) de réaliser tout seul. Si les DEV n'accèdent pas à l'information ou à la ressource recherchée auprès des autres équipes, leur CS en serait affectée, et par conséquent leur tentative de réaliser leur objectifs sera compromise.

Le modèle de dépendance stratégique que nous présentons dans cette section nous permet de capturer la structure intentionnelle du processus dans lequel les DEV et les autres acteurs sont impliqués. Il met en évidence ce qui compte pour l'acteur (ou l'équipe) dans l'accomplissement de ses objectifs. Nous avons ainsi une meilleure connaissance des vulnérabilités, des opportunités et une meilleure reconnaissance des patterns dans ces relations, telles que les mécanismes pour réduire les vulnérabilités et améliorer la CS. La connaissance des zones de vulnérabilité est nécessaire, si nous voulons proposer des solutions alternatives pour améliorer la CS des DEV.

Cet analyse s'articule autour de certains concepts clés de l'ingénierie des exigences tels que proposés par Yu (2011) dans ses travaux sur la modélisation du social, soient (a) vulnérabilité, (b) dépendance, (c) routines, et (d) capacité. Les définitions proposées pour ces termes sont libellées comme suit :

Routines : La caractérisation interne d'un agent tourne autour des routines détenues par l'agent et les éléments qui composent la routine. Une routine est le principal moyen par lequel un agent peut accomplir ce qu'il veut. Elle est un modèle pour les activités récurrentes de l'agent. Elle se compose d'éléments qui comprennent des sous-objectifs, des sous-tâches, des ressources et des soft-goal.

Il faut noter cependant que nous ne faisons usage de « *Routines* » seulement dans un contexte de relation de dépendance stratégique entre des « agents du monde »

Capacité: Quand un agent a une routine qui peut servir à quelque chose, par exemple, pour atteindre un certain objectif, nous disons que l'agent a une capacité à atteindre cet objectif

Vulnérabilité : Le concept de vulnérabilité exprime la multi-dimensionnalité des “catastrophes”. On concentre l'attention sur la totalité des relations dans une situation sociale donnée qui constituent une condition qui, en combinaison avec les données environnementales, produit une “catastrophe”. La détermination de la vulnérabilité d'une relation est basée sur la force de la relation. Elle est faible, forte ou critique selon que la relation soit de catégorie Ouvert (en anglais *Open*), Engagée (en anglais *Committed*) ou Critique (en anglais *Critical*)

Dépendance : On en distingue 4 types.

- Les dépendances d'objectifs (en anglais *Goal dependency*)
- Les dépendances de tâches (en anglais *Task dependency*)
- Les dépendances de ressources (en anglais *Resources dependency*)
- Les dépendances non fonctionnelles (en anglais *Soft-goal dependency*)

4.4.8.1 Formulation des relations de dépendance des DEV

Nous proposons dans le tableau 4.25, toutes les dépendances que nous avons identifiées lors de l'analyse des relations qu'entretiennent les acteurs du système. Des versions successives de ce tableau de dépendance ont été élaborées et présentées aux parties prenantes pour discussion et validation.

Tableau 4.25 Tableau des dépendances entre les acteurs

N	Partie en charge	Partie à charge	Objet de la dépendance	Nom de l'objet de la dépendance
1	DEV	Architectes	Pour objectif	Spec. technique
2	DEV	AT	Pour objectif	Exigences
3	AT	DEV	Pour objectif	Document d'utilisateur
4	DEV	PO	Pour tâche	PBI à travailler
5	PO	DEV	Pour objectif	Travailler les PBI
6	DEV	CM	Pour ressource	Déployer codes
7	CM	DEV	Pour objectif	Mettre à jour base de codes
8	DEV	DEV	Pour ressource	Compatibilité sous-application
9	INT	DEV	Pour objectif	Consommation web service
10	DEV	AT	Pour ressource	Clarification des règles
11	DEV	FMK	Pour ressource	Mise à jour contrôle
12	DEV	IT	Pour ressource	Matériel informatique fonctionnel
13	AT	DEV	Pour ressource	Assistance SQL
14	DEV	AT	Pour soft-goal	Produire des exigences rapidement
15	DEV	CM	Pour soft-goal	Déployer les « builds » rapidement
16	DEV	Superviseur	Pour ressource	Validation de l'architecture
17	DEV	Superviseur	Pour ressource	Support de priorisation
18	AT	PO	Pour ressource	Statut de l'application
19	AT	PM	Pour ressource	État des travaux
20	AT	SUP	Pour ressource	Statut de l'application
21	DEV	SUP	Pour ressource	Vérification architecture
22	FMK	SUP	Pour ressource	Support priorisation
24	INT	SUP	Pour ressource	Support priorisation
25	SUP	DEV	Pour tâche	PBI à travailler

La version que nous présentons ici en est le résultat. Chaque ligne du tableau est atomique et représente une relation complète de dépendance entre 2 acteurs. Il n'y a pas d'ordre de classement dans l'énumération des lignes du tableau.

4.4.8.2 Le diagramme de dépendance stratégique

Dans cette section, nous présentons en détail, certaines relations qui nous paraissent importantes dans le modèle de dépendance stratégique des DEV. La figure 4.22 présente le modèle dans son ensemble.

- Les développeurs sont dépendants des équipes d'architectes pour des spécifications d'implémentation techniques (dépendance d'objectif)

Selon l'évaluation fournie par les DEV sur leurs relations avec d'autres équipes, leur dépendance vis-à-vis des architectes est de force *ouverte* (*dependance ouverte*). Dans cette relation, le *dependor* a des désirs et aimerait voir l'objectif « Spec techniques » atteint afin de l'utiliser pour son travail. Les architectes ont toujours la latitude de décider sur la réalisation ou non du *dependum*. Selon la force de cette dépendance, le non réalisation du *Dependum* aura un certain effet sur les routines et les objectifs du *Depender* mais les conséquences ne seront pas sérieuses. Dans cette relation **ouverte**, la vulnérabilité des DEV peut être qualifiée de faible.

- Les DEV sont dépendants des AT pour des spécifications (dépendance d'objectif). A leur tour, les AT sont dépendants des DEV pour des documents d'utilisateur (dépendance d'objectif).

La dépendance des DEV vis-à-vis des AT est de force *Critique* (*Critical dependency*). Dans cette relation, le *dependor* a des désirs et aimerait voir l'objectif « Exigences » atteint afin de l'utiliser pour son travail. Si AT ne délivre pas le *dependum* « Exigences », les DEV ne seront pas en mesure de poursuivre leur travail. Leurs routines en seront affectées. Il est en effet inconcevable pour les DEV de commencer à

développer si les exigences ne sont pas ajoutées aux PBI ou si elles sont incorrectes ou incomplètes. Cette dépendance est *critique* puis que dans la plupart des cas, DEV n'a pas d'autres alternatives. Le degré de vulnérabilités est aussi très élevé. Le non-satisfaction de ce *dependum* implique que les DEV ne seront pas en mesure de donner les services attendus nécessaires à d'autres acteurs du système qui dépendent d'eux à leur tour.

Entre ces 2 acteurs, il y a une relation de dépendance réciproque dont chaque partie pourrait stratégiquement faire recours pour obtenir son *dependum* (nous reviendrons sur les solutions stratégiques dans le chapitre 5). Il semble aussi que la force de la dépendance des AT vers les DEV soit *critique*, c'est-à-dire que l'acteur AT croit qu'il n'y a pas d'autres issues pour lui d'obtenir ses documents d'usager. La vulnérabilité des AT envers les DEV est forte pour ce *dependum*

- Les DEV sont dépendants des PO pour des PBI à travailler (dépendance d'objectif). Les PO sont dépendants des DEV pour travailler les PBI (dépendance d'objectif)

Ici aussi, il y a une relation d'interdépendance entre les PO et les DEV. Ces 2 relations sont qualifiées de **critique** puisque dans la configuration actuelle du processus, les DEV ne sont pas en mesure de faire leur travail si les PO ne leur communiquent pas les listes des items à travailler. De l'autre côté, les PO ont besoin que les DEV accomplissent leur travail pour amener le système d'un état à l'autre. Nous remarquerons qu'il n'y a presque pas de problèmes relevés dans ce type de relations dans les barrières à la CS identifiées par les DEV.

C'est une relation de force **critique** ou les 2 parties ont une grande vulnérabilité vis-à-vis de l'autre.

- Les développeurs DEV sont dépendants des CM pour le déploiement des items à tester (dépendance de ressource).

Les DEV ont besoin des CM pour promouvoir les codes dans les environnements QA. Cette dépendance affecte la réalisation des objectifs des DEV. La dépendance est **critique** et la vulnérabilité des DEV pour ce *dependum* est élevée

- Les CM sont dépendants des développeurs DEV pour la mise à jour des bases de codes pour les « builds » à déployer (dépendance d'objectif)

Les CM ont une grande vulnérabilité par rapport au DEV. Si les codes ne sont pas sauvegardés dans les bases de codes ou qu'ils ne sont pas localisés dans les bons répertoires, il serait difficile au CM de mettre en œuvre leurs routines et réaliser correctement leurs objectifs. Ils n'ont pas d'autre recours que d'attendre après la réalisation du *dependum* par les DEV afin de poursuivre la leur. Le non réalisation de cet objectif a des impacts considérables sur les CM, et a des répercussions sur les acteurs qui dépendent des CM.

La dépendance est *Critique* et la vulnérabilité des CM pour ce *dependum* est élevée

- Les développeurs d'une unité particulière sont dépendants d'autres développeurs pour les questions de compatibilité des versions des sous-applications (dépendance de ressource).

Cette dépendance de ressource entre les équipes de DEV ne semble pas empêcher les DEV de faire leur travail. Il est certain que le non satisfaction de ce *dependum* peut occasionner des délais supplémentaires dans la livraison des objectifs ou la qualité du produit fini, mais il ne semble pas être critique pour aucun des groupes. La force de cette *dependum* est faible.

- L'équipe d'intégration est dépendante des développeurs pour savoir comment consommer les Web services (dépendance d'objectif)

Il y a une dépendance d'objectif entre les DEV et les équipes d'intégration. Si les web services ne sont pas suffisamment documentés, les équipes d'intégration auront certaines difficultés à respecter leurs propres objectifs. Cependant ils parviennent dans la plupart des cas à trouver des moyens alternatifs pour exécuter leurs routines.

La dépendance est *Engagée* et la vulnérabilité des INT par rapport aux DEV pour ce *Dependum* est faible

- Les développeurs sont dépendants des gestionnaires de compte pour des clarifications de travail (dépendance de ressource).

Les développeurs utilisent les AT comme ressources pour valider certaines informations et connaissances sur les règles de gestions et sur les clients. Même si dans la plupart des cas, la finition de leur travail n'est pas liée à l'obtention de ces informations, il faut quand même noter que les DEV ont un certain degré de vulnérabilité vis-à-vis des AT. Cette dépendance entre DEV et AT est de type *Ouvert*.

La vulnérabilité des DEV vis-à-vis des AT pour ce *dependum* est faible

- Les développeurs sont dépendants des FMK pour des contrôles d'interface et leurs mises à jour (dépendance de ressources)

Les DEV ont une vulnérabilité par rapport au FMK. Si les contrôles d'interface ne sont fournis, il ne serait pas facile aux DEV de réaliser correctement leurs objectifs. Il faut noter cependant que le processus de développement donne une flexibilité au DEV de mettre ou non à jour les contrôles d'interface utilisés dans leur application. Les DEV peuvent toujours réaliser leurs objectifs même si ce *dependum* n'est pas livré entièrement par le *dependeur*. Les DEV ont des solutions alternatives. La non réalisation

de cet objectif a des impacts modérés sur les DEV et n'a aucune répercussion sur les acteurs qui dépendent des DEV.

La vulnérabilité des DEV pour ce *dependum* est faible.

La dépendance est Ouverte pour les DEV

- Les développeurs dépendent des IT pour des ordinateurs de travail et des mises à jour de logiciel de travail (dépendance de ressource)

La vulnérabilité des DEV pour ce *dependum* semble élevée. Même si la satisfaction de cet objectif par les IT ne fait pas partie directement des processus que nous avons identifiés dans le cycle de vie de développement, les DEV ont besoin de la mise à jour de leur logiciel de travail. Le non satisfaction de ce *dependum* impacte le travail des DEV.

Cette dépendance de ressources est force *Critique*.

La vulnérabilité est forte pour les DEV pour ce *dependum*

- Les AT ont besoin des DEV(DBA) pour des rapports et des instructions SQL Ad hoc leur permettant d'interroger les bases de données (dépendance de ressource)

Les AT utilisent les DBA de DEV pour produire des requêtes SQL Ad hoc afin de mieux comprendre la nature des données. Dans la plupart des cas, les données générées ne sont pas envoyées aux clients et ne servent que pour des usages internes. Les AT n'ont aucune vulnérabilité par rapport à ce *dependum*. Si les DEV ne le satisfont pas, les AT peuvent utiliser d'autres sources.

La vulnérabilité des AT dans cette relation de dépendance vis-à-vis des DEV est faible

La dépendance est Ouverte pour les DEV

- Les AT dépendent des DEV pour des travaux de maintenance de base de données (dépendance de tâche).

Dans cette relation de dépendance où les DEV sont les *dependee*, les AT dépendent des DEV pour exécuter une tâche. Les AT précisent comment la tâche doit être faite mais pas pourquoi. Le *dependee* est tout aussi vulnérable dans ce cas, puisque les DEV peuvent ne pas réaliser avec succès cette tâche dont ils ne comprennent pas le contexte d'affaire. Ils ne peuvent pas être en mesure de proposer des solutions alternatives.

La réalisation de ce *dependeum* doit être interprétée comme une contrainte pour les DEV plutôt que l'application d'un savoir-faire quelconque et c'est justement une des raisons pour laquelle sa réalisation peut ne pas réussir. Aussi, les DEV peuvent simplement décider de ne pas le faire en prétextant qu'ils ont autres tâches plus importantes à réaliser. La figure 4.22 récapitule les dépendances stratégiques des DEV vis-à-vis des autres équipes. Il s'agit de la version corrigée et validée par les parties prenantes.

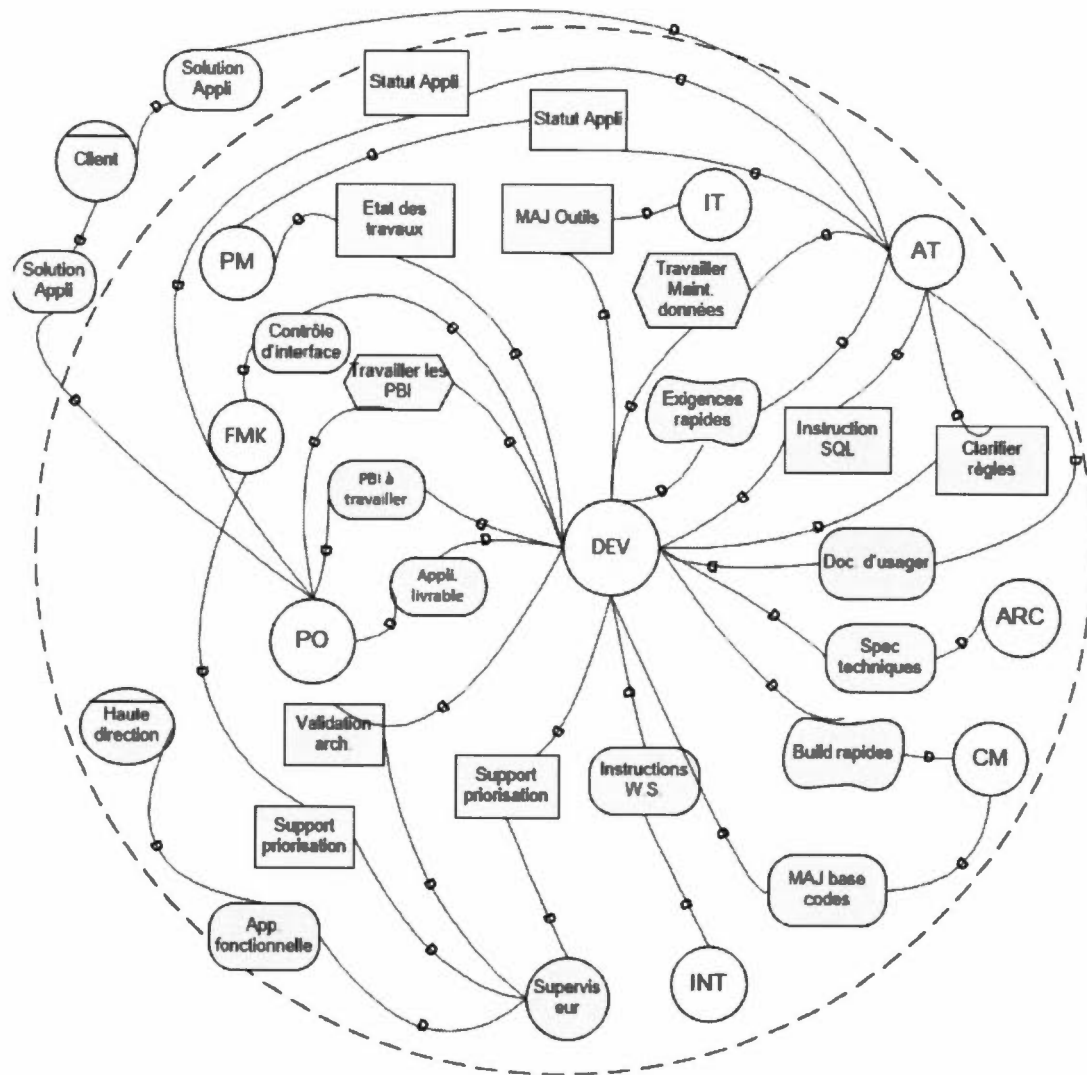


Figure 4.9 Le modèle de dépendance stratégique des DEV

Légendes des diagrammes SD et SR

	Lien de décomposition
	Lien de dépendance
	Lien <i>Means-ends</i>
	But
	Tache
	Dans une dépendance de ressources, le dépendant dépend du <i>Dependee</i> pour la disponibilité d'une entité
	Un <i>Softgoal</i> est similaire à un but sauf que les critères de succès ne sont pas définis à priori. Les <i>Softgoals</i> peuvent être traités comme des besoins non fonctionnels.
	Une entité active qui réalise des actions pour atteindre des objectifs en exerçant son savoir-faire. Désigne de manière générique toute unité à laquelle des dépendances intentionnelles peuvent être attribuées. Les agents, les rôles et les positions sont des sous-unités d'un acteur social complexe, chacun étant un acteur dans un sens plus spécialisé.
	Acteur avec des manifestations concrètes et physiques, comme un individu humain. Peut aussi représenter des artefacts technologiques

4.5 Conclusion

À partir de l'analyse du processus de développement d'application décrit ici, nous avons pu identifier les acteurs, leurs objectifs et les interactions qui existent entre eux. Nous avons porté notre attention à la fois sur les comportements et nous nous sommes interrogés sur ce que chaque acteur veut, comment atteint-il ce qu'il veut et sur qui dépende-t-il pour atteindre ce qu'il veut.

Le modèle de dépendance stratégique proposé pour l'ensemble des relations dans lesquelles est impliqué notre groupe cible, nous a aidés à découvrir et à mettre en évidence pour tous les acteurs du système, les opportunités qui leur sont offertes, les vulnérabilités avec lesquelles ils ont à composer dans leurs relations externes intentionnelles les uns avec les autres. Dans le chapitre suivant, nous identifierons plus spécifiquement les mécanismes qui aideraient à fortifier les relations de dépendance et à atténuer les vulnérabilités entre ces acteurs intentionnels afin de les rapprocher davantage vers de meilleures consciences de la situation.

CHAPITRE V

PROPOSITION DE SOLUTION D'AMÉLIORATION DE LA CS

5.1 Introduction

Les problèmes mis au jour par cette analyse sont innombrables. De toute évidence, de nombreuses et différentes stratégies utiles peuvent être proposées pour améliorer la CS et la circulation de l'information au sein de l'entreprise. Cependant, améliorer la CS au sein d'une organisation complexe telle que celle du développement d'application, exigera également que le personnel soit en mesure d'utiliser correctement les recommandations et directives reçues pour opérer des évaluations de la situation à tous les niveaux (diagnostic, compréhension et projection). La mise en application des solutions proposées permettra au personnel d'être mieux outillé pour communiquer des informations à l'intérieur et entre les équipes impliquées dans le développement. Pour cette raison, en plus des actions directes sur la manière dont l'information est échangée au sein des équipes de développement d'application, un programme de formation pour améliorer la CS d'équipe a également été développé.

Les solutions d'amélioration que nous proposons s'inscrivent aussi dans un modèle de réingénierie des processus qui introduit des changements organisationnels dans le but d'améliorer la performance de l'entreprise dans son ensemble. En matière de réingénierie des processus, il est généralement reconnu que l'intervention de l'expert s'apparente plus à de l'art qu'à de la science pure, et que les résultats peuvent être non prédictibles (Keen, 1991; Davenport, 2013). Concrètement, nos recommandations visent à changer la manière de travailler au sein de l'entreprise, mais il n'y a pas une garantie totale que la solution proposée fonctionne telle que prévue. En plus, la nature

dynamique de l'entreprise et la complexité des relations entre les acteurs, peuvent créer de nouvelles réalités et conditions qui affectent le résultat attendu.

Ce chapitre présente les solutions que nous proposons pour l'amélioration de CS d'équipe. Il s'agit des actions concrètes dans l'organisation et le fonctionnement des équipes et aussi de l'élaboration d'un programme de formation en CS d'équipe. La méthodologie de leurs exécutions est inspirée de celle proposée par Endsley (Endsley, 2000) dans ses travaux sur la formation en CS d'équipe dans le domaine de l'aviation.

5.2 Solutions d'amélioration de la CS d'équipe

Le tableau 5.1 présente plusieurs changements et améliorations à la fois technologiques et organisationnels qui peuvent être mis en œuvre pour améliorer la diffusion de l'information. Bien que ces facteurs soient spécifiques à cette entreprise, beaucoup de solutions proposées peuvent également être applicables à d'autres organismes de développement d'application.

Tableau 5.1 Les améliorations au système

Les interventions technologiques
• Soutien à l'échange d'informations entre les unités de développeurs • Meilleure intégration des bases de données des applications commerciales utilisées dans les équipes de développeurs • Soutien à cognition de l'équipe • Régulation des tâches et des objectifs individuels et d'équipe
Les améliorations organisationnelles et individuelles
○ Un meilleur échange d'information entre les unités de développeurs ○ Un meilleur échange d'information entre les unités de développeurs et les autres unités ○ Amélioration de la documentation des requêtes pour assurer que les exigences appropriées soient disponibles et fournies pour le travail ○ Une meilleure description des rôles et attributions de chacun et de chaque équipe ○ Une meilleure compréhension entre les équipes impliquées dans le développement ○ Un meilleur esprit d'équipe et travail d'équipe ○ Un plan de formation adéquat pour accompagner les nouveaux recrues ○ Améliorer le suivi des temps d'exécution des tâches du personnel. ○ Participation accrue du personnel à tous les niveaux dans la résolution des problèmes ○ Améliorer la stabilité organisationnelle ○ Rationalisation de l'organisation ○ Ajustement des principes de la méthodologie de développement Agile

5.3 Les stratégies de fortification des relations entre les acteurs

5.3.1 Agir sur les dépendances pour une meilleure CS d'équipe

L'analyse narrative et le modèle de dépendance stratégique que nous avons présentés dans la figure 4.22 nous ont permis de distinguer des dépendances de diverses natures (objectif, ressource, tâche) et de divers degrés (Ouvert, Engagé, Critique) entre les DEV et les équipes de support. Du côté des DEV, une dépendance forte signifie que le *dependor* (DEV) est plus vulnérable et qu'il peut prendre des mesures plus sérieuses pour réduire sa vulnérabilité. Du côté des *dependee* (par exemple, équipes de support), une forte dépendance implique qu'ils devront faire de plus grands efforts pour essayer de livrer le *dependum*. Connaissant donc la nature et la force de toutes les relations dans lesquelles sont impliqués les DEV, il est plus aisé d'envisager des solutions appropriées.

5.3.2 Atténuer les vulnérabilités

Il existe divers mécanismes qui contribuent à fortifier une dépendance et à atténuer la vulnérabilité (Yu, 1994).

« Enforcement mechanism » : Un *Dependum* peut être exécutoire s'il y a une manière pour le *dependor* de causer un échec pour un certain objectif du *dependee*. C'est-à-dire qu'il existe entre les deux parties, une dépendance réciproque.

« Assurance mechanism » : Indépendamment du mécanisme qui rend exécutoire le *dependum*, un *dependum* peut être traité avec diligence par le *dependee* si ce dernier sait que l'accomplissement du *dependum* rentre dans son propre intérêt. Dans ce cas, le *dependor* n'a presque pas d'effort à faire pour exiger le *dependum*. Il y a donc ici, un mécanisme naturel d'application du *dependum*. Ce mécanisme aussi appelé « Assurance mechanism », fortifie la relation du *dependor* en réduisant le degré de dépendance et la vulnérabilité.

« Insurance mechanism » : Par ce mécanisme, on réduit la vulnérabilité du *dependor* en réduisant le degré de dépendance sur le *dependum*. Un *dependor* peut augmenter ses chances de rendre exécutoire le *dependum* s'il est capable de se le procurer d'une autre manière que par l'intermédiaire du *dependee* actuel. Autrement dit, la stratégie consiste à avoir plusieurs *dependee*.

Toute combinaison de ces trois mécanismes peut être utilisée mais notre approche de solution se repose sur le « Assurance mechanism » et le « Insurance mechanism ».

5.4 Le modèle stratégique de dépendance proposé

En appliquant les mécanismes de fortification de dépendance décrits précédemment à la section 5.3.2, nous en déduisons le modèle de dépendance stratégique de la figure 5.1. Les détails de l'application des mécanismes sont présentés dans les sections qui suivent.

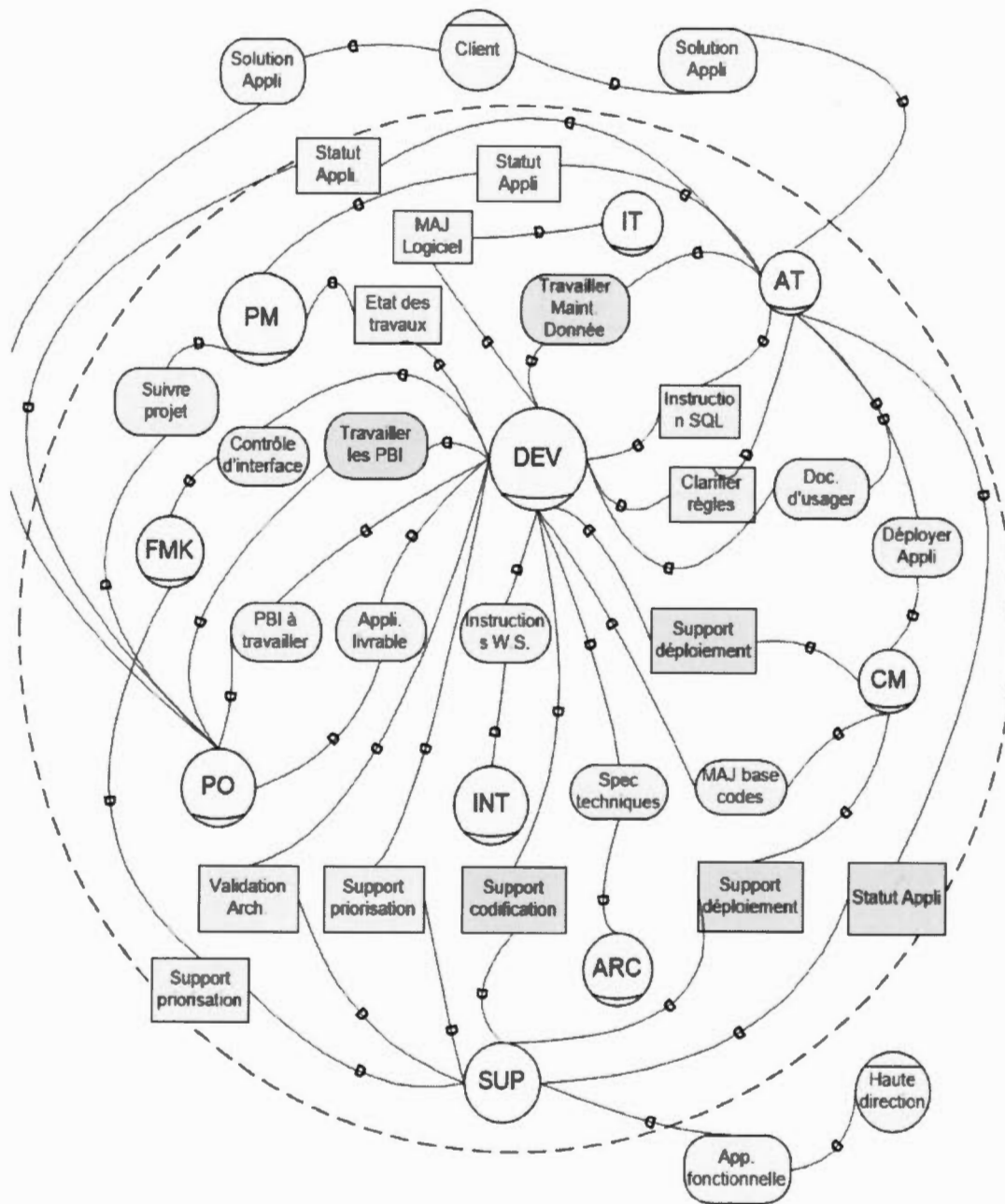


Figure 5. 1 Modèle de dépendance stratégique proposé

Dans le modèle SD proposé, nous avons inclus les améliorations suivantes :

1) Les relations de dépendance de tâche envers les DEV ont été retirées

Les AT doivent documenter leurs requêtes et donner une explication et mise en contexte de ce qu'ils demandent. Cela permettra aux DEV de comprendre le contexte d'affaire du travail qui leur est assigné. Le travail sera donc moins mécanique si les DEV connaissent le rationnel qui motive les requêtes. Ainsi, les procédures de complétude des PBI peuvent être ajustées afin d'amener les AT à inscrire en claire, le rationnel de leur demande soutenu par une description du contexte d'affaire qui a motivé le PBI.

Les SUP devront aussi adoucir leur relation avec les DEV. Il est plus avantageux pour les 2 équipes si les SUP renforcent leur rôle de support envers les DEV au lieu de leur dire comment exécuter leurs tâches.

Aussi, les relations de dépendance de tâches qui lient les DEV au PO ont été retirées. En mettant en application cette recommandation du modèle dans le cycle de vie de développement, les PO pourront toujours continuer d'attribuer du travail de développement aux DEV, mais ils devront éviter de leur dire comment ils doivent l'exécuter.

2) Les *soft-goal* ont été fusionnés à des dépendances d'objectif et à des dépendances de ressources

Le concept de *soft-goal* est basé sur la notion d'exigence non-fonctionnelle en ingénierie de logiciel. Lorsque les DEV disent qu'ils veulent que les documents d'exigence soient produits rapidement par les AT, le terme "rapidement" reste vague et interprétable par chaque partie. Les AT de leur côté peuvent ne pas avoir les mêmes notions d'urgence que les DEV, ou peuvent dire qu'ils ont d'autres tâches à exécuter.

Les *soft-goal* sont donc fusionnés avec les dépendances d'objectif et les dépendances de ressource pour les rendre plus opérationnels et les introduire dans le processus normal de travail. Dans certains cas, des dépendances d'objectif et de ressource additionnelles ont été proposées lorsque nécessaire pour prendre en considération les impératifs de qualité exprimés par les *soft-goal*. Les mécanismes pour les rendre exécutoires seront mis en place à la fois par la formation et par les actions organisationnelles à introduire.

5.5 Modèle de raisonnement stratégique des DEV

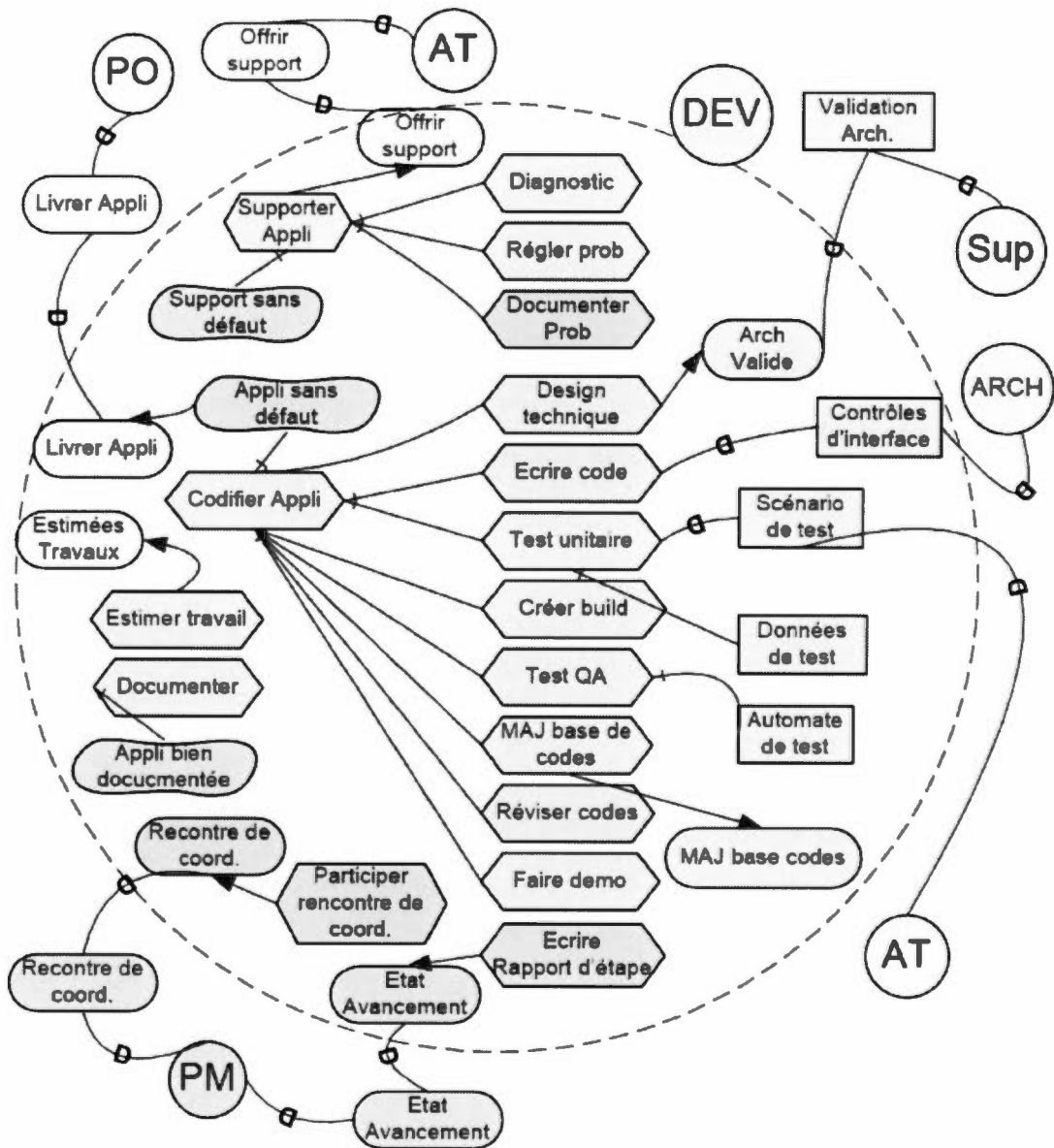


Figure 5. 2 Modèle de raisonnement stratégique des DEV

Le modèle de raisonnement stratégique (SR) de la figure 5.2 montre ce qui se passe à l'intérieur des frontières des acteurs DEV, y compris les objectifs, les soft-goals, les tâches et les ressources qui contribuent à l'analyse des alternatives et à l'accomplissement des dépendances. Ce modèle SR montre aussi comment les dépendances sont satisfaites entre les acteurs. Il met en exergue les tâches de communication et d'échanges d'information entre les DEV et les autres équipes. Les activités de documentations aussi y sont représentées.

5.6 Les changements dans le processus et dans les relations entre les acteurs.

5.6.1 Décloisonnement des équipes pour de meilleurs modèles mentaux partagés

De l'analyse, il ressort que les différentes équipes n'ont pas un bon modèle mental de ce que les autres équipes savent ou ne savent pas, ou ce dont les autres équipes ont besoin de savoir. Une bonne CS au niveau de l'équipe dépend du fait d'avoir une compréhension claire de ce que signifie l'information lorsqu'elle est convoyée à travers les membres de l'équipe. Ainsi, les équipes ont besoin de partager non seulement des données, mais également des niveaux plus élevés de CS, y compris la signification des données pour les objectifs de l'équipe et pour la projection de l'information. Ce processus est grandement renforcé par la création d'un modèle mental partagé qui fournit un cadre commun de référence pour les actions des membres de l'équipe et permet aux membres de l'équipe de prédire les comportements des uns et des autres.

Un modèle mental partagé peut fournir des communications plus efficaces en offrant un moyen commun d'interpréter et de prédire les actions basées sur des informations limitées, et peut ainsi constituer une base cruciale pour un travail d'équipe efficace. Un bon modèle mental aiderait à réduire le long processus de demande de clarification dans lequel s'engagent les équipes lorsqu'un problème particulier surgit. Lorsque les modèles mentaux partagés ne sont pas présents, une équipe ne peut pas comprendre

pleinement les implications de l'information transmise d'une autre équipe, ce qui fait que les malentendus, les erreurs et l'inefficacité sont susceptibles de se produire. En fournissant à chaque équipe une meilleure information sur les objectifs d'autres équipes, la façon dont elles exécutent leurs tâches, et quels sont les facteurs qu'elles prennent en compte dans leurs processus de décision, un modèle mieux partagé peut être développé. Cela devrait grandement améliorer non seulement les interactions interpersonnelles entre les équipes, mais aussi la qualité des processus de décision.

- Des rencontres de *planning* et de *grooming* élargies

Dans le sens d'amélioration des modèles mentaux partagés, l'intervention que nous proposons est de procéder au décroisonnement des unités de DEV. Décroisonner les unités de DEV en leur offrant des opportunités de tenir des rencontres de *planning* et de *grooming* ensemble; en suivant la cadence des activités du cycle de vie ensemble. Ces rencontres doivent aussi inclure les équipes de support telles que les PM et les OPS. L'idée est de mettre chacun dans des situations où il est au courant de ce qui se passe dans les autres équipes et travaillent en prenant en compte les réalités des autres. Vue la taille des équipes, cette rencontre pourrait se limiter aux chefs d'équipes DEV, aux PM et au PM. Les autres membres des équipes DEV peuvent joindre au besoin.

5.6.2 Amélioration de la communication entre les équipes.

- Instituer un outil de communication qui informe de la dernière version du module sur lequel chaque équipe a travaillé.

L'analyse a fait ressortir, entre autres, des problèmes de compatibilité entre les versions des sous-applications. Chaque unité de DEV doit produire et communiquer à toutes les équipes, la version la plus récente de l'application à sa charge. Ce changement agira sur les modèles mentaux de chaque acteur puisqu'il permettra à tous de connaître quelle version d'une sous-application donnée est compatible avec quelle version. Ils pourront

ainsi poser des diagnostics plus facilement et plus juste lorsque des problèmes surviennent dans leurs modules pour des raisons de dépendance avec d'autres modules.

- Instituer un outil de communication qui informe les DEV des cédulas de déploiement en production.

Les informations sur les déploiements, notamment, la date et l'heure du déploiement, le nom de la fonctionnalité, la liste des items, l'opérateur en charge, le numéro de la version, la version du système hôte (par exemple le système d'opération, le système de gestion de base de données), etc... doivent être documentées et mises à la disposition de toutes les équipes de DEV.

- Amélioration de la documentation des PBI

En plus de travailler sur la documentation des exigences qui figurent dans les PBI, des actions peuvent être entreprises par les PO en concertation avec les AT, afin que ces derniers inscrivent des scénarios de test appropriés dans les PBI. Il s'agit de scénarios de test reçus directement des clients et non ceux imaginés par les AT. Ce changement apparemment négligeable a des effets sur la compréhension de ce que le client désire réellement, et réduira les recours constants à des demandes de clarification des DEV vis-à-vis des AT. Les relations entre les DEV et les PO seront fortifiées et chaque partie y gagne stratégiquement.

5.6.3 Ajustement des principes Agile au contexte de l'Entreprise

- Équilibrer l'attention donnée au développement du logiciel et la documentation des fonctionnalités développées

La documentation des fonctionnalités du logiciel est distribuée dans la description des tâches et est difficile à rassembler. La raison principale en est que peu d'attention est portée à cette étape pourtant importante du cycle de vie du logiciel. La responsabilité de la documentation devrait être confiée aux analystes ou aux chefs d'équipe pour les équipes qui n'ont pas d'analyste attitré. Le répertoire de document des spécifications ainsi constitué aidera les

employés novices et même les seniors à comprendre plus rapidement non seulement les règles de gestion mais aussi les procédures de travail. Une attention plus grande à la documentation contribue grandement à l'amélioration des modèles mentaux des équipes.

- Documenter l'essentiel des spécifications avant le début du Sprint

Il n'est pas nécessaire de dissocier entièrement l'étape de la documentation des fonctionnalités, de celui du sprint. L'idée est de mobiliser les analyses afin qu'ils commencent très tôt leur travail et apprennent toute la documentation en collaboration avec les clients avant que les DEV n'entrent en scène. Même si toute la documentation n'est pas prête avant le début Sprint, cette nouvelle procédure aura pour avantage de faciliter les séances de planification puisque les DEV auront en ce moment suffisamment d'information sur chaque tâche pour faire des estimations plus réalistes. L'autre avantage de cette mesure est qu'un travail d'analyse plus élaboré au début du processus, donnera aux développeurs une plus grande indépendance par rapport aux AT et ainsi, réduira les situations d'interruption fréquentes qu'ils rencontrent.

- Étendre la durée des itérations des Sprint

La durée actuelle des Sprint de 2 à 3 semaines est excessivement courte pour la taille des équipes et la quantité d'items qui sont livrés à chaque Sprint. Des Sprints d'une durée de 4 à 5 semaines donneront plus de temps aux QA de couvrir plus de scénarios de tests, et aux AT de faire plus de tests d'acceptation. L'itération actuel de 2 à 3 semaines met tout simplement trop de pression sur les équipes et est à l'origine de nombreux problèmes dont des livraisons hâtives des logiciels. Ainsi, cette nouvelle disposition donnera plus de temps au logiciel de mûrir avant son déploiement en environnement de production.

- Limiter l'accès des clients au développeur

Les discussions intensives entre les DEV et les clients internes ou externes doivent être contrôlées et réduites au minimum possible durant la phase de l'exécution des Sprint. Durant cette période, les analyses, les PM et les PO doivent être les points de contact des clients pour les clarifications sur les spécifications. Ici Aussi l'avantage est de laisser le temps au DEV de se concentrer sur leur travail.

5.6.4 Un système d'évaluation du personnel basé sur des mesures objectives

Cette action répond à la stratégie de « Assurance mechanism » abordée précédemment et qui consiste à mettre le *depend* dans une situation où l'exécution du *dependum* va dans son propre intérêt. Si l'entreprise met en place des outils et des procédures d'évaluation du rendement des employés basée sur des objectifs mesurables, chaque employé devra être évalué sur des critères objectifs qui font clairement ressortir, de façon chiffrée, les attentes et les réalisations. Avec de pareils objectifs de résultat, les employés devront mettre des efforts pour satisfaire les *dependum* dans les délais prescrits.

5.6.5 Clarification et redéfinition des rôles et responsabilités des équipes

- Déléguer aux DEV certaines responsabilités d'autres équipes.

Il va falloir redéfinir les rôles des DEV et des CM et ajuster le processus actuel pour que les CM se libèrent de certaines de leurs attributions au profit des DEV. Cette action rentre dans la stratégie de « Insurance mechanism » qui consiste à réduire les vulnérabilités en diversifiant les *depend*. La relation de dépendance entre les DEV et les CM pour le *dependum* « déployer les codes pour QA » est l'un de plus problématiques du cycle. Elle peut être fortifiée suite à certains ajustements aux procédures qui permettraient aux DEV de faire les « builds » et de les déployer en environnement QA. Pour leur part, les DEV réduiront leur vulnérabilité par rapport à ce *dependum*. Ce changement fera avancer les tests et réduire les délais de livraison des applications.

- Clarifier les responsabilités des AT concernant la qualité des données reçues des clients.

L'intégration des données directement reçues des clients doit suivre un processus contrôlé et être confiée à une équipe qui devra en avoir la responsabilité. Pour le

moment, la responsabilité n'est pas clairement définie et ce travail tombe entre les mailles du filet.

5.6.6 Un système d'information conforme à la méthodologie agile

Sur le plan des solutions technologiques, l'implantation d'un système d'information conforme aux principes agile aurait l'avantage de supporter à la fois la cognition de l'équipe et de réguler les tâches et objectifs individuels et de l'équipe

- Soutien à la cognition de l'équipe

Le système devra aider les membres d'équipe sur plusieurs points, dont :

- Informer lorsque les enregistrements des codes informatiques dans la base de codes sont effectués;
- Soutenir la communication entre les membres de l'équipe, en l'initiant lorsque cela est nécessaire. Par exemple envoyer des alertes pour les demandes de révisions de codes lorsque les révisions sont complétées;
- Identifier les tâches dépendantes qui sont confiées à un membre d'un autre une autre équipe, informer sur l'état d'avancement;
- Garder les plans de test conçu par les QA et les envoyer au DEV;
- Notifier les DEV lorsque leurs codes sont promus d'un environnement à l'autre. Ils pourront de cette façon suivre la progression de leur travail de l'environnement DEV vers celui de production

- Réguler les tâches et les objectifs individuels et d'équipe

- Aider à visualiser et surveiller le progrès des tâches individuelles tous les jours;
- Aider à visualiser et surveiller le progrès collectif tâches planifiée versus tâches exécuté par itération;
- Indiquer les retards dans les livraisons en rapport avec ce qui est planifié;

- En fin de cycle, fournir des statistiques sur les exécutions et le progrès réalisé sur chaque tâche;
- Regrouper les tâches par objectif pour l'équipe;
- Soutien pour le suivi du temps d'exécution des travaux du personnel

5.7 Les recommandations pour une formation des équipes sur la CS

Plusieurs concepts de formation ont été identifiés pour améliorer la CS d'équipe ou des équipes de développeurs sur la base de la présente analyse et discussion.

5.7.1 Modèles mentaux partagés et communication inter-équipes

Nous avons abordé la question des actions pour l'amélioration des modèles mentaux à la section 5.5.1. Il ressort qu'il y a une relation directe entre la similitude des modèles mentaux auprès des membres de l'équipe et la performance de l'équipe.

Ces connaissances sur les modèles mentaux constitueront l'essentiel de la formation et communication sur ce sujet. Il ne s'agira pas d'aborder des notions académiques très avancées mais il s'agira plutôt de travailler sur des exemples concrets du quotidien des équipes. Par exemple, expliquer quel mode de partage d'information peut aider à éviter des changements dans une partie de l'application qui a un grand impact sur les travaux des autres. Aussi, il sera question d'élaborer sur les types d'organisations de la communication qui peuvent aider à mieux comprendre les requêtes des autres équipes et à comprendre comment les autres équipes utilisent les données échangées. Enfin, la mise en place des normes et processus de travail, leur compréhension et leur mise en pratique sont essentiels pour l'édification des modèles mentaux partagés.

5.7.2 Contextualisation des décisions

La réduction des dépendances de tâches passe par une verbalisation des décisions. Les équipes ont exprimé des besoins de faire un meilleur travail de transmission des informations à d'autres équipes en ce qui concerne la raison pour laquelle elles décident

(ou non) de prendre des actions particulières (par exemple, les reports des tâches planifiées sur d'autres cycles, la nature des requêtes de maintenance des données, etc.). À moins que la justification et les raisons ne soient communiquées, des malentendus considérables peuvent survenir. En outre, ceci fait rater des opportunités d'avoir de meilleures informations d'autres équipes, qui pourraient posséder des informations pertinentes pour une solution plus optimale.

Dans la section des barrières et problèmes du chapitre 4, nous avons évoqué l'absence du contexte d'utilisation dans les requêtes de maintenance de base de données. Communiquer pourquoi une décision particulière a été prise, fournit un niveau beaucoup plus élevé de CS (en particulier au niveau de la compréhension). Il permet à d'autres équipes soit de comprendre et d'accepter la décision ou de proposer d'autres solutions qui peuvent être mieux dans la réalisation des objectifs organisationnels.

Aussi, lorsque les PBI passent d'une équipe à une autre, plus d'informations doivent également être véhiculées sur la nature du diagnostic faite par l'équipe précédente. À ce niveau, il existe un besoin d'améliorer les communications entre les équipes et particulièrement entre celles qui sont géographiquement distribuées. Une formation qui met l'accent sur l'enseignement de la verbalisation du raisonnement derrière les décisions et la nécessité de fournir plus de détails au sujet des activités de diagnostic, devrait contribuer à améliorer considérablement la CS d'équipe.

5.7.3 Meilleure communication entre les chefs d'équipes

Les chefs d'équipes doivent renforcer leurs rôles de « personne ressource » et de relai d'information. Les chefs d'équipe devraient avoir des rencontres sur certains aspects de la gestion des travaux. Ils doivent recevoir une formation explicite sur la façon 1) d'exécuter une réunion pour transmettre les informations sur des objectifs communs des équipes, 2) fournir au groupe une compréhension commune de qui fait quoi, 3) renforcer le mécanisme actuel qui facilite une compréhension de la nature des

interrelations entre les tâches et les activités des développeurs et 4) fournir des attentes en matière de travail d'équipe.

Les chefs d'équipe doivent également recevoir une formation spécifique sur l'importance de transmettre des informations sur l'état du travail au sein des équipes au cours des périodes de transition (passage d'un cycle à l'autre). Sans ce type de rétroaction, les gens peuvent facilement perdre de vue comment ils progressent d'un cycle à un autre et pour des tâches qui chevauchent plusieurs cycles et impliquent plusieurs intervenants. Cette rétroaction est importante pour la performance individuelle et la CS, et aussi pour favoriser un esprit d'équipe dans la réalisation des activités.

Dans le processus en place actuellement, la rencontre de rétrospective durant laquelle les acteurs se demandent ce qui a fonctionné et ce qui a moins bien fonctionné au cours du cycle, constitue un instrument de communication efficace dont il faut tirer avantage. La formation peut recommander l'utilisation de cette même plateforme de communication comme lieu d'échange entre les chefs d'équipes. Une équipe peut donc inviter les autres chefs d'équipe à sa rencontre de rétrospective. Chacun peut y partager les leçons apprises dans sa propre équipe. Cette information est cruciale pour permettre aux membres des équipes d'avoir un bon modèle mental sur ce que chacun fait et comment les tâches sont inter-reliées afin qu'ils puissent avoir une bonne CS relative à l'impact de leurs actions et tâches sur d'autres personnes et sur l'objectif global de l'entreprise.

5.7.4 Feedback

Les développeurs ont exprimé le désir de savoir si les clients utilisent effectivement les fonctionnalités sur lesquelles ils ont travaillé et quelles en sont les appréciations. Actuellement, les DEV reçoivent peu de commentaires sur la suite d'une solution particulière déployée en production. Les commentaires arrivent au niveau des AT mais

ne descendent pas aux DEV. Un diagnostic particulièrement difficile suivi d'un développement peuvent être totalement couronnés de succès, ou peuvent ne pas fonctionner dans un autre environnement. À l'heure actuelle, il est très difficile de suivre le rendement d'une action. Cette rétroaction est cruciale pour le développement de meilleurs modèles mentaux sur les parties de l'application sur lesquelles ont travaillé les DEV. Sans cette rétroaction, il est très difficile d'améliorer les compétences de diagnostic.

Bien que des améliorations du système soient recommandées pour régler ce problème, il est également recommandé que le personnel soit formé pour fournir de telles évaluations. Non seulement les gestionnaires et les « leads » doivent jouer un rôle actif dans la fourniture de ce feedback, mais les DEV peuvent également être formés pour fournir plus de commentaires (soit lors des rencontres physiques ou par tout autre moyen de communication en leur possession) sur ce qui a fonctionné et ce qui n'a pas fonctionné.

5.7.5 Formation sur la CS

Des problèmes communs peuvent être liés à des défaillances de la CS dans un certain nombre de systèmes, y compris 1) des oublis d'informations ou des étapes, souvent en association avec des interruptions de tâches (développeurs qui se font interrompre dans leur travaux), 2) de ne pas transmettre des informations entre les membres de l'équipe (unité DEV), 3) les informations critiques manquantes due à d'autres distractions liées aux tâches, et 4) une interprétation erronée des informations en raison des attentes.

Une formation peut être utilisée pour fournir une prise de conscience de ces problèmes et les moyens de les combattre. Par exemple, les interruptions de tâches sont un problème commun menant à des erreurs CS. Souvent, de telles interruptions conduisent à des omissions d'étapes ou les activités manquantes. Les DEV ainsi que les « leads », les PO et les PM peuvent être formés pour prendre des mesures particulières à la suite

d'une interruption de tâche (par exemple, une double vérification des travaux antérieurs effectués, etc.). Ce type de formation peut être utile pour aider les DEV à s'assurer qu'ils ne manquent pas d'informations essentielles à l'accomplissement de leurs tâches.

5.8 Programme de formation sur la conscience de la situation d'équipe

5.8.1 Le contenu du cours

Le programme de formation que nous proposons sur la CS d'équipe est inspiré de celui proposé par Robertson et Endsley, 1997. Il est développé pour répondre aux cinq buts et objectifs de la formation sur la CS listés dans les sections précédentes :

- Modèles mentaux partagés,
- Contextualisation des décisions,
- Échanges d'informations,
- Rétroactions,
- Conscience de la situation individuelle

En outre, le cours offre une revue des meilleures pratiques techniques de base dans les divers domaines applicables à l'entreprise (Meilleures pratiques pour le SQL, meilleures pratiques pour le Business Intelligence). Les connaissances de ces techniques de bases sont considérées comme des exigences préalables pour les participants. Elles ne sont pas incluses dans le syllabus.

Du contenu original du programme de formation proposé par Robertson et Endsley, nous avons apporté de nombreux ajouts et adaptations sans toutefois nous éloigner de l'esprit du cours. En termes d'adaptation, la formation a été nommée « **Comment être productif en environnement complexe?** » afin de tourner l'attention vers la

productivité des équipes et la productivité individuelle. Aussi, nous l'avons structurée en 6 modules comme suit.

- Module 1 : Que signifie environnement complexe?
- Module 2 : Introduction à la Conscience de la situation
- Module 3 : Performance et obstacles à la performance
- Module 4 : Démarches de résolution de problème
- Module 5 : Les normes organisationnelles
- Module 6 : L'information et la perception

Le syllabus que nous avons proposé pour la formation est présenté dans l'annexe C.

Les modules 1, 2 et 3 sont propres à notre étude. Les modules 4, 5 et 6 ont été adaptés pour inclure des exemples, concepts et études de cas issus du monde de gestion de projets informatiques et de la vie réelle de l'entreprise.

Le module 1 explore les critères qui définissent la complexité de l'environnement de travail des participants. Souvent, la complexité de la gestion de projets informatiques dans les organisations n'est pas comprise ou est minimisée. L'objectif de ce module est d'amener les participants à prendre conscience de la complexité des opérations de leur environnement et à savoir identifier et interpréter les signaux de bonne ou mauvaise gouvernance.

Le module 2 présente les constituants de la CS dans le domaine de la gestion de projets informatiques. Il s'agit d'analyse des besoins en informations et des dépendances de tâches des différentes équipes tout au long du processus de production de l'entreprise.

Le module 3 analyse les problèmes et obstacles à la CS d'équipe et à la CS individuelle. Il s'agit d'exemples et d'études de cas issus de l'expérience particulière de l'entreprise.

Le module 4 traite des démarches de résolution de problème technique, de partage des résultats de diagnostics et de solutions; le travail en groupe et le travail en équipe; et les décisions collectives.

Le module 5 traite des normes et des procédures de travail. Les normes peuvent être aussi des règles non écrites mises en place par les équipes. Les normes jouent un rôle crucial dans le fonctionnement des organisations et ont un impact direct sur la performance des équipes et la qualité de la production.

Le module 6 traite de la question de la perception, des suppositions et opinions personnelles dans la résolution de problèmes opérationnels. Une supposition doit être testée soit par partage d'informations, soit par demande d'avis contraire, ou écoute attentive des autres.

Étant donné qu'il est difficile de rassembler toutes les équipes dans une même salle de classe pour plusieurs heures de cours magistral, le cours sur la CS d'équipe a été conçu pour être présenté comme une prestation en petit groupe et suivi par des rencontres de renforcement individuel pour le personnel de trois équipes de DEV (CLAIM, BI, DBA). Il s'agira d'un programme pilote qui pourrait, plus tard, être étendu à toutes les équipes de l'entreprise.

Il est préférable de dispenser le cours à une classe composée des participants de la même équipe pour les développeurs, et d'équipes distinctes pour ce qui est des chefs d'équipe. Les chefs d'équipe sont mis ensemble car le cours se concentre sur la réduction des écarts et des malentendus qui peuvent survenir entre les équipes, et des gaps dans les informations ou mauvaises communications qui surviennent entre ces différentes équipes.

Il est prévu que la majorité du bénéfice du cours viendra des interactions qui surviendront lorsque les participants partageront leurs points de vue et informations durant les rencontres.

5.8.2 Le support de cours

Un ensemble de diapositives PowerPoint couvrant les six notions de la CS d'équipe avec des exemples de projets de développement et des études de cas sont inclus dans le cours pour encourager l'apprentissage actif.

5.8.3 Les stratégies d'enseignement

La stratégie d'enseignement utilisée pour le cours fait recours à la méthode d'apprentissage des adultes « Discovery learning ». Dans ce type d'apprentissage, une réponse exacte n'est pas donnée au participant, mais il reçoit plutôt le matériel et les indices minimum pour trouver les réponses lui-même. Le « Discovery learning » a lieu dans des situations de résolution de problèmes où l'apprenant se fonde sur sa propre expérience et ses connaissances préalables. Il s'agit d'une méthode d'enseignement à travers laquelle les participants interagissent avec leur environnement en explorant et en manipulant des objets, aux prises avec des questions et des controverses, ou réalisent des expériences.

L'organisation de l'ensemble du cours permet d'atteindre un haut niveau d'interaction et de participation parmi les participants et crée ainsi un processus d'apprentissage expérientiel. Le cours sur la CS d'équipe encourage fortement la participation à la résolution de problèmes, à des réponses aux questions ouvertes, favorisant l'acquisition et le traitement de l'information.

5.9 Conclusion

Aux nombreux problèmes relevés dans notre analyse, nous avons avancé un certain nombre de solutions à la fois (a) sur le plan stratégique, en proposant des changements organisationnels (redéfinition des rôles; adoption de nouveaux outils de communication; révision du processus de production) et (b) sur le plan de la formation

sur la CS d'équipe. Les buts et objectifs de ces approches de solutions sont l'amélioration de la CS des individus et la CS des équipes étudiées afin d'aider les équipes à se rapprocher davantage de leurs propres objectifs de performance.

Toutefois, une réingénierie des processus peut avoir des impacts considérables sur le fonctionnement organisationnel et sur le personnel, et il est nécessaire que le personnel ait conscience de ce fait. Aussi, comme mentionné précédemment, il est documenté que les effets de telles interventions sur l'entreprise peuvent être non prédictibles. L'étude des mécanismes de gestion des impacts d'un changement organisationnels sont cependant, hors de la portée de la présente thèse.

Nous ne nous attendons pas à introduire des changements drastiques dans l'entreprise en faisant appliquer les recommandations de façon immédiate sur une courte période. Pour cette raison, et pour les besoins de notre recherche, il ne nous sera possible que d'évaluer de façon subjective la **valeur** et les **utilités attendus** de la formation. Pour les changements organisationnels qui sont immédiatement appliqués, il nous sera possible de proposer des mesures plus objectives. Ces éléments de mesures seront abordés dans le chapitre 6.

CHAPITRE VI

ÉVALUATION DE LA SOLUTION PROPOSÉE

6.1 Introduction

Ce chapitre traite de l'application et de la mesure des recommandations issues du chapitre 5. Il s'agit ici de notre 2ème travail de terrain au cours duquel nous avons eu à solliciter une fois de plus, la participation des employés de l'entreprise étudiée. Nous présentons ici, la méthodologie d'évaluation que nous avons utilisée ainsi que les résultats et leurs interprétations.

6.2 Reformulation des objectifs et impacts de la formation

Les objectifs de la formation étaient de doter les participants des techniques et habiletés pour la compréhension de ce que constitue la CS d'équipe et son importance dans l'entreprise conformément à nos objectifs de recherche. En termes de retombée immédiate, la formation permettra aux participants de faire :

- 1) Une meilleure utilisation des ressources en personnel et en technologie pour une communication et coordination efficace;
- 2) Une meilleure compréhension des objectifs de l'entreprise et des objectifs de chaque équipe;
- 3) Une meilleure compréhension des effets des normes, pratiques et processus de production, sur les objectifs et la performance;
- 4) Une amélioration des capacités des relations interpersonnelles.

6.3 Méthode d'évaluation

Les actions organisationnelles ainsi que la formation sur la CS d'équipe ont été livrées à l'entreprise, au lieu de travail des équipes de développeurs. Le personnel des équipes de DEV, et de support n'avait jamais reçu de formation sur la CS d'équipe. On peut donc considérer qu'ils n'ont pas de connaissances spécifiques sur ce qui constitue la CS d'équipe et ses bénéfices sur le travail d'équipe avant ce programme de formation.

La prestation a été développée pour 4 h et prévue pour être dispensée sur 2 séances en petit groupe pour permettre plus d'exercices de groupe, des interactions, et des études de cas tirés des exemples réels de l'entreprise.

6.3.1 Échantillonnage

Il ne nous a pas été possible de dispenser la formation à l'ensemble des équipes de l'entreprise, vu les exigences en moyens logistiques requis par un projet d'une telle envergure. Nous avons circonscrit le programme à l'étape de prototype en travaillant spécifiquement avec un petit groupe de participants assez représentatif de la dynamique de l'entreprise. Douze personnes provenant de 3 équipes différentes de DEV ont participé aux séances de formation qui fait l'objet de cette évaluation. La participation à la formation était volontaire ainsi que la participation à son évaluation. Les participants provenaient majoritairement des équipes de DBA, BI et CLAIM. Les développeurs d'application étaient les plus représentés.

Les participants étaient très expérimentés dans leur travail au sein de l'entreprise (en moyenne ils ont 7 années d'expérience dans leur domaine respectif)

6.3.2 Le processus d'évaluation

Le processus d'évaluation contenait trois niveaux :

- 1) Mesure de la valeur et de l'utilité de la formation;
- 2) Mesure des comportements auto-déclarés pré/post formation;

3) Mesure des changements dans le comportement au travail.

Dans la littérature sur la CS, les plus récentes études consacrées à la mesure des effets d'une formation en CS d'équipe sur la performance des équipes sont celles de Taylor et Robertson (1995), de Endsley et Robertson (2000) (nous en avons parlé dans la section 1.4 sur les objectifs de notre thèse), et de O'Connor (2013). De ce fait, la méthodologie d'évaluation que nous appliquons dans la présente étude est largement inspirée de celles proposées par ces auteurs.

- Taylor et Robertson(1995) ont évalué les effets d'une formation en CS de l'individu et les modes de communication en milieu organisationnel sur la performance de l'équipe. Dans ce programme dénommé CRM (Crew Resource Management) réalisé en 1994 à la NASA, les auteurs décrivaient trois années d'évaluation de l'effet de la formation en CRM sur l'équipe d'opération de vol et de maintenance d'aéronef. L'évaluation consistait à mesurer les attitudes post-formation des gestionnaires et techniciens, leurs comportements auto-déclarés ainsi que les performances à la tâche de leur unité de travail. Les résultats ont révélé un impact notable sur la performance.
- Une démarche similaire à celle du CRM a été utilisée par Endsley et Robertson (2000) dans leur programme « *Team SA training* », pour l'évaluation de la formation sur la CS d'équipe, toujours dans le domaine de l'aviation, auprès des équipes de maintenance d'aéronef.
- O'connor (2013), dans une étude intitulée « *Excuse me: teaching junior doctors to speak up* », a pour sa part appliqué la même méthodologie que le CRM, dans un programme de formation de jeunes médecins sur des aptitudes non-techniques telles que la CS, la communication et la prise de décision.

Dans cette thèse, nous examinons les attitudes des participants, c'est-à-dire, celles qui sont auto-déclarées. Par attitude, on entend celles mesurées immédiatement avant et après la formation ainsi que les attitudes des participants 1 mois après la formation.

Afin d'évaluer l'efficacité de la formation et des changements organisationnels, une évaluation initiale et une post-évaluation ont été menées auprès des mêmes participants :

- 1) Un questionnaire d'évaluation a été fourni avant la formation pour évaluer les connaissances et les comportements des participants, en relation à leur CS;
- 2) Un questionnaire a été administré immédiatement après la formation pour obtenir les opinions subjectives des participants sur sa valeur et son utilité et pour mesurer les changements dans les attitudes selon les déclarations des participants. Le niveau de connaissance acquise en termes d'attitudes et de comportements liés à leur CS a également été mesuré;
- 3) Le formulaire a été administré à nouveau un mois plus tard pour évaluer les changements de comportement au travail.

6.3.2.1 Usage des réponses individuelles des participants.

Il n'y avait pas eu de question de groupe. Toutes les réponses recueillies ont été exprimées par chaque participant de façon individuelle. A l'aide de l'identification des nom et prénom du participant, nous avons été en mesure d'apparier les réponses pré et post formation de chacun des répondants.

6.3.2.2 L'analyse des données de l'équipe

Lorsque nécessaire, les données individuelles recueillies ont été regroupées par équipe ou pour l'ensemble des répondants en faisant abstraction de l'individu. Dans ce cas, les

chiffres avec lesquels nous avons travaillé sont les moyennes des points attribués aux répondants pour chacune des questions à échelle.

6.3.2.3 Le questionnaire d'évaluation

Le questionnaire d'évaluation utilisé est une adaptation du CRM/TOQ proposée par Taylor et Robertson (1995). En choisissant de concevoir notre questionnaire sur la base du CRM/TOQ, nous estimons obtenir une « bonne » mesure des effets de la formation en CS d'équipe puisque le questionnaire CRM/TOQ a déjà été validé comme un outil de mesure efficace pour les formations sur la CS. Une analyse factorielle effectuée sur les questions du questionnaire a révélé qu'elles peuvent être traitées comme un ensemble d'éléments indépendants. Aussi, toutes les mesures du questionnaire du groupe « Questions sensibles à la formation » sont réputées être sensibles à la formation, c'est-à-dire que si la formation a un impact sur les participants, il devrait être reflété dans les réponses fournies par ces derniers.

Les changements que nous avons opérés dans le questionnaire CRM/TOQ consistaient à son ajustement par rapport au domaine spécifique de l'informatique et au cas particulier des équipes en observation.

Dans le reste de cette thèse, nous utilisons le terme « questionnaire d'évaluation de la CS d'équipe informatique » (QECSI) lorsque nous faisons référence à notre questionnaire.

Le questionnaire QECSI comporte 3 catégories de thèmes et 6 thèmes organisés de la manière suivante:

Questions sensibles à la formation	Numéro de Question
- Partage des responsabilités de gestion	6, 8, 11, 13, 19
- Communication et coordination	5, 12, 14, 16, 17
- Gestion du stress	9, 20
- Affirmation de soi	1,2
Questions non sensibles à la formation	Numéro de question
- Partage des objectifs de son propre équipe	21, 22, 23, 24
- Partage des objectifs des autres équipes	25, 26
Questions sur l'expérience de la formation	27 à 37

Au total, le questionnaire QECSI contient 20 éléments à réponse multiple dans la catégorie « Questions sensibles à la formation »; 6 dans la catégorie des « Questions non sensibles à la formation » et 10 sur l'évaluation du déroulement de la formation. Les questions de la catégorie « Questions sensibles à la formation » se rapportent spécifiquement aux thèmes abordés dans la formation. Dans le thème « Communication et coordination » de cette catégorie, nous avons retiré 2 questions du questionnaire original de Taylor et Robertson car elles sont trop spécifiques au domaine de l'aviation. Le retrait était nécessaire parce que le travail de gestionnaire de projet informatique diffère de celui de la maintenance d'aéronef notamment en termes d'objectif de performance à atteindre. Nous les avons remplacées par les questions No5 et No18. En termes de pratique de recherche dans le domaine de l'évaluation des formations CRM, il est fréquent de voir des variations mineures au questionnaire CRM/TOQ y compris à l'intérieur du milieu de l'aviation où les formations CRM ont fait leur apparition il y a près de 3 décennies (Salas *et al.*, 2006)

Le questionnaire QECSI complet est présenté dans l'annexe B

6.3.3 Comparaison des données pré et post formation

Pour la mesure de la compréhension des concepts de CS d'équipe pré et post formation, nous avons eu recours au Wilcoxon signed-rank test. Le Wilcoxon signed-rank test est un test d'hypothèse statistique non paramétrique utilisé lorsque l'on compare deux échantillons connexes, appariés, ou des mesures répétées sur un seul échantillon.

Ici, nous utilisons le Wilcoxon signed-rank test pour comparer deux ensembles de scores qui viennent des mêmes participants sur la période pré et post formation. Cela nous permet donc de noter tout changement statistiquement significatif dans les scores sur un intervalle de temps.

Le Wilcoxon signed-rank test nous permet de répondre à la question de savoir s'il y a eu effectivement une différence dans la compréhension des concepts de CS et les intentions de leur application au travail quotidien avant et après le programme de formation. Dans ce cas, notre variable dépendante est « les attitudes relatives à la CS d'équipe ». Nos deux groupes connexes sont les valeurs de ces attitudes CS auprès de notre groupe cible **avant** et **après** le programme de formation sur la CS.

Nos variables dépendantes sont mesurées à l'ordinal sur une échelle de 1 à 5 ou 1 représente une valeur faible et 5 représente une valeur forte.

Nous disposons de 2 groupes connexes indiquant que les mêmes sujets sont présents dans les deux groupes. La raison pour laquelle il est possible d'avoir les mêmes sujets dans chaque groupe est que chaque sujet a été mesuré deux fois sur la même variable dépendante.

6.4 Résultat de l'évaluation

Nous avons eu un taux de réponse de 100% sur les questionnaires remis aux participants. Il a été possible de réaliser un tel résultat parce que nous avons demandé aux participants de fournir leurs réponses lorsqu'ils étaient encore présents dans la session de formation. Les résultats sont présentés ci-dessous selon le processus d'évaluation proposé à la section 6.3.2

6.4.1 Valeur et utilité

L'évaluation post-formation a été utilisée pour mesurer le niveau d'utilité et la valeur perçue de la formation. Les participants ont marqué chaque section du cours sur une échelle de cinq points, qui variait de 1-perte de temps à 5-extrêmement utile.

Les participants présents à la formation ont montré un enthousiasme général pour l'idée de cette formation et ont exprimé une satisfaction unanime d'avoir pu y prendre part. Nous avons tiré ces conclusions en analysant les résultats fournis pour les questions d'ordre général sur la formation, à savoir « dans quelle mesure la formation sur la CS d'équipe changera votre comportement au travail? » et « dans quelle mesure la formation sur la CS d'équipe serait utile pour les autres employés? ». La note moyenne pour le cours était globalement de 4,3, correspondant à mieux que « très utile ». 89% des participants ont considéré que le cours était soit « très utile » ou « extrêmement utile », ce qui représente un niveau élevé d'enthousiasme pour le cours. Aucun des participants n'a évalué le cours comme « une perte de temps ». Il n'y avait donc pas de faible note pour le cours dans son ensemble. Plus de 91% estiment que le cours serait très ou extrêmement utile aux autres (avec une note moyenne de 4.2).

Plus de 92% des participants ont estimé que le cours était soit « très utile » ou « extrêmement utile » pour accroître la qualité des applications et l'efficacité du travail en équipe. Sur ce sujet, la note moyenne est de 4.6. En moyenne, ils ont évalué chacun

des grands sujets abordés durant la formation comme « très utile » (les scores moyens sont compris entre 3.8 et 4.8).

La figure 6.1 montre la distribution des fréquences recueillies pour l'évaluation de l'ensemble de la formation.

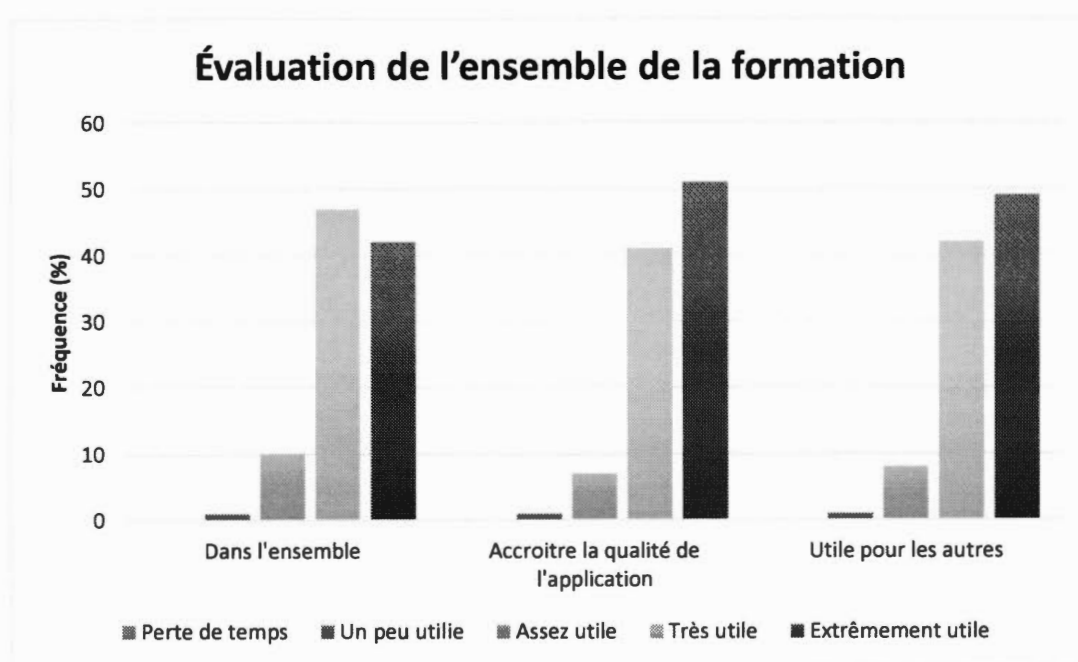


Figure 6. 1 Évaluation de l'ensemble de la formation par les participants.

6.4.2 Changement dans les comportements et attitudes

Il faut préciser ici que la mesure des comportements et attitudes est basée sur les intentions de changement exprimées par les participants à travers les questions d'auto-évaluation de leurs attitudes et comportements. Lorsqu'on leur demande dans quelle mesure les acquis de la formation pourraient avoir une incidence sur leur comportement au travail, 86% estimaient qu'ils feraient un « changement modéré » ou « un grand

changement », comme le montre la figure. 6.2. Évidemment pour cette question, nous ne disposons que des données post-formation.

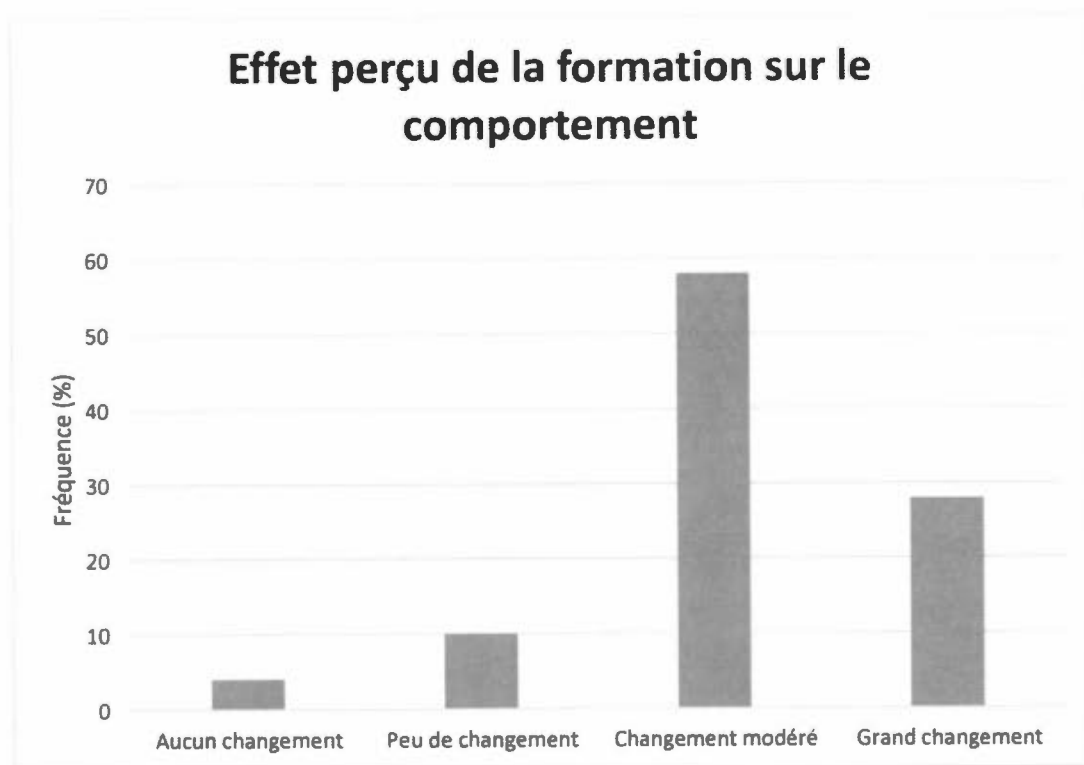


Figure 6. 2 Effet perçu de la formation sur le comportement des participants

Le changement moyen dans l'évaluation post-formation par rapport à l'évaluation pré formation sur chaque comportement décrit dans le formulaire d'auto évaluation de la mesure comportementale pré / post a également été évalué. Les modifications sur chaque sujet ont été comparées pour chaque répondant en utilisant une analyse de comparaison par paires (pré formation et post formation). L'analyse statistique non paramétrique de Wilcoxon a révélé que les comportements auto-déclarés ont changé de manière significative sur 9 des 26 sujets évalués ($p(0,05)$).

6.4.2.1 Amélioration de la CS dans l'ensemble des équipes

Les participants ont signalé après la formation qu'ils seraient plus susceptibles de garder informés les membres des autres équipes ainsi que ceux de leurs propres équipes, sur le statut de leur travail à chaque étape (une augmentation de 19%).

Ils étaient également sensiblement plus susceptibles de suivre le statut des activités sur lesquelles les autres ont travaillé dans le cycle (une augmentation de 11%).

Ils étaient aussi plus sensibles à la question « Lorsqu'un employé d'une autre équipe est en désaccord avec les buts et priorités établies pour votre équipe, peut-il manifester son désaccord? ». Ici, nous avons eu une augmentation de 12%.

6.4.2.2 Développement de meilleurs modèles mentaux partagés en ce qui concerne d'autres équipes

Les participants ont indiqué qu'ils seraient plus enclins à comprendre les points de vue des autres lorsqu'ils sont engagés dans un désaccord avec d'autres équipes (soit une augmentation de 12%).

Les participants seraient d'avis de participer à la résolution des problèmes dans d'autres équipes s'ils sont sollicités (soit une augmentation de 14%)

6.4.2.3 Meilleure communication

Les participants ont signalé des changements dans plusieurs comportements liés à l'amélioration des communications et au travail d'équipe. Entre autres :

- Ils étaient plus susceptibles d'améliorer leur communication écrite en documentant davantage les tâches qui leurs sont assignés (une augmentation de 21%).

- Ils étaient plus susceptibles de documenter plus amplement leur contribution dans une tâche avant qu'elle ne soit livrée à d'autres équipes (une augmentation de 36%).
- Ils étaient également plus susceptibles de s'assurer que tous les problèmes et activités soient discutés lors des réunions de planning (une augmentation de 23%).
- Enfin, il y a eu une augmentation de 12% lorsqu'il s'agit d'encourager les autres à donner leur point de vue sur leurs préoccupations ou problèmes en ce qui concerne le fonctionnement du processus de production, lors des réunions de planning.

Les résultats du test de Wilcoxon sont présentés dans la figure 6.3 pour la variable de communication « Dans quelle mesure accepteriez-vous de documenter plus amplement votre contribution dans une tâche avant qu'elle ne soit livrée à d'autres équipes? ». Le résultat est significatif à $p \leq 0.05$ pour $n=12$.

Treatment 1	Treatment 2	Sign	Abs	R	Sign R
3	4	-1	1	4	-4
3	5	-1	2	9	-9
4	4	n/a	0	n/a	n/a
2	3	-1	1	4	-4
3	4	-1	1	4	-4
4	5	-1	1	4	-4
2	4	-1	2	9	-9
3	5	-1	2	9	-9
3	4	-1	1	4	-4
4	5	-1	1	4	-4
2	2	n/a	0	n/a	n/a
3	4	-1	1	4	-4

Significance Level:

☐ 0.01

☒ 0.05

1 or 2-tailed hypothesis?:

☐ One-tailed

☒ Two-tailed

Result Details

W-value: 0

Mean Difference: -2.4

Sum of pos. ranks: 0

Sum of neg. ranks: 55

Z-value: -2.8031

Mean (W): 27.5

Standard Deviation (W): 9.81

Sample Size (N): 10

Result 1 - Z-value

The Z-value is -2.8031. The p-value is 0.00512. The result is significant at $p \leq 0.05$.

Result 2 - W-value

The W-value is 0. The critical value of W for N = 10 at $p \leq 0.05$ is 8. Therefore, the result is significant at $p \leq 0.05$.

Figure 6. 3 Résultats du test statistique non paramétrique de Wilcoxon

6.4.2.4 Réponses aux questions ouvertes concernant l'intention d'utilisation des acquis de la formation

Une question ouverte a été posée aux participants concernant leurs intentions d'utiliser les acquis de la formation et leurs applications dans leur travail. Cette question, uniquement incluse dans le questionnaire post-formation (Annexe B, Q V), a été traitée comme des auto-évaluations de la perception de l'intention et de changement de comportement au travail.

Les participants ont été invités à écrire leurs réponses à la question « Comment comptez-vous utiliser les acquis de cette formation dans votre travail? ». La codification des réponses à cette question a donné les trois principales catégories de réponses suivantes : 1) « Être un meilleur auditeur » 2) « Être plus conscients des autres » et, 3) « Traiter mieux avec les autres ». Ces trois catégories représentent deux types de changements potentiels de comportement auto-perçu : passive et active. Nous qualifions les 2 premières catégories comme des comportements « passifs ».

Des améliorations intentionnelles et voulues sont faites à la personnalité des répondants, comme, « Je suis sensibilisé que la qualité de mon travail affecte les autres » ou « J'accepte plus les opinions des autres ».

Inversement, l'autre catégorie, classée comme des comportements « actives », contient des comportements désirés tels que, « mieux communiquer en documentant les résultats de mes investigations sur les tâches qui me sont assignées », ainsi que « utiliser plus fréquemment les périodes de planning pour la résolution de problèmes », ou, « mieux interagir avec les autres ».

De façon générale, il ressort que les participants ont montré de meilleures intentions d'apporter des changements dans leur comportement au travail suite à la formation. Ces intentions de changements ont rapport avec l'amélioration de la CS au travail à

travers l'amélioration de la communication intra-équipe et entre les équipes, ainsi que la collaboration avec les autres équipes.

6.5 Changements dans le travail

Un mois après la formation, nous avons administré le même questionnaire aux participants afin de vérifier si les changements qu'ils ont prévu incorporer dans leur habitude de travail ont été effectivement appliqués. 100% des participants étaient encore dans leur emploi après la session de formation et avaient répondu aux questions. Durant la même période de 1 mois, nous avons également fait de nombreuses observations des procédures de travail et des comportements des mêmes participants afin d'identifier les points de changement. Pour ces participants (n=12), nous avons utilisé le test de Wilcoxon pour effectuer une comparaison par paires des réponses sur chacun des mesures du questionnaire post-formation et le questionnaire de 1 mois. Au niveau de signification de 0.05, cette analyse n'a révélé aucun changement dans les réponses obtenues. Ces réponses, soutenues par des observations indiquent que les comportements que les participants avaient l'intention d'adopter, ont effectivement été réalisés dans la pratique, au moins pour ce petit échantillon. Nous devons cependant souligner que les chefs d'équipe et superviseurs de l'entreprise ont fortement contribué en introduisant dans le processus de travail certains changements de procédures suggérés.

Dans les tableaux 6.1 à 6.4, nous avons répertorié quelques exemples de changements que nous avons observés.

Exemple 1 : Échanges d'information entre équipes sur l'état des applications.

Le tableau 6.1 est un exemple de définition des « builds » préparé par le responsable QA de BILLING et envoyé à toutes les autres équipes de DEV afin de les mettre au courant de la version la plus récente de l'application BILLING et de ses sous-

composantes. Les informations contenues dans ce tableau adressent les problèmes liés aux échanges de données entre applications.

Tableau 6.1 Définition des « builds » de BILLING

From : Billing team
Please use the following builds as the official builds of last sprint
Billing Mainline: 1.0.1.195
Billing Support: 1.0.1.048-HF.038
Policy QA Alpha: 2.4.1.204
DocumentManagment-Mainline: 1.0.0.046
PaymentGateway : 1.0.0.59

Exemple 2 : Échanges d'information entre équipes sur l'état des applications.

Le tableau 6.2 est un exemple de définition des « builds » préparée par le responsable QA de CLAIM et envoyée à toutes les autres équipes de DEV afin de les mettre au courant de la version la plus récente de l'application CLAIM et de ses sous-composantes. Les informations contenues dans ce tableau adressent les problèmes liés aux échanges de données entre applications.

Tableau 6.2 Définition des « builds » de CLAIM

From : Claims team Hi All, Please use the following build as the Official Claims Build of Sprint 14 Claims Mainline: 1.0.0.334 Latest builds that are compatible with this release: Policy QA Alpha: 2.4.1.204 + Billing: 1.0.1.165 + Notes and Follow-Ups: 1.0.0.024 + Document Management: 1.0.0.022 + Foundation IA: 1.0.1.034 + **Please note that all build numbers (except for the Claims build) are not necessarily official builds, those build numbers show the minimum build number to be used in order to get the new Claims functionalities. See with the corresponding teams for official build numbers. **
--

Exemple 3 : Changement organisationnel : mise en commun des rencontres de planning des équipes de développement (CLAIM et BILLING)

Le tableau 6.3 représente une note d'information envoyée par le directeur de produit aux membres des équipes CLAIM et BILLING, les informant de la fusion des 2 équipes. Cette décision a permis à chaque individu de mieux connaître les 2 applications et de réduire ainsi les erreurs liées à la dépendance des 2 applications.

Tableau 6.3 Note sur la fusion des équipes CLAIM et BILLING

Hi,

In order to improve the operational efficiency of the billing and claims scrum teams, we've merged the two teams into one effective this sprint. This will allow us to have a dedicated prod support resource to deal with triage items that may come up during sprint, and also spread the knowledge of the two modules across more people facilitating backups when people are out on vacation.

As a result, we've consolidated the two sprint reviews into one meeting and I'll be sending out the sprint commit list in one email. We're still working on a consolidated view for the team to work with, so this time I'm going to list out billing and claims in explicit lists – next go around it should be one list (with an indication of what is billing vs. claims).

Billing pts committed: 16.5 pts

ID	Work Type	Item	Title
96946	Bug		[CLIENT] PROD - Invoice outstanding amount not reduced as a result of receipt application
85596	Product Backlog Item		[Claims 91586] As Amy, I want to be able to retrieve custom claim field data for document generation
84056	Product Backlog Item		As Mary, I want to configure when automated reports should run

Tableau 6.3 (Suite)

Claims pts committed: 17.5 pts			
Note for claims: The mid-sprint items listed below are the remaining pieces for our initial go-live. This build will become our support branch while the rest of the features will go to mainline for a future release.			
ID	Work Type	Item	Title
97298	Product Backlog Item		Upgrade to foundation IA Client API
96900	Bug		[APP] Phase 1: Add & New Button does not add multiple documents
91586	Product Backlog Item		[Billing 85596] As Amy, I want to be able to retrieve custom claim field data for document generation
If you have any questions / concerns, please do not hesitate to follow up with me.			
Thanks,			
D			

Exemple 4 : Ajout d'informations de contexte dans les requêtes de travail.

Le tableau 6.4 est un exemple de requête de tâche à effectuer (PBI) envoyée à l'équipe DBA. Cette requête est documentée et comporte des informations sur les motivations du demandeur et le contexte général du problème. Ceci résout le problème de dépendances de tâches qui existait entre les développeurs et les gestionnaires de compte.

Tableau 6.4 Exemple de requête de tâche à effectuer (PBI)

Change Request(s):

(1) Change the assured company attached in the endorsement record for Policy [PLOCICY NUMBER] from [ASSURED NAME 1] (CompanyID: 174960) to [ASSURED NAME 2] (CompanyID: 6074711) both in Submission and Underwriting. Make sure to change it to all associated shipments and invoices, if there are any.

SubmissionID: 5684231

PolicyID: 2988431

****If this needs to be ran in UAT Support the assured company id to attach is 6041981**

(2) Change the created date from August 4, 2015 to August 13, 2015 for the policy event record where policy type is Endorsement. For the Endorsement. Change the Status from Excluded to Pending.

(3) Generate the IVOS file for August 13, 2015. [CLIENT] wants to test this first before going to production.

Background:

The user changed the name of the assured company from [ASSURED NAME 1] to [ASSURED NAME 2] instead of creating a new one. This is now causing an error in their downstream system because the ID associated to [ASSURED NAME 1] is the same with [ASSURED NAME 2], hence the system is saying that it does not match since this ID is for [ASSURED NAME 1]. We are only updating the last endorsement record because this is the endorsement to change the assured. Unfortunately, user did not create a new assured company prior to endorsing but attached the Eastman company that was renamed.

We also need to send the information downstream which is why we are requesting to change the created date of the policy event record. This will trigger the integration to include this policy in the August 11 file.

6.6 Discussion sur l'évaluation de la formation

Au lieu de demander aux employés de l'entreprise « De quelle formation avez-vous besoin? », la première étape de notre démarche a plutôt consisté en une analyse des besoins de formation par la réalisation d'un diagnostic élaboré de ce qui doit être enseigné, pour qui, et dans quel type de système d'organisation. Les résultats de cette

étape nous ont permis 1) d'anticiper les résultats de l'apprentissage, 2) de générer un guide pour la conception de la formation et sa livraison, 3) d'avoir des idées pour l'évaluation de la formation, et 4) d'avoir des informations sur les facteurs organisationnels qui impacteront négativement ou positivement l'efficacité de la formation.

Les observations que nous avons faites à la suite de la formation et de son évaluation, notamment concernant les améliorations introduites dans les communications entre les équipes, la documentation des travaux, les relations inter équipe, ainsi et les résultats observés sur la performance, nous amènent à dire que la formation sur la CS d'équipe a très bien réussie. Ce constat est soutenu par l'analyse des données qui montre que les méthodes et les moyens de formation, y compris les études de cas, les mises en situation, ont réussi et constituent un support intéressant dans l'atteinte des objectifs d'apprentissage. La stratégie d'enseignement mise en avant dans cette formation, basée sur le partage d'expériences, a permis à chacun de tirer le meilleur parti des expériences individuelles de chaque participant. Elle a permis à chacun de se faire une représentation de ce qui constitue le processus de travail des autres équipes et les difficultés qu'elles rencontrent dans l'atteinte de leurs objectifs.

Les cours ont été administrés à des équipes de développeurs d'application assez expérimentés et qui possèdent des compétences et expériences diverses dans le domaine des projets informatiques. Les participants ont particulièrement apprécié un tel effectif et l'ont considéré comme un ingrédient essentiel dans son succès. Même si les participants étaient issus d'équipes distinctes, structurellement cloisonnées, la familiarité et les relations sociales existantes entre les membres ont permis à chacun de partager plus facilement ses propres expériences de travail avec le reste du groupe, contribuant ainsi à l'élaboration de modèles mentaux partagés.

Le contenu du cours, associé à tous les grands objectifs de la formation, a été bien apprécié par la grande majorité des participants qui ont évalué chacune des sections comme « très utile » ou « extrêmement utile ».

Sur la question de la valeur perçue de la formation sur la qualité des applications, le cours a été considéré entre « très utile » et « extrêmement utile » dans l'ensemble. Les participants étaient prêts à la recommander à d'autres collègues puisqu'à la question « Dans quelle mesure une telle formation peut être utile à d'autres? », la formation a été jugée « très utile » pour les autres.

D'autres analyses montrent qu'il y a eu des améliorations significatives dans les comportements tels que « partage de l'information et résolution de problème dans l'équipe », « participation à la résolution de problème en équipe », « partage des expériences et résolution de problème entre équipes », « documentation des travaux » et « communication et coopération ». Les mêmes comportements mesurés à intervalle d'un mois ont révélé que ces comportements sont restés au même niveau. Les résultats du test de Wilcoxon sur les évaluations comparatives pré/post formation sont également des preuves convaincantes de l'effet positif de la formation sur la CS d'équipe. Les réponses à la question ouverte « comment comptez-vous utiliser les acquis de la formation dans votre travail? » confirment la signification réelle des chiffres obtenus des évaluations post-formation. L'influence de la formation sur les attitudes des participants peut être considérée comme stable et solide du moins sur une période d'un mois.

Il y a une similitude entre les résultats de notre étude et ceux obtenus à la suite du programme de formation CRM qui a été couronné de succès (Taylor et Robertson, 1995). Il y a également similitude avec les résultats obtenus par Endsley et Robertson (2000) auprès des techniciens de maintenance d'aéronef qui ont participé à l'étude sur la CS d'équipe *Team SA Training*. Dans le premier cas, près des deux tiers des participants issus des groupes d'opérateurs aériens ont indiqué que la formation CRM

était « très utile » ou « extrêmement utile ». Dans le cas des résultats de *Team SA Training*, c'est près 89% du personnel de maintenance formé qui l'ont trouvé « très utile » ou « extrêmement utile ».

Les figures 6.4 et 6.5 illustrent le niveau d'acceptation du *Team SA Training* par les participants de l'équipe de maintenance tel que mesuré immédiatement après la formation. Cet enthousiasme se compare à la réponse recueillie dans notre étude pour la formation sur la CS d'équipe pour une entreprise informatique :

- La figure 6.4 fait une comparaison des changements de comportement perçus de cette formation et celle de *Team SA Training* de Endsley et Robertson (2000). L'échantillon dans l'étude de Endsley et Robertson est beaucoup plus grande comparé au nôtre.
- La figure 6.5 compare l'enthousiasme pour cette formation et celle de *Team SA Training* de Endsley et Robertson (2000).

Dans la présente évaluation, 91% des participants ont trouvé que le cours est « très utile » ou « extrêmement utile » et sont prêts à la recommander aux autres. Sur la base de ces résultats, nous pouvons conclure que la formation sur la CS d'équipe en milieu informatique est considérée comme très utile et a un impact favorable sur la CS d'équipe.

Les résultats présentés dans ce chapitre sont basés sur un échantillon assez petit issu des équipes informatiques de l'entreprise. Même si nos conclusions se comparent assez bien à celles d'autres études similaires, elles seraient davantage validées et consolidées si l'étude pouvait être élargie à l'ensemble de l'entreprise en y incluant toutes les équipes de développeurs ainsi que les équipes connexes avec lesquelles les développeurs entretiennent des relations stratégiques. Aussi, l'évaluation peut être

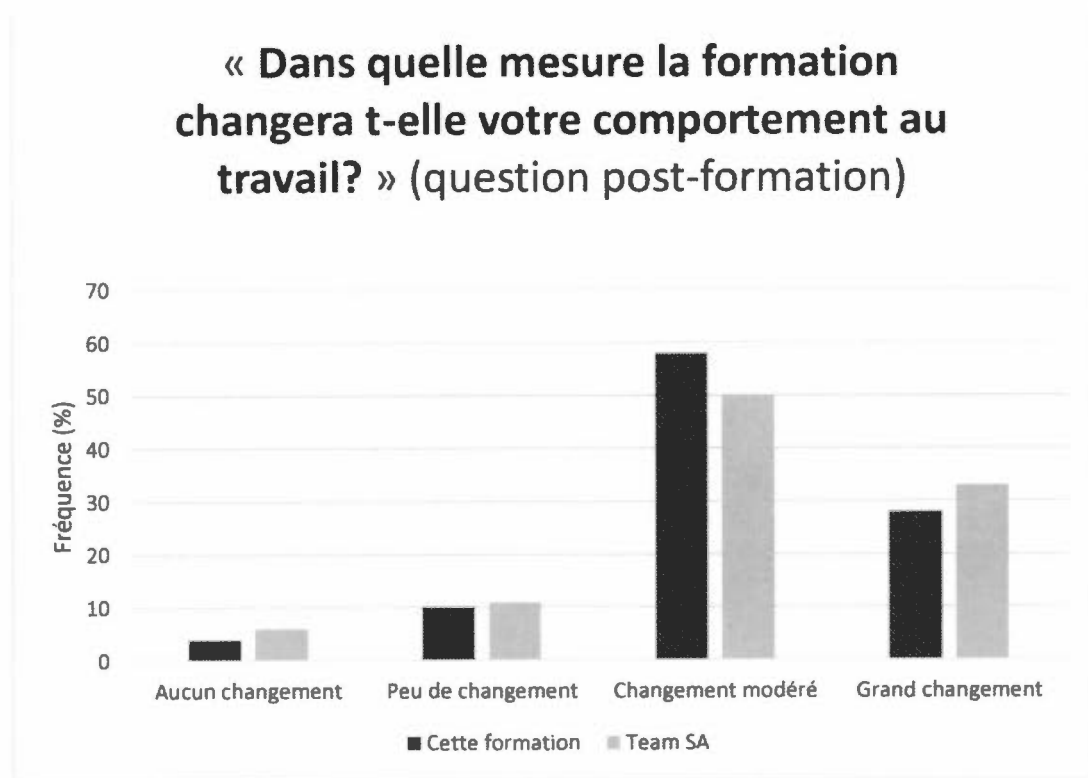


Figure 6. 4 Comparaison des changements de comportement perçus

davantage étendue sur une période plus vaste que celle de 1 mois. Une évaluation de suivi pourrait donc être souhaitable pour valider les résultats des changements de comportement sur le lieu de travail. Dans leur recherche sur la formation CRM dont nous avons fait référence dans cette section même, Taylor et Robertson arrivaient à une conclusion similaire en ces termes :

« Associations between positive post-training attitudes and subsequent performance were even stronger when attitudes were measured at 2 months and at 12 months after the training. Overall, the survey results from these 2 month and 12 month follow-up surveys are remarkably strong and indicate that lessons learned from the CRM training consolidate and strengthen in the months afterward, and in more "active" or practical ways. » (Taylor et Robertson, 1995, page 51)

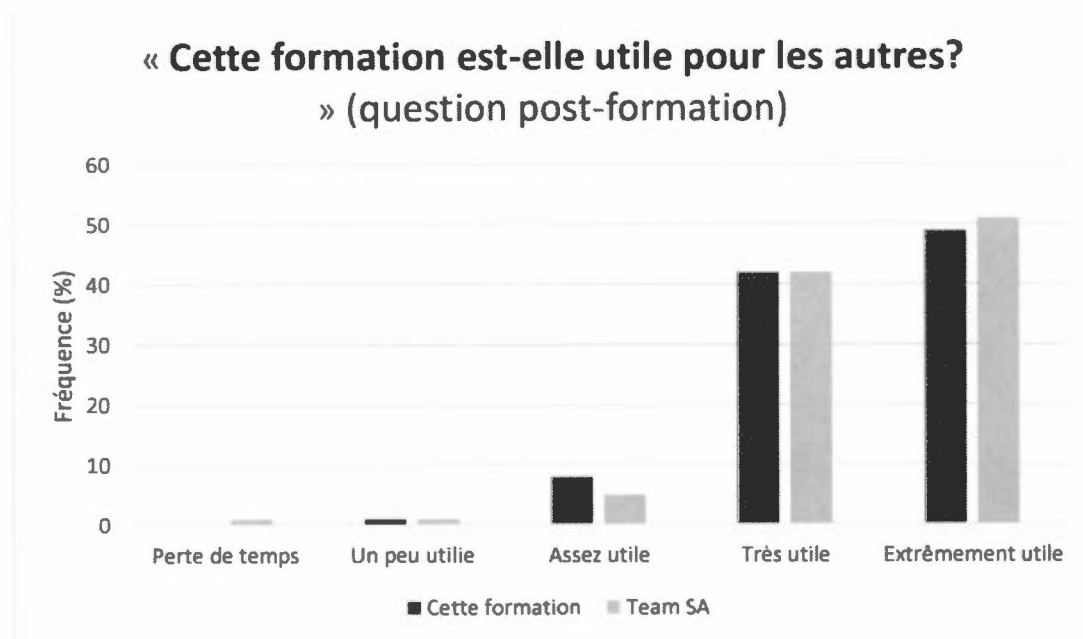


Figure 6. 5 L'enthousiasme pour cette formation comparé à *Team SA Training*

En répétant le suivi-évaluation à une période de 3 mois et de nouveau à des durées plus longues, des résultats plus fiables peuvent être obtenus concernant la mesure dans laquelle la formation a une influence sur les comportements au travail, liés à la CS d'équipe.

Aussi, à l'intérieur de cette période d'évaluation, il serait souhaitable de répéter la même formation au même groupe de participants afin d'aiguiser leurs compétences en favorisant un encrage des concepts et comportements enseignés.

Enfin, il serait hautement souhaitable de déterminer la mesure dans laquelle la formation a un impact sur des mesures critiques de performance dans le développement d'application dans l'entreprise. L'objectif ultime étant de réduire les erreurs de décision dans le cycle de vie des applications, d'améliorer la sécurité des applications et d'améliorer la performance des équipes et du processus dans son ensemble.

6.7 Conclusion

Pour le chercheur, l'évaluation de la formation a montré qu'elle a eu un effet bénéfique inattendu sur la compréhension des objectifs de l'entreprise et la définition des besoins CS des parties prenantes. En effet, les études de cas discutées durant les séances et les interactions qui en ont résulté entre les participants, nous ont permis de mieux comprendre plusieurs réalités de l'entreprise.

La formation a eu un impact sur le comportement des participants mais il est difficile de conclure une relation de cause à effet entre la formation et la performance des équipes. En effet ces séances de formation ont été des opportunités pour amener les parties prenantes à discuter et à se mettre en accord sur leurs besoins, c'est-à-dire, explorer des solutions qui fonctionnent pour tous. L'évaluation de la formation a montré qu'elle a été d'une grande utilité pour les employés et qu'elle constitue un outil pouvant les aider à relever le niveau de CS des équipes. Cependant, vu que la formation avait été administrée à si peu de participants, faire une évaluation significative de son effet sur les résultats de performance n'a pas été possible dans cette thèse. A l'avenir, cependant, l'effet de la mise en œuvre de la formation et des changements organisationnels sur plusieurs mesures de performance clés devrait être évalué. Il s'agira d'avoir des mesures pour les équipes (unités de travail) sur un certain nombre de critères tel que le niveau de compréhension du travail assigné, la compréhension des objectifs des autres équipes, la précision des estimés sur les travaux à effectuer, et la livraison à temps, ainsi que le taux de report des travaux. Nous reformulerons cette proposition dans la conclusion générale de notre thèse sous le titre de « recherches futures »

L'efficacité de cette formation, telle que mesurée par ses évaluations, peut aider les gestionnaires et plus particulièrement l'équipe de gestion des ressources humaines à élaborer des plans de formation du personnel de l'entreprise, qui tiennent compte des concepts non techniques comme ceux de la CS d'équipe.

CONCLUSION

Dans ce chapitre, nous présentons les principaux résultats obtenus de notre recherche, les conclusions que nous en avons tirées et les recherches futures que nous envisageons d'explorer. Mais avant, nous rappelons l'originalité et la contribution du travail, ainsi que la problématique, les objectifs que nous nous étions fixés, et la démarche méthodologique que nous avons adoptée.

Originalité et contribution de la recherche sur la CS

La CS et l'analyse informatique des besoins : L'application du concept de CS à l'analyse des besoins en informatique, qui constitue le socle de cette thèse, est en soit une innovation et est une contribution importante dans ce domaine de recherche. En effet, nous avons porté l'étude de la CS au-delà de ses domaines de prédilection que constituent l'aéronautique, les plateformes pétrolières, les urgences médicales, et les casernes de pompiers. Nous avons démontré l'applicabilité et l'importance de la CS dans la gestion des projets informatiques. Notre travail est le tout premier à explorer en quoi consiste réellement la CS dans ce domaine apparemment éloigné de ceux mentionnés précédemment, mais qui en réalité est confronté aux mêmes réalités de ce qu'on appelle un environnement dynamique et complexe.

CS d'équipe : Nous avons exploré le concept de CS au-delà de la sphère traditionnelle où l'objet d'intérêt est le processus cognitif au sein de l'individu. Les développeurs informatiques ont été présentés comme une équipe interdisciplinaire d'individus engagés dans le projet commun de « services informatiques aux clients ». Le comportement généré par la CS est ici dirigé vers un but externe à l'individu, celui de la performance de la tâche exécutée.

Sur le plan de l'ingénierie des exigences, à l'aide des travaux de Yu (2011), la méthodologie informatique iStar a permis d'effectuer une analyse qui a pris en compte les standards du design préliminaire des projets informatiques. Dans ce sens, en tant qu'analyste, nous nous étions engagés dans une approche systématique et incrémentale de l'élaboration des exigences. Ceci a été rendu possible par un processus coopératif et itératif d'analyse du problème et de documentation des observations résultantes. Les différents formats de représentation qui ont résulté ont été vérifiés et validés auprès des parties prenantes pour en assurer l'exactitude.

Application du modèle de CS de Endsley (Endsley, 1995a) : Cette recherche a permis d'appliquer certains aspects du modèle de CS de Endsley et d'en faire la compréhension. Il s'agit particulièrement de ses composantes de **buts**, et de **formation** :

- Les buts et les plans d'un acteur déterminent quels aspects de l'environnement sont pris en compte dans la CS. Ils influenceront la façon dont l'attention est dirigée et la façon dont l'information est perçue et la manière de l'interpréter.
- La formation est perçue comme moyen d'améliorer la CS en la mettant à jour avec des éléments nouveaux. Elle permet d'atteindre les buts.

D'un côté, les buts et l'apprentissage font partie des facteurs individuels qui influencent la CS, au même titre que les facteurs de la tâche, les préconceptions, la connaissance, l'expérience et les habiletés (Voir Figure 2.1). De l'autre côté, nous soutenons que le fait de modifier l'environnement de l'entreprise par une recherche-action en y adoptant un programme de formation, fait lui-même partie des facteurs environnementaux.

En terme de contribution, à travers le chapitre 5, l'étude a fourni un riche ensemble de lignes directrices qui permettent d'améliorer la conception des systèmes pour soutenir la CS chez les équipes. Il s'y dégage une contribution importante à la recherche sur la

CS d'équipe : Les acteurs comme participants actifs dans la mise à jour de leurs modèles mentaux et le développement de leur propre CS.

Comme il apparaît dans l'évaluation effectuée après la formation, la formation a provoqué et accéléré des changements de comportement qui soutiennent la mise à jour des modèles mentaux des équipes.

En amont, iStar, la méthodologie informatique d'analyse des exigences, a mis en évidence un nombre considérable de nouveaux éléments de CS (type de relations entre acteurs, les relations de vulnérabilité et de dépendance, et leurs impacts sur le fonctionnement des équipes) pour alimenter les modèles mentaux à travers l'analyse orienté-but des acteurs. Tous ces éléments ont par la suite trouvé leur voie dans un programme de formation dont l'évaluation a fait ressortir des signes évidents de mise à jour des connaissances et des modèles mentaux des acteurs autour des questions de coordination des activités et des relations de dépendance dans leur environnement.

Les acteurs utilisent leurs modèles mentaux pour définir à prime abord ce qui constitue pour eux, une « donnée » (Klein, 1995). Lorsqu'un individu a un modèle mental à jour sur les caractéristiques d'un système particulier, il est capable de diriger son attention sur les indices critiques, avoir des attentes réalistes sur les états futures de l'environnement (incluant à quoi s'attendre et à quoi ne pas s'attendre). Un modèle mental se construit au fil du temps à mesure que les individus accumulent de l'expérience sur l'environnement dans lequel ils opèrent (Endsley, 2000).

Mais, il semble, suite à l'évaluation de la formation, que le processus de construction du modèle mental n'est pas un processus passif qui s'opère uniquement au gré des événements et des leçons apprises. La construction des modèles mentaux peut tout aussi être un processus accéléré dans lequel les acteurs sont activement impliqués. Les acteurs sont des participants actifs en termes de comment ils organisent et modifient leur environnement. Ici, l'adoption d'un programme de formation a été l'élément

changeant de l'environnement. L'aspect dynamique des réalités du monde réel fait partie des dimensions temporaires de la CS (voir définition proposée par Endsley). Cette nature dynamique fait qu'à mesure que la situation évolue et change en incorporant d'éléments nouveaux (via la formation), ainsi change aussi la CS d'équipe.

Problématique et objectifs

Les équipes de développeurs informatiques sont aidées par de nombreuses autres équipes de l'entreprise dans la réalisation de leurs objectifs. Chacune de ces équipes a un impact majeur sur l'atteinte de ces objectifs. Dans ce contexte, il nous a paru nécessaire d'examiner comment l'information circule entre et parmi les membres des équipes afin d'identifier les facteurs du système et du personnel qui ont un impact sur la capacité des membres de l'équipe à maintenir une image précise de l'état de l'application informatique qu'ils produisent. Cette information semble être cruciale pour leur capacité à effectuer des tâches (étant donné qu'il y a des interdépendance entre les tâches en cours et celles exécutées par d'autres membres d'équipe), leur capacité à faire des évaluations correctes (par exemple, si un problème détecté doit être réglé maintenant ou plus tard), et leur capacité à se projeter correctement dans l'avenir pour prendre de bonnes décisions (par exemple le temps nécessaire pour effectuer la tâche, la disponibilité des ressources, la nouvelle composante de l'application, etc.). Tout en demeurant dans la dynamique des équipes, la question revient donc à se demander « qu'est-ce qui doit être perçu par chaque personne dans son travail, qu'est-ce qui doit être compris et quelles projections doivent être faites ? »

Il était donc important de déterminer comment et à quel degré la CS est présente dans des équipes d'une entreprise informatique et d'étudier les mécanismes qui sont utilisés pour acquérir et partager la CS au sein et entre les équipes.

Dans la perspective systémique de la CS, dans laquelle nous avons inscrit notre thèse, il nous revenait d'aborder les questions de coordination, de partage d'information et de dépendance entre les acteurs. Il ne suffit pas qu'un membre de l'équipe ait un morceau

de l'information. Il est plutôt nécessaire que chaque personne soit au courant de ce qui est pertinent pour la réalisation des rôles et responsabilités de son équipe; c'est-à-dire, ce qui est nécessaire pour ses responsabilités et comment acquérir ces informations de qualité. C'est dans ce sens que la connaissance et l'amélioration des mécanismes de communication et des relations interpersonnelles ont été importantes pour nous. Cependant, pour éviter une surcharge cognitive, il n'a pas été nécessaire que toutes les informations soient partagées avec chaque individu impliqué.

Rappel de la méthodologie appliquée

Pour réaliser nos objectifs, nous avons choisi de mener une recherche-action où nous avons effectué d'importantes activités de recherche :

- 1) La détermination des besoins en CS et l'analyse des ressources en information dans les équipes de développeurs d'application;
- 2) L'évaluation des besoins de formation pour améliorer la CS d'équipe;
- 3) L'application des méthodologies d'analyse dites « sociales » à l'étude de la CS;
- 4) La validation de la solution proposée.

A l'aide d'enquêtes contextuelles, d'investigations d'événements récents, d'entrevues et d'observations, la première étape du travail a consisté à déterminer des exigences spécifiques des individus en matière de CS dans le domaine informatique. Cela a été résolu par une analyse des tâches orientées vers un but qui a évalué 1) les objectifs et sous-objectifs associés aux équipes de développeurs, 2) les exigences de décision associées à ces objectifs, et 3) les exigences en CS nécessaires pour traiter les décisions à tous les trois niveaux - détection, compréhension, et projection. La deuxième partie de l'analyse de la CS d'équipe a été consacrée à l'identification des ressources en informations utilisées dans l'environnement de développement pour atteindre les exigences CS identifiées dans l'analyse des tâches orientées-but. Deux grandes

catégories de ressources ont été étudiées : 1) autres membres du personnel utilisés en tant que source d'informations et 2) les technologies utilisées comme sources d'information.

Les résultats

L'analyse des besoins en CS des équipes de développeurs informatiques, nous a amené à découvrir leurs objectifs et sous objectifs, à travers la compréhension des processus de production de l'entreprise étudiée. Les résultats nous ont permis de mieux comprendre ce qu'est la CS dans ce domaine particulier et la manière dont elle fait partie intégrante des comportements orientés-but des informaticiens. L'efficacité de la CS est donc manifestement limitée par le niveau d'attention des informaticiens à des informations pertinentes dans leur environnement, à la nature des relations, et à l'importance de la communication qu'ils entretiennent entre eux. Les résultats révèlent qu'au-delà des connaissances techniques, les informaticiens doivent posséder des connaissances non-techniques pouvant leur permettre de développer leurs aptitudes à communiquer leurs expériences et ont aussi besoin de posséder des capacités qui leur permettent de reconnaître quelles informations doivent être échangées entre et parmi les membres de l'équipe.

Dans le chapitre 4, dédié aux résultats de notre recherche, plusieurs obstacles à l'atteinte d'un niveau viable de CS ont été révélés. Nous avons élaboré sur les obstacles au niveau des relations interpersonnelles, au niveau technologique et au niveau organisationnel et avons proposé des pistes d'amélioration.

Nous avons particulièrement analysé les relations de dépendance et de vulnérabilités qui existent entre les abstractions d'agents du système en appliquant les concepts de la méthodologie d'analyse du social proposés par iStar. L'utilisation de cet outil de modélisation orienté-agent, nous a permis d'engager plus ouvertement des

conversations intéressantes avec les parties prenantes (développeurs, gestionnaires et autres équipes), à propos des concepts abstraits autour de la notion principale de “buts” et “objectifs”. Nous en avons déduit un modèle de dépendance stratégique qui représente les différentes relations d’interdépendance qui existent entre les acteurs. Le rôle de ce modèle n’est pas simplement de nous amener vers des considérations fonctionnelles d’un système informatique mais aussi de nous amener à réfléchir plus en profondeur, à un niveau d’abstraction plus élevé, sur des problèmes qui existent réellement dans l’entreprise, et à découvrir des pistes de solutions.

De plus, en impliquant les parties prenantes dans le processus d’analyse aussi longtemps que possible et en leur fournissant une représentation de la situation dans un langage accessible, il nous a été possible d’accroître la qualité des solutions proposées. Concrètement, l’application de cette méthodologie d’analyse des exigences nous a particulièrement permis 1) de réaliser des représentations en langage informatique de la dynamique sociale dans laquelle s’opèrent les relations entre les acteurs du système; 2) de documenter formellement toutes les exigences du système et d’éviter les écarts d’interprétation qu’il serait possible d’avoir lorsqu’on fait des analyses informelles basées uniquement sur l’usage de langages naturels; 3) d’améliorer la compréhension d’un système opaque complexe dans une spécification complète du système

Nous avons également procédé à une meilleure prise en charge des parties prenantes en introduisant un outil simple, comme le tableau de vulnérabilité qui nous a permis 1) de susciter des raisonnements à propos des connaissances du système; 2) d’identifier et de consolider les différentes visions d’amélioration du système, de toutes les parties prenantes, dans un format non-ambiguë, vérifiable et consistant qui peut être manipulé et modifié.

L’approche de recherche-action nous recommande de réinvestir les résultats de la recherche directement dans la pratique au bénéfice des acteurs impliqués (Denscombe, 2014). En recherche-action, le chercheur travaille avec les acteurs et parties prenantes

pour proposer une nouvelle ligne de conduite dans le but d'améliorer leurs stratégies, pratiques et connaissances sur les environnements dans lesquels ils pratiquent leur travail. (Wicks, 2009). Pour ce qui nous concerne, une bonne partie de nos conclusions et recommandations a été mise en action et a contribué aux changements et amélioration des pratiques organisationnelles qui ont lieu à l'intérieur des équipes étudiées.

Les solutions issues de cette analyse sont essentiellement des concepts de formation qui sont avérés très prometteurs comme un moyen de renforcer les compétences et les aptitudes nécessaires pour atteindre un niveau élevé de CS dans un environnement d'équipe.

Nous avons utilisé ces concepts de formation pour créer un programme de formation qui a été prototypé et livré à la même entreprise informatique. En améliorant les compétences en CS des différentes équipes de développeurs et en nous concentrant spécifiquement sur les zones où des problèmes de CS sont susceptibles de se produire, des améliorations dans l'efficacité et la performance des opérations de développement d'application peuvent être atteints.

La mise en œuvre du prototype de la formation a été utilisée pour obtenir une indication de la valeur et de l'utilité de la formation de cette nature. L'idée d'utiliser la formation pour améliorer la CS des équipes de développeurs d'application est nouvelle. A notre connaissance, à ce jour, il n'y a pas eu d'autres formations similaires. Aussi, l'évaluation que nous avons menée ici doit être considérée comme la première évaluation d'un prototype de formation de ce type.

Le fait que le programme ait reçu un accueil positif et jugé utile aux développeurs est révélateur de son succès.

Les limites de l'approche de la recherche-action

- Accès aux participants 'In Situ'

La collecte de données de façon générale, ou la collecte des bonnes données au bon moment, dans les lieux naturels de travail des participants est un défi majeur de la recherche-action. En effet, la présence du chercheur dans le milieu de travail peut être considérée parfois comme une intrusion par les employés et par conséquent, engendrer une distorsion dans le déroulement normal des processus de travail sous observation. Cela peut occasionner des réticences et des changements de comportement de manière à introduire des biais dans les données collectées. Dans notre cas, nous n'avons pas eu à vivre véritablement cette expérience puisque, l'environnement de travail n'a jamais été altéré par notre présence particulièrement lors des de nos séances d'observation. Plusieurs participants ayant presque oublié notre présence en tant que chercheur. Cependant, en tant que figure familière, nous avons noté quelques retenues de la part de certains participants lors des séances d'entrevues. Les réticences émanaient plus souvent des participants issus des départements autres que le nôtre. Mais le fait de combiner plusieurs sources de collecte de données nous a aidés à recouper les informations et ainsi minimiser les biais. Il faut souligner que ce risque est réel pour de nombreux chercheurs en recherche-action.

- Les ressources

Une formation telle que nous l'avons menée requiert beaucoup de ressources car le projet implique la mobilisation de plusieurs employés qui doivent choisir entre leurs activités normales et les séances de formation. Malgré notre position d'architecte système et de responsable d'équipe dans l'entreprise, et avec notre connaissance des habitudes internes, nous avons quand même été contraint à ne réaliser qu'un prototype du projet de formation. Le déploiement de la formation à toute l'entreprise aurait occasionné la mobilisation d'énorme ressource. De plus, le chercheur nécessite beaucoup de temps pour concevoir et dispenser la formation, et procéder à l'évaluation de son impact immédiat et longitudinale.

- Généralisation des conclusions

L'applicabilité du concept de CS n'a été validée que dans une étude avec un échantillon relativement réduit de personnel dans l'industrie de l'informatique. Ainsi, la généralisation des conclusions présentées ici est limitée en attendant de nouvelles études. Il faudra toutefois noter que contrairement à la recherche traditionnelle, la validité des résultats et conclusions dans la recherche-action ne dépendent pas du fait de mesurer l'étendue et la fréquence des phénomènes sur une période de temps, sur un grand nombre d'échantillons et dans plusieurs études similaires afin de justifier des énoncés de cause à effet précis (et donc étroitement définis) (Brydon-Miller, 2003).

Les limites de notre méthode de formation

Notre solution de formation a été exécutée en dehors des considérations de la méthodologie de développement qui est appliquée dans l'Entreprise. En effet, une solution de partage de connaissances fidèle aux méthodes agiles pourrait avoir plus d'impact. Par exemple, nous aurions pu exécuter les séances de formation directement dans les rencontres de rétrospective qui constituent une excellente plateforme d'échange. Nous en prenons bonne note pour nos travaux futurs.

Les limites du modèle de la CS de Endsley

Dans une certaine mesure, on pourrait qualifier les trois niveaux de CS qui constituent le cœur du modèle de CS de Endsley (Figure 2. 12), de strictement linéaires. Certains auteurs comme Chiappe *et al* (2012) vont plus loin en le caractérisant de modèle sériel. Dans la description du modèle, Endsley affirme que sans un développement de la perception, l'on ne peut comprendre les situations et être en mesure de projeter l'état du système dans le futur. Même si parfois les trois niveaux de CS sont présentés dans le modèle comme des niveaux ascendants de CS, il n'en demeure pas moins que le modèle suggère que la perception, la compréhension et la projection se produisent forcément dans des étapes linéaires distinctes. Les informations de CS seraient séparées

mentalement dans trois catégories (Perception, compréhension et projection). En fait, le modèle n'indique pas clairement que ce n'est pas nécessairement le cas. « Une personne qui comprend la situation actuelle a plus de CS que celle qui peut lire les données sur un écran mais ne sait pas ce que cela signifie. De même, une personne qui peut projeter les événements futurs probables et les états du système et de l'environnement ont une meilleure CS que celle qui ne peut pas » (Endsley, 2015, p.8). La réalité, cependant, est qu'une simple progression de 1-2-3 n'est pas un mécanisme de traitement efficace dans un système complexe et dynamique où l'expertise de l'acteur et le traitement axé sur les objectifs entrent en jeu. Comme corollaire de cette affirmation, Sorensen *et al* (2011) soutient aussi que: « Pour ces experts, il n'est pas possible de diviser leur CS en trois niveaux de manière significative. Le modèle n'explique pas les situations où la CS est un processus continu » (p.453).

Les novices divisent leur compréhension du monde dans ces catégories soignées et rangées mais ce n'est pas le cas chez les experts (Chi, 2014). Dans notre présente recherche, nous avons utilisé la définition des 3 niveaux de CS plutôt comme descriptif des types d'informations contenues dans le concept. Il n'a pas servi à tester le mode de cognition des individus.

A notre point de vue, la 2ème limite du modèle de CS de Endsley est qu'il est conçu pour décrire les mécanismes de cognition de l'individu. En tant que tel, le modèle ne nous paraît pas facilement applicable aux équipes. Pour évaluer l'applicabilité du concept de CS en équipe à l'aide de ce modèle, il est nécessaire de prendre en compte les mécanismes de coordinations et de communications entre les individus; les mécanismes d'échange d'informations et de connaissances, ainsi que leurs impacts sur les décisions individuelles et collectives. Il faudra par la suite prendre en compte les procédures et les normes que les équipes utilisent dans leur environnement pour faciliter la coordination entre les membres.

Application de la méthodologie iStar : les leçons apprises

Cette recherche nous a amené à conclure que l'ingénierie des exigences n'est pas seulement un processus dans lequel l'ingénieur collecte et documente les besoins des parties prenantes. D'une part, il consiste à un processus cognitif dans lequel les parties prenantes parviennent de façon collaborative à se mettre d'accord sur ce qui doit être fait, en comprenant la nature des problèmes et en apprenant des autres à travers des négociations et discussions de points de vue différents et parfois contradictoires. D'autre part, en tant qu'analyste, nous avons besoin de comprendre comment les choses fonctionnent dans le système actuel et pourquoi elles fonctionnent ou ne fonctionnent pas et selon quels critères. Pour produire des exigences qui correspondent aux besoins des parties prenantes, il est nécessaire de comprendre les motivations derrière les besoins exprimés. Il ne suffit pas simplement de répertorier ce que les parties prenantes disent vouloir, car il n'est pas si facile pour eux d'exprimer clairement leurs besoins. Il nous revient donc, en tant qu'analyste du système, de mettre en place des techniques et approches efficaces pour aider les parties prenantes à mieux s'exprimer et à mieux se comprendre.

Le choix d'utiliser iStar comme support méthodologique à l'analyse de la CS, nous a permis d'appliquer ses modèles SD et SR, modèles conceptuels de haut niveau. Le SD, avec ses artefacts et concepts de communication (but, objectif, rôle, ressources, etc...) pour expliquer le « comment » des choses, et le modèle SR pour décrire les relations intentionnelles qui sont « internes » aux acteurs et pour fournir une représentation explicite de « pourquoi » et « comment », et des alternatives. Cependant, nous avons trouvé que iStar ne permet pas de susciter des conversations profondes et sérieuses entre toutes les parties. iStar ne propose pas de techniques spécifiques pour l'engagement des parties prenantes dans le processus d'ingénierie des exigences. À la limite, sa méthode peut être présentée comme un outil pour faire des notations et enregistrer les motivations des acteurs. Il est certain que la méthodologie iStar aide l'analyste à comprendre les comportements et les motivations de l'acteur dans ses

« frontières » de relation, mais le problème est qu'elle semble être plus utile à l'analyste qu'aux acteurs.

Pour accroître la qualité des exigences, nous estimons que le rôle de toute méthodologie de but et d'agent n'est pas seulement de conduire à l'élaboration d'un système, mais elle doit aussi servir comme un moyen de réfléchir à des problèmes et de parvenir à des accords entre les parties prenantes. Pour y parvenir, un tel outil devra 1) permettre une représentation du système dans un langage accessible à toutes les parties prenantes; 2) garder les parties prenantes aussi longtemps que possible dans les discussions.

Première leçon : une représentation tabulaire des dépendances

Sur ce point, nous avons opté d'accompagner le diagramme stratégique de dépendance de iStar par une représentation tabulaire des relations de dépendance répertoriées dans le système. Ce type de représentation est celle que nous avons utilisée dans le tableau 4.25. Elle est simple et offre de nombreux avantages dont les suivants :

- 1) Elle donne un résumé de toutes les relations de dépendance entre les acteurs ;
- 2) Chaque ligne du tableau est atomique et exprime une relation entière entre 2 acteurs du système. Cela oblige les parties prenantes à examiner les relations, une à la fois et à engager des discussions sur les points d'accord et de désaccord;
- 3) Elle se conçoit facilement à l'aide d'un simple logiciel de traitement de texte ou d'un tableur;
- 4) Elle suscite la contribution des parties prenantes car elle est facile à modifier;
- 5) Elle est bâtie autour des concepts familiers aux agents (but, objectif, ressource, tâche);

6) Enfin, elle suscite des raisonnements plus aisément car facile à lire.

Deuxième leçon : une approche collaborative pour réduire la dominance de l'analyste

Nous avons mentionné à la section 6.5 que les séances de formation ont eu des effets surprenants et bénéfiques autant pour l'analyste dans son rôle d'ingénieur des exigences, que pour les acteurs du système dans leur rôle de parties prenantes. Le fait de mettre ensemble les intervenants, nous a permis d'avoir une vue du système et des besoins, qui est complémentaire à celle obtenue suite à des observations et à l'application des questionnaires. Les rencontres nous ont surtout permis de mieux catégoriser les types d'acteurs que comporte le système selon la nature de leur contribution au processus d'ingénierie des exigences. Deux catégories d'acteurs se dégagent: 1) les rapporteurs d'histoire et 2) les facilitateurs. Il nous paraît important de dissocier le rôle que ces acteurs ont à jouer dans le processus d'ingénierie des exigences, des rôles qui leur reviennent dans le système lui-même.

À partir de cette expérience, nous proposons d'intégrer à la méthodologie iStar, une « démarche collaborative ». Ce concept en soit, n'est pas nouveau en ingénierie des systèmes. Il a été proposé par un certain nombre de recherches notamment la méthode WinWin de Boehm et Ross (1989) qui a plus tard évoluée en EasyWinWin (Boehm *et al.*, 2001); la méthode CoREA de Geisser et Hildenbrand (2006) ou la BPCRAR (*Business Process oriented Collaborative Requirements Acquisition and Refining*) de Lai *et al* (2014). Cependant, le fait d'intégrer le concept dans la méthodologie iStar pour l'étude de la CS d'équipe est unique à cette thèse. Soulignons cependant qu'aucune des méthodes précitées n'a été conçue pour être utilisée dans l'ingénierie d'un système sociotechnique à prédominance cognitive. Leurs approches sont plus orientées vers la conception de logiciel.

Les descriptions des deux catégories d'acteurs identifiées sont les suivantes :

- **Les rapporteurs d'histoire**

Ils sont importants au succès du système. Ce sont les acteurs qui peuvent décrire leurs activités, énoncer leurs besoins, décrire les relations qu'ils entretiennent dans le système avec les autres acteurs, parler de leurs connaissances du système, raconter leur mésaventures et leurs problèmes, exprimer leurs intentions et proposer des configurations de système qui vont dans le sens de leurs besoins stratégiques. Dans le système de l'entreprise étudiée ici, cette catégorie inclut par exemple les développeurs d'application (collectivement ou individuellement), les gestionnaires de comptes, les gestionnaires de configuration. Dans d'autres systèmes, les clients peuvent faire partie des rapporteurs d'histoires.

- **Les facilitateurs**

Cette catégorie comporte des acteurs qui jouent le rôle de personne ressource. Ils sont médiateurs et font le lien entre les équipes pour s'assurer que les objectifs du système sont atteints et que les normes opérationnelles sont appliquées. Ils trouvent des liens entre les faits rapportés par les rapporteurs d'histoire.

Les détails des rôles de ces deux types d'acteurs et leur mode de fonctionnement à l'intérieur du processus d'ingénierie des exigences sont cependant hors de la portée de cette thèse.

Recherches futures

- Évaluation de l'impact des changements organisationnels sur les composantes sociotechniques de l'entreprise.

Le véritable défi de l'ingénierie des processus réside dans sa phase d'implantation, c'est-à-dire la phase où l'expert traite avec les questions de gestion du changement lié à l'introduction de processus nouveaux ou à la reconfiguration des processus existants. Il est relativement facile de proposer une nouvelle manière de faire les choses, mais il est plus compliqué de la faire fonctionner et de réussir à amener toutes les parties prenantes à l'adopter. Cette réalité semble indiquer que les réels besoins des parties prenantes sont souvent mal identifiés et mal analysés durant le processus de design, ce qui compromet sérieusement les chances de succès. Le déficit d'utilisation d'outils de modélisation et d'analyse des intérêts stratégiques des parties prenantes a certainement contribué à la persistance de telles situations indésirables. En utilisant le Framework de iStar, le modèle de dépendance stratégique que nous avons élaboré a supporté une identification systématique des intérêts des parties prenantes concernées. Il serait maintenant intéressant de porter cette recherche à un niveau plus loin en étudiant les manières dont les changements organisationnels pourraient être effectivement opérés et comment en évaluer les impacts à long terme sur les individus, les équipes, et les systèmes.

- Expérience de l'équipe et niveau de CS

La présente thèse n'a pas fait cas des impacts de l'expérience et de l'expertise de l'équipe sur le niveau de CS d'équipe. Les développeurs acquièrent leur CS en partie à travers l'information disponible dans leur environnement et à travers les relations avec les ressources humaines, et technologiques, ainsi que des méthodes de communication à l'intérieur de leur équipe. Il est évident que la manière dont l'information est recueillie et traitée dépend aussi de l'expertise de l'individu (Hogan *et al.*, 2006). En effet, certains développeurs sont plus aptes à atteindre facilement un haut niveau de CS avec peu d'information de l'environnement, alors que d'autres font un usage insuffisant de ces mêmes informations. Il s'avère que cette différence est attribuable à la différence dans les capacités cognitives individuelles, telles que l'attention, les connaissances

techniques acquises, etc. Les experts sont caractérisés par leur habilité à atteindre un bon niveau de CS avec peu d'effort, rapidement.

En termes de recherche future, il serait intéressant d'explorer, à travers une enquête auprès des équipes informatiques, ce qui se passe au niveau de l'expérience de l'équipe et les liens qui existent entre la qualité des méta-connaissances de l'équipe et son modèle mental.

Pour finir, nous aimerions commenter cette définition (opérationnelle) de la CS retrouvée dans les documents officiels de la *Federal Aviation Administration* des Etats-Unis.

La conscience de la situation est définie comme une extraction continue de l'information environnementale, l'intégration de cette information avec des connaissances antérieures pour former une image mentale cohérente et l'utilisation de cette image pour orienter la perception et anticiper les événements futurs. Autrement dit, la conscience de la situation signifie savoir ce qui se passe autour de vous. (U.S. Department of Transportation, Federal Aviation Administration, 2014, Watch Supervision section, para. 10)

Cette définition traduit l'interprétation du concept de la CS dans le monde réel opérationnel. Pour la communauté opérationnelle, il semble que ce qui importe ultimement, c'est la compréhension et l'amélioration de la performance humaine aussi bien dans les situations routinières que dans les situations exceptionnelles, à l'intérieur d'un système réel, naturel, dynamique et complexe. Mais, affirmer qu'un bon niveau de CS implique nécessairement une bonne performance est un peu hâtif car le lien entre la performance et la CS n'est pas déterministe.

Aussi, les résultats et conclusions tirées de la présente recherche peuvent ne pas être applicables à certaines entreprises informatiques ayant des objectifs de productivité et de performance similaires. Cependant, le fait de ne pas pouvoir reproduire nos conclusions dans d'autres circonstances ne devrait en aucun cas entacher l'utilité et l'efficacité des résultats que nous avons obtenus, car, la recherche-action porte sur

l'exploration des multiples déterminants des actions, des interactions et des relations interpersonnelles dans des contextes uniques. Son objectif est d'approfondir la compréhension par les praticiens des situations complexes dans lesquelles ils vivent et travaillent, de sorte que leurs actions sont mieux informées (Checkland *et al.*, 1998; Brydon-Miller, 2003)

La démonstration de l'applicabilité du concept de CS d'équipe en environnement organisationnel, contribuera à documenter et à fournir au monde de la recherche, des exemples du monde réel sur les problèmes de performance humaine.

Mais, il est fondamental de noter qu'avant tout, la recherche-action permet de combler le fossé entre la recherche et la pratique. Elle le fait en rejetant le concept d'un processus en deux étapes dans lequel la recherche est menée en premier par les chercheurs puis, dans une seconde étape distincte, les connaissances générées par la recherche sont appliquées par les praticiens. Au lieu de cela, les deux processus de recherche et d'action sont intégrés (Shotter, 2007). C'est un peu dans cette dynamique que s'inscrit toute notre thèse.

ANNEXE A

QUESTIONNAIRE CDM PROBES (O'Hare *et al.*, 2000)

Goal Specification	What were your specific goals at the various decision points?
Expectancy	Were you expecting to make this sort of decision during the course of the event? Describe how this affected your decision-making process.
Conceptual	Are there any situations in which your decision would have turned out differently? Describe the nature of these situations and the characteristics that would have changed the outcome of your decision.
Influence of uncertainty	At any stage, were you uncertain about either the reliability of the relevance of the information that you had available? At any stage, were you uncertain about the appropriateness of the decision?
Information integration	What was the most important piece of information that you used to formulate the decision?
Situation Awareness	What information did you have available to you at the time of the decision? Where did you get that information from? What parts of the information comes from your interaction with other teams
Situation Assessment	Did you use all of the information available to you when formulating the decision? Was there any additional information that you might have used to assist in the formulation of the decision?
Options	Were there any other alternatives available to you other than the decision you made?
Decision blocking - stress	Was there any stage during the decision-making process in which you found it difficult to process and integrate the information available? Describe precisely the nature of the situation

QUESTIONNAIRE CDM PROBES (O'Hare *et al.*, 2000)

Basis of choice	Do you think that you could develop a rule, based on your experience, which could assist another person to make the same decision successfully? Why/Why not?
Analogy/ generalization	Were you at any time, reminded of previous experiences in which a similar decision was made? Were you at any time, reminded of previous experiences in which a different decision was made?

ANNEXE B

QUESTIONNAIRE D'ÉVALUATION PRE SEMINAR / POST FORMATION

Please answer by writing beside each item the number that best reflects your personal attitude. Choose the number from the scale below.

**** Scale ****

- | 1 | 2 | 3 | 4 | 5 |
|----------------------|----------------------|---------|----------------|-------------------|
| Disagree
Strongly | Disagree
Slightly | Neutral | Agree Slightly | Agree
Strongly |
-
- ___ 1. Members of development team should avoid disagreeing with others.
 - ___ 2. It is important to avoid negative comments about the procedures and techniques of other team members.
 - ___ 3. Casual, social conversation on the job during periods of low workload can improve Application development team coordination.
 - ___ 4. Good communications and team coordination are as important as technical proficiency for operational effectiveness.
 - ___ 5. **Written communication and detailed work description are important for operational effectiveness.**
 - ___ 6. The manager, supervisor, or team leader in charge should take hands-on-control and make all decisions in emergency and non-standard situations.
 - ___ 7. The manager, supervisor, or team leader in charge should verbalize plans for procedures or actions and should be sure that the information is understood and acknowledged by the other Application development team members.
 - ___ 8. Application development team members should not question the decisions or actions of the manager, supervisor, or team leader except when they threaten the performance and the quality of the application.
 - ___ 9. Even when fatigued, I perform effectively during critical phases of work.

- ___ 10. Managers, supervisors, and team leaders should encourage questions during normal operations and in special situations.
- ___ 11. There are no circumstances where the subordinate should assume control of a project.
- ___ 12. A debriefing and critique of procedures and decisions after each major task is an important part of developing and maintaining effective team coordination.
- ___ 13. Overall, successful Development management is primarily a function of the manager's, or team leader's technical proficiency.
- ___ 14. Training is one of the manager's most important responsibilities.
- ___ 15. Because individuals function less effectively under high stress, good team coordination is more important in emergency or abnormal situations.
- ___ 16. Effective team coordination requires each person to take into account the personalities of other team members.
- ___ 17. The responsibilities of the manager, supervisor, or team leader include coordination between his or her work team and other support areas.
- ___ 18. **We should be aware of and sensitive to the activities and technical problems of other Development teams.**
- ___ 19. A truly professional manager, supervisor, or team leader can leave personal problems behind
- ___ 20. My decision-making ability is as good in abnormal situations as in routine daily operations.

Please answer by writing beside each item the number that best reflects your personal attitude. Choose the number from the scale below.

**** Scale ****

1	2	3	4	5
Disagree Strongly	Disagree Slightly	Neutral	Agree Slightly	Agree Strongly

In the following questions, “my management group” refers to those people who report to the same manager that I do.

- ____ 21. I am kept informed by others in my management group about the goals and objectives of this organization (e.g., work estimate, quality, application component, etc.).
- ____ 22. Work goals and priorities are understood and agreed to by members of my management group.

In the following items, “my work group”, refers to those people who report to me.

- ____ 23. Employees in my work group receive detailed feedback regarding the organization's performance.
- ____ 24. If employees in my work group disagree with the goals and priorities that have been established, they feel free to raise their concerns with supervision.
- ____ 25. Employees in other groups within Development team plan and coordinate their activities effectively together with people in my work group.
- ____ 26. Employees in other groups, team throughout the company act as if they share many of the same organizational goals that we do.

TRAINING EXPERIENCE AND EVALUATION

I. For each of the topic areas or training techniques listed below, please rate the value of this aspect of the training to you. Rate each item by choosing the number on the scale below which best describes your personal opinion and then write the number beside the item.

1	2	3	4	5
Waste of time	Slightly Useful	Somewhat Useful	Very Useful	extrêmement Useful

- | | | |
|----|-------|---|
| 27 | ----- | Training in interpersonal communications and skills |
| 28 | ----- | Training in skills using role play. |
| 29 | ----- | Case studies of application incident |
| 30 | ----- | Norms |
| 31 | ----- | Overall, how usefull did you find the training? |

II. Development team Coordination Concepts training potential to increase application quality and performance and team effectiveness. (circle one from list below)

32	1 Disagree Strongly	2 Disagree Slightly	3 Neutral	4 Agree Slightly	5 Agree Strongly
----	---------------------------	---------------------------	--------------	------------------------	------------------------

III. How useful will such training be for others? (circle one from list below)

33	1	2	3	4	5
	Waste of Time	Slightly Useful	Somewhat Useful	Very Useful	Extremely Useful

IV. Is the training going to change your behavior on the job? (circle one from the list below)

34	1	2	3	4
	No change	A slight change	A moderate change	A large change

V. How will you use this training on your job?

35 _____

VI. What aspects of the training were particularly good?

36 _____

VII. What do you think could be done to improve the training?

37 _____

Questionnaire adapté du questionnaire CRM/TOQ de Taylor (Taylor et Robertson, 1995)

ANNEXE C

SYLLABUS DU COURS SUR LA CS

Titre du cours : Comment être productif dans un environnement complexe?

But et Objectifs de la formation

Le but de la formation est de doter les participants, des compétences non techniques pour améliorer la conscience de la situation des équipes informatiques. Les objectifs suivants seront réalisés :

- 1) Connaitre et comprendre les "normes" organisationnelles et leurs effets sur la performance des équipes.
- 2) Promouvoir un comportement assertif.
- 3) Comprendre les styles de leadership individuels.
- 4) Comprendre et gérer les situations d'urgence.
- 5) Promouvoir les démarches de résolution rationnelle des problèmes et la prise de décision consensuelle
- 6) Améliorer les compétences interpersonnelles

- Module 1 : Que signifie environnement complexe? – 30 min

L'objectif de ce module est d'amener les participants à prendre conscience de la complexité des opérations de leur environnement et à savoir identifier et interpréter les signaux de bonne ou mauvaise gouvernance.

- Diversité des équipes.
- Rôles et responsabilité divers.
- Mobilité du personnel.
- Difficile d'appliquer des solutions déjà expérimentées à de nouveaux problèmes.
- Réalité changeante / Exigences changeantes.
- Complexité de la tâche.
- Dépendance des tâches et des équipes.
- Présence de personnalité diverse au sein de l'équipe.
- Contrainte des règles d'affaire et de l'industrie

- Module 2 : Introduction à la Conscience de la situation – 30 min

Qu'est-ce que la conscience de la situation pour les équipes de projet informatique. Analyser les besoins en informations (perception, diagnostic, projection), et les dépendances de tâches des différentes équipes.

- Modèle de la CS
 - Perception
 - Compréhension
 - Projection

- Module 3 : Performance et obstacles à la performance – 60 minutes

Les chefs d'équipes doivent renforcer leurs rôles de « personne ressource » et de relai d'information. L'importance de la transmission des informations sur l'état du travail au sein des équipes. Comprendre les problèmes et obstacles à la conscience de la situation.

- Que signifie performance?
- Identifier ensemble les problèmes liés au cycle de production actuel dans l'entreprise.
- Comprendre les différents comportements au bureau devant les situations qui s'y déroulent.
- Importance de transmettre des informations sur l'état du travail au sein des équipes.
- Effets des feedbacks sur les compétences de diagnostic.
- **Analyse des problèmes et obstacles à la conscience de la situation à partir des études de cas**

**- Module 4 : Les approches de résolution de problèmes et prise de décision – 1
Heure 45 minutes**

Comprendre les démarches de résolution de problèmes techniques, de partage des résultats de diagnostics et de solutions; le travail en groupe, le travail en équipe, et les décisions collectives. Les dépendances entre équipes.

- Une équipe a des objectifs. Les membres de l'équipe doivent connaître les objectifs de l'équipe.
- Connaître les objectifs et les modes de résolutions de problème des autres équipes avec lesquelles on travaille.
- Comprendre les forces et faiblesses des réactions/comportements par rapport à l'atteinte des objectifs des équipes.
- Les acteurs ont des personnalités différentes et les comportements individuels influencent la communication, la perception des autres, la prise de décisions et les méthodes de résolution de conflits
- La résolution de problèmes implique un processus rationnel et consensuel.

- Les décisions consensuelles sont meilleures aux décisions individuelles.
- Les relations interpersonnelles ont des impacts sur les processus de prise de décision.
- Connaissance des rôles et attributions des autres équipes et leurs impacts sur nos décisions et la performance de notre équipe.
- Connaissance des modes de résolutions de problème des autres équipes avec lesquelles on travaille.

- **Module 5: Les normes – 45 Min**

Les normes jouent un rôle crucial dans le fonctionnement des organisations et ont un impact direct sur la performance des équipes et la qualité de la production. Les normes peuvent être aussi des règles non écrites mises en place par les équipes.

- Identification du rôle des normes dans le processus de gestion de projet informatique
 - Introduire le concept / réviser les normes en place;
 - Donner des exemples
 - La connaissance des normes montre notre implication et notre appartenance à l'équipe.
 - La compréhension des normes de l'équipe nous aide à comprendre plus facilement ce qui se passe dans l'équipe.
- Quelles sont les normes qui ont conduit à des incidents
- Bonnes et mauvaises normes
- Comment gérer les normes et prévenir les incidents

- **Module 6 : L'information et la perception – 45 Min**

Prise de conscience de sources d'erreurs au niveau personnel (oublis d'information ou des étapes en association avec des interruptions de tâches; défaut de transmission d'informations entre les membres de l'équipe; diagnostics incomplets, informations critiques manquantes due à des distractions, interprétation erronée des informations) Importance de la documentation et transmission des résultats de diagnostic.

- Les interruptions et distractions lors de l'exécution des tâches
- Les suppositions doivent être testées.
- Comment et quand documenter notre travail
- La documentation aide les autres à mieux comprendre notre démarche de résolution de problème.
- Tirer avantage de l'utilisation des ressources (personnel et technologique) pour la transmission des informations aux bonnes personnes.
- **Analyse des rapports d'incidents en environnement de production**

Syllabus, Adapté de Robertson et Endsley, 1997

ANNEXE D

QUESTIONNAIRE DES ENQUETES CONTEXTUELLES

SECTION 1

1. Comment définissez-vous le rôle et les objectifs de votre équipe.
2. Quelles équipes dépendent du travail de votre équipe?
3. Pour quelle information ou pour quel service?
4. Comment communiquez-vous avec cette équipe (processus, outils) ?
5. Rencontrez-vous des problèmes dans l'obtention de l'information ou des services de cette équipe?
6. De quelles équipes dépendez-vous pour accomplir votre travail?
Pour quelles informations ou quels services?
7. Comment communiquez-vous avec cette équipe (processus, outils) ?
8. Rencontrez-vous des problèmes à faire parvenir des informations ou vos services à cette équipe?
9. Pensez-vous que le processus de travail actuel fonctionne bien pour votre équipe?
10. S'il y a des améliorations à faire au processus, que feriez-vous différemment?
11. Quelle est la fréquence de communication que vous avez avec chacun des membres de l'équipe DEV ou des équipes de support?
12. Sur une échelle de 1 à 4, quelle est l'importance de vos interactions avec chaque membre des autres équipes pour l'atteinte de vos objectifs?

13. Quels sont selon vous les problèmes que vous rencontrez dans votre travail et dans vos interactions avec le reste du personnel dans l'atteinte de vos objectifs de travail.

SECTION 2

1 Fortement en désaccord	2 En désaccord	3 Neutre	4 En accord	5 Fortement en accord
--------------------------------	-------------------	-------------	----------------	-----------------------------

1. Je suis informé par les autres membres de mon unité de travail sur les buts et objectifs de l'entreprise
2. Les buts et priorités de l'entreprise sont compris et acceptés par les membres de mon unité de travail
3. Les employés de mon unité de travail reçoivent des feedback détaillés sur les performances de l'entreprise
4. Si les employés dans mon unité de travail sont en désaccord vis-à-vis des objectifs et priorités de l'entreprise, ils se sentent libres de l'exprimer à leur superviseur.

SECTION 3

1 Fortement en désaccord	2 En désaccord	3 Neutre	4 En accord	5 Fortement en accord
--------------------------------	-------------------	-------------	----------------	-----------------------------

1. Les employés dans d'autres équipes planifient et coordonnent leurs activités effectivement ensemble avec les membres de mon unité de travail
2. Les employés dans d'autres équipes et départements de l'entreprise agissent comme s'ils partagent les mêmes objectifs organisationnels que nous.

BIBLIOGRAPHIE

- Antón, A. (1996, April). Goal-based requirements analysis. In *Requirements Engineering, 1996., Proceedings of the Second International Conference on*, 136-144). IEEE.
- Arenas, A. E., Massonet, P., Ponsard, C. et Aziz, B. (2015). Goal-Oriented Requirement Engineering Support for Business Continuity Planning. In *Advances in Conceptual Modeling*, 259-269. Springer International Publishing.
- Awad, M. A. (2005). A comparison between agile and traditional software development methodologies. University of Western Australia.
- Baccarini, D. (1999). The logical framework method for defining project success. *Project management journal*, 30(4), 25-32.
- Baskerville, R., et Myers, M. D. (2004). Special issue on action research in information systems: Making IS research relevant to practice: Foreword. *Mis Quarterly*, 329-335.
- Baskerville, R., et Wood-Harper, A. T. (1998). Diversity in information systems action research methods. *European Journal of information systems*, 7(2), 90-107.
- Bechky, B. A. (2006). Gaffers, gofers, and grips: Role-based coordination in temporary organizations. *Organization Science*, 17(1), 3-21.
- Berio, G., Di Leva, A., Giolito, P. et Vernadat, F. (1995). The M*-OBJECT methodology for information system design in CIM environments. *Systems, Man and Cybernetics, IEEE Transactions on*, 25(1), 68-85.
- Black, R. (2006). The psychology behind true project success. *Computing Canada*, 32(6), 19
- Blaikie, N. (2009). Designing Social Research. Cambridge: Polity Press.
- Bots, P. W., Van Twist, M. J., et Van Duin, R. (1999, January). Designing a power tool for policy analysts: dynamic actor network analysis. In *Systems Sciences, 1999. HICSS-32. Proceedings of the 32nd Annual Hawaii International Conference on* (pp. 10-pp). IEEE.

- Brydon-Miller, M., Greenwood, D., et Maguire, P. (2003). Why action research?
- Boehm, B. W. et Ross, R. (1989). Theory-W software project management principles and examples. *Software Engineering, IEEE Transactions on*, 15(7), 902-916.
- Boehm, B., Grünbacher, P. et Briggs, R. O. (2001, July). EasyWinWin: a groupware-supported methodology for requirements negotiation. In *Proceedings of the 23rd International Conference on Software Engineering*, 720-721. IEEE Computer Society.
- Bolman, L. (1979). Aviation accidents and the "Theory of the situation." In G.E. Cooper, M.D. White, and J.K. Lauber (Eds), *Ressource management on the flight deck: Proceedings of a NASA/industry workshop*, 31-58. Moffett Field, CA: NASA Ames Research Center
- Borgida, A., Greenspan, S. et Mylopoulos, J. (1985). *Knowledge representation as the basis for requirements specifications*, 152-169. Springer Berlin Heidelberg.
- Bradbury-Huang, H. (2010). What is good action research? Why the resurgent interest? *Action Research*, 8(1), 93-109.
- Brandel, M. (2006). The NEW project manager. *Computerworld*, 40(15), 43-45.
- Buchanan, D., et Huczynski, A. (1997). *Organizational Behavior*. Vol. 7.
- Burtscher, M. J. et Manser, T. (2012). Team mental models and their potential to improve teamwork and safety: A review and implications for future research in healthcare. *Safety science*, 50(5), 1344-1354.
- Business Dictionary (2017). Récupéré de : <http://www.businessdictionary.com/definition/team-management.html>
- Cailliau, A., Damas, C., Lambeau, B. et van Lamsweerde, A. (2013, July). Modeling car crash management with KAOS. In *Comparing Requirements Modeling Approaches Workshop (CMA@RE), 2013 International*, 19-24. IEEE.
- Cameron, E. et Green, M. (2015). *Making sense of change management: A complete guide to the models, tools and techniques of organizational change*. Kogan Page Publishers.

- Cannon-Bowers, J. A., Prince, C., Salas, E., Owens, J. M., Morgan Jr, B. B. et Gonos, G. H. (1989). Determining aircrew coordination training effectiveness. In *Proceedings of the 11th Annual Meeting of the Interservice/Industry Training Systems Conference*. Ft Worth, TX, 128-36.
- Cannon-Bowers, J. A., Tannenbaum, S. I., Salas, E. et Volpe, C. E. (1995). Defining team competencies: Implications for training requirements and strategies. In R. Guzzo, & E. Salas (Eds.), *Team effectiveness and decision-making in organizations*, San Francisco: Jossey-Bass.
- Caroll, C. F. (2013). IT Success and Failure—the Standish Group CHAOS Report Success Factors. Récupéré de <https://www.projectsmart.co.uk/white-papers/chaos-report.pdf>
- Casagrande, E., Woldeamlak, S., Woon, W. L., Zeineldin, H. H. et Svetinovic, D. (2014). NLP-KAOS for Systems Goal Elicitation: Smart Metering System Case Study. *Software Engineering, IEEE Transactions on*, 40(10), 941-956.
- Cicmil, S., et Hodgson, D. (2006). Making projects critical: an introduction. *Making projects critical*, 1-28.
- Chalandon, X. (2007). *Conscience de la situation: invariants internes et invariants externes. Contributions de l'ergonomie cognitive et de l'ingénierie à la conception de systèmes d'aide à la gestion des environnements dynamiques* (Doctoral dissertation, Thèse de Doctorat en Ergonomie. CNAM, Paris).
- Chi, M. T. (2014). *The nature of expertise*. Psychology Press.
- Chiappe, D. L., Strybel, T. Z., et Vu, K. P. L. (2012). Mechanisms for the acquisition of situation awareness in situated agents. *Theoretical Issues in Ergonomics Science*, 13(6), 625-647.
- Checkland, P. et Holwell, S. (1998). Action research: its nature and validity. *Systemic Practice and Action Research*, 11(1), 9-21.
- Coghlan, D., et Brannick, T. (2014). *Doing action research in your own organization*. Sage.
- Cohen, S. G. et Bailey, D. E. (1997). What makes teams work: Group effectiveness research from the shop floor to the executive suite. *Journal of management*, 23(3), 239-290.

- Cooke, N. J., Gorman, J. C., Myers, C. W. et Duran, J. L. (2013). Interactive team cognition. *Cognitive science*, 37(2), 255-285.
- Cr    , F. (2006). *La recherche-action et les transformations sociales*. Editions L'Harmattan.
- Crowston, K., et Prestopnik, N. R. (2013, January). Motivation and data quality in a citizen science game: A design science evaluation. In *System Sciences (HICSS), 2013 46th Hawaii International Conference on* (pp. 450-459). IEEE.
- Daniels, C. B., et LaMarsh, W. J. (2007, April). Complexity as a cause of failure in information technology project management. In *System of Systems Engineering, 2007. SoSE'07. IEEE International Conference on* (pp. 1-7). IEEE.
- Dardenne, A., Van Lamsweerde, A. et Fickas, S. (1993). Goal-directed requirements acquisition. *Science of computer programming*, 20(1), 3-50.
- Davenport, T. H. (2013). *Process innovation: reengineering work through information technology*. Harvard Business Press.
- Davies, A., et Brady, T. (2000). Organisational capabilities and learning in complex product systems: towards repeatable solutions. *Research policy*, 29(7), 931-953.
- DeFranco, J. F., Neill, C. J. et Clariana, R. B. (2011). A cognitive collaborative model to improve performance in engineering teams—A study of team outcomes and mental model sharing. *Systems Engineering*, 14(3), 267-278.
- DeMarco, T. (2002). Structured analysis and system specification. In *Software pioneers*, 529-560. Springer Berlin Heidelberg.
- Demir, M., McNeese, N. J., et Cooke, N. J. (2017). Team situation awareness within the context of human-autonomy teaming. *Cognitive Systems Research*, 46, 3-12.
- Denscombe, M. (2014). *The good research guide: for small-scale social research projects*. McGraw-Hill Education (UK).
- Doolin, B., et Lowe, A. (2002). To reveal is to critique: actor-network theory and critical information systems research. *Journal of information technology*, 17(2), 69-78.
- DiVincenzo, T. (2006). Project managers stay in charge and out front. *Occupational Outlook Quarterly*, 50(2), 19-25.

- Drury, M., Conboy, K., et Power, K. (2012). Obstacles to decision making in Agile software development teams. *Journal of Systems and Software*, 85(6), 1239-1254.
- Endsley, M. R. (1988, October). Design and evaluation for situation awareness enhancement. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 33(2), 97-101. SAGE Publications.
- Endsley, M. R. (1989). Final report: Situation awareness in an advanced strategic mission (NOR DOC 89-32). *Hawthorne, CA: Northrop Corporation*.
- Endsley, M. R. (1990, October). Predictive utility of an objective measure of situation awareness. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 34(1), 41-45. SAGE Publications.
- Endsley, M. R. (1993). A survey of situation awareness requirements in air-to-air combat fighters. *The International Journal of Aviation Psychology*, 3(2), 157-168.
- Endsley, M. R. (1995a). Toward a theory of situation awareness in dynamic systems. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 37(1), 32-64.
- Endsley, M. R. (1995b). Measurement of situation awareness in dynamic systems. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 37(1), 65-84.
- Endsley, M. R.. et Robertson, M. M. (2000). Situation awareness in aircraft maintenance teams. *International Journal of Industrial Ergonomics*, 26(2), 301-325.
- Endsley, M. R. (2000). Theoretical underpinnings of situation awareness: A critical review. *Situation awareness analysis and measurement*, 3-32.
- Endsley, M. R., Sollenberger, R. et Stein, E. (2000). Situation awareness: A comparison of measures. *Proceedings of the Human Performance, Situation Awareness and Automation: User-Centered Design for the New Millennium*, 15-19.
- Endsley, M. R.. et Jones, W. M. (2001). A model of inter-and intra-team situation awareness: Implications for design, training and measurement. *New trends in cooperative activities: Understanding system dynamics in complex environments*, 7, 46-67.
- Endsley, M. R. (2006). Expertise and situation awareness. *The Cambridge handbook of expertise and expert performance*, 633-651.

- Endsley, M. R. (2012). *Designing for Situation Awareness: An Approach to User-Centered Design*, Second Edition. CRC Press
- Endsley, M. R. et Jones, W. (2013). Situation awareness. *The Oxford Handbook of Cognitive Engineering*, 88-108.
- Endsley, M. R. (2015). Situation awareness misconceptions and misunderstandings. *Journal of Cognitive Engineering and Decision Making*, 9(1), 4-32.
- Engwall, M. (2003). No project is an island: linking projects to history and context. *Research policy*, 32(5), 789-808.
- España, S., González, A. et Pastor, Ó. (2009, January). Communication Analysis: a requirements engineering method for information systems. In *Advanced Information Systems Engineering*, 530-545. Springer Berlin Heidelberg.
- Espinosa, A., Lerch, F. J., Kraut, R. E., Salas, E., et Fiore, S. M. (2004). Explicit vs. implicit coordination mechanisms and task dependencies: one size does not fit all. *Team cognition: understanding the factors that drive process and performance*. American Psychological Association, Washington, DC, 107-129.
- Ewusi-Mensah, K. (1997). Critical issues in abandoned information systems development projects. *Communications of the ACM*, 40(9), 74-80.
- Fayaz, A., Kamal, Y., Amin, S., et Khan, S. (2017). Critical success factors in information technology projects. *Management Science Letters*, 7(2), 73-80.
- Feather, M. S., Fickas, S. et Helm, B. R. (1991, September). Composite system design: the good news and the bad news. In *Knowledge-Based Software Engineering Conference, 1991. Proceedings., 6th Annual*, 16-25. IEEE.
- Fioratou, E., Flin, R., Glavin, R.. et Patey, R. (2010). Beyond monitoring: distributed situation awareness in anaesthesia. *British journal of anaesthesia*, 105(1), 83-90.
- Foushee, R. L. et Helmreich . H. C. (1993). Why Crew Resource Management? Empirical and Theoretical Bases of Human Factors Training in Aviation. *Cockpit Resource Management*, 1.
- Fowler, M., et Highsmith, J. (2001). The agile manifesto. *Software Development*, 9(8), 28-35.

- Fracker, M. L. (1988, October). A theory of situation assessment: Implications for measuring situation awareness. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 32(2), 102-106. SAGE Publications.
- Fracker, M.L. (1989) - Attention gradients in situation awareness. In AGARD conference proceedings 478 *La perception de la situation au cours des opérations aériennes*. Neuilly-sur-Seine : AGARD.
- Fracker, M. L. (1991). *Measures of situation awareness: Review and future directions* (No. AL-TR-1991-0128). LOGUE (GEORGE E) INC MONTROUSEVILLE PA.
- Independent.co.uk, (Fevrier, 2017) (<http://www.independent.co.uk/travel/news-and-advice/amsterdam-airport-in-chaos-air-traffic-control-system-failure-delays-cancelled-flights-a7556906.html>).
- Garg, V. et Camp, L. J. (2015). Risk Characteristics, Mental Models, and Perception of Security Risks.
- Geisser, M., et Hildenbrand, T. (2006). A method for collaborative requirements elicitation and decision-supported requirements analysis. In *Advanced software engineering: expanding the frontiers of software technology*, 108-122. Springer US.
- Geneca (2011). Doomed from the start? Why a majority of business and IT teams anticipate their software development project will fail. Geneca, Oakbrook Terrace, IL. Winter 2010/2011. Industry Survey. Récupéré de http://www2.geneca.com/l/24412/2013-0725/3fvr/24412/5010/Doomed_From_the_Start.pdf
- Gentner, D. et Stevens, A. L. (2014). *Mental models*. Psychology Press.
- Gillard, S. (2009). Soft skills and technical expertise of effective project managers. *Issues in informing science & information technology*, 6.
- Gillespie, B. M., Gwinner, K., Fairweather, N. et Chaboyer, W. (2013). Building shared situational awareness in surgery through distributed dialog. *Journal of multidisciplinary healthcare*, 6, 109.
- Glavin, R. J. (2011). Human performance limitations (communication, stress, prospective memory and fatigue). *Best Practice & Research Clinical Anaesthesiology*, 25(2), 193-206.

- Grabher, G. (2002). The project ecology of advertising: tasks, talents and teams. *Regional studies*, 36(3), 245-262.
- Graham, S. E. et Matthews, M. D. (2000). Modeling and measuring situation awareness. In *Workshop on assessing and measuring training performance effectiveness*, 14-24.
- Gramopadhye, A. K., Drury, C. G., et Sharit, J. (1993, October). Training for decision making in aircraft inspection. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (Vol. 37, No. 18, pp. 1267-1271). Sage CA: Los Angeles, CA: SAGE Publications.
- Gugerty, L. (2011). Situation awareness in driving. *Handbook for driving simulation in engineering, medicine and psychology*.
- Hogan, M. P., Pace, D. E., Hapgood, J. et Boone, D. C. (2006). Use of human patient simulation and the situation awareness global assessment technique in practical trauma skills assessment. *Journal of Trauma and Acute Care Surgery*, 61(5), 1047-1052.
- Hood, A. C., Bachrach, D. G. et Lewis, K. (2014). Transactive Memory Systems, Conflict, Size and Performance in Teams. *Journal of Leadership, Accountability and Ethics*, 11(3), 11.
- Holtzblatt, K. et Beyer, H. (2014). Contextual Design: Evolved. *Synthesis Lectures on Human-Centered Informatics*, 7(4), 1-91.
- Hutchins, E. et Klausen, T. (1996). Distributed cognition in an airline cockpit. *Cognition and communication at work*, 15-34.
- IATA, (2015) : SAFETY REPORT 2014, Issued April 2015. Récupéré de <http://www.iata.org/publications/Documents/iata-safety-report-2014.pdf>
- Ifenthaler, D. et Pirnay-Dummer, P. (2014). Model-based tools for knowledge assessment. In *Handbook of research on educational communications and technology*, 289-301. Springer New York.
- Jacobsson, M., Lundin, R. A., et Söderholm, A. (2015). Researching projects and theorizing families of temporary organizations. *Project Management Journal*, 46(5), 9-18.

- Järvinen, P. (2007). Action research is similar to design science. *Quality & Quantity*, 41(1), 37-54.
- Jo, I. H. (2012). Shared mental models on the performance of e-learning content development teams. *Journal of Educational Technology & Society*, 15(1), 289-297.
- Johannesen, L., Sarter, N., Cook, R., Dekker, S. et Woods, D. D. (2012). *Behind Human Error*. Ashgate Publishing, Ltd.
- Johnson, D. W. et Johnson, F. P. (1991). *Joining together: Group theory and group skills*. Prentice-Hall, Inc.
- Jones, D. G. et Endsley, M. R. (2000). Can real-time probes provide a valid measure of situation awareness. *Proceedings of the Human Performance, Situation Awareness and Automation: User-Centered Design for the New Millennium, Savannah, GA*.
- Jones, P. E. et Roelofsma, P. H. (2000). The potential for social contextual and group biases in team decision-making: Biases, conditions and psychological mechanisms. *Ergonomics*, 43(8), 1129-1152.
- Jones, D. G. et Kaber, D. B. (2004). Situation awareness measurement and the situation awareness global assessment technique. *Handbook of human factors and ergonomics methods*, 42-1.
- Jonker, C. M., Van Riemsdijk, M. B. et Vermeulen, B. (2011). Shared mental models. In *Coordination, Organizations, Institutions, and Norms in Agent Systems VI*, 132-151. Springer Berlin Heidelberg.
- Kaber, D. B., Riley, J. M., Endsley, M. R., Sheik-Nainar, M., Zhang, T. et Lampton, D. R. (2013). Measuring situation awareness in virtual environment-based training. *Military Psychology*, 25(4), 330.
- Kahneman, D. et Tversky, A. (1982). On the study of statistical intuitions. *Cognition*, 11(2), 123-141.
- Kalargiros, E. M. et Manning, M. R. (2015). Divergent Thinking and Brainstorming in Perspective: Implications for Organization Change and Innovation. In *Research in Organizational Change and Development*, 293-327. Emerald Group Publishing Limited.

- Kass, S. J., Herschler, D. A., et Companion, M. A. (1990, October). Are they shooting at me?: An approach to training situational awareness. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 34(18), 1352-1356. SAGE Publications.
- Loucopoulos, P., et Kavakli, E. (1995). Enterprise modelling and the teleological approach to requirements engineering. *International Journal of Cooperative Information Systems*, 4(01), 45-79.
- Keen, P. G. (1991). Redesigning the organization through information technology. *Planning Review*, 19(3), 4-9.
- Kemmis, S., McTaggart, R., et Nixon, R. (2013). *The action research planner: Doing critical participatory action research*. Springer Science & Business Media.
- Kendra, K. A., et Taplin, L. J. (2004). Change agent competencies for information technology project managers. *Consulting Psychology Journal: Practice and Research*, 56(1), 20.
- Kerzner, H. (1989). Systems project management: A case study at the IRS. *Journal of Systems Management*, 40(1), 7.
- Kerzner, H. (2013). *Project management: a systems approach to planning, scheduling, and controlling*. John Wiley & Sons.
- Klein, G., (1989). A recognition-Primed Decision (RPD) Model of Rapid Decision Making, Gary A. Klein. Klein Associates Inc. Fairborn, OH, 1989
- Klein, G. (1995). Studying situation awareness in the context of decision-making incidents. *Experimental Analysis and Measurement of Situation Awareness*, Daytona Beach, FL.
- Kurapati, S., Kolfshoten, G., Verbraeck, A., Drachsler, H., Specht, M., & Brazier, F. (2012). A theoretical framework for shared situational awareness in sociotechnical systems. In *Proc. 2nd Workshop on Awareness and Reflection in Technology-Enhanced Learning* (pp. 47-53).
- Klimoski, R., et Mohammed, S. (1994). Team mental model: Construct or metaphor?. *Journal of management*, 20(2), 403-437.
- Latour, B. (2005). *Reassembling the social: An introduction to actor-network-theory*. Oxford university press.

- Ladas, C. (2009). Scrumban-essays on kanban systems for lean software development. Lulu. com.
- Lai, H., Peng, R. et Ni, Y. (2014). Evaluating the BPCRAR Method: A Collaborative Method for Business Process Oriented Requirements Acquisition and Refining. In *Requirements Engineering*, 90-104. Springer Berlin Heidelberg.
- Lapresse.ca, (Septembre, 2014)
(<http://www.lapresse.ca/actualites/sante/201409/11/01-4799125-vingt-hopitaux-touchees-par-une-panne-informatique.php>), Accédé en septembre 2017
- Lapresse.ca, (Mars 2007), <http://affaires.lapresse.ca/economie/200901/06/01-676861-limpot-federal-en-panne-informatique.php>). Accédé en septembre 2017
- Law, J. (2009). Actor network theory and material semiotics. *The new Blackwell companion to social theory*, 141-158.
- Lewandowsky, S. et Kirsner, K. (2000). Knowledge partitioning: Context-dependent use of expertise. *Memory & Cognition*, 28(2), 295-305.
- Lewis, K. (2003). Measuring transactive memory systems in the field: scale development and validation. *Journal of Applied Psychology*, 88(4), 587.
- Lewis, K. et Herndon, B. (2011). Transactive memory systems: Current issues and future research directions. *Organization Science*, 22(5), 1254-1265.
- Lezcano, R., Alfonso, L., Guzmán, L., Alberto, J., Gómez, A. et Alonso, S. (2015). Extraction of goals and their classification in the KAOS model using natural language processing. *Ingeniare. Revista chilena de ingeniería*, 23(1), 59-66.
- Lim, B. C. et Klein, K. J. (2006). Team mental models and team performance: a field study of the effects of team mental model similarity and accuracy. *Journal of Organizational Behavior*, 27(4), 403.
- Liu, M. (1997). Fondements et pratiques de la recherche-action.
- Manning, S. (2008). Embedding projects in multiple contexts—a structuration perspective. *International Journal of Project Management*, 26(1), 30-37.
- Marks, M. A., Sabella, M. J., Burke, C. S., et Zaccaro, S. J. (2002). The impact of cross-training on team effectiveness. *Journal of Applied Psychology*, 87(1), 3.

- Matthews, M. D. et Beal, S. A. (2002). *Assessing situation awareness in field training exercises*. MILITARY ACADEMY WEST POINT NY OFFICE OF MILITARY PSYCHOLOGY AND LEADERSHIP.
- Maynard, M. T. et Gilson, L. L. (2014). The role of shared mental model development in understanding virtual team effectiveness. *Group & Organization Management*, 39(1), 3-32.
- McGuinness, B. et Ebbage, L. (2002). *Assessing human factors in command and control: workload and situational awareness metrics*. BAE SYSTEMS BRISTOL (UNITED KINGDOM) ADVANCED TECHNOLOGY CENTRE.
- McGuinness, B. et Foy, L. (2000, October). A subjective measure of SA: the Crew Awareness Rating Scale (CARS). In *Proceedings of the first human performance, situation awareness, and automation conference, Savannah, Georgia*.
- McManus, S. (2008). Organizational resilience in New Zealand (Ph. D. Thesis). *University of Canterbury, Christchurch, New Zealand*.
- McLean, C., et Hassard, J. (2004). Symmetrical absence/symmetrical absurdity: Critical notes on the production of actor - network accounts. *Journal of Management Studies*, 41(3), 493-519.
- McLeod, G., et Smith, D. (1996). *Managing information technology projects*. Course Technology.
- Miah, J., S., Kerr, D., et von Hellens, L. (2014). A collective artefact design of decision support systems: design science research perspective. *Information Technology & People*, 27(3), 259-279.
- Mohammed, S. et Dumville, B. C. (2001). Team mental models in a team knowledge framework: Expanding theory and measurement across disciplinary boundaries. *Journal of Organizational Behavior*, 22, 89-106.
- Meister, D. (1985). *Behavioural foundations of system development* (2nd Ed.). Malabar: Robert E. Kreiger.
- Mertler, C. A. (2008). *Action research: Teachers as researchers in the classroom*. Sage.
- Molesworth, B. R. C. et Estival, D. (2015). Miscommunication in general aviation: The influence of external factors on communication errors. *Safety Science*, 73, 73-79.

- Moray, N. et Sheridan, T. B. (2005). Où sont les neiges d'antan?. *Human Performance, Situation Awareness, and Automation: Current Research and Trends HPSAA II, Volumes I and II*, 1.
- Morgan Jr, B. B., Glickman, A. S., Woodard, E. A., Blaiwes, A. S. et Salas, E. (1996) *Measurement of team behaviors in a Navy environment*. BATTELLE COLUMBUS LABS RESEARCH TRIANGLE PARK NC.
- Morgeson, F. P., Aiman-Smith, L. D. et Campion, M. A. (1997). Implementing work teams: Recommendations from organizational behavior and development theories. *Advances in interdisciplinary studies of work teams: Issues in the implementation of work teams*, 4, 1-44.
- Mullen, B., Johnson, C. et Salas, E. (1991). Productivity loss in brainstorming groups: A meta-analytic integration. *Basic and applied social psychology*, 12(1), 3-23.
- Nazira, S., Sorensenb, L. J., Øvergårdb, K. I. et Manca, D. (2014). How distributed situation awareness influences process safety. *CHEMICAL ENGINEERING*, 36.
- Neisser, U. (1976). *Cognition and reality: Principles and implications of cognitive psychology*. WH Freeman/Times Books/Henry Holt & Co.
- Nofi, A.A. (2003) – *Defining and measuring shared situational awareness*. Technical report CRM D0002895.A1, Virginia : Center for Naval Analyses.
- Nullmeyer, R. T., Stella, D., Montijo, G. A. et Harden, S. W. (2005). Human factors in Air Force flight mishaps: Implications for change. In *Proceedings of the 27th Annual Interservice/Industry Training, Simulation, and Education Conference*.
- Nuseibeh, B. et Easterbrook, S. (2000, May). Requirements engineering: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, 35-46. ACM.
- O'Connor, P., Byrne, D., O'Dea, A., McVeigh, T. P. et Kerin, M. J. (2013). "Excuse Me:" Teaching Interns to Speak Up. *Joint Commission Journal on Quality and Patient Safety*, 39(9), 426-431.
- O'Hare, D., Wiggins, M., Williams, A. et Wong, W. (2000). Cognitive task analysis for decision centred design and training. *Task analysis*, 170-190.
- Orasanu, J. (1990). Shared mental models and crew decision making (Report No. 46). Princeton, NJ: Princeton University, Cognitive Science Laboratory

- Orasanu, J., Martin, L. et Davison, J. (2001). Cognitive and contextual factors in aviation accidents: decision errors. In E. Salas & G. Klein (Eds.), *Linking expertise and naturalistic decision-making*, 209-243. Mahwah, NJ: Lawrence Erlbaum Associates.
- Orlikowski, W. J., et Yates, J. (2002). It's about time: Temporal structuring in organizations. *Organization science*, 13(6), 684-700.
- Patton, M. Q. (2005). *Qualitative research*. John Wiley & Sons, Ltd.
- Patel, P., Ranabahu, A. H., et Sheth, A. P. (2009). Service level agreement in cloud computing.
- Pant, I., et Baroudi, B. (2008). Project management education: The human skills imperative. *International journal of project management*, 26(2), 124-128.
- Paris, C. R., Salas, E. et Cannon-Bowers, J. A. (2000). Teamwork in multi-person systems: a review and analysis. *Ergonomics*, 43(8), 1052-1075.
- Pertet, S., et Narasimhan, P. (2005). Causes of failure in web applications (cmu-pdl-05-109). *Parallel Data Laboratory*, 48.
- Pinelle, D., Gutwin, C., et Greenberg, S. (2003). Task analysis for groupware usability evaluation: Modeling shared-workspace tasks with the mechanics of collaboration. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 10(4), 281-311.
- Pohl, K. et Ulfat-Bunyadi, N. (2013). The Three Dimensions of Requirements Engineering: 20 Years Later. In *Seminal Contributions to Information Systems Engineering*, 81-87. Springer Berlin Heidelberg.
- Potts, C., Takahashi, K. et Anton, A (1994) : Inquiry-Based Requirements Analysis. *IEEE Software*, March 1994, 21-32
- Potts, L. (2008). Diagramming with actor network theory: a method for modeling holistic experience. In *IEEE International Professional Communication Conference (IPCC 2008), Montreal, Quebec, Canada.*, (pp. 1-6). Washington, DC: IEEE.
- Prince, C. et Salas, E. (1997) - The role of situation assessment in the conduct of flight and in decision making. In *Engineering psychology and cognitive ergonomics, volume 1, Transportation systems, 1*.

- Qureshi, S., Liu, M., et Vogel, D. (2006). The effects of electronic collaboration in distributed project management. *Group decision and negotiation*, 15(1), 55-75.
- Regal, D. M., Rogers, W. H. et Boucek, G. P. (1988). *Situational awareness in the commercial flight deck: Definition, measurement, and enhancement* (No. 881508). SAE Technical Paper.
- Respect-IT. (2007) : A KAOS Tutorial. Récupéré de <http://www.objectiver.com/fileadmin/download/documents/KaosTutorial.pdf>
- Riemenschneider, C. K., Hardgrave, B. C., et Davis, F. D. (2002). Explaining software developer acceptance of methodologies: a comparison of five theoretical models. *IEEE transactions on Software Engineering*, 28(12), 1135-1145.
- Riss, U. V., Rickayzen, A., Maus, H., et van der Aalst, W. M. (2005). Challenges for business process and task management. *Journal of Universal Knowledge Management*, 2, 77-100.
- Robertson, M. M., Taylor, J. C., Stelly, J. W. et Wagner, R. (1995). A systematic training evaluation model applied to measure the effectiveness of an aviation maintenance team training program. In *Proceedings of the Eighth International Symposium on Aviation Psychology*, 631-636.
- Robertson, M. M., et Endsley, M. R. (1997, October). Development of a situation awareness training program for aviation maintenance. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (Vol. 41, No. 2, pp. 1163-1167). Sage CA: Los Angeles, CA: SAGE Publications.
- Robinson, W. N. et Volkov, V. (1998). Supporting the negotiation life cycle. *Communications of the ACM*, 41(5), 95-102.
- Rognin, L., Salembier, P., et Zouinar, M. (2000). Cooperation, reliability of socio-technical systems and allocation of function. *International Journal of Human-Computer Studies*, 52(2), 357-379.
- Rook, L. (2013). Mental models: a robust definition. *The Learning Organization*, 20(1), 38-47.
- Rose, K. H. (2013). A Guide to the Project Management Body of Knowledge (PMBOK® Guide)—Fifth Edition. *Project management journal*, 44(3), e1-e1.

- Ross, D. T. (1978). Structured Analysis (SA): A Language for Communicating Ideas. In *Programming Methodology*, 388-421. Springer New York.
- Rumbaugh, J., Jacobson, I. et Booch, G. (2004). *Unified Modeling Language Reference Manual, The*. Pearson Higher Education.
- Sacco, R. (2006). Yogi Berra, PMP. *Computerworld*, 40(43), 44.
- Sampson, B. (2007). Get with the project. *Professional Engineering*, 20(12), 41-42.
- Sandhåland, H., Oltedal, H. et Eid, J. (2015). Situation awareness in bridge operations -A study of collisions between attendant vessels and offshore facilities in the North Sea. *Safety science*, 79, 277-285.
- Sandhåland, H., Oltedal, H. A., Hystad, S. W. et Eid, J. (2015). Distributed situation awareness in complex collaborative systems: A field study of bridge operations on platform supply vessels. *Journal of occupational and organizational psychology*, 88(2), 273-294.
- Salas, E., Prince, C., Baker, D. P. et Shrestha, L. (1995). Situation awareness in team performance: Implications for measurement and training. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 37(1), 123-136.
- Salas, E., Cooke, N. J. et Rosen, M. A. (2008). On teams, teamwork, and team performance: Discoveries and developments. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 50(3), 540-547.
- Salas, E., Wilson, K. A., Burke, C. S., et Wightman, D. C. (2006). Does crew resource management training work? An update, an extension, and some critical needs. *Human Factors*, 48(2), 392-412.
- Salmon, P. M., Stanton, N. A., Walker, G. H., Jenkins, D., Ladva, D., Rafferty, L. et Young, M. (2009). Measuring Situation Awareness in complex systems: Comparison of measures study. *International Journal of Industrial Ergonomics*, 39(3), 490-500.
- Salmon, P., Stanton, N., Walker, G. et Green, D. (2006). Situation awareness measurement: A review of applicability for C4i environments. *Applied ergonomics*, 37(2), 225-238.

- Salmon, P. M., Stanton, N. A., Walker, G. H., Baber, C., Jenkins, D. P., McMaster, R., et Young, M. S. (2008). What really is going on? Review of situation awareness models for individuals and teams. *Theoretical Issues in Ergonomics Science*, 9(4), 297-323.
- Sarter, N. B. et Woods, D. D. (1991). Situation awareness: A critical but ill-defined phenomenon. *The International Journal of Aviation Psychology*, 1(1), 45-57.
- Schmidt, R., Lyytinen, K., et Mark Keil, P. C. (2001). Identifying software project risks: An international Delphi study. *Journal of management information systems*, 17(4), 5-36.
- Schulz, C. M., Endsley, M. R., Kochs, E. F., Gelb, A. W. et Wagner, K. J. (2013). Situation awareness in anesthesia: concept and research. *Anesthesiology*, 118(3), 729-742.
- Schwaber, K., et Beedle, M. (2002). *Agile software development with Scrum (Vol. 1)*. Upper Saddle River: Prentice Hall.
- Schwartz, D. (1990). Training for situational awareness. Houston, TX: *Flight Safety International*
- Shotter, J. (2007). With what kind of science should action research be contrasted? *International Journal of Action Research*, 3(1+ 2), 65-92.
- Smith, K. et Hancock, P. A. (1995). Situation awareness is adaptive, externally directed consciousness. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 37(1), 137-148.
- Smith-Jentsch, K. A., Mathieu, J. E. et Kraiger, K. (2005). Investigating linear and interactive effects of shared mental models on safety and efficiency in a field setting. *Journal of Applied Psychology*, 90(3), 523.
- Smolensky, M. W. (1993). Toward the physiological measurement of situation awareness: The case for eye movement measurements. In *Proceedings of the Human Factors and Ergonomics Society 37th annual meeting*, 41-42.
- Singh, H., Giardina, T. D., Petersen, L. A., Smith, M. W., Paul, L. W., Dismukes, K., Bhagwath, G. et Thomas, E. J. (2012). Exploring situational awareness in diagnostic errors in primary care. *BMJ quality & safety*, 21(1), 30-38.

- Sorensen, L. J., Stanton, N. A., et Banks, A. P. (2011). Back to SA school: contrasting three approaches to situation awareness in the cockpit. *Theoretical Issues in Ergonomics Science*, 12(6), 451-471.
- Sonesh, S., Rico, R., et Salas, E. (2013). Team decision making in naturalistic environments: A framework for and introduction to illusory shared cognition. *Judgment and decision making at work*, 199-227.
- Souza, J., Botega, L., Segundo, J. E. S. et Berti, C. (2015). A Methodology for the Assessment of the Quality of Information from Robbery Events to Enrich Situational Awareness in Emergency Management Systems. *Procedia Manufacturing*, 3, 4407-4414.
- Stephenson, G. M. (2015). Group Remembering. *Psychology of Group Influence*, 22, 357.
- Stanton, N. A., Salmon, P. M., Walker, G. H. et Jenkins, D. P. (2010). Is situation awareness all in the mind?. *Theoretical Issues in Ergonomics Science*, 11(1-2), 29-40.
- Stanton, N. A., Stewart, R., Harris, D., Houghton, R. J., Baber, C., McMaster, R. et Green, D. (2006). Distributed situation awareness in dynamic systems: theoretical development and application of an ergonomics methodology. *Ergonomics*, 49(12-13), 1288-1311.
- Stepniczka, I., Tomova, L., Niedermeier, D., Peschl, F. M. et Lamm, C. (2015). Social Situation Awareness: Empathic Accuracy in the Aircraft Cockpit. In *Proceedings of the 37th Annual Meeting of the Cognitive Science Society (COGSCI 2015)*. P.
- Subba Rao, S. (2000). Enterprise resource planning: business needs and technologies. *Industrial Management & Data Systems*, 100(2), 81-88.
- Sundstrom, E., De Meuse, K. P. et Futrell, D. (1990). Work teams: Applications and Effectiveness. *American psychologist*, 45(2), 120.
- Suraweera, T., Pulakanam, V., et Guler, O. (2006, December). Managing the implementation of IT projects in SMEs: An exploratory investigation. In *Digital Information Management, 2006 1st International Conference on* (pp. 381-388). IEEE.
- Tesch, D., Kloppenborg, T. J., et Frolick, M. N. (2007). IT project risk factors: the project management professionals perspective. *Journal of Computer Information Systems*, 47(4), 61-69.

- Thomas, G., et Fernández, W. (2008). Success in IT projects: A matter of definition? *International journal of project management*, 26(7), 733-742.
- Thompson, L. L. et Wilson, E. R. (2015). Creativity in Teams. *Emerging Trends in the Social and Behavioral Sciences: An Interdisciplinary, Searchable, and Linkable Resource*.
- Taylor, J. (2003). Managing information technology projects: applying project management strategies to software, hardware, and integration initiatives. Amacom.
- Taylor, R. M. (1990). Situation awareness rating technique (SART): the development of a tool for aircrew systems design. In *Situational Awareness in Aerospace Operations* (Chapter 3). France: Neuilly-sur-Seine, NATO-AGARD-CP-478.
- Taylor, J.C. et Robertson, M.M. (1995). *The effects of crew resource management (CRM training in airline maintenance: results following three year's experience*. NASA Ames Research Center Office of Life and Microgravity Sciences and Applications, Moffett Field, CA.
- U.S. Department of Transportation, Federal Aviation Administration, (2014). Facility operation and administration (Order JO 7210.3Y). Récupéré de <https://www.faa.gov/documentLibrary/media/Order/7210.3Y.pdf>
- Van den Bossche, P., Gijselaers, W., Segers, M., Woltjer, G. et Kirschner, P. (2011). Team learning: building shared mental models. *Instructional Science*, 39(3), 283-301.
- Vidulich, M. (1994). Cognitive and performance components of situation awareness: SAINT team task one report. *Situation awareness: Papers and annotated bibliography*, 17-28. Armstrong Laboratory, Wright-Patterson AFB: OH.
- Vaughan, G. et Hogg, M. A. (2005). *Introduction to social psychology*. Pearson Education Australia.
- Waag, W. L. et Houck, M. R. (1994). Tools for assessing situational awareness in an operational fighter environment. *Aviation, space, and environmental medicine*.
- Walker, G. H., Gibson, H., Stanton, N. A., Baber, C., Salmon, P. et Green, D. (2006). Event analysis of systemic teamwork (EAST): a novel integration of ergonomics methods to analyse C4i activity. *Ergonomics*, 49(12-13), 1345-1369.

- Wang, X., Conboy, K., et Cawley, O. (2012). "Leagile" software development: An experience report analysis of the application of lean approaches in agile software development. *Journal of Systems and Software*, 85(6), 1287-1299.
- Wegner, D. M., Erber, R. et Raymond, P. (1991). Transactive memory in close relationships. *Journal of personality and social psychology*, 61(6), 923.
- Wellens, A. R. (1993). Group situation awareness and distributed decision making: From military to civilian applications. *Individual and group decision making: Current issues*, 267-287.
- Wickens, C. D. (1995). Situation awareness: Impact of automation and display technology. In *Aerospace Research and Development Conference Proceedings* 575, 71-78. Neully-Sur-Seine, France: Advisory Group for Aerospace Research and Development.
- Wicks, P. G., et Reason, P. (2009). Initiating action research: Challenges and paradoxes of opening communicative space.
- Wooldridge, M., Jennings, N. R. et Kinny, D. (2000). The Gaia methodology for agent-oriented analysis and design. *Autonomous Agents and multi-agent systems*, 3(3), 285-312.
- Yip, C., Fiorenzo, P. J., Jung, K. D., Tupper, J. et Santos, J. (2015, April). A requirements engineering framework of the evolving US energy landscape. In *Systems and Information Engineering Design Symposium (SIEDS)*, 2015, 348-353. IEEE.
- Yu, E. et Mylopoulos, J. (1994, December). Towards modelling strategic actor relationships for information systems development—with examples from business process reengineering. In *Proceedings of the 4th Workshop on Information Technologies and Systems, Vancouver, BC, Canada*, 21-28.
- Yu, E. S. (2011). Modelling strategic relationships for process reengineering. *Social Modeling for Requirements Engineering*, 11, 2011.
- Yu, E. S. (2002). Agent-oriented modelling: software versus the world. In *Agent Oriented Software Engineering II*, 206-225. Springer Berlin Heidelberg.
- Zhang, Q., Cheng, L., et Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1), 7-18.
- Zsombok, C. E. et Klein, G. (2014). *Naturalistic decision making*. Psychology Press.