MASTER SIC
EM SORBONNE

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

MASTER MANAGEMENT DES SYSTEMES D'INFORMATION –

SPECIALITE PROFESSIONNELLE : EXECUTIVE MANAGEMENT DES SYSTEMES D'INFORMATION ET DE CONNAISSANCE

MEMOIRE

Comment accorder méthodes agiles et gestion raisonnée des systèmes d'information.

Proposition d'un modèle développé empiriquement dans le secteur bancaire centré sur l'ingénierie

REDIGE ET SOUTENU PAR :

KHALIL BEN AHMED

PROMOTION JB 2020

DIRECTEUR DE MEMOIRE :

SAMUEL PARFOURU

DATE DE LA SOUTENANCE :

8 DECEMBRE 2021

L'UNIVERSITE N'ENTEND DONNER AUCUNE APPROBATION NI IMPROBATION AUX OPINIONS EMISES DANS CE MEMOIRE : CES OPINIONS DOIVENT ETRE CONSIDEREES COMME PROPRES A LEUR AUTEUR.

Remerciements

Je tiens, tout d'abord, à remercier mon directeur de mémoire M. Samuel PARFOURU, pour sa patience, sa disponibilité et surtout ses judicieux conseils, qui ont largement contribué à alimenter ma réflexion.

Mes remerciements vont également à tous mes professeurs dans le cadre du master « Managment des systèmes d'information » à l'Université de Paris-Sorbonne. Leur compétence et la qualité de leur enseignement m'ont fourni les outils indispensables à la conduite de cette recherche.

Je remercie vivement les équipes CA Consumer Finance Service Numérique France Partenariats Bancaires Groupe et Architecture d'entreprise, au sein desquelles j'ai pu développer et appliquer les travaux du présent mémoire.

Je tiens à remercier également les membres du jury pour l'honneur qu'ils me font en acceptant d'évaluer le présent travail.

Table des matières

Table des figures

Introduction générale

De nos jours, toutes les grandes entreprises ont entrepris une transformation numérique afin d'automatiser leurs activités et opérations. Cette course vers l'informatisation des processus métiers fait qu'au fil des ans les entreprises se sont constitué un portefeuille applicatif conséquent qui pilote toutes leurs activités et embarque toute leur logique métier.

En effet « l'informatique joue aujourd'hui un rôle décisif dans la réalisation des objectifs et de la mission de l'entreprise : elle est impliquée dans sa stratégie globale. Avec des orientations et des intentions clairement définies, il sera plus facile de la mettre en parfaite adéquation avec les besoins métier » Chris Ferguson, DSI, Elders Rural Services.

Par conséquent, face à l'évolution constante des besoins fonctionnels, des stratégies commerciales et objectives de l'organisation et face à l'obsolescence et déficience des applications, les entreprises se doivent de faire évoluer et transformer leur parc applicatif afin de répondre à ces nouvelles exigences et créer de la valeur. Cela se traduit par le développement d'un ensemble de nouvelles fonctionnalités qui viennent s'intégrer au SI, la modification du processus métier du SI existant ou la migration applicative.

Ce besoin perpétuel d'évolution maintient une pression constante sur les entreprises et leurs DSI. En effet, tout changement entraîne des risques et des coûts qui pèsent sur la stratégie de croissance de l'entreprise. Ceux qui ne disposent pas de la flexibilité nécessaire pour implémenter ces transformations se voient confrontés à la problématique de l'inertie et par extension à une perte de productivité.

Ainsi, toute entreprise qui a pour ambition de pérenniser ses activités et augmenter ses bénéfices se doit de s'inscrire dans cette dynamique d'évolution constante ; ce qui requiert agilité et efficacité. Le patrimoine applicatif des entreprises est un des actifs le plus stratégique et son maintien à jour face aux exigences métiers est d'une importance capitale.

C'est dans ce contexte de quête perpétuelle d'évolution que l'avènement de l'Agile a été vécu comme une révolution. En effet, ce nouveau paradigme casse les lignes en proposant une démarche adaptative qui rompt avec le système de pensée déterministe. Cette approche itérative incrémentale centrée sur le client s'est rapidement propagée à l'ensemble des organes de l'entreprise passant par l'organisation et allant jusqu'à transcender la culture d'entreprise. L'Agile n'est plus une

affaire de la DSI, mais celui de l'ensemble de l'entreprise. Ce phénomène inhérent au développement logiciel et très local à sa naissance s'est propulsé au sommet des organisations et a accaparé alors toutes les attentions. Il est indéniable que la promesse de l'Agile ne pouvait pas laisser indifférentes les organisations : Assurer l'alignement stratégique avec les métiers tout en optimisant les couts de développement. En effet, disposer d'un système d'information à l'image des métiers est devenu indispensable dans un contexte extrêmement compétitif. L'absence d'alignement se répercute directement sur la performance de l'entreprise.

Néanmoins, aussi paradoxal que cela puisse paraître, l'aspect le moins représenté dans ce nouveau paradigme, né des entrailles du monde du développement, est le système d'information. Cette sous-représentation de l'ingénierie ne semble pas inquiéter les organisations qui ont enclenché des transformations massives pour espérer capitaliser sur toutes les promesses du paradigme Agile.

Or, il devient aujourd'hui essentiel de se poser la question et d'établir le lien entre ce nouveau système de pensée, d'organisation et de management et le système d'information et l'ingénierie. A travers ce mémoire, nous décrivons les travaux visant à établir ce lien entre le mouvement Agile dans toutes ces dimensions (méthode de projet, organisation et culture d'entreprise) et le système d'information comme étant un ensemble organisé de ressources humaines et matérielles. Nous cherchons ainsi à mettre en lumière les mécanismes d'interactions et la résultante de ces derniers. Nous proposons d'explorer les modèles d'alignement stratégique des systèmes d'information et de les étudier sous le prisme de l'Agile. Cela va nous conduire à proposer un nouveau modèle qui vise à combler l'écart constaté entre le rôle du métier représenté par le client et l'ingénierie dans la gouvernance du SI.

La structure du mémoire se présente comme suit. Le mémoire débute par la présentation de la mouvance Agile, sa naissance son évolution et sa domination sur le paysage actuel. Cela nous amènera à centrer la problématique autour de la relation et l'articulation entre ce nouveau paradigme, les systèmes d'information et l'ingénierie.

Ensuite, dans le deuxième chapitre, nous faisons le lien entre le système d'information et les structures sociales afin d'instancier la théorie de structuration de Giddens. Ce parallèle nous permettra d'établir les liens profonds entre la culture d'entreprise, l'organisation et les systèmes d'information. Nous analysons ce modèle sous le prisme de l'Agile en appliquant les différentes dimensions et en évaluant les impacts de chaque dimension sur les différents organes de l'organisation.

Dans le troisième chapitre, nous étudions les différentes formes d'alignements stratégiques des systèmes d'information en les articulant avec l'Agile. Cela nous amène à identifier un déséquilibre dans la gouvernance du SI entre le métier représenté par le client et l'ingénierie. Nous traitons de cette problématique en proposant un modèle qui puise ses références dans les modèles de gestion de la connaissance. L'objectif étant de maintenir l'alignement métier du SI tout en insufflant les exigences d'ingénierie propres au SI afin d'assurer la pérennité du système d'information sur le temps long.

Le quatrième chapitre est une mise en pratique de ce modèle de connaissance de l'ingénierie instancié aux différentes formes de l'Agile, c'est-à-dire, la méthode de gestion de projet, l'organisation et la culture d'entreprise. L'objectif est de proposer un lien cohérent entre ces dimensions de l'Agile et l'ingénierie dans un modèle qui peut être mis en pratique par les organisations.

Enfin, nous terminons ce manuscrit par une conclusion synthétisant les travaux réalisés et en proposant différentes perspectives.

Il est important de noter que cette étude est le reflet d'un travail empirique au sein d'une entreprise financière du CAC40 qui a conduit à la mise en œuvre, dans un contexte réel, c'est-à-dire au sein de l'entreprise, du modèle proposé au sein de l'organisation.

Chapitre 1 : Naissance et évolution du paradigme Agile

Le terme Agile est aujourd'hui indissociable de la publication du « Manifest Agile » par les grands spécialistes du développement logiciel en 2001. Cependant, l'histoire de l'Agile est, quant à elle, antérieure à la publication du manifeste.

1.1 Emergence de l'Agile

Pour comprendre l'émergence de cette méthode, il faut remonter au début des années 1990, avec l'avènement de la crise du logiciel. En effet, les années 90 marquent le déploiement de l'informatique à grande échelle au sein des entreprises. Ce déploiement massif ne s'est pas traduit par une informatisation rapide et instantanée des processus métiers. Bien au contraire, les entreprises ont été confrontées à « la crise du développement des applications », qui désigne les lourds délais de mise en production des applications estimées en moyenne par les experts à l'époque à 3 ans. Ce délai de 3 ans est une moyenne qui est largement dépassé par certaines organisations. On peut notamment citer l'industrie aérospatiale : le programme de la navette spatiale américaine qui a démarré en 1982 a utilisé les outils de développements des années 1960. Or, les processus métiers sont loin d'être immuables. C'est pourquoi énormément d'applications se retrouvent en déphasage par rapport aux nouveaux besoins métiers après leurs livraisons. Cela a conduit les organisations à abandonner certains projets, même s'ils répondaient parfaitement aux exigences du démarrage des projets, et faire le constat des pertes sèches qui ont fragilisé ces structures.

Cette situation a causé beaucoup de frustration au sein des développeurs ; ce qui les a poussés à réfléchir à d'autres alternatives afin d'amener du changement au cours des développements et de ne pas se retrouver condamnés par les décisions prises au démarrage des projets. Cela s'est traduit par l'émergence d'une communauté de développeurs experts soucieux de trouver une solution à cette problématique.

Cela s'est traduit évidement par la conférence SnowBird en Utah en 2001. A l'issu de cette conférence est né le manifeste Agile et la communauté agile. Cependant, le groupe d'experts, composé notamment de Kent Beck (pionnier de la méthode Extreme Programming), Ward Cunningham, Arie van Bennekum, Alistair Cockburn et douze autres experts, n'étaient pas unanimes à l'idée d'utiliser le terme « Agile ». En effet, l'objectif était de promouvoir des valeurs et une ligne directrice pour les développeurs et d'amener de la valeur de manière itérative et incrémentale aux clients.

1.2 Les valeurs et principes du Manifeste Agile

Le Manifeste Agile repose sur quatre valeurs fondamentales et douze principes [Beck et al 2001], qui appuient cette nouvelle approche autour de l'agilité.

1. **Les individus et les interactions plutôt que les processus et les outils** : La première valeur du Manifeste centre l'attention sur les personnes. Les processus passent ainsi au second plan pour mettre en avant la communication entre les individus. En effet, les processus souffrent d'inertie et de bureaucratie, cela a pour effet direct d'handicaper l'organisation dans sa quête d'évolution destinée à répondre rapidement aux changements. Ainsi, libérer les individus de ce formalisme a pour effet direct de fluidifier les échanges et de créer une dynamique qui suit le changement perpétuel imposé par le contexte des organisations.
2. **Logiciel fonctionnel plutôt que documentation exhaustive :** Historiquement, un temps considérable était consacré à la documentation : de la conception jusqu'à la description du produit final, il y avait une documentation pour tout. Cela tient du fait que le développement logiciel était associé aux méthodes conceptuelles d'analyse, de conception et de gestion de projet (exemple Merise). Ces méthodes reposaient énormément sur la documentation afin de transiter la connaissance entre les différents étages développement. C'est ce point que cible le Manifeste. En effet, en maximisant l'interaction entre les personnes et en œuvrant à avoir un logiciel fonctionnel, on tend à rationaliser les documentations. Néanmoins, il n'était jamais question de supprimer la documentation. Or, cette dérive, qui résulte de la mauvaise interprétation du manifeste, est malheureusement constatée aujourd'hui.
3. **La collaboration avec le client plutôt que la négociation de contrat :** L'interaction entre les équipes de développements et le client étaient placés sous l'égide de la contractualisation. Le contrat était le seul point d'articulation ; ce qui transforme inéluctablement la relation avec le client en relation conflictuelle entre ce qui est mentionné dans le contrat et ce qui a été réalisé par les équipes. C'est bien cette dynamique, qui se rapproche d'un cercle vicieux, que le Manifeste propose de rompre en amenant de la collaboration entre les différents acteurs. Le client est incité à participer activement dans l'élaboration et le développement de la solution finale, ce qui instaure une relation de confiance entre les acteurs et *in fine* participe fortement à la réussite du projet.
4. **Réagir au changement plutôt que de suivre un plan :** Cette valeur est ce qui se rapproche le plus de la définition sémantique de l'agilité. Derrière cette valeur, se cache un nouveau système de pensée, celui des approches

adaptatives. En effet, les organisations étaient obsédées par le besoin de tout analyser, calculer et planifier. Il était question de tout prédire et déterminer. Or, l'avenir est par essence indéterminé. Il devient donc évident que l'approche adaptative, qui propose de réagir au changement au lieu de tout planifier, évite cette dépense couteuse dans la production de plans stratégiques qui finissent par devenir obsolètes et ainsi se consacrer plutôt à naviguer dans la complexité en proposant des solutions ciblées, adaptées aux situations qui se présentent.

Les piliers du Manifeste sont donc 4 valeurs, qui ne définissent aucun cadre précis d'application. Le Manifeste est avant tout un système de valeurs, une manière de réfléchir, d'appréhender la complexité et d'interagir en groupe.

Douze principes viennent compléter le Manifeste afin de proposer des principes directeurs méthodologiques pour étayer les valeurs du Manifeste, mais cette fois-ci sous le prisme du développement logiciel.

1. **Satisfaction des clients grâce à la livraison précoce et continue des logiciels** : Une livraison continue, à l'instar d'une livraison en fin de projet, permet de rectifier et d'adapter la vision du produit final définie par le client, ce qui contribue à la satisfaction du client.
2. **Tenir compte de l'évolution des exigences tout au long du processus de développement :** Principe qui découle de l'approche adaptative, en accompagnant l'évolution des exigences et du besoin fonctionnel tout au long du projet.
3. **Livraison fréquente de logiciels fonctionnels :** L'aspect itératif incrémental est décliné de ce principe de livraison continue de logiciel fonctionnel. L'aspect fonctionnel est déterminant. En effet, cela implique que chaque livraison est utilisable par le client et apporte une valeur notable par rapport à la dernière livraison. Cela maintien l'intérêt du client et son implication dans le projet
4. **Collaboration entre les acteurs commerciaux et les développeurs tout au long du projet :** La collaboration est indéniablement un facteur clé de la réussite d'un projet. Elle est requise entre les différents acteurs, notamment entre la partie commerciale et la partie technique. Cela amène à une meilleure prise de décision et *in fine* à la réussite du projet.
5. **Assister, faire confiance et motiver les personnes concernées :** Le cœur du projet est l'humain. Cet aspect est effectivement le barycentre du Manifeste, et ce principe ne fait qu'appuyer l'idée qu'il essentiel de construire avant tout un collectif d'individu basé sur la confiance et l'entraide avant de construire du logiciel.

6. **Favoriser les interactions en face à face :** La communication directe est la plus efficace pour échanger de la connaissance sans perte et ainsi maximiser l'apport de valeur entre les différents acteurs.
7. **Le logiciel fonctionnel est la principale mesure du progrès :** *In fine*, ce qui compte pour le client, c'est le produit fonctionnel délivré. C'est donc l'ultime mesure qui permet de déterminer l'avancement réel du projet.
8. **Des processus Agile pour soutenir un rythme de développement cohérent :** Un principe encore une fois centré sur l'humain. Il est en effet impossible de développer correctement du logiciel en imposant un rythme insoutenable aux équipes. Il est donc question d'établir un équilibre entre les capacités des individus, l'équipe en général et les exigences du projet. Cela passe notamment par l'inclusion des membres de l'équipe dans les processus d'estimation des efforts.
9. **L'attention portée aux détails techniques et à la conception améliore l'agilité :** Mettre l'accent sur la qualité du logiciel est un facteur important pour la réussite du projet et cela passe inévitablement par une bonne conception pour soutenir les changements sur le long terme.
10. **Simplicité :** La simplicité est la meilleure arme contre la complexité
11. **Les équipes auto-organisées favorisent l'élaboration d'architectures, d'exigences et de conceptions de qualité :** Ce principe est fondateur dans l'émergence de nouvelles formes d'organisation autour d'équipes auto-organisées. Cela induit une délégation des responsabilités vers les équipes et un changement profond dans le management. Ce modèle est en synergie avec l'approche adaptative. En effet, l'équipe en tant qu'entité, est l'organe le plus à même de prendre des décisions pour palier ses propres problématiques.
12. **Réflexions régulières sur la manière de devenir plus efficace :** Être dans une démarche d'auto-amélioration continue conduit à perfectionner les processus, la technique et la communication au sein de l'équipe afin de maximiser les chances de réussir le projet.

Le Manifeste est donc une réflexion globale proposée par un groupe de passionnés autour de valeurs et de principes. Ce document, malgré son aspect synthétique, a eu un impact considérable sur le monde du développement logiciel dans un premier temps puis sur l'ensemble des organisations dans un second temps. En effet, il a largement conduit à l'émergence de multiples méthodes de conduites de projets, ainsi qu'a de nouveaux modèles d'organisation. Il a ainsi permis d'établir une nouvelle culture. Néanmoins, il faut toujours garder à l'esprit que c'est avant tout un système de

valeur qui ne prétend nullement apporter une solution miracle « silver bullet » à l'ensemble des situations et organisations.

1.3 Chronologie de l'évolution de l'Agile

A partir de l'émergence de l'Agile, on peut assimiler son évolution à un mouvement composé de trois phases distinctes. Chaque phase adresse une problématique à son échelle, elle se construit autour d'une communauté qui essaie de proposer de nouvelles approches et solutions à ses problématiques cohérentes avec les préceptes de l'Agile. Or, comme toute nouvelle approche, au lancement, les opinions ont tendance à diverger. En effet, la recherche de solutions implique l'opposition de plusieurs idées mais qui, avec le temps, vont tendre à converger et former un consensus, ce temps marque la fin d'une phase. Le schéma qu'on retrouve ci-dessous reprend les trois phases de l'évolution du mouvement Agile.

Figure 1: Les 3 mouvements Agile [Rudd 2016]

1.3.1 Phase 1 : Agile Teams

Cette première phase avait pour but de répondre à « la crise du développement des applications », évoquée dans la section précédente, en apportant de nouvelles pratiques et un système de valeur au processus de développement logiciel. C'est donc bien la publication du manifeste Agile en 2001, qui a déclenché cette phase. Si aujourd'hui les pratiques comme Scrum ou Extreme Programming sont considérées

comme des standards sur lesquels il y a un consensus, cela n'était pas aussi évident au tout début.

Deux événements permettent d'établir qu'il y a bien un consensus et marquent la fin de la phase : la fondation de l'Agile Coaching Institute (ACI) en 2010 et la mise en place de la PMI ACP Agile certification en 2012.

En effet, la création d'un nouveau rôle « coach Agile » et sa démocratisation au sein des organisations ainsi que l'introduction de l'agile au sein de la PMI confirment l'existence d'un réel besoin des organisations d'être accompagnées pour structurer leurs équipes de développement autour de ce nouveau paradigme.

In fine, cette phase a établi les bases d'une méthode de gestion de projet organisée autour d'une équipe Agile qui respecte les préceptes du Manifeste Agile.

1.3.2 Phase 2 : Agile à l'échelle

Avec l'émergence de ce nouveau modèle d'équipes Agile centré sur des projets, une nouvelle problématique s'est posée : Comment instaurer une coordination entre toutes ces équipes Agile tout en maintenant une cohérence au niveau de l'organisation ?

En effet, plusieurs experts ont constaté la défaillance de la méthode Agile pour maintenir une cohérence à large échelle et se sont proposés d'établir de nouveaux modèles afin de répondre à ces nouveaux enjeux, notamment Larman et Vodde, qui ont publié en 2008 « Practices for Scaling Lean & Agile Development ». Ils ont, grâce à cet ouvrage, formalisé leur expérience en tant que responsables de la transformation Agile de Nokia et proposé une démarche pour la mise en place d'une telle organisation.

Cela a amené la communauté à formaliser des modèles. Ainsi, on a vu apparaitre de nouveaux Framewok comme SaFe, Agile of Agile ou un nouveau modèle comme celui de Spotify.

1.3.2.1 Le modèle Spotify

L'entreprise Spotify, spécialisée dans le streaming de contenu musical/audio qui a démarré en tant que Startup avec un effectif réduit, a connu une croissance exponentielle qui l'a conduit à détenir aujourd'hui plus de 4000 employés dans le monde. Cette croissance fulgurante l'a confrontée à la difficulté d'allier agilité et passage à l'échelle. Pour répondre à cette équation, l'entreprise a innové avec son propre modèle « Spotify ».

Les piliers du modèle :

- Un Framework d'organisation à taille humaine ;
- Une approche holistique de l'agilité sur tous les aspects de la gouvernance d'entreprise ;
- Une forte culture d'entreprise.

La composition du modèle :

- **Les squads** : sont l'unité de base du modèle Spotify. Une squad correspond plus ou moins à une équipe Scrum ;
- **Les tribus** : sont un ensemble de squads qui travaillent collectivement sur un sujet commun. Réunissant jusqu'à 100 collaborateurs, les tribus partagent une même localisation et disposent d'un Tribe Lead ;
- **Les chapitres** : chaque employé fait partie d'un chapitre qui correspond à un domaine de compétence métier ;
- **Les Guildes** : correspondent à des « communautés d'intérêt », elles sont communes à l'ensemble des tribus et réunissent des personnes souhaitant partages des outils, des pratiques et des connaissances.

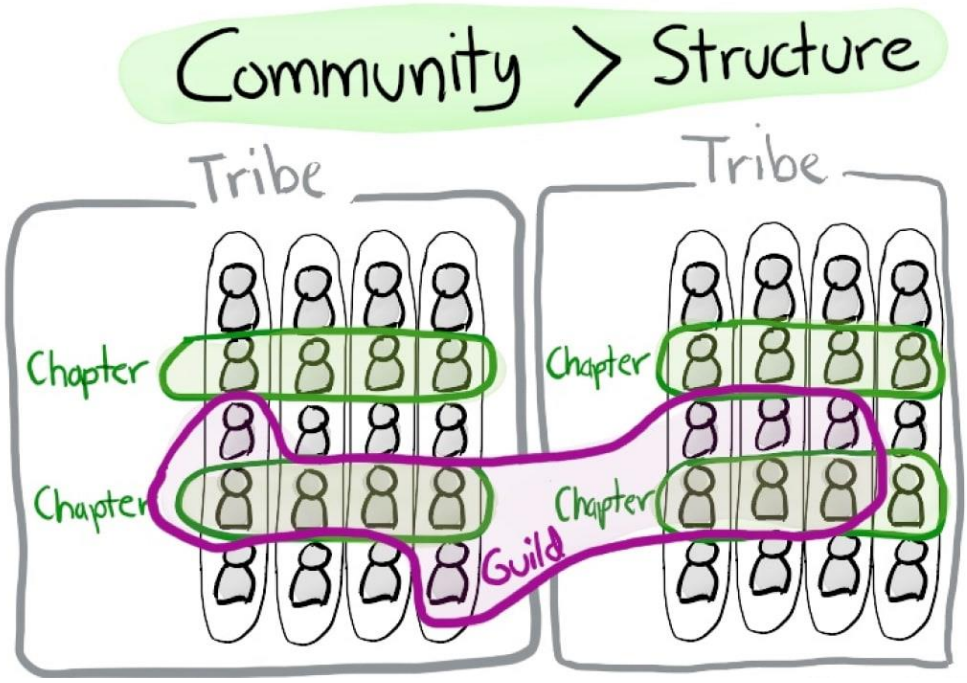

Figure 2 : Modèle d'organisation à l'échelle Spotify [Kniberg 2014]

Le modèle prône l'agilité et l'autonomie des équipes et en fait un moteur d'innovation permettant aux équipes de s'adapter rapidement aux besoins du marché. Le principe de pollinisation croisée, intrinsèque au modèle Spotify, favorise le développement des compétences des individus et représente aussi un vecteur de réussite du modèle.

Néanmoins, l'autonomie des équipes peut rapidement se transformer en chaos sans un alignement clair. C'est d'ailleurs l'un des facteurs que nous abordons dans notre réflexion.

1.3.2.2 La méthode SAFe

SAFe est avant tout un modèle opérationnel ; il n'est pas à mettre en opposition avec le modèle Spotify, qui est plutôt un modèle d'organisation. Il est tout à fait possible de les mettre en œuvre de manière complémentaire.

Le cadre SAFe (Scaled Agile Framework) a été créé en 2011 par Dean Leffingwell qui, fort de ses expériences dans des grandes entreprises, a mis en pratique cette connaissance pour aider les organisations à mettre en pratique l'agilité dans une pensée systémique à l'échelle de l'entreprise.

La force de SAFe est sa capacité à proposer une solution « clé en main » afin de réaliser cette transition à l'échelle sans être en rupture avec l'ancien modèle d'organisation.

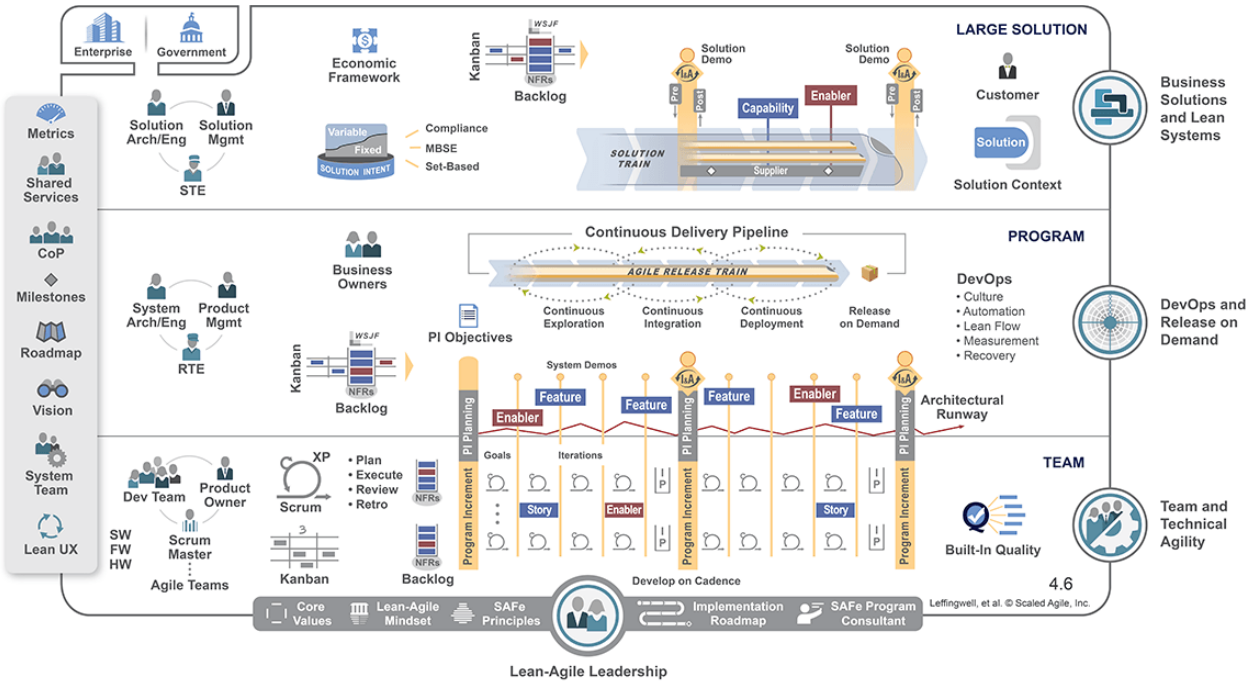

Figure 3 : La méthodologie SAFe

SAFe définit 3 niveaux de gouvernance pour l'Agile à l'échelle :

- **La gestion de portefeuille projet** : opéré par le top management de l'entreprise, il a pour objectif de faire converger la stratégie d'entreprise avec la stratégie d'investissement afin de définir un portefeuille contenant des « Buisness Epics » ;

- **La gestion de programmes** : ce niveau est le terrain de jeu des « products managers » et des « Buisness owners ». C'est à ce niveau qu'on découpe les « Buisness Epics » pour instancier des « Trains » *via* des « Program Epics » et « Features » ;
- **La gestion d'équipe** : est l'équivalent d'une équipe Scrum, c'est à ce niveau que la production s'opère pour délivrer les projets.

L'articulation entre ces niveaux se fait grâce au concept de « Train » ou « Art » (Agile Release Train). Cette notion représente le Backlog au niveau programme pour atteindre les objectifs définis. Il est donc au cœur de la méthode vu qu'il embarque l'ensemble des acteurs SAFe.

La force de SAFe peut être aussi vu comme une faiblesse, car en maintenant une certaine cohérence avec les anciennes approches hiérarchique, il définit un nombre important de couches de gouvernances qui ne contribuent pas à responsabiliser les acteurs.

1.3.2.3 Conclusion

Le point commun entre ces nouveaux Framework ou modèles, c'est qu'ils sont axés sur des enjeux d'organisation et de structure. L'Agile n'est plus seulement une affaire d'équipe restreinte autour d'un projet, mais bien un enjeu d'organisation d'entreprise dans son ensemble.

1.3.3 Phase 3 : Business Agility

Cette troisième phase propose d'instancier le paradigme Agile à l'ensemble de l'entreprise. L'agilité intègre la culture d'entreprise pour devenir au centre de l'organisation. Le management est ainsi amené à évoluer et à prendre en compte cette nouvelle dimension, c'est-à-dire endosser le système de valeur de l'agilité et l'instancier à tous les niveaux de l'organisation.

In fine, l'objectif est d'apporter de la cohérence entre les équipes opérationnelles, l'organisation et le management afin de maximiser l'apport de valeur. Cela se traduit au niveau du management par la favorisation de la compétence et de l'autonomie des individus ainsi que la favorisation de l'auto-organisation des équipes. Il est donc question d'un modèle ou **l'adaptabilité remplace la prédictibilité** à toutes les échelles de l'entreprise.

Ce nouveau paradigme est développé par Jurgen Appelo dans son livre « Management 3.0 » [Appelo 2011], dans lequel il définit le rôle du manager dans les organisations

agiles. Il développe l'idée que le manager à lui tout seul est incapable d'amener toutes les solutions aux problèmes complexes que rencontre l'organisation. Cela l'amène à repenser le système de prise de décision hiérarchique en proposant de le décentraliser afin d'inclure l'ensemble des équipes dans ce processus. Cet effet de levier a comme résultat immédiat celui de démultiplier les capacités de raisonnement et d'amener des angles de réflexions inédits au sein de l'organisation. *In fine*, la pollinisation croisée des idées conduit à augmenter les capacités d'innovation et d'adaptation face à un contexte incertain.

La chercheuse Saras Sarasvathy (2001) a, quant à elle, développé cette notion pragmatique du management, qui considère que l'avenir est extrêmement incertain et complexe à prédire, afin de l'appréhender avec des approches prédictives en proposant le paradigme **Effectual** [Sarasvathy 2008]. En effet, ce paradigme est en rupture avec le système de pensée déterministe, qui repose sur un plan initial et des objectifs à atteindre. A contrario, l'effectuation se concentre sur les moyens et détermine, à partir des moyens, les effets atteignables. Cinq principes permettent de définir et cerner la logique « effectuelle » :

- **Démarrer à partir des moyens** : un entrepreneur est toujours limité en quelques sorte par les moyens dont il dispose. Il devient donc critique de fonder sa réflexion à partir de ces moyens et de considérer tout le champ du possible induit par ces moyens ;
- **Limiter les risques** : considérer ce qu'on est prêt à perdre et limiter les risques afin de naviguer dans un environnement incertain au lieu d'axer son raisonnement sur le retour sur investissement ;
- **Saisir les opportunités** : considérer tout évènement imprévu comme une opportunité plutôt que d'essayer de minimer son incidence ;
- **Développer les partenariats** : aucun entrepreneur ne peut prédire la forme finale d'une opportunité. Ainsi, former des partenariats permet de réduire les incertitudes. Il devient donc important de ne pas considérer la concurrence comme un obstacle et un rival à affronter ;
- **Piloter au lieu de prédire** : « le futur n'est ni trouvé, ni prédit, mais bien inventé ». Partant de ce principe, il n'est pas nécessaire de prévoir la tendance si on arrive à piloter le présent pour construire le futur.

Il devient donc évident, qu'à travers ces travaux, qui présentent des nouveaux paradigmes de pensée à l'échelle d'une organisation, qu'une rupture épistémologique s'est établie avec le système de pensée déterministe. L'agilité s'est hissée au sommet

de l'organisation pour s'étendre sur l'ensemble des organes de l'entreprise. On ne parle plus de méthode ou de Framework, mais d'un système de pensée et de décision.

En analysant cette troisième vague sous le prisme du Manifeste Agile, une cohérence d'ensemble peut être décelée. Le Manifeste n'a jamais été un simple guide pour le développement logiciel, mais un système de valeur qui, instancié à l'ensemble de l'organisation, dévoile la puissance d'un modèle où l'agilité et l'adaptabilité remplacent le déterminisme.

1.3.4 Synthèse de l'évolution du mouvement Agile

Depuis la publication du Manifeste en 2001, il est indéniable que le paradigme Agile a évolué et revêt une tout autre dimension.

Le mouvement est né après avoir établi des concepts fondateurs pour le développement logiciel qui redéfinissent les pratiques, les méthodes et les relations et interactions humaines au sein d'un projet.

Ces concepts ont été repris et déployés dans un premier temps au niveau de l'organisation afin de produire des modèles d'organisation qui puissent opérer à l'échelle et dans un second temps au niveau de la culture d'entreprise et ainsi transformer l'ensemble de l'organisation.

Cette évolution repose sur les valeurs et principes énoncés dans le Manifeste.

1.4 Critiques

1.4.1 Agile et agilité

Dans un article [Thomas 2014] au nom provocateur « Agile est mort », Dave Thomas, un des fondateurs du Manifeste Agile, donne le ton. Il développe une critique autour des dérives du terme « Agile ».

En effet, à l'origine, le manifeste était composé d'une liste de valeurs et de pratiques dont le nom original était « Le manifeste pour un développement agile de logiciels ». Cependant, le terme retenu par la communauté est le « Manifeste Agile ». Ce raccourci a contribué à déformer la substance du message de départ et ouvert la porte à l'exploitation de ce terme « Agile ».

Dave Thomas défend l'agilité en affirmant que l'agilité est une attitude, un état d'esprit qui repose sur un système de valeurs. Or la transformation de cet adjectif en substantif a permis de créer une industrie, l'industrie « Agile ».

Une industrie est donc née au tour du Manifeste. En instrumentalisant ce terme « Agile », plusieurs nouvelles méthodes de gestion de projet ont vu le jour et sont devenues indispensables pour tous les professionnels de l'informatique et de la programmation.

Du jour au lendemain, cette industrie s'est positionnée comme la seule alternative viable pour la gestion de projets informatiques et se retrouve donc dans une situation de monopole. Les organisations se sont senties donc obligées de consommer de l'Agile : formations de coaching, consultances etc.

Par conséquent, une sorte de dictature de l'Agile s'est installée, alimentée par la peur : la peur d'être dépassé, de ne pas suivre les dernières tendances et de rater un tourant décisif dans le monde de l'informatique et de la gestion de projet.

La question ne se posait plus alors au sein des directions informatiques, tous les projets devaient être « Agile » nécessairement. Ainsi, on a forcé certaines équipes à travailler en « Agile » et à suivre une méthode qui devrait solutionner tous les problèmes de l'informatique moderne.

L'exemple qui reflète au mieux cette situation est la méthode « Water Scrum Fall ». Cette méthode est parvenue à se hisser parmi les méthodes les plus adoptées par les entreprises [Theocharis et al 2015]. Elle propose d'encapsuler la méthode Scrum au sein de la méthode « Waterfall ». Elle se décompose donc en trois phases : Water, Build et Fall, respectivement phase de rédaction des exigences, phase de développement et phase de déploiement.

La phase de Build se caractérise par l'utilisation de Scrum. Cette phase intervient après avoir déterminé l'ensemble des exigences et fixé le périmètre au cours de la phase « Water ». Cela implique que la méthode utilise un modèle adaptatif (Scrum) dans un contexte prédictif.

Cette incohérence structurelle dénote l'obstination de certaines organisations à tout transformer en Agile, quitte à utiliser des modèles qui sont intrinsèquement dysfonctionnels.

C'est cette dérive que critique Dave Thomas, car, quelle que soit finalement la méthode Agile déployée au sein d'une organisation, si celle-ci ne s'accompagne pas d'un changement réel du système de pensée en puisant dans les valeurs du Manifeste, cette méthode n'aura pas contribué à rendre l'organisation plus « Agile ».

Il est donc essentiel de recentrer ces méthodes sur les valeurs du Manifeste afin de tirer parti de la puissance de l'agilité.

1.4.2 L'ingénierie Business-driven

Afin de démontrer la supériorité des méthodes agiles, il est devenu de coutume de les comparer à la méthode « Waterfall ».

CHAOS RESOLUTION BY AGILE VS WATERFALL				
SIZE	**METHOD**	**SUCCESSFUL**	**CHALLENGED**	**FAILED**
All Size Projects	Agile	39%	52%	9%
	Waterfall	11%	60%	29%
Large Size Projects	Agile	18%	59%	23%
	Waterfall	3%	55%	42%
Medium Size Projects	Agile	27%	62%	11%
	Waterfall	7%	68%	25%
Small Size Projects	Agile	58%	38%	4%
	Waterfall	44%	45%	11%
The resolution of software projects from FY2011-2015 within the new CHAOS database, segmented by the Agile process and waterfall method. The total number of software projects is over 10,000				

Figure 4: Chaos resolution by Agile VS Waterfall FY2011-2015

En effet, cette dernière a démontré ses limites dans le monde du développement logiciel. Outre le fait d'être condamnée par les décisions prises en amont et l'incapacité de s'adapter au changement, cette méthode met en place une organisation du travail très hiérarchisée et obsolète. Ainsi, les développeurs et les testeurs qui se situent en bout de chaine de la cascade (projet défini par la direction, design réalisé par les architectes et délais fixés par les manager) se retrouvent dans une position d'exécutants. Une position qui ne leur donne aucune latitude pour amener de la valeur et apporter réellement leurs expertises.

Les méthodes Agile proposent d'éliminer cette organisation hiérarchique en mettant en avant des rôles complémentaires. Cependant, le dysfonctionnement n'est pas pour autant corrigé : « Waterfall reproduit le modèle social d'une organisation

dysfonctionnelle avec une hiérarchie définie. Bien souvent, Agile reproduit le modèle social d'une organisation dysfonctionnelle — sans hiérarchie clairement définie » [Michael O. Church]

En effet, même si sur le papier Agile évince le modèle hiérarchique, il persiste tout de même dans une forme implicite. Car, en mettant le client au centre de la méthode, il acquiert *de facto* un pouvoir de décision bien plus important que le reste de l'équipe. De ce fait, le choix des experts se trouve relégué au second plan face à des clients qui disposent finalement d'une totale autorité sur le déroulement des travaux.

Cet effet est accentué par des cérémoniaux qui, dans la pratique, relèvent plus du micro-management pour suivre l'avancement des développeurs que d'un réel espace de partage de l'information et de la connaissance.

In fine, que ce soit la méthode « Waterfall » ou les méthodes Agile, l'ingénierie n'est plus pilotée par les ingénieurs, mais plutôt par la direction (Waterfall) ou par les clients (méthode Agile).

Or, la mission de l'ingénierie dans le monde du SI ne consiste pas seulement à répondre aux besoins ponctuels du métier/clients, mais à se projeter, préparer l'avenir en mettant en place des architectures robustes, éliminer la dette technique et anticiper les technologies à venir. Ces actions, bien que essentielles, sont délaissées dans ces méthodes de projet en faveur du besoin client.

Cette situation pose un problème de gouvernance du SI : comment maintenir un SI fiable alors qu'il n'est pas piloté par l'ingénierie ? Est-ce qu'on n'est pas en train de déconstruire le SI ?

Chapitre 2 : Le SI une projection de la structure sociale d'une organisation

Afin d'établir le lien entre le système d'information et l'organisation, il est important de partir d'une définition de référence du système d'information.

2.1 Qu'est-ce qu'un système d'information ?

La notion de système d'information demeure une notion abstraite qui, en fonction de l'angle dont on l'observe, peut-être définie selon des dimensions différentes : technique, sociotechnique, social ou stratégique.

Reix et Rowe [Reix et Rowe 2002] propose une première définition technique du système d'information *« un ensemble organisé de ressources : matériel, logiciel, personnel, données, procédures [...] permettant d'acquérir, de traiter, de stocker des informations (sous forme de donnée, textes, images, sons, etc.) dans et entre des organisations »*

Avec cette définition, Reix détaille les quatre fonctions principales d'un système d'information :

- **Collecter** : acquérir l'information d'un environnement interne ou externe ;
- **Stocker** : capacité à conserver l'information et la rendre disponible ;
- **Transformer/traiter** : construire la nouvelle information à partir de la chaine de valeur de l'organisation ;
- **Diffuser, restituer et transmettre** l'information dans son environnement interne ou externe.

Il est important de noter que cette définition n'impose pas une forme spécifique au système d'information, c'est-à-dire que le système d'information ne se limite pas à sa forme informatisée, mais qu'il peut tout à fait être mis en œuvre grâce à une pratique manuelle (documents papiers etc ...).

In fine, cette définition technique et généraliste permet d'établir que le système d'information est un ensemble complexe de ressources de natures diverses qui manipule / est manipulé par des fonctions ayant pour résultat de l'information.

Une définition sociotechnique du système d'information est proposée par Reix & Rowe, 2002 : « *ensemble d'acteurs sociaux qui mémorisent et transforment des représentations via des technologies de l'information et des modes opératoires* ». Cette définition

identifie une dimension sociale, une dimension organisationnelle (par la mémorisation et la transformation des représentations) et une dimension matérielle (par l'usage des technologies de l'information).

Selon Hirshein et al. [Hirshein et al 1995], « *Le système d'information est un système social de significations partagées* ». Cette définition permet d'établir le caractère social d'un système d'information et positionne les relations sociales au centre du système.

S. Alter (1992) propose une vision stratégique du système d'information : « *Le système d'information est une combinaison de pratiques de travail, d'informations, d'individus et de technologies de l'informations (aujourd'hui nous devrions ajouter de la communication et de la connaissance) en vue d'atteindre certains objectifs* ».

Également, d'autres approches permettent de définir les systèmes d'information telles que les approches systémique, fonctionnelle, et structurelle.

Concernant l'approche systémique, Le Moigne [Le Moigne 1997], considère le système d'information comme un système vivant et réalisant des objectifs à part entière : c'est un système qui englobe tous les composants dont les interactions sont de type informationnel. Ainsi, Le Moigne propose un « découpage » de l'organisation entre un système opérant (transformation des ressources et des flux), et un système de pilotage (prise de décision, définition des objectifs, etc.). Le système d'information relie les deux systèmes précédents en collectant, traitant, stockant et diffusant l'information.

Bernus & Nemes [Bernus et Nemes 1996], proposent une approche fonctionnelle où ils considèrent qu'un système d'information « *doit garantir que la bonne information est disponible au bon endroit et au bon moment* ». La valeur ajoutée du système d'information dans l'entreprise est de permettre aux acteurs de travailler ensemble d'une façon coordonnée.

Une approche structurelle définit le système d'information en distinguant deux sous-systèmes Morley [Morley et al 2005] : « *le système de traitement de l'information (comprenant les acteurs, les données, et les processus), et le système informatique (comprenant les ressources matérielles et logicielles, les bases de données et les fonctions* ».

Toutes ces définitions permettent de mieux cerner le concept de système d'information et d'appréhender toutes les dimensions qui le compose.

Il devient alors évident que le système d'information ne se limite pas à un simple outil de gestion de l'information, mais s'impose comme un outil structurant de l'organisation.

2.2 Théorie de la structuration

La théorie de la structuration proposée par Giddens [Giddens 1984], s'oppose à l'antagonisme avec les perspectives objectives (vision axée sur les structures qui ne prend pas en compte l'aspect humain) et subjectives (attention exclusive à l'individu ou au groupe sans considération des aspects socio-structurels) de la réalité, mais fait cohabiter ces deux dimensions et mets en lumière une dualité de structure et d'action.

Cette dualité est décrite par Deltour & Vaast (2000) : « La structure et les propriétés institutionnelles des systèmes sociaux sont créés par l'action humaine et contribuent à former les futures actions humaines. Les structures sont produites et reproduites par interactions entre action et structure ».

Clark [Clark et al 1990] synthétise cette théorie de la structuration dans une démonstration axée sur quatre points liés par une relation d'implication [Marina 2004] :

1. Le vrai axe de réflexion de la théorie sociale n'est pas l'action individuelle, mais les pratiques sociales qui forgent les individus et sont au cœur des sociétés ;
2. Les pratiques sociales sont accomplies par des agents humains. Un agent humain est défini par la capacité d'un individu à agir de manière indépendante et être maitre de ses propres choix. Par opposition, la structure représente les facteurs qui délimitent ou limitent l'agent dans ses choix ;
3. Les pratiques sociales se caractérisent par leurs aspects routinier et récurrent qui, *in fine,* représentent les propriétés structurelles des sociétés. Or, celles-ci sont produites par des agents qui sont libres de toute structure ;
4. La structure dépend donc de l'action ou de l'activité. Ainsi, la structure est le moyen et le résultat d'un processus de structuration, c'est-à-dire qu'il existe une relation récursive entre les actions des acteurs et la structure. Giddens représente cette relation avec la notion de « Double hermeneutic » et met en lumière cette interaction entre action et structure en affirmant que « Nous créons la société en même temps qu'elle nous construit ».

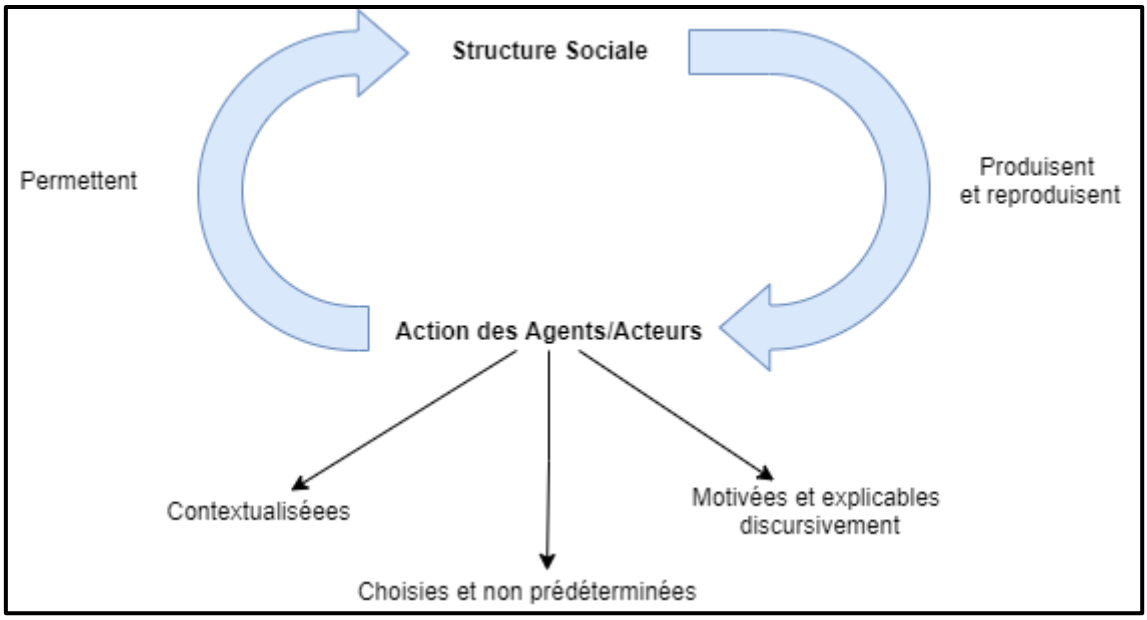

Figure 5 : Articulation entre structure sociale et l'action des acteurs

Cette théorie transposée au monde du système d'information permet de faire le lien entre le système d'information et l'organisation. En effet, si l'on considère qu'une organisation est un cas particulier d'un système social (aspect routinier et récurent), on peut dresser une représentation de l'ensemble des interactions grâce à ce schéma [système d'information et management].

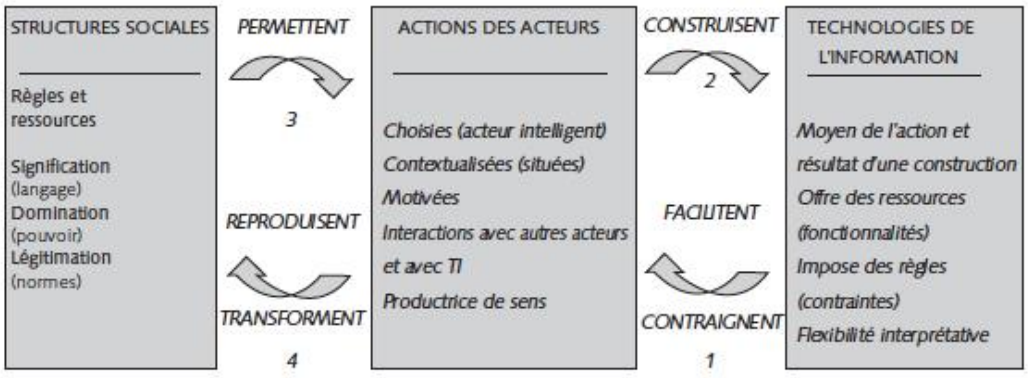

Figure 6 : Schématisation des interactions [Reix et al 2016]

Ce modèle met en évidence quatre types de relations décrites dans l'ouvrage [système d'information et management] :

- Relation 1 : les technologies de l'information facilitent et contraignent l'action des individus ;
- Relation 2 : les actions des acteurs construisent la technologie ;

- Relation 3 : les structures sociales, les propriétés institutionnelles affectent les utilisateurs dans leurs interactions avec les technologies de l'information ;
- Relation 4 : les interactions entre acteurs et entre acteurs et technologies reproduisent ou transforment les structures sociales.

L'interaction entre le système d'information et structure sociale se matérialise grâce à ce modèle. En effet, le système d'information transforme indirectement les structures sociales des organisations à travers les actions des acteurs et se retrouve, par ailleurs, transformé par celles-ci. Ce lien de dépendance entre système d'information et organisation implique que tout changement appliqué à l'organisation ou aux acteurs de l'organisation a un impact direct ou indirect sur le système d'information et vice-versa.

Par ailleurs, en raison du caractère partiellement aléatoire des actions des acteurs, ce modèle révèle le caractère **indéterministe** des technologies de l'informations.

2.3 Agilité et système d'information

Face à cet aspect indéterminisme des systèmes d'information, l'agilité s'est retrouvée au-devant de la scène grâce à son approche adaptative. En effet les approches adaptatives tendent à mieux appréhender la complexité générée par l'incertitude. Il devient donc intéressant de transposer les trois phases de l'agilité développées précédemment sur le modèle de structuration appliqué au système d'information et d'identifier les liens d'interactions.

2.3.1 Agile Teams & théorie de la structuration du SI

Cette phase se limite aux équipes projets, elle définit des lignes directrices pour le développement logiciels et l'interaction entre les acteurs au sein du projet.

Ainsi, par rapport au modèle de structuration, le déploiement de l'Agile Team a directement un impact sur les actions des acteurs et leurs interactions.

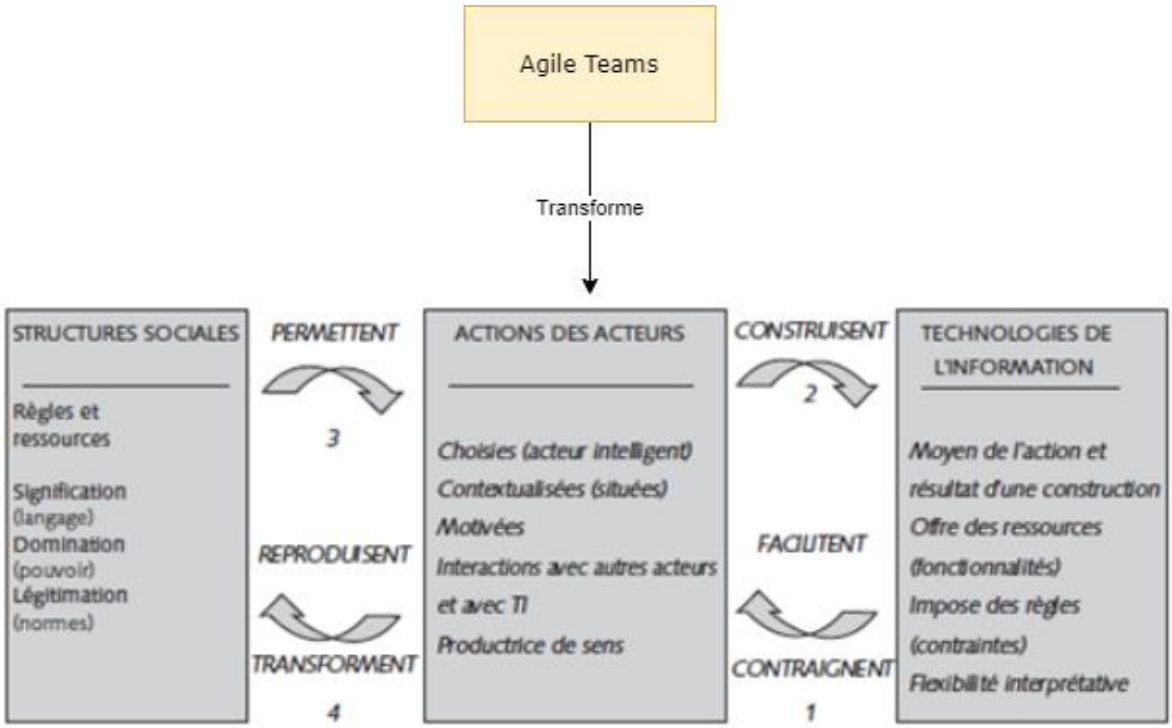

Figure 7 : Agile teams et structuration

Il devient perceptible avec la représentation ci-dessus que l'impact de l'instanciation de l'Agile Teams au sein des équipes projet ne se limite pas aux acteurs, mais concerne également le système d'information et les structures sociales de l'organisation, grâce à cette relation de récursivité. L'agilité au sein des équipes n'est donc pas simplement une méthode de gestion de projet, mais plutôt une pratique qui s'infuse au sein de l'organisation et finit par impacter l'ensemble de ses organes. Il devient donc déterminant d'identifier le résultat de cette transformation, qui est opérée sur les acteurs. Or, comme indiqué dans les chapitres précédents, il n'existe pas une seule forme d'agilité appliquée aux équipes. Cela implique des résultats différents et donc un effet de rétroaction (au sens de la systémique) qui varie en fonction de la méthode appliquée.

Il devient donc complexe de traiter toutes les formes d'agilité avec ce modèle, néanmoins l'analyse de ce modèle permet d'amener une première observation : il est bel et bien possible de déconstruire le système d'information en appliquant une forme d'Agile qui détériore l'action des acteurs. Ces formes d'Agile ont été abordées dans les chapitres précédents, elles s'appuient sur le dogme de la solution parfaite pour résoudre tous les problèmes de projets et sont généralement proposées aux organisations par des cabinets de conseils sans prendre en compte les spécificités des organisations. Cet Agile qualifié de « Dark Agile », par certains détracteurs, devient tout aussi dangereux pour ces pratiquants que pour le système d'information en général.

La deuxième observation porte sur la cohérence d'ensemble du système. L'agilité n'est appliquée que sur un organe du modèle qui se retrouve ainsi seul à porter ce nouveau paradigme. Ce déséquilibre modifie le rôle des acteurs concernés qui ne consiste plus à pratiquer l'agilité mais également à la promouvoir. Ce rôle n'étant pas naturel pour ces acteurs, cela peut amener des fractures au sein de l'organisation entre pratiquants et non pratiquants et entre adhérents et non adhérents, ce qui a pour effet de détériorer la qualité des relations entre les acteurs et, par effet de rétroaction, de déconstruire le système d'information.

2.3.2 Agile à l'échelle & théorie de la structuration du SI

L'Agile à l'échelle est une instanciation de l'organisation d'une équipe agile à l'ensemble de la structure. Cette nouvelle forme d'organisation permet d'articuler l'ensemble des équipes autour de ce nouvel état d'esprit et cette nouvelle dynamique. Cela a pour effet immédiat de démocratiser l'agilité et d'instaurer un référentiel commun. Néanmoins, l'agilité à l'échelle ne permet pas de transformer fondamentalement les structures sociales, elle contribue à les faire évoluer partiellement, mais c'est bien les actions des acteurs qui se retrouvent principalement changés grâce à cette nouvelle forme d'organisation, qui redéfinie les interactions entre les individus au sein de la structure.

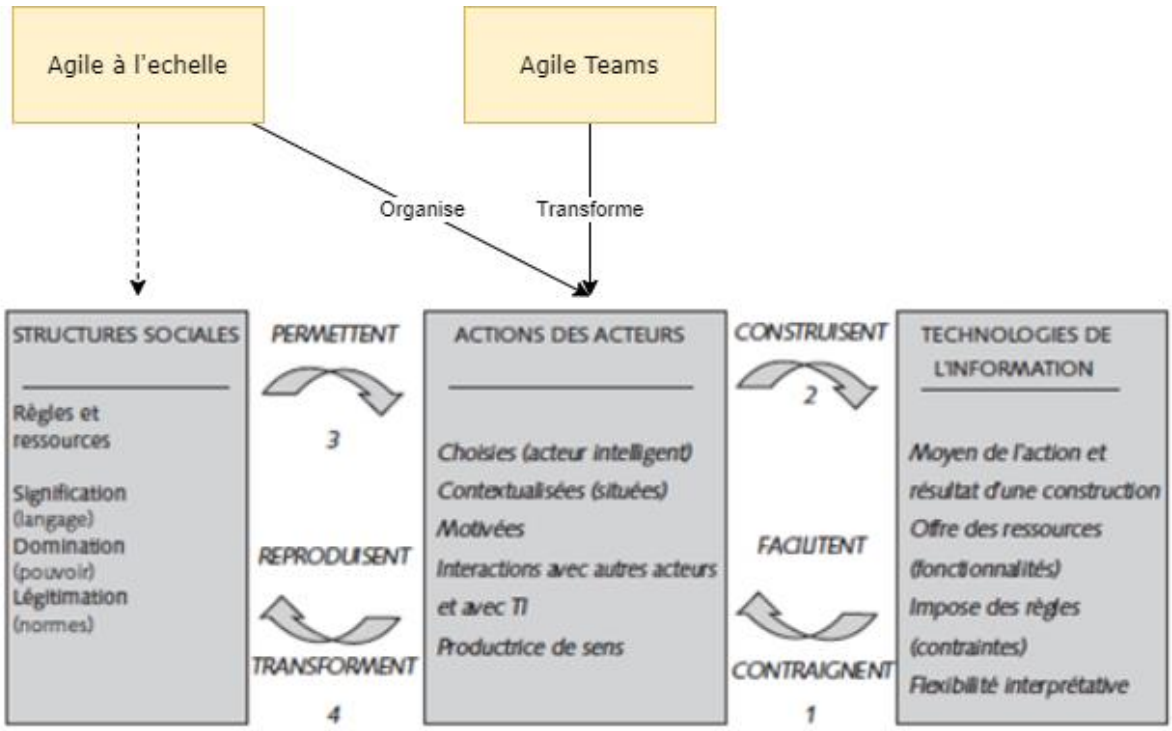

Figure 8 : Agile à l'échelle & théorie de la structuration du SI

Cette deuxième vague apporte une meilleure cohérence au système sans pour autant éliminer la dissonance entre les différents organes du modèle. En effet, les structures sociales, même si elles sont légèrement transformées par l'agilité à l'échelle et par l'effet de rétroaction induit, restent néanmoins portées sur un système de pensée qui ne découle pas du paradigme agile.

2.3.3 Business Agile & théorie de la structuration du SI

En transposant la troisième phase de l'Agile au modèle de structuration, il devient plus facile de comprendre le réel enjeu de cette phase et son impact sur l'organisation. En effet, le Buisness Agile vise à tranformer les structures sociales des entreprises en redéfinissant le langage, en réalisant une transition d'un modèle héarchique vers un modèle plus décentralisé et en mettant en place de nouvelles normes. Ainsi, les structures sociales autant que les acteurs deviennent animés par le même paradigme. Ce qui a pour effet d'autoalimenter (au sens systémique) le système au tour de ce nouveau système de pensée et d'accélérer la mise en pratique de l'agilité.

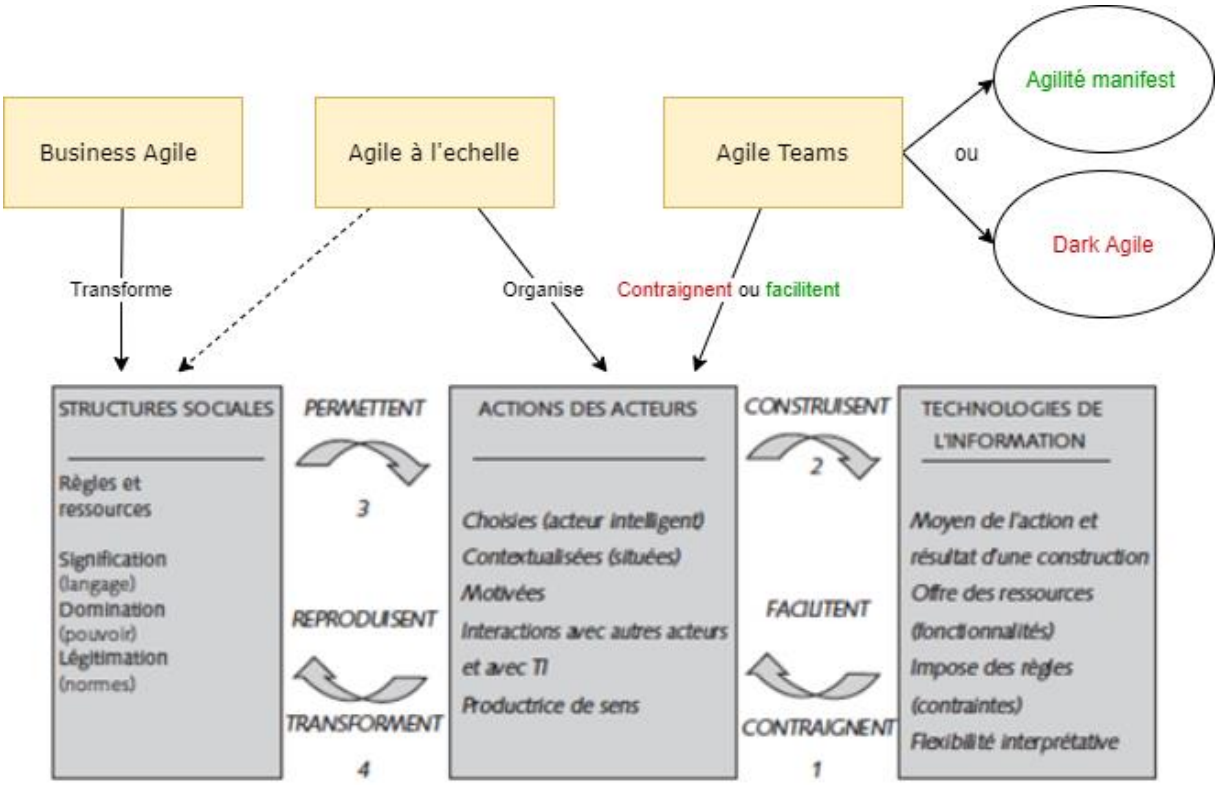

Figure 9 : Culture agile et structuration

C'est la culture d'entreprise dans son ensemble qui est transformée en adéquation avec les actions de ses acteurs.

2.3.4 Conclusion

Les trois phases de l'Agile réunies permettent d'insuffler ce paradigme à deux organes de notre modèle de structuration, c'est-à-dire les structures sociales et les actions des acteurs. Cependant, le troisième organe « technologies de l'information » n'est qu'indirectement affecté par ces phases à travers les liens de récursivité.

Chapitre 3 : Intégration d'une base de connaissance pour le recentrage des processus agiles autour de l'ingénierie du SI

L'amélioration des performances des systèmes d'information en termes de stabilité et de robustesse, de maintenabilité et d'adaptabilité aux évolutions rapides des besoins de l'entreprise, de sécurité et de coût, nécessite une valorisation du rôle de l'ingénieur dans le processus de construction et/ou de réorganisation des systèmes d'information.

La plupart des approches de construction et d'adaptation des systèmes d'information se focalisent essentiellement sur la satisfaction des besoins du client traduits par les processus métiers. Les processus métiers guident l'équipe de développement à tous les niveaux de la structure hiérarchique des systèmes d'information. Au niveau du développement des applications informatiques, les méthodes agiles préconisent d'ajuster les besoins à chaque itération en incluant le client dans ce processus itératif. Par exemple, pour le processus agile Scrum, le client est représenté par le « Product owner », qui traduit les exigences du client à chaque réunion d'évaluation du produit. Ces approches centrées sur les besoins et exigences des clients sont certes très réactives et répondent à la variabilité croissante des besoins, mais font perdre à l'ingénieur son rôle essentiel dans les choix des architectures aussi bien logicielles que matérielles, dans la mise en œuvre des politiques de sécurité, essentielles pour tout système d'information confronté à la malveillance de son environnement d'exploitation, dans la vision globale de la solution permettant l'intégration et la cohérence globale de l'application et dans la traçabilité en ingénierie. L'ingénieur, ayant aussi un rôle capital dans la mise en œuvre de méthodes de résolution de problèmes et d'optimisation, il apparait nécessaire de recentrer le processus de construction et/ou réorganisation des systèmes d'information autour de l'ingénierie.

Pour atteindre cet objectif de recentrage du processus de développement agile autour de l'ingénierie, nous préconisons l'intégration dans les systèmes d'information d'une base de connaissance en ingénierie. L'intégration d'une base de connaissances dans le processus de développement vise le rééquilibrage du processus de construction et/ou réorganisation des systèmes d'information de tout client vers un pilotage mixte qui prend en considération, certes les besoins du client mais aussi les exigences de l'ingénierie pour l'amélioration des performances globales du système d'information.

L'introduction d'un modèle de connaissance dans la construction des systèmes d'information a déjà été envisagée dans plusieurs travaux [Arduin et al 2012], ce qui a donné naissance aux systèmes d'information et de connaissance (SICO) [Rosenthal et

Grunddstein 2009]. Pour ces systèmes, le système d'information est considéré comme un ensemble organisé de ressources incluant non seulement des ressources numériques (logiciels, machines, données), mais aussi des ressources humaines. L'humain n'est pas un simple utilisateur, mais un composant essentiel du système porteur de connaissance. Notre approche de rééquilibrage des processus de développement s'intègre dans cette démarche de valorisation des ressources humaines et plus précisément de l'ingénierie. La construction de la base de connaissance s'appuie essentiellement sur les modèles, les expériences et expertises générés par l'architecture du logiciel. Le rééquilibrage passe par la valorisation du rôle de l'architecte du logiciel, et le développement doit être centré autour de ce rôle. Notre approche se situe dans le cadre du développement des systèmes d'information et de connaissance. Nous préconisons l'ajout d'une base de connaissance au modèle de structuration des systèmes d'information pour capitaliser les connaissances de l'architecte du logiciel. En exploitant cette base de connaissance, nous montrons comment s'effectuent le rééquilibrage des processus agiles et le recentrage autour de l'ingénierie.

Dans ce chapitre, nous commençons par présenter les notions de connaissance, de gestion des connaissances et les processus de formalisation et d'intégration des systèmes de gestion des connaissances dans les systèmes d'information. Nous détaillons par la suite notre approche d'intégration du modèle de connaissances dans la structuration hiérarchique des systèmes d'information. Nous commençons par identifier les connaissances d'ingénierie nécessaires à la construction et réorganisation des systèmes d'information, avant d'en donner les formalisations possibles. La construction de la base de connaissance s'appuie sur le rôle de l'architecte du logiciel. La formulation des connaissances nécessite l'introduction des styles architecturaux, des langages de description des architectures et des ontologies.

3.1 Proposition du recentrage de l'Agilité autour de l'ingénierie

Toutes les approches de construction et/ou de réorganisation des systèmes d'information sont pilotées par les besoins métiers et visent essentiellement à satisfaire les besoins du client. Le client s'accapare toutes les intentions des développeurs, c'est pourquoi il est devenu impératif de l'inclure dans tous les niveaux de construction/réorganisation des systèmes d'information. Il est même nécessaire de faire participer le client dans les différentes phases des processus agile et de tenir compte de l'évolution de ses besoins. Certes, cela permet de répondre aux objectifs attendus en termes de fonctionnalité, mais n'offre aucune garantie sur la satisfaction des exigences de qualité de développement. Cette exigence de qualité concerne les

aspects de robustesse, sécurité, évolutivité, modularité, maintenabilité et vision globale de la conception pour garantir la cohérence globale du système. Ces aspects sont essentiels pour l'amélioration des performances des systèmes d'information. La performance des systèmes d'information se mesure, certes, à sa capacité à s'adapter et à répondre aux évolutions des besoins (alignement des systèmes d'information), mais elle dépend aussi de la qualité de son développement.

Il faut repenser la place de l'ingénieur dans le processus de construction/ réorganisation des systèmes d'information. Celui-ci ne doit pas être un simple développeur qui doit mettre en œuvre les fonctionnalités des systèmes d'information dictées par le client, mais doit être au cœur du processus de construction /réorganisation de ces systèmes ; car il doit effectuer des choix stratégiques qui participent à l'amélioration de la qualité. Ces choix doivent être pris aux différents niveaux de mise en œuvre des systèmes d'information.

3.1.1 Pilotage des approches d'ingénierie des SI par les besoins clients

Dans notre étude nous distinguons deux grandes familles dans les approches d'ingénierie des SI : (i) les processus de développement et (ii) les approches de réorganisation et d'adaptation des SI (les processus d'alignement stratégique des SI). Nous essayons, à travers cette étude, de montrer la focalisation des approches d'ingénierie sur la satisfaction des exigences du client.

3.1.1.1 Les approches d'alignement stratégique

L'alignement stratégique (ou la réorganisation) vise à faire évoluer le système informatique pour l'adapter à l'évolution des stratégies métiers de l'entreprise. Pour ce faire, une conception nouvelle du système informatique doit être adoptée. L'alignement de la stratégie de l'entreprise et de la stratégie du système d'information repose sur deux conditions préalables [Cigref 2003] :

- Compréhension et intégration de la stratégie de l'entreprise par la fonction système d'information dans son ensemble ;
- Prise en compte des contraintes et des opportunités de l'informatique dans la stratégie de l'entreprise.

La plupart des approches d'alignement sont centrées autour des métiers. Leurs préoccupations majeures sont d'aligner les objectifs du système d'information sur ceux du métier. Les aspects ingénieries sont relégués au second plan. Pour renforcer nos propos, nous citons quelques exemples significatifs d'approches d'alignement :

- **BALES** (Architecture of Integrated Information System) [Papazoglou et Heuvel 2000] : C'est une approche basée sur la modélisation des applications métiers à l'aide d'un cadre d'intégration d'entreprise. Ainsi, elle relie les modèles métiers/modèles d'entreprise aux modèles d'application et aux SI.
- **ARIS** (Architecture of Integrated Information System) [Scheer et Nüttgens 2000] : Elle propose une architecture générale des processus métiers qui se base sur deux approches pour l'adaptation des SI : (i) une approche qui permet le développement d'un système organisationnel fonctionnel, et (ii) une approche qui vise à développer un système technique. ARIS couvre les phases allant de la re-conception des processus métier à la conception et au déploiement du système d'information technique, en se basant sur une conception du logiciel orientée processus.
- **SEAM** (Systemic Entreprise Methodology) [Wegmann et al 2002] : C'est une approche qui vise à aligner les processus métiers et les SI supports. Elle permet de représenter les ressources disponibles au sein de l'entreprise et de son environnement, tout en les reliant aux processus.
- **Approche de Wieringa** [Wieringa et al 2003] : c'est une approche qui propose un cadre d'analyse pour la conception des processus métiers et de l'architecture des SI. Grâce à ce cadre, la stratégie d'entreprise et ses applications logicielles support forment un système réactif qui répond à l'évolution de l'environnement.
- **B-SCP** (Business Strategy, Context, and Process) [Bleistein, et all, 2006] : C'est une approche d'ingénierie des besoins. Elle vise à permettre la validation des besoins en phase amont de la conception des systèmes d'information afin d'assurer l'alignement avec la stratégie, le contexte, et les processus de l'organisation.
- **Urbanisation des SI** (approche de longépé) Longépé C. [Longépé 2001] [Longépé 09] définit l'urbanisme des SI comme « *un moyen pour sauvegarder la cohérence et améliorer l'efficacité du système d'information c'est-à-dire la qualité de sa contribution à l'atteinte des objectifs de l'entreprise* ». Cette approche vise à rendre le SI le plus réactif possible. Le modèle d'urbanisation des SI représente les systèmes d'information de manière hiérarchique en quatre couches interagissant avec le niveau stratégique : la couche métier, la couche fonctionnelle, la couche applicative et la couche technique (voir figure 1).

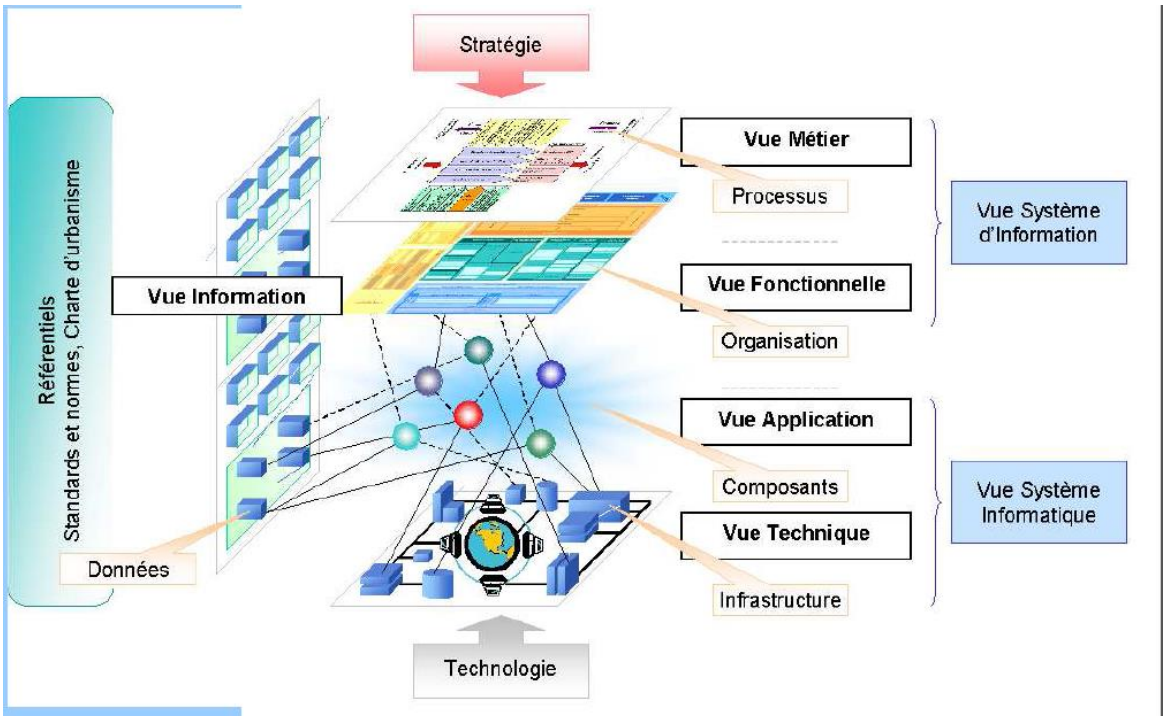

Figure 10 : Modèle hiérarchique de l'urbanisation des systèmes d'information

La démarche d'urbanisation recentre le pilotage de l'évolution du système d'information sur la stratégie et les besoins métiers de l'entreprise [Ben Amira 2015]. A partir d'objectifs stratégiques clairement identifiés au niveau stratégique, les processus métier à mettre en œuvre sont alors identifiés. Puis chaque processus métier est traduit en termes de fonctions et informations, ce qui forme le niveau fonctionnel. Ces deux niveaux constituent la vue système d'information. Les applications et l'architecture technique permettent l'implémentation des fonctionnalités du système d'information et des données s'y afférant, ce qui constitue les niveaux applicatif et technique qui forment la vue système informatique. Cela constitue une démarche de conception descendante « top-down ». Il apparait ainsi que le rôle essentiel de l'urbanisation est de passer d'un système d'information existant à un système d'information cible par des étapes successives de description ou de construction d'architectures.

A partir de cette analyse des approches d'alignement, il apparait que :

- Toutes les approches de modélisations des systèmes d'information proposent de les représenter par des niveaux hiérarchiques qui partent des niveaux stratégiques jusqu'aux niveaux techniques.
- Ces approches sont exclusivement pilotées par les processus métiers, c'est-à-dire la satisfaction des besoins du client.

Nous préconisons de reconsidérer le rôle de l'ingénieur et d'accorder de l'importance à ses décisions tout au long du processus de décomposition hiérarchique à travers les différents niveaux d'un système d'information. Ces décisions concernent les choix architecturaux pour les solutions logicielles et matérielles, la définition et la mise en œuvre de politiques de sécurité et l'élaboration de méthodes d'optimisation pour l'amélioration des performances des systèmes d'information. Ces décisions doivent être formalisées et mises à la disposition des équipes de développement. Ainsi, nous parvenons à un équilibrage du processus d'alignement des SI en déplaçant le pilotage exclusif par les processus métier (client) vers l'ingénierie.

3.1.1.2 Les processus de développement

Nous nous sommes intéressé, dans cette étude, aux processus de développement basés sur les modèles UML. Deux grandes familles se distinguent : les Processus Unifié (PU), ou Unified Process (UP) et les processus agiles.

a) Les processus unifiés

Les processus unifiés est une famille de méthodes de développement de logiciels orientés objets. Elle se caractérise par une démarche itérative et incrémentale, pilotée par les cas d'utilisation, et centrée sur l'architecture et les modèles UML (voir figure 2).

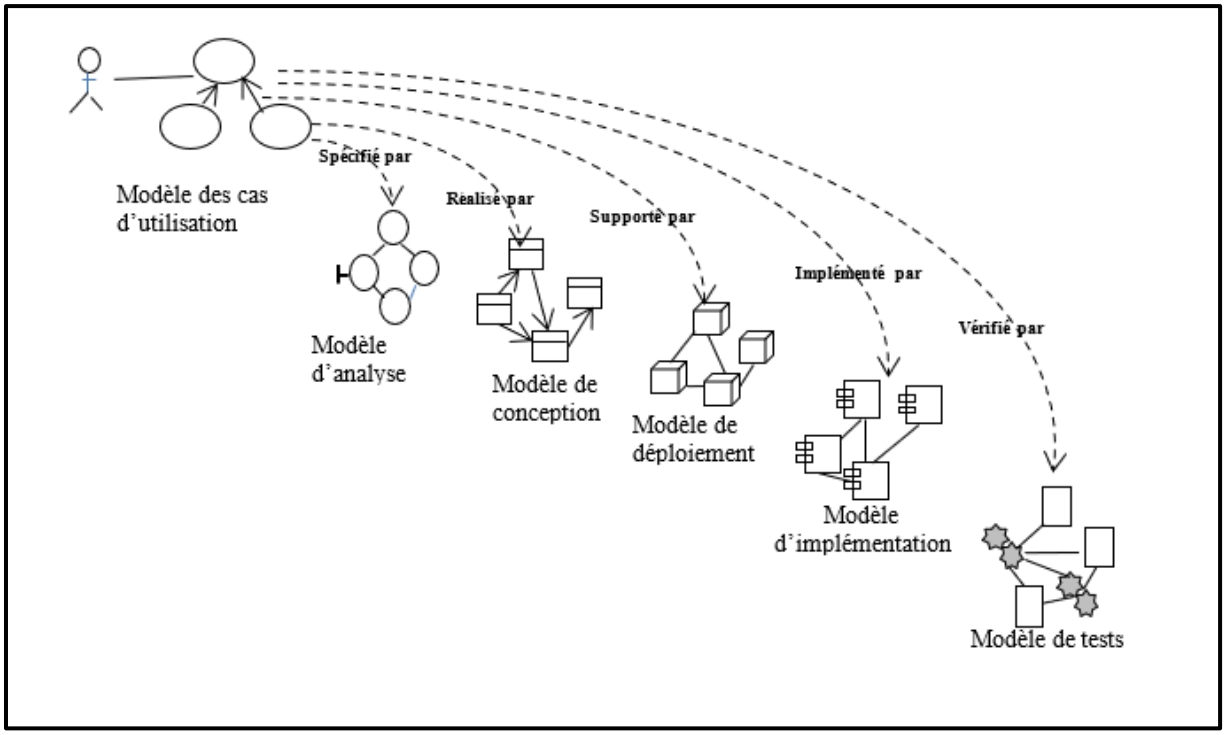

Figure 11 : Pilotage des processus unifiés par les cas d'utilisation

La démarche préconisée est la suivante :

- Créer une ébauche grossière de d'architecture ;
- Travailler sur les cas d'utilisation représentant les fonctions essentielles du système ;
- Adapter l'architecture pour qu'elle prenne en compte ces cas d'utilisation ;
- Sélectionner d'autres cas d'utilisation et refaire de même.

Les processus unifiés préconisent un découpage d'un projet en « mini-projets », ce qui constitue les itérations. Chaque itération comprend un certain nombre de cas d'utilisation et doit traiter en priorité les risques majeurs. Une itération reprend les livrables dans l'état où les a laissés l'itération précédente et les enrichit progressivement d'une manière incrémentale.

Pour chaque itération, les PU proposent l'enchainement des activités suivantes :

- Exigences : recherche des acteurs et des cas d'utilisation, priorisation, description des cas d'utilisation, prototypage de l'interface utilisateur, et structuration du modèle des cas d'utilisation ;
- Analyse : analyse de l'architecture, analyse d'un cas d'utilisation, analyse d'une classe, et analyse d'un package ;
- Conception : conception de l'architecture, conception d'un cas d'utilisation, conception d'une classe, et conception d'un sous-système ;
- Mise en œuvre (implémentation) : implémentation de l'architecture, intégration du système, implémentation d'un sous-système, implémentation d'une classe, et exécution de tests unitaires ;
- Test : planification des tests, conception des tests, mise en œuvre des tests, exécution des tests d'intégration, exécution de tests systèmes, et évaluation des tests.

Le processus unifié définit également des rôles pour les acteurs intervenant pour ces activités (par exemple architecte, ingénieur composant, designer d'interface utilisateur, etc.). Il est à noter qu'il s'agit bien de rôles et non de personnes ; un même membre d'équipe peut donc cumuler plusieurs rôles.

b) Les processus agiles

En ingénierie logicielle, les pratiques agiles mettent en avant la collaboration entre des équipes auto-organisées et pluridisciplinaires et leurs clients. Elles s'appuient sur l'utilisation d'un cadre méthodologique léger mais suffisamment centré sur l'humain et la communication. Elles préconisent une planification adaptative, un développement

évolutif, une livraison précoce et une amélioration continue, et elles encouragent des réponses flexibles au changement.

La méthodologie Agile nécessite une forte implication du client et implique des évaluations fréquentes de l'état d'avancement avec l'équipe de projet et le client.

Les avantages de l'adoption de la méthodologie agile sont les suivants :

- Contact continue avec le client : avec les méthodes agiles, il y a un contact permanent tout au long du processus et des livraisons itératives, ce qui permet de s'assurer de la satisfaction des attentes du client.
- Capacité d'adaptation : avec les méthodologies agiles les changements peuvent être intégrés sans grand effort à la prochaine itération.
- Livraison plus rapide : les méthodes agiles intègrent une approche de développement continue qui assure la livraison à des intervalles courts (2 à 4 semaines) de produits exploitables.
- Risques moindres : le fait qu'avec les méthodes agiles les différentes versions du produit soient validées par le client, cela diminue le risque d'échec du projet.
- Innovation continue : les méthodes agiles sont basées sur la collaboration et l'amélioration continue, ce qui peut conduire à l'innovation et au développement de nouveaux produits et fonctionnalités.

A partir de cette étude sur les processus de développement, nous constatons que ces deux approches se focalisent essentiellement sur la satisfaction des exigences du client. Les processus unifiés sont pilotés par les cas d'utilisation qui constituent les exigences fonctionnelles imposées par le client. Tous les autres modèles sont générés à partir de l'étude de la réalisation des cas d'utilisation. Les modèles d'architectures sont introduits pour l'implémentation des cas d'utilisation. Les méthodes agiles placent le client au centre du développement. A chaque itération, le client est consulté pour valider la version actuelle du produit et prendre en compte d'éventuelles évolutions du produit ou de ses fonctionnalités. La réussite du projet dépend de la collaboration et de la communication fréquente des différents intervenants du projet, en particulier du client ou de la personne qui le représente dans l'équipe projet. Là aussi les aspects ingénieries sont légués au second plan des préoccupations.

3.1.2 Intégration d'une base de connaissance pour le recentrage de l'agilité autour de l'ingénierie

Pour la mise en œuvre d'un pilotage mixte de la construction et ou réorganisation des systèmes d'information par les processus métiers et l'ingénierie, nous préconisons l'ajout au modèle de structuration des systèmes d'information d'une base de

connaissance qui capitalise toutes les décisions d'ingénieries. Ces décisions peuvent être prises à différents niveaux du modèle de structuration et exploitées par les équipes de développement dans le cadre de processus agile. En partant d'un modèle de structuration des SI en cinq niveaux hiérarchiques, adopté par l'approche d'urbanisation, la figure 3 fournit notre proposition d'intégration de la base de connaissance. Dans la suite de ce chapitre, nous allons formaliser cette base de connaissance en fournissant les types d'informations et de connaissances qu'elle doit contenir et la manière de les décrire.

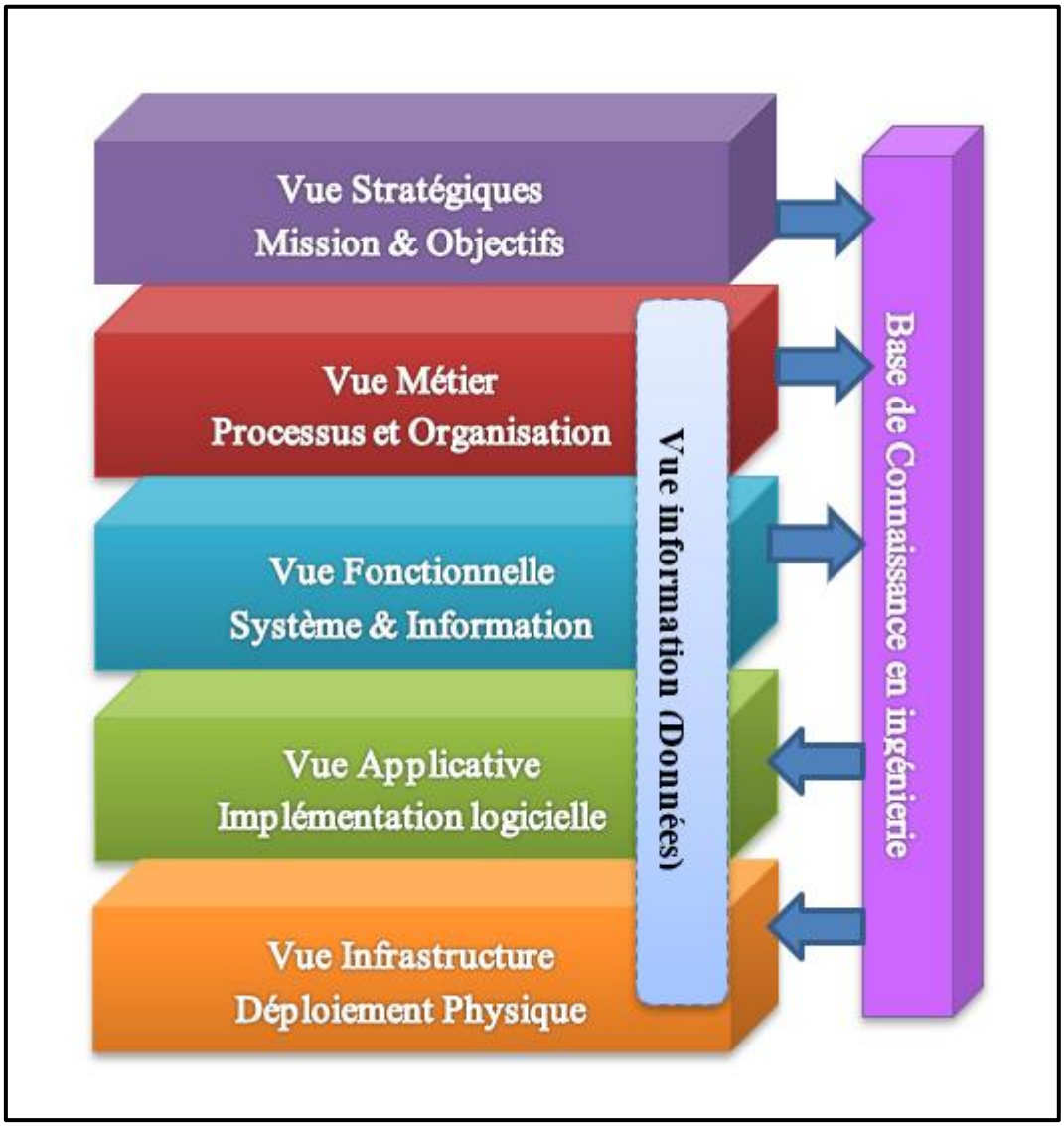

Figure 12 : Intégration d'une base de connaissance dans la construction d'un Système d'Information

3.2 Gestion des connaissances dans les systèmes d'information

La connaissance est devenue le patrimoine le plus important pour les entreprises. Selon Drecker & Fontaine [Drucker et Fontaine 1993] « *la connaissance constitue une*

ressource critique et un facteur de compétitivité déterminent pour un pays, une industrie, une entreprise ». Ainsi, le développement des connaissances et des compétences est devenu un facteur déterminant et incontournable pour l'amélioration de la productivité et de la compétitivité des entreprises.

Selon l'étude Foresight [Foresight 2020] [Prax 2007], réalisée en 2005 par le magazine The Economist auprès de 1650 gestionnaires à l'échelle mondiale, la gestion des connaissances (knowledge management) est reconnue comme le défi majeur pour les 15 prochaines années en regard des tendances lourdes de l'économie mondiale : mondialisation, atomisation des sources d'information et d'expertise et personnalisation des produits et des services.

3.2.1 Définition des connaissances

Baizet définit dans [Baizet 2004] les connaissances comme des informations raffinées, synthétisées, systématisées, ou comme des informations associées à un contexte d'utilisation. Cette définition montre qu'il y a une différence entre les mots : données, informations et connaissances. Tsuchiya [Tsuchiya 1993] différencie ainsi ces termes : « *Bien que les termes « donnée », « information » et « connaissance » soient souvent utilisés de manière interchangeable, il existe une distinction claire entre eux. Lorsque la donnée est sensée à travers un cadre interprétatif, elle devient information, et lorsque l'information est liée au sens à travers un cadre interprétatif, elle devient connaissance* ».

La connaissance peut être définie comme le résultat d'une interaction entre des informations et un système d'interprétation dans un domaine d'application donné. Elle peut être caractérisée par le triplet « information/utilisateur/domaine » [Bouzid 2017].

La plupart des auteurs classent les connaissances en deux types : les connaissances explicites et les connaissances tacites (savoir-faire) [Grundstein 2004] [Grundstein et al 2003].

Les connaissances explicites font référence aux savoirs transmissibles à travers un langage « formel et systématique » [Lalouette 2014]. Elles peuvent être une donnée technique ou universitaire ou une information qui est décrite dans un langage formel, comme les manuels, les expressions mathématiques, les brevets, etc. [Smith 2001].

Les connaissances tacites font référence aux savoirs-faire qui sont difficiles à formaliser et à communiquer et ne peuvent être transférés que par la volonté des gens à partager leurs expériences. Elles sont généralement acquises sur une longue période d'apprentissage et d'expérience [Grundstein et al 2003].

3.2.2 La gestion des connaissances

La gestion des connaissances est un domaine interdisciplinaire qui fait appel aux sciences cognitives, aux systèmes experts, au génie cognitif, aux réseaux sémantiques, aux bases de données, aux sciences de la documentation et aux sciences de la gestion. Jean-Yves Prax propose trois définitions de la gestion des connaissances [Prax 2007] :

- Une définition fonctionnelle : « *Manager le cycle de vie de la connaissance depuis l'émergence d'une idée : formalisation, validation, diffusion, réutilisation, valorisation* » ;
- Une définition opérationnelle : « *Combiner les savoirs et les savoir-faire dans les processus, produits, organisations, pour créer de la valeur* » ;
- Une définition économique : « *Valoriser le capital intellectuel de la firme* ».

Swan, Scarbrough, & Preston, 1999 dans [Swan et al 1999] définissent la gestion des connaissances comme un ensemble de démarches, méthodes et outils pour la création, collecte, formalisation, capitalisation, partage et utilisation de connaissances (figure 4).

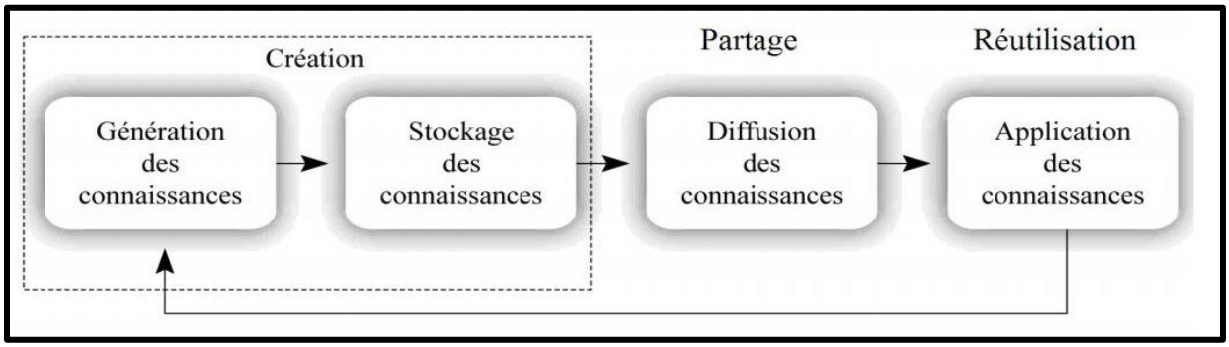

Figure 13 : Cycle de vie de la connaissance

La gestion des connaissances exige le passage des connaissances personnelles à des connaissances collectives qui peuvent être largement partagées dans l'organisation.

- **La création des connaissances** : Le capital lié aux connaissances de l'organisation est constitué de connaissances explicites et tacites, sur le plan des individus qui composent l'organisation et sur le plan collectif de l'organisation elle-même. Le schéma de la figure 5 présente ces « états » de la connaissance et leurs transitions selon une théorie développée par Nonaka [Nonaka et Takeuchi 1995], l'un des fondateurs de la gestion des connaissances. Dans ses travaux, Nonaka montre que le développement des savoirs dans l'organisation est fondé sur une dynamique de transfert entre les différents états de la connaissance. La connaissance est prise comme unité de base pour expliquer le comportement de l'organisation, laquelle est vue non seulement

comme un système de « traitement » de connaissances, mais aussi de création de connaissances. Le développement des savoirs dans l'organisation implique des transitions entre les quatre états de la connaissance [Bouzid 2017] :

- La transition *du tacite vers le tacite* est appelée *socialisation*. Elle représente l'interaction des individus au sein d'un groupe. C'est un processus d'ajustement mutuel à partir des connaissances tacites de chacun ;
- La transition du tacite vers l'explicite est appelée conscientisation ou articulation. Elle désigne l'explicitation, par le discours ou l'écrit, des pratiques et des croyances. Sa difficulté réside dans l'adoption d'un langage et de concepts partagés. C'est le but notamment des technologies sémantiques de fournir un tel langage appelé ontologie ;
- La transition *de l'explicite vers le tacite* est appelée *intériorisation ou assimilation*, sur le plan individuel. L'intériorisation représente l'enracinement de la connaissance explicite dans des séquences qui peuvent atteindre le stade du réflexe, de l'automatisme, et qui doivent normalement s'accompagner de gains d'efficience.

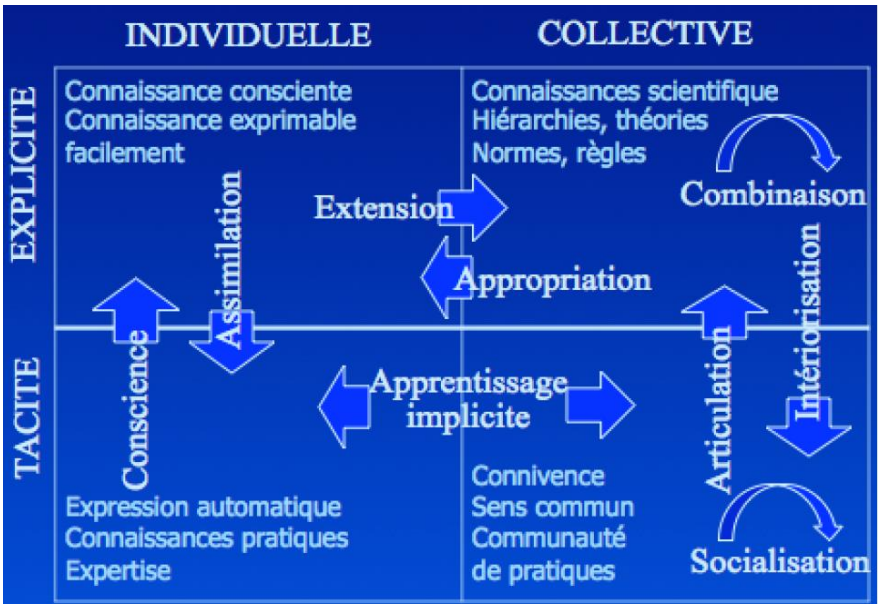

Figure 14 : Matrice des états de la connaissance et de leurs transitions [Bouzid 2017]

- **La capitalisation des connaissances** : pour Grundstein [Grundstein 2004] « *capitaliser les connaissances, c'est considérer certaines connaissances utilisées et produites par l'entreprise comme un ensemble de richesses et en tirer des intérêts contribuant à augmenter la valeur de ce capital* ». Il caractérise la problématique de la capitalisation des connaissances par cinq facettes et leurs interactions, représentées par la Figure 6.

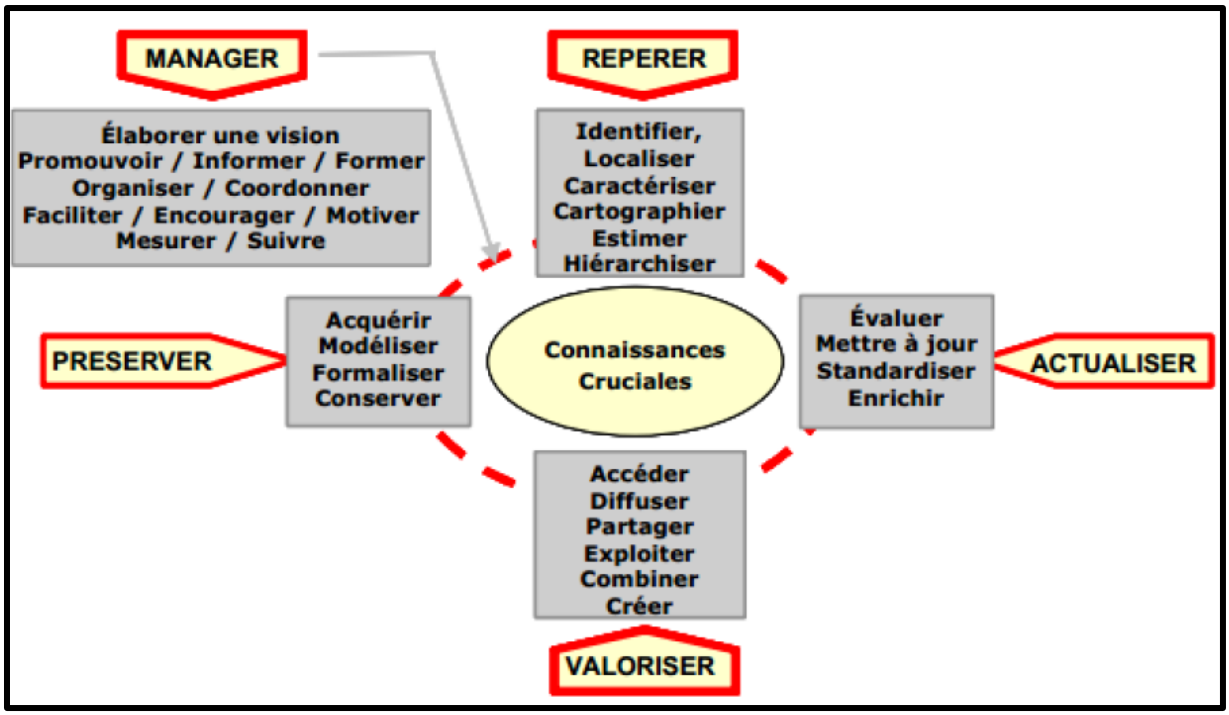

Figure 15 : Les cinq facettes de la problématique de capitalisation des connaissances en entreprise [Grundstein 2004]

- **Le partage des connaissances** est défini comme l'échange de connaissances entre les individus, les équipes, les unités organisationnelles et les organisations. Cet échange peut se fixer ou non un but, mais il n'a pas généralement un objectif *a priori* clair [Paulin et Suneson 2015]. Le but principal du partage ou du transfert des connaissances est de fournir des connaissances explicites des acteurs (comme les formules, les processus et routines), ainsi que des connaissances tacites (partage d'expériences et savoir-faire) aux autres acteurs pour les aider à accomplir les objectifs, collaborer pour résoudre les problèmes, développer de nouvelles idées, ou mettre en œuvre des politiques ou des procédures [Wang et al 2014].

3.2.3 Méthodes de formalisation des connaissances

Plusieurs méthodes sont aujourd'hui disponibles pour soutenir la formalisation des savoirs détenus par des individus dans des « bases de connaissances », des « recueils de connaissances » ou des bibliothèques de modèles ou de cas, sous une forme explicite, donc accessible aux autres membres d'une organisation.

J. Y. Prax présente cette démarche de formalisation du savoir tacite comme une pyramide d'abstraction-concrétisation (figure 7) [Prax 2007]. Le producteur explicite et abstrait ses représentations personnelles en une représentation partagée, puis la formalise sous forme d'un modèle de connaissances. Ce modèle est ensuite intégré

dans une bibliothèque de modèles et diffusé à l'aide d'un outil de gestion des modèles de la bibliothèque.

Inversement, un utilisateur, confronté à un cas réel, se sert de l'outil de diffusion pour repérer des modèles utiles à l'aide de l'outil de gestion des modèles. Il obtient une méthode ou une théorie générique, extrapolable à une classe de problèmes. Il en extrait des règles et des procédures qu'il applique au cas réel qui l'intéresse.

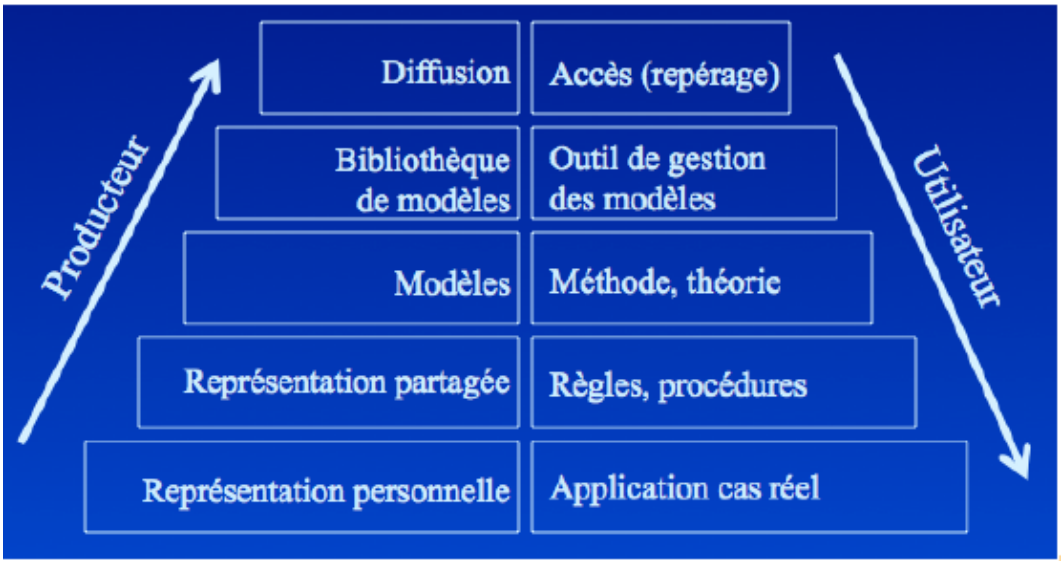

Figure 16 : Méthodologie de formalisation des connaissances [Prax 2007].

Prax classifie les méthodes de formalisation des connaissances en six catégories [Prax 2007] :

1. *Méthodes visant à partager une connaissance tacite ou implicite contenue dans une situation vécue ou dans l'exécution d'une action. Ces méthodes utilisent divers moyens :* récits de situations vécues, socialisation dans des communautés de pratique, jeux d'animation de groupe tels que le remue-méninge (brainstorming), la carte cognitive (mindmapping), les jeux de rôles et d'imitation, l'usage de métaphores ;
2. *Méthodes visant à créer une méta-mémoire ou une mémoire de la mémoire de l'organisation.* Ces méthodes ne capitalisent pas la mémoire elle-même, mais son repérage « cartographique » par des métadonnées ou en utilisant une ontologie d'un domaine ;
3. *Typologie et structure des mémoires d'entreprise* : Il s'agit ici de catégoriser et d'établir des associations avec les connaissances de l'entreprise dans une mémoire à base de cas, ou dans une mémoire de projet.

4. *Méthodes de capitalisation des retours d'expérience :* Elles permettent une capitalisation ponctuelle des connaissances d'un expert et l'échange des bonnes pratiques.
5. *Méthodes de modélisation de l'entreprise partant d'une analyse systémique ou d'un flux de connaissances appliquées au processus.*
6. *Méthodes d'aide à la décision basées sur des modèles logiques ou probabilistes :* Ces méthodes codifient l'expertise d'un secteur de l'entreprise sous forme de système à base de règles logiques ou de réseaux bayésiens, auxquels on fournit une description factuelle d'une situation pour en déduire des recommandations pour la prise de décision à l'aide d'un moteur d'inférence logique ou probabiliste.

3.3 Processus d'intégration d'une base de Connaissance dans la construction du système d'information

3.3.1 Rôle central de l'architecte logiciel dans la génération de la base de connaissance

Avec la complexité de plus en plus croissante des systèmes à développer, l'architecture logicielle est devenue une discipline incontournable au cœur du développement. L'architecture logicielle permet d'exposer de manière compréhensible et synthétique la complexité d'un système logiciel et de faciliter l'assemblage de pièces logicielles. Elle permet, en effet, au concepteur de raisonner à un haut niveau d'abstraction sur les propriétés fonctionnelles et non fonctionnelles. L'architecte est le pilier essentiel de tout développement logiciel. Il conçoit des solutions architecturales qui permettent le passage de l'expression du besoin à l'implémentation du logiciel tout en garantissant la qualité requise. Ainsi, toutes les décisions prises par l'architecte doivent être stockées dans la base de connaissance et exploitées à tous les étages du développement. En accordant à l'architecte un rôle central dans le développement des systèmes, nous recentrons le développement vers l'ingénierie.

Traditionnellement, le processus de développement de logiciel a été modélisé sous forme d'étapes incrémentales mettant en jeu le raffinement, démarrant sur une évaluation du problème à résoudre et aboutissant à l'implémentation du système. La figure 8 illustre un modèle de développement guidé par l'architecture [Le Goaer 2009]. Ainsi, on trouve les quatre étapes suivantes :

- *L'analyse des besoins* vise à déterminer, d'une part l'information et le traitement de cette information et d'autre part les besoins non fonctionnels concernant les aspects de sécurité, confidentialité, robustesse, modificabilité, maintenabilité, utilisabilité, coût et délais, etc.

- *La conception architecturale* concerne la sélection des éléments architecturaux et de leurs interactions, en précisant les contraintes qui pèsent sur ces éléments et leurs interactions. L'architecture fournit un cadre qui satisfait aux besoins et sert de base pour la conception.
- *La conception détaillée* traite de la modularisation, de la spécification détaillée des interfaces des éléments de conception, leurs algorithmes et procédures, et des types de données nécessaires pour soutenir l'architecture. La conception doit satisfaire les propriétés spécifiées à la fois dans l'architecture et par les besoins.
- *L'implémentation* est liée aux représentations des algorithmes et des types de données. Elle doit satisfaire la conception, l'architecture, et les besoins.

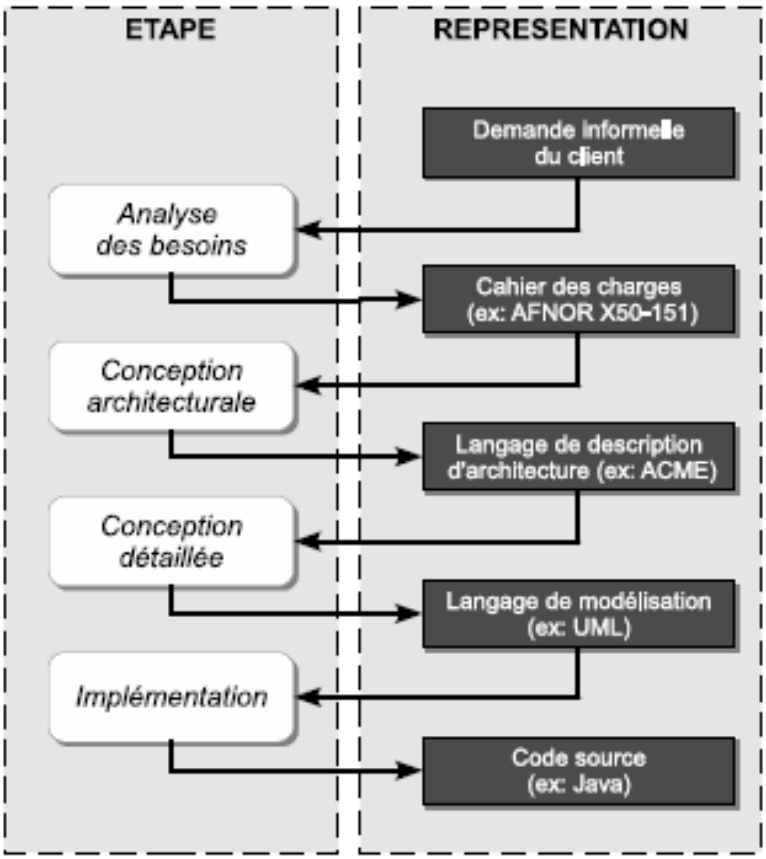

Figure 17 : Etapes du modèle de développement guidé par les architectures

Dans ce processus itératif de développement, des informations et connaissances sont transmises entre les étages. L'étage analyse des besoins et identifie les besoins fonctionnels et non fonctionnels. L'étage de conception architecturale définit les choix architecturaux et les contraintes et les exigences de qualité et de fiabilité. Il détaille les éléments nécessaires pour guider l'implémentation. Chaque étage a besoin des informations générées par les étages qui le précèdent. Nous voyons ainsi qu'il est nécessaire de disposer d'une base d'informations et de connaissances qui sera

alimentée par les modèles, les contraintes, les directives et l'expertise générés, à chaque étage, à partir du modèle de développement centré architecture. Ce besoin devient crucial pour les processus agiles qui sont des processus itératifs et incrémentales, où le système global est développé par partie, ce qui induit un risque de perte d'information et d'expertise significatif.

Pour un équilibrage du processus de développement entre les besoins métiers (les besoins client) et les exigences de l'ingénierie, nous préconisons un modèle de développement centré architecture avec l'ajout d'une base d'information et de connaissance (voir figure 9). Les décisions prises au niveau du processus de développement, qui concerne essentiellement les aspects architecturaux et de conception, sont modélisées pour alimenter la base de connaissance. Les étapes de conception détaillée et d'implémentation sont guidées par les métiers (qui traduisent les besoins du client) et les choix architecturaux. Les informations utiles pour le déroulement de ces étapes sont extraites de la base de connaissance.

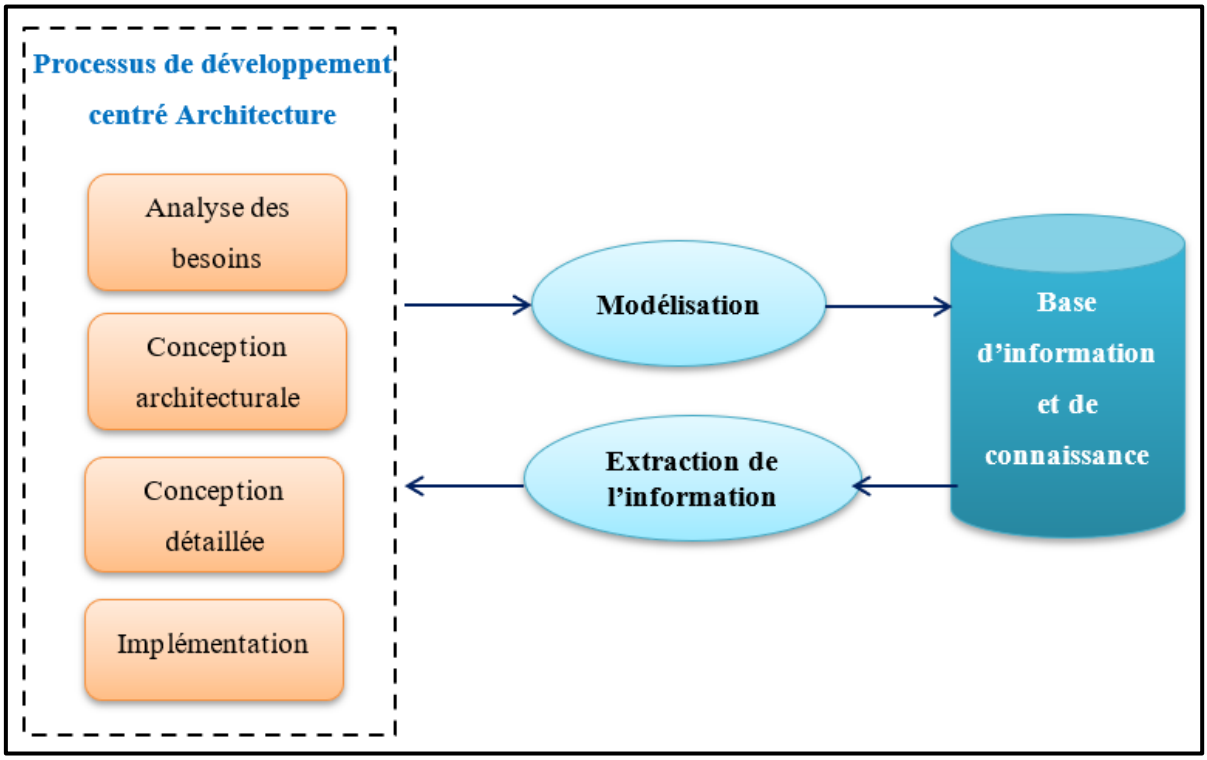

Figure 18 : Intégration d'une base de connaissance dans le processus de développement centré architecture

3.3.2 Modélisation des informations et des connaissances d'ingénierie

Les informations et connaissances à modéliser concernent : (i) les architectures logicielles adoptées, (ii) les modèles de conception et (iii) les connaissances, les bonnes

pratiques et les expertises acquises par les architectes du logiciel et qui doivent être capitalisées.

La modélisation des architectures du logiciel s'appuie sur les styles architecturaux et les formalismes de description d'architectures. Quant à la modélisation de la conception, elle s'appuie essentiellement sur le langage UML (Unified Modeling Langage). Pour la modélisation des connaissances et expertises, les ontologies sont les plus appropriées.

Dans ce paragraphe, nous allons d'abord introduire la notion d'architecture du logiciel, ce qui nous permettra de définir les styles architecturaux et les langages ADL. Ensuite, nous donnons une définition des ontologies et leur rôle dans les systèmes d'information.

3.3.2.1 Définition de l'Architecture Logicielle

La complexité croissante des systèmes a amené progressivement les chercheurs à repenser la façon de construire les systèmes logiciels. Face à l'accroissement de la complexité de ces systèmes, la vision de l'ingénierie des logiciels a évolué en passant de la perspective de développement d'un système monolithique (monobloc) à un système construit comme un assemblage de morceaux logiciels. Ainsi, de nouvelles catégories d'architectures ont émergé, offrant chacune des unités de granulaires différentes pour le partitionnement d'un système (voir figure 10).

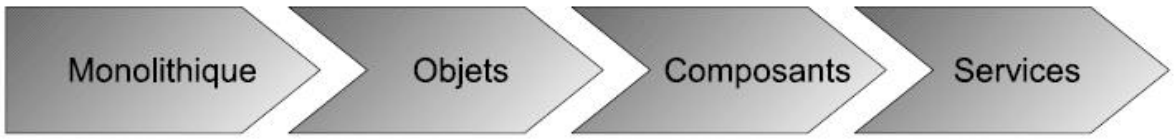

Figure 19 : Catégorie d'architecture face à la complexité croissante des systèmes.

Les objets constituent les briques de base pour les approches de conception orientées objets. Le système est représenté par un ensemble d'objets communicants qui échangent des messages et des données pour la réalisation des besoins fonctionnels du système. La conception orientée objets a introduit certes la propriété de réutilisabilité, mais n'offre pas le niveau de granularité suffisant pour la construction à plus large échelle. Les composants constituent les briques de construction à plus large échelle et qui sont vus comme le prolongement des objets. L'architecture à base de composants vise à construire un système à travers l'assemblage de composants. Puis, la complexité de certains systèmes en termes d'hétérogénéité dans les technologies utilisées, et de distribution à travers les réseaux a fait naître les architectures à base de

services [Le Goaer 2009]. Les services constituent les briques de construction à très large échelle caractérisés par un environnement distribué et hautement dynamique. Ces catégories d'architectures forment des couches conceptuelles successives. La conception d'un service peut reposer sur des composants, eux-mêmes construits à partir d'objets. Chaque catégorie d'architecture offre une couche d'abstraction construite sur la précédente permettant de masquer progressivement les détails pour se focaliser sur les préoccupations relatives à la vue que l'on a du système. L'introduction de la hiérarchie d'abstraction permet la maîtrise de l'accroissement constant de la complexité des systèmes.

L'architecture logicielle se définit comme une spécification abstraite d'un système en termes de composants logiciels ou modules qui le constituent, des interactions entre ces composants (connecteurs) et d'un ensemble de règles qui gouvernent cette interaction [Hadj Kacem 2008]. Un composant est une unité d'abstraction dont la nature dépend de la structure considérée dans le processus de conception architecturale. Il peut être un service, un module, une bibliothèque, un processus, une procédure, un objet, une application, etc. Un composant encapsule typiquement l'information ou la fonctionnalité tandis que les connecteurs assurent la communication entre les composants. Cette architecture possède, généralement, un ensemble de propriétés d'ordre topologique ou fonctionnel qu'elle doit respecter tout au long de son évolution.

Notons qu'un système peut avoir plusieurs structures correspondantes dont chacune a un point de vue, donc une finalité ou classe de problèmes précis à résoudre. En somme, comme en architecture du bâtiment, un logiciel est représenté par plusieurs plans et schémas destinés chacun à un corps de métier et dépendants de l'étape du processus de développement.

Ainsi, une architecture logicielle doit générer plusieurs modèles pour couvrir les différents points de vue du système, mais elle doit aussi capitaliser les expertises et les bonnes pratiques de l'architecte pour guider le développeur à concrétiser les besoins fonctionnels et non fonctionnels du système. Ainsi, l'architecture repose moins sur les fonctions que sur la qualité du logiciel.

Pour aller plus loin dans la capitalisation de l'expertise de conception et pour une plus grande réutilisabilité dans un domaine particulier, la notion de style architectural a été introduite. Un style architectural est un patron de conception qui définit les caractéristiques communes à une catégorie d'architecturales à travers un ensemble de contraintes et de propriétés.

Les styles d'architecture sont des schémas d'architectures logicielles caractérisés ainsi par [Hadj Kacem 2008] :

- Un ensemble de types de composants.
- Un ensemble de connecteurs.
- Une répartition topologique de ces composants indiquant leurs relations.
- Un ensemble de contraintes sémantiques.

Ils forment un ensemble de modèles éprouvés et enrichis par l'expérience de plusieurs développeurs et constituent une aide générique à l'élaboration de structures architecturales. Ils sont à la base de la réutilisabilité et de l'amélioration de la compréhension de l'organisation de l'architecture et des performances du logiciel.

Parmi les principaux styles architecturaux, nous citons l'architecture pipeline, l'architecture avec référentiel, l'architecture Modèle-Vue-Contrôleur (MVC), l'architecture multicouches et l'architecture n-niveaux.

3.3.2.2 Formalismes de description d'architectures logicielles

La description des architectures logicielles peut être réalisée en utilisant un des cinq formalismes suivants (voir figure 11) :

- Les Langages de Description d'Architecture (ADLs)
- Le Langage de Modélisation Unifié (UML)
- Notation de gestion des processus métiers (BPMN)
- Les Techniques formelles
- Les Multi-formalismes

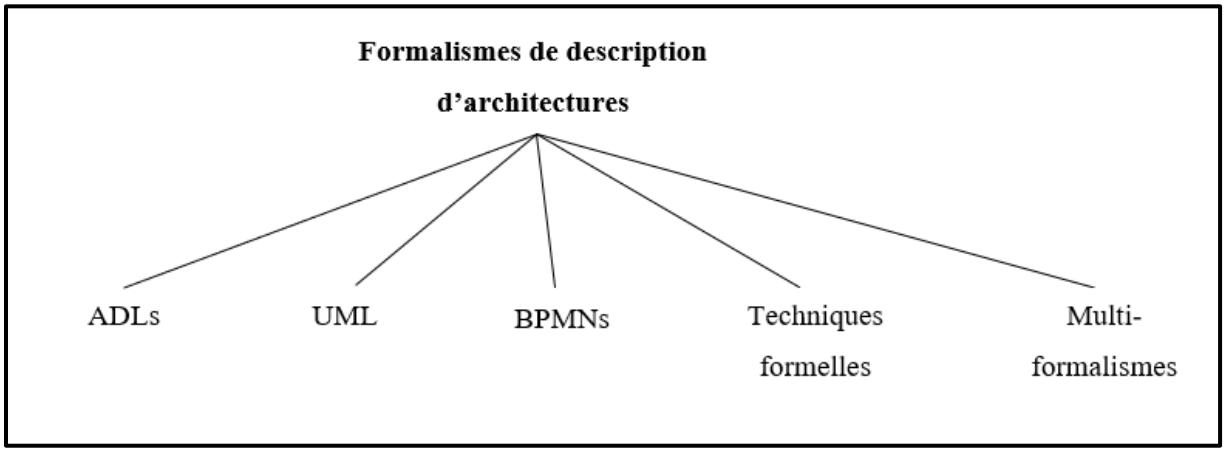

Figure 20 : Formalismes de description d'architectures logicielles

a) Les Langages de Description d'Architecture (ADLs)

Les ADLs offrent les formalismes (graphiques, textuelles, …) nécessaires pour permettre la description explicite des éléments qui caractérisent l'architecture à savoir les composants, les connecteurs et la configuration de l'architecture. Ils doivent offrir aussi des outils de support pour valider et analyser les modèles.

Dans un ADL, un composant est une unité de calcul ou de mémorisation. Une description explicite de son interface est indispensable [Medvidovic et Taylor 2000]. De plus, les composants peuvent être typés, avoir une sémantique formelle ou informelle, exporter des contraintes d'utilisation, permettre l'évolution ou présenter des propriétés non fonctionnelles.

L'interface d'un composant consiste en un ensemble de points d'interactions entre le composant et le monde extérieur qui permettent l'invocation des services, le type du composant, c'est-à-dire une abstraction qui permet l'encapsulation de fonctionnalités dans des blocs réutilisables, et l'instanciation de plusieurs composants similaires.

L'évolution des composants se traduit par des modifications de certaines propriétés (interface, comportement). L'utilisation d'un langage orienté objet permet, notamment grâce à l'héritage, de garantir une évolution avec une réutilisation optimale. Les contraintes sur les services déterminent les conditions d'utilisation d'un composant.

Dans les ADL, les connecteurs sont des blocs de construction utilisés pour modéliser l'interaction entre composants ainsi que les règles qui gouvernent cette interaction. Leurs caractéristiques sont les mêmes que celles définies pour les composants. Cependant, alors que les composants constituent les entités à concevoir, les connecteurs quant à eux représentent des entités de communication à utiliser, c'est-à-dire qui ne sont pas à concevoir ou à modifier par rapport à l'architecture étudiée, mais dont la modélisation est indispensable pour l'analyse du système complet. Ce sont, par exemple, les réseaux ou les connexions physiques quelconques.

La configuration de l'architecture (topologie) est un graphe de composants et de connecteurs pour réaliser une architecture. Cette information est nécessaire pour déterminer si les composants et connecteurs sont composés correctement. Elle permet aussi de réaliser un comportement composite. Dans un ADL, la configuration doit être compréhensible et permettre une description avec différents niveaux d'abstraction avec une continuité entre les niveaux et idéalement jusqu'à la création d'un système exécutable.

Bien que les outils de support ne fassent pas explicitement partie du langage, l'utilité d'un ADL est directement liée aux outils supportés par l'environnement associé. Les environnements classiques basés sur un ADL proposent en partie des outils de spécification active, de gestion des aspects multi-facettes de certains composants, d'analyse, de raffinement et de génération de code exécutable.

Un certain nombre d'ADL ont été proposés pour la modélisation d'architectures, à la fois dans un domaine particulier et comme langages de modélisation d'architecture à usage général. Nous pouvons citer à titre d'exemple Darwin [Magee et al 1994], [Magee et al 1993], Rapide [Luckham 1996] et Wright [Allen et Garlan 1997].

Un ADL est donc un langage qui fournit des fonctionnalités pour modéliser l'architecture conceptuelle d'un système logiciel, distinguée de la mise en œuvre du système. ADL fournit à la fois une syntaxe concrète et un cadre conceptuel pour caractériser les architectures. Le cadre conceptuel reflète généralement les caractéristiques du domaine pour lequel l'ADL est destiné et/ou le style architectural.

b) UML : Unified Modeling Language

UML a été conçu suite à l'unification des différentes notations graphiques des modélisations objets (OMT, BOOCH, OOSE, Fusion, HOOD, etc …). La première version d'UML a été adoptée et publiée par le groupe OMG en 1997. Depuis, UML est devenu la référence pour la conception de logiciels. Il est maintenant utilisé dans la grande majorité des entreprises développant du logiciel à des fins non seulement de documentation et d'analyse, mais de plus en plus à des fins productives de génération automatique ou semi-automatique de code, de tests ou de documentation.

La version 2.0 d'UML a apporté des capacités de modélisation de systèmes qui manquaient aux versions précédentes :

- une séparation claire entre la modélisation de la structure (avec en particulier le diagramme de structure composite) et du comportement du système ;
- une sémantique beaucoup plus précise améliorant l'utilisation productive des modèles dans une démarche d'ingénierie dirigée par les modèles ;
- une amélioration des mécanismes d'extension comme les profils.
- l'introduction de nouvelles constructions qui le rendent plus adapté au développement à base de composants et à la spécification des descriptions architecturales.

La notation UML est décrite sous forme d'un ensemble de diagrammes. La première génération d'UML (UML 1.x) définit neuf diagrammes pour la spécification des applications. Dans UML 2.0, quatre nouveaux diagrammes ont été ajoutés : il s'agit des diagrammes de structure composite (Composite structure diagrams), des diagrammes de paquetages (Packages diagrams), des diagrammes de vue d'ensemble d'interaction (Interaction overview diagrams) et des diagrammes de synchronisation (Timing diagrams). Les diagrammes UML sont regroupés dans deux classes principales :

- **Les diagrammes statiques** : regroupent les diagrammes de classes, les diagrammes d'objets, les diagrammes de structure composite, les diagrammes de composants, les diagrammes de déploiement, et les diagrammes de paquetages.
- **Les diagrammes dynamiques** : regroupent les diagrammes de séquence, les diagrammes de communication (nouvelle appellation des diagrammes de collaboration d'UML 1.x), les diagrammes d'activités, les machines à états, les diagrammes de vue d'ensemble d'interaction, et les diagrammes de synchronisation.

UML 2.0 apporte des améliorations pour représenter les architectures logicielles. Le diagramme de composants a été révisé et le concept de classificateur structuré (structured classifier) a été intégré. Les classes, les collaborations et les composants sont des classificateurs structurés. UML 2.0 introduit la notion de composant (component) ainsi que la notion de connecteur (connector : assembly connector et delegation connector). L'introduction de ces concepts, ainsi que ceux de port ou encore la distinction entre interfaces offertes et requises fournissent une palette d'éléments de notation intéressante pour la modélisation de l'architecture logicielle [Hadj Kacem 2008].

c) Notation de gestion des processus métiers (BPMN) :

BPMN (Business Process Management Notation) a été développé par BPMI (Business Process Management Initiative) et maintenu à présent par le groupe OMG/BPMI [Object Management Group 2008]. C'est un standard pour la modélisation des processus métiers ainsi que des processus des services Web. Son but principal est de fournir un cadre permettant de décrire un processus d'une manière commune à tous les utilisateurs et ce indépendamment de l'outil utilisé. Il permet de définir principalement trois objets de base, tâche, événement, et branchement. Une tâche est un élément granulaire caractérisé par un début et une fin. Un événement identifie un état particulier dans le processus (c.-à-d. début, intermédiaire, fin). Un branchement

sert à représenter les routages entre les flux d'entrées et les flux de sorties (p. ex. branchement exclusif, parallèle, conditionnel, etc.). BPMN se compose d'un diagramme appelé le diagramme de processus métiers (BPD). Ce dernier a été conçu pour être facile à utiliser et à comprendre, mais offre également la possibilité de modéliser des processus métiers complexes.

d) Les techniques formelles :

Un cahier des charges décrit essentiellement les données manipulées et leur évolution, c'est-à-dire les opérations qui les transforment. Les deux principales approches de la spécification formelle diffèrent quant à l'accent mis sur ces deux aspects [Almeida et al 2011] :

- Le comportement du système modélisé peut être exprimé en se concentrant sur ses opérations, les mécanismes disponibles (services) ou les actions qui peuvent être effectuées. De ce point de vue, l'élément crucial est une définition claire des modifications ou des changements effectués par chaque opération de l'état interne du système modélisé. Les langages de spécification qui décrivent de tels systèmes sont appelés langages de spécification basés sur des états ou basés sur des modèles.
- Le comportement du système cible peut plutôt être exprimé en se concentrant sur les données manipulées, la manière dont elles évoluent ou la manière dont elles sont liées. Cette classe de spécifications comprend les spécifications algébriques, connues parfois aussi comme spécifications axiomatiques.

Une classification des langages formels est donnée par Mesli [Mesli 2017], qui s'appuie sur les travaux d'Almeida et al [Almeida et al 2011] et ceux de Lamsweerde [Lamsweerde 2000]. Elle est illustrée par la figure 12.

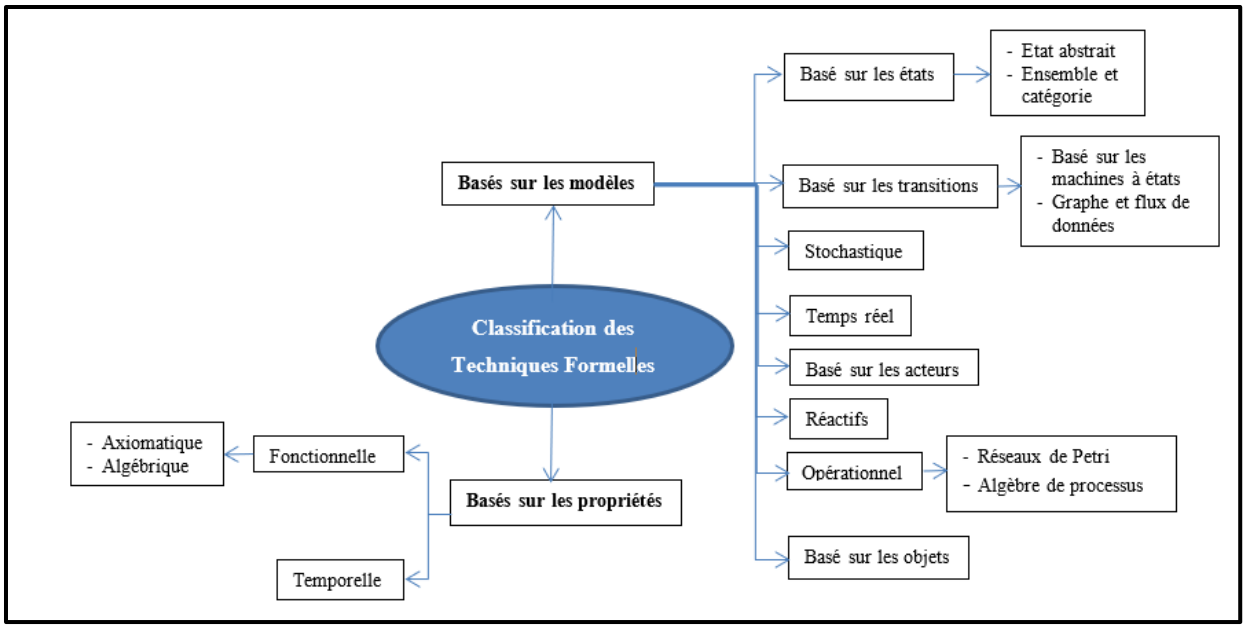

Figure 21 : Classification des techniques formelles.

- **Spécification basée sur un modèle**

Les langages utilisés pour cette classe de spécifications se caractérisent par leur capacité à décrire la notion d'état interne du système cible, et par leur focalisation sur la description de la façon dont les opérations du système modifient cet état. Les fondements sous-jacents sont les mathématiques discrètes, la théorie des ensembles, la théorie des catégories et la logique [Almeida et al 2011].

Les langages basés sur les modèles sont classés comme suit par Mesli [Mesli 2017] :

- **Langages basés sur les états** qui peuvent être divisés en deux catégories : (i) les langages basés sur les états abstraits comme ASM et B et (ii) les langages basés sur les ensembles, comme VDM et Z.
- **Langages basés sur les transitions** qui se focalisent sur les transitions entre états. Ces langages peuvent être divisés en langages basées sur les machines à états, comme les automates, statecharts ou bien les langages basés sur les graphes et les flux de données. Des langages comme Lustre et SDL s'appuient sur ce paradigme.
- **Langages stochastiques** s'appuient sur des modèles probabilistes pour spécifier les systèmes. Chaque transition est étiquetée par une probabilité. Les chaines de Markov font partie de cette catégorie de langage.
- **Langages pour le temps réel** qui doivent être capable de spécifier les aspects temporels. Les automates temporisés font partie de cette catégorie.

- **Langages basés sur les acteurs.** Ils permettent de spécifier les aspects structurels d'un système en spécifiant les relations et les dépendances entre les acteurs du système. Un acteur est une entité active qui exécute des actions pour atteindre des objectifs du système. Parmi ces langages nous pouvons citer TROPOS [Fuxman 2001].
- **Langages réactifs** qui modélisent le système par des entités qui s'exécutent en parallèle. Ils doivent traiter en permanence les événements externes qui proviennent de l'environnement et qui ont pour rôle le déclenchement de l'exécution des entités qui modélisent le système. Parmi ces langages nous pouvons citer le langage ESTEREL [Berry et Gonthier 1992].
- **Langages opérationnels** qui spécifient le système par un ensemble de processus qui s'exécutent en parallèle. Ces langages peuvent être subdivisés en deux catégories : les réseaux de Petri et l'algèbre de processus.
- Langages basés sur les objets qui modélisent le système à travers ses objets. Object-Z [Smith 2012] qui est une extension de la méthode Z fait partie de cette catégorie.

- **Spécification algébrique basée sur les propriétés**

Une seconde approche classique de la spécification repose sur l'utilisation d'algèbres multi-triées. Une algèbre multi-triée consiste en un ensemble de données regroupées en ensembles (un ensemble pour chaque type de données), une collection de fonctions sur ces ensembles correspondant aux fonctions du programme en cours de modélisation, et un ensemble d'axiomes spécifiant les propriétés de base des fonctions dans l'algèbre. Un tel formalisme permet de faire abstraction, à partir des algorithmes utilisés, pour coder les propriétés souhaitées, pour se concentrer sur les représentations des données et sur le comportement d'entrée-sortie des fonctions [Almeida et al 2011]. Ces spécifications basées sur les propriétés sont classées par Mesli [Mesli 2017] en deux catégories :

- **Langages fonctionnels** qui permettent de décrire le système sous forme de collections mathématiques de fonctions. La notion de fonction est essentielle dans ce type de langage. Ils sont subdivisés en deux types : algébrique et axiomatique. dans les langages algébriques les fonctions sont groupées selon les types d'objets et leur domaine et les propriétés sont représentées par des équations. Le langage LOTOS [Bolognesi et Brinksma 1987] fait partie de cette catégorie. Les langages axiomatiques spécifient les fonctions du système par des logiques généralement d'ordre supérieur. Parmi ces langages nous pouvons citer PVS et COQ.

- **Langages basés sur les propriétés temporelles** qui décrivent les traces admissibles du comportement du système à travers le temps. Les propriétés sont décrites par des assertions logiques portant sur les objets du système. Ces langages peuvent être temporels ou stochastiques. Les logiques temporelles telles que LTL et CTL permettent de décrire les traces temporelles du système.

e) Multi-formalismes

Plusieurs travaux de recherche ont élaboré des approches de spécifications et de validation de systèmes en combinant plusieurs formalismes existants. L'objectif principal est de tirer profit des avantages de chaque formalisme en augmentant le pouvoir expressif et de permettre l'application de techniques de vérification et de validation de certaines propriétés critiques. Il s'agit dans la majorité des cas de fusionner un formalisme semi formel type UML avec des formalismes formels de type B, Z ou les réseaux de Petri.

3.3.2.3 Les Ontologies

a) Définition

De nombreuses définitions d'ontologies sont proposées par les différentes communautés scientifiques. Cependant, la définition la plus couramment citée est celle de Grüber [Grüber 1995]. Selon Gruber « *une ontologie est une spécification explicite d'une conceptualisation* ». Cependant, de nombreux chercheurs reprochent à cette définition son caractère très générique. C'est pourquoi, c'est la définition de Studer et al [Studer et al 1998] qui est privilégiée. Celle-ci complète la définition de Gruber en introduisant la notion de « partage ». Ainsi, et d'après Studer et al « *une ontologie est une spécification formelle et explicite d'une conceptualisation partagée* ».

Cette définition dégage quatre notions importantes dans le domaine des ontologies :

- **Formel** : précise que la conceptualisation et la représentation du domaine doivent être standardisées et exploitables par un système informatique.
- **Explicite** : signifie que les concepts utilisés ainsi que les contraintes sont définis de façon déclarative.
- **Conceptualisation** : indique qu'une ontologie est une abstraction du monde réel et que les termes utilisés de même que leurs relations doivent être décrits sans ambiguïté.
- **Partage** : souligne que les ontologies favorisent une connaissance consensuelle.

D'une manière plus formelle, une ontologie est composée d'un vocabulaire spécifique qui permet la description du domaine sous forme d'un graphe constitué de concepts et de relations [Vandecsteele 2012]. Ainsi, une ontologie peut être définie par un ensemble hiérarchique de termes et de relations logiques qui existent entre ces termes à l'aide d'un langage formel. Elle favorise la communication à l'intérieur des organisations.

b) Les composants d'une ontologie

Principalement deux approches existent pour la construction des ontologies : (i) l'approche des cadres (frame) et la logique du premier ordre ; et (ii) l'approche des logiques de description.

- **Modélisation des ontologies par les cadres et la logique de premier ordre :** Selon Grüber [Grüber 1995], une ontologie modélisée par des cadres et de la logique de premier ordre est constituée de cinq principales composantes :
    - *les classes* : elles représentent les concepts généraux d'un domaine donné. Il s'agit d'abstractions pertinentes d'un fragment du monde réel d'un domaine d'application. Une classe peut représenter une tâche, un processus etc. Les classes sont organisées en taxonomie hiérarchique.
    - *les relations* : elles représentent les types d'interaction possibles entre les concepts du domaine. Dans une ontologie, les relations permettent de structurer les connaissances et de définir les interrelations entre les concepts. Deux grands types de relation peuvent être distingués : les relations taxonomiques et les relations associatives. Les premières, appelées également relation de spécificité/généricité permettent d'organiser hiérarchiquement un ensemble de concepts. Les secondes désignent toutes les relations entre les concepts qui ne sont pas des relations de taxonomies.
    - *Les fonctions* : ce sont des cas particuliers de relations dans lesquelles un élément est défini en fonction des éléments précédents.
    - *Les axiomes formels* : ils permettent de représenter les assertions qui sont toujours vraies. Ils permettent de combiner des concepts, des relations et des fonctions pour inférer de nouvelles connaissances.
    - *les instances* : elles permettent de représenter les éléments ou les individus des classes de l'ontologie. Ces individus correspondent à une instance concrète de la classe à laquelle ils appartiennent. Dans le cas d'une ontologie contenant des instances, celle-ci devient alors une base de connaissances [Vandecsteele 2012].

- **Modélisation d'une ontologie par la logique de description :** La théorie de la logique de description permet de construire les ontologies à l'aide de trois composantes : les concepts, les attributs (rôles) et les individus [Vandecsteele 2012].
    - *Les concepts* : ils ont la même définition que dans l'approche des cadres. Ils représentent les classes d'objets.
    - *Les attributs* : permettent de représenter les relations entre les concepts et les propriétés des concepts.
    - *Les individus* : représentent les instances des classes.

c) Rôle des ontologies dans les systèmes d'information

Les ontologies servent de cadre unificateur et un moyen pour la capitalisation et le transfert de connaissances. Gandon précise dans [Gandon, 2008] que les ontologies jouent un rôle aux différents niveaux d'un système d'information : « *Une ontologie informatique offre un cadre unificateur et fournit des primitives améliorant la communication entre les personnes, entre les personnes et les systèmes, et entre les systèmes* ».

Ainsi l'ontologie apparait comme un élément essentiel au cœur des systèmes d'information. Une ontologie offre, en effet, un cadre unificateur dans la mesure où une ontologie spécifie un langage commun, un corpus de connaissance dans un domaine donné. Cela permet d'éviter les ambigüités et de favoriser la communication. Cette communication concerne non seulement les personnes entre elles, mais aussi les personnes et le système ou les systèmes eux-mêmes. Il est nécessaire de souligner que les ontologies favorisent la représentation et la manipulation des informations, c'est-à-dire la capitalisation des connaissances et leur réutilisation.

Plus précisément, dans un système informatique, les ontologies interviennent à différents niveaux. Guarino [Guarino, 1998] en dénombre sept :

- la spécification et l'analyse des besoins du système ;
- la maintenance du système ;
- la coopération et le partage de connaissance ;
- la recherche d'information ;
- l'interopérabilité entre diverses sources de données hétérogènes ;
- la compréhension du schéma conceptuel et du vocabulaire du système à travers la visualisation ;
- l'exécution et le traitement de requêtes exprimées en langage naturelle.

L'ontologie joue un rôle central dans les systèmes informatiques. Elle constitue un support de connaissance entre les différentes composantes du système.

d) Typologie des ontologies

Plusieurs typologies d'ontologies ont été proposées par les chercheurs selon le degré de conceptualisation ou le niveau de formalisation des connaissances.

- **Typologie selon l'objet de conceptualisation** : la typologie de ce type le plus couramment citée est celle proposée par Guarino [Guarino, 1998]. Celle-ci est construite en quatre niveaux dépendants de conceptualisation (voir figure 13) :
  - *Ontologies de haut niveau* : Elles sont aussi désignées aussi « top level ontologies » [Guarino 1998], ou encore ontologies génériques [Van Heijst et al 1997]. Elles modélisent les concepts les plus génériques valables pour plusieurs domaines. Par exemple, les concepts de temps et d'espace peuvent être représentés dans une ontologie générique.
  - *Ontologies de tâches* : Elles fournissent un ensemble de termes et de relations qui permettent de décrire la manière de résoudre un problème. Une typologie de tâche est la structure conceptuelle d'un système à base de connaissance [Mizoguchi et Bourdeau 2000].
  - *Ontologies du domaine* : Elles décrivent les concepts spécifiques à un domaine donné. Elles sont construites par spécialisation des termes et des notions véhiculées par les ontologies de haut niveau pour un domaine spécifique.
  - *Ontologies d'application :* Elles sont utilisées pour modéliser les concepts d'un domaine particulier dans le cadre de la réalisation d'une activité spécifique. Elles sont à la fois la spécialisation et l'union des ontologies de tâches et de domaines.

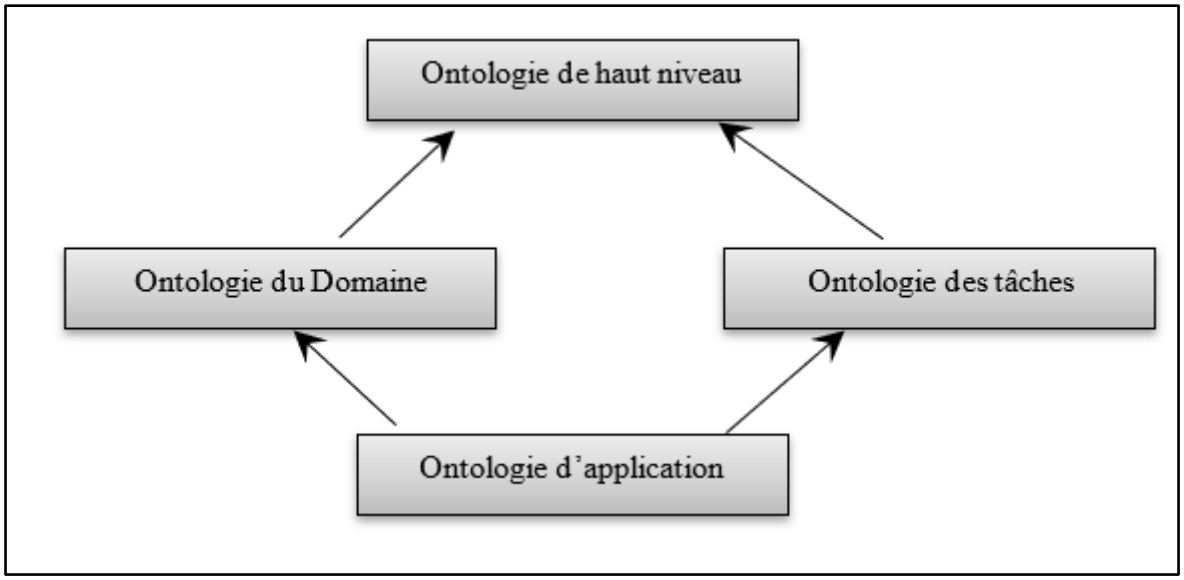

Figure 22 : Typologie selon le niveau de conceptualisation [Guarino, 1998]

- **Typologie selon le degré de formalisation :** Par rapport au degré de formalisation de la représentation, Uschold et Gruninger [Uschold et Gruninger 1996] proposent une classification comprenant quatre niveaux :
  - *Ontologie informelle* **:** L'ontologie est exprimée dans un langage naturel ;
  - *Ontologie semi-informelle* : L'ontologie est exprimée dans un langage naturel structuré ;
  - *Ontologie semi-formelle* : L'ontologie est exprimée dans un langage artificiel défini formellement ;
  - *Ontologie formelle* : L'ontologie est exprimée dans un langage artificiel disposant d'une sémantique formelle permettant de procéder à des vérifications.

e) Méthodologies pour la construction d'ontologies

Comme toutes les méthodes de génie logiciel, les ontologies doivent offrir un processus qui définit des règles précises pour leur construction. Cependant, il n'y a pas encore de consensus autour d'un processus unique. Il existe actuellement plusieurs méthodologies de construction des ontologies. Un point commun se dégage de ces méthodologies. Il concerne leur structuration en trois phases principales : (i) la spécification sémantique, (ii) la conceptualisation et (iii) la formalisation. Il faut préciser que la construction d'ontologie n'est pas linéaire, mais comprend de nombreux aller-retour entre les phases de conceptualisation et de formalisation. Le processus de développement des ontologies est donc un processus itératif et incrémental.

3.4 Conclusion

Dans ce chapitre, nous avons présenté notre proposition pour le pilotage mixte de l'agilité dans les SI par les besoins métiers (client) et les décisions d'ingénierie générées essentiellement par les architectes logiciels. Pour atteindre cet objectif, nous avons proposé l'intégration d'une base de connaissance qui rassemble toutes les décisions concernant les choix architecturaux, les politiques de sécurité et la vision globale du système pour l'intégration et les évolutions futures. Cette base de connaissance constitue l'élément central d'un système d'information. C'est un cadre unificateur qui améliore la communication entre les différents intervenants et fourni les standards et les expertises nécessaires.

Nous avons entrepris la formalisation de cette base de connaissance à travers les langages de descriptions des architectures logicielles ADLs et les ontologies. Les ontologies sont très prometteuses dans la mesure où elles offrent un langage commun pour la représentation et l'exploitation des connaissances dans un domaine donnée.

Chapitre 4 : Proposition d'un modèle d'agilité centré sur l'ingénierie appliqué au secteur bancaire

L'enjeu qui se présente aujourd'hui est celui d'amener dans ce nouveau modèle Agile organisé autour d'équipes projets, de communautés et d'une culture d'entreprise, une nouvelle composante d'ingénierie du SI qui s'opère au cœur même de ce modèle. On peut croire aujourd'hui que l'architecture d'entreprise remplie ce rôle transverse de convergence et de cohérence entre les processus métiers et l'implémentation IT. Cela fut le cas dans des modèles d'organisations et de méthodes de projets (MERISE entre autres) compartimentés, ou chaque étape nécessitait la validation de l'étape précédente et ou les architectes devaient valider l'architecture en amont de l'implémentation et intervenaient ainsi directement sur les projets. Dans ces modèles déterministes et prédictifs, la trajectoire était maintenue du début jusqu'à à la fin et on pouvait tendre à réduire énormément les écarts entre les choix d'architecture définies par les architectes du démarrage, jusqu'à la solution finale. Or, le modèle a évolué et la pratique s'est métamorphosée avec les approches adaptatives Agile en accordant beaucoup plus de responsabilités aux équipes opérantes, notamment en délégant une part importante des choix d'architectures à ces mêmes équipes. En effet, l'axe de réflexion basé sur la satisfaction client amène les équipes à modifier la trajectoire en cours de route et ainsi à devoir rapidement trancher de nouveaux choix conceptuels. Dans la pratique, les équipes d'architectures sont rarement interpellées pour accompagner ces choix du fait de leurs récurrences importantes et du nombre important de projets. Il devient plus évident que l'équipe d'architecture est dans une position périphérique par rapport au cœur du réacteur représenté par les équipes projets. Son rôle s'est dilué, et se limite, au final, à constater les écarts entre ce qui a été livré par les projets et la cible qui a été définie en s'efforçant à maintenir manuellement un référentiel qui, par l'accélération des transformations du SI, semble atteindre ses limites.

Dans ce contexte, nous proposons de repenser le rôle des architectes logiciels et d'entreprise sous le prisme du modèle Agile afin de contrebalancer cette vision court-termiste dans les projets. En effet, en intégrant l'architecture dans le modèle, on porte les besoins sur SI au cœur de la méthode et on réaligne la balance afin de résonner non plus seulement en besoin client mais aussi en besoin du SI. Construire le SI de demain doit être une préoccupation de l'ensemble des acteurs de l'organisation et non seulement une activité restreinte aux équipes d'architecture.

L'objectif est donc d'exploiter la base de connaissance présentée dans le chapitre précédent et de l'instancier dans un modèle qui repend aux différentes vagues de l'Agile, c'est-à-dire en débutant par les équipes projets, puis l'organisation et en terminant avec la culture d'entreprise.

4.1 Méthode Agile et ingénierie du SI

Il est primordial que les décisions d'ingénierie prises aux différents niveaux de structuration soient prises en compte dans la mise en œuvre des systèmes d'information. Dans une phase de constructions des SI, ses choix peuvent concerner des choix de styles architecturaux pour le logiciel qui aura une répercussion certaine sur la qualité du logiciel. Ils peuvent concerner aussi les choix des politiques de sécurité qui doivent être implémentées. Mais cela pourra aussi concerner d'une manière générale les méthodes et approches d'optimisation et d'amélioration des performances.

Nous proposons d'intégrer ce modèle de connaissance à l'une des méthodes agiles les plus pratiquées : la méthode Scrum.

Pourquoi le choix de Scrum en particulier ? Parce que Scrum est de loin la méthodologie la plus utilisée parmi les méthodes agiles existantes, elle est donc plus éprouvée, plus documentée et supportée.

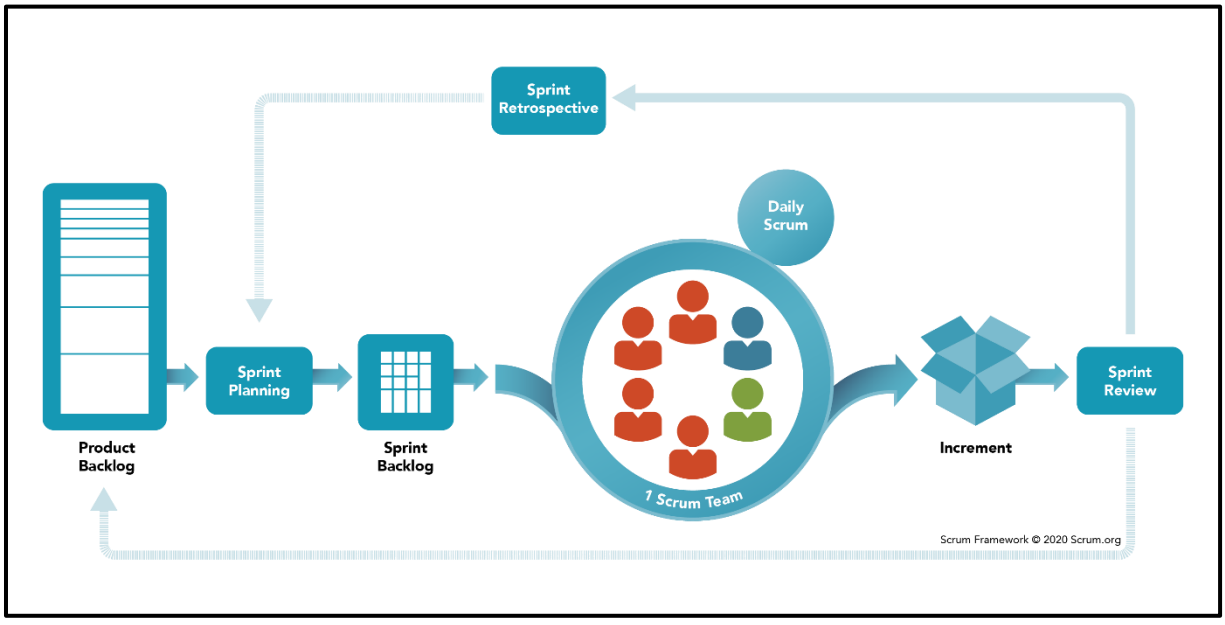

Figure 23 : Méthode Scrum

Scrum est considéré comme un cadre ou « framework » de gestion de projets. Ce cadre est constitué d'une définition des rôles, de réunions et d'artefacts.

Scrum définit 3 rôles :

- **Le « Product Owner »,** qui porte la vision du produit à réaliser (représentant généralement le client).
- **Le « Scrum Master »** garant de l'application de la méthodologie Scrum.
- **L'équipe de développement** qui réalise le produit.

La vie d'un projet Scrum est rythmée par un ensemble de réunions clairement définies et strictement limitées dans le temps :

- **Planification du Sprint** (Sprint = itération) : au cours de cette réunion, l'**équipe de développement** sélectionne les éléments prioritaires du « **Product Backlog** » (liste ordonnancée des exigences fonctionnelles et non fonctionnelles du projet) qu'elle pense pouvoir réaliser au cours du sprint (en accord avec le « **Product Owner** »).
- **Revue de Sprint** : au cours de cette réunion, qui a lieu à la fin du sprint, l'**équipe de développement** présente les fonctionnalités terminées au cours du sprint et recueille les feedbacks du **Product Owner** et des utilisateurs finaux. C'est également le moment d'anticiper le périmètre des prochains sprints et d'ajuster, au besoin, la planification de release (nombre de sprints restants).
- **Rétrospective de Sprint** : la rétrospective qui a généralement lieu après la revue de sprint est l'occasion de s'améliorer (productivité, qualité, efficacité, conditions de travail, etc) à la lueur du "vécu" sur le sprint écoulé (principe d'**amélioration continue**).
- **Mêlée quotidienne** : il s'agit d'une réunion de synchronisation de l'équipe de développement qui se fait debout (elle est aussi appelée "stand up meeting") en 15 minutes maximum au cours de laquelle chacun répond principalement à 3 questions : « Qu'est-ce que j'ai terminé depuis la dernière mêlée ? Qu'est-ce que j'aurai terminé d'ici la prochaine mêlée ? Quels obstacles me retardent ? ».

L'intégration du modèle de connaissance dans l'approche agile Scrum nécessite l'ajout d'un autre rôle, celui de **l'expert architecture** qui a une connaissance parfaite du modèle de connaissance et qui participe aux différentes réunions et veille à son respect et à sa mise en œuvre.

A travers ce modèle de connaissance et l'expert ingénierie, les approches agiles se rééquilibrent en faveur de l'ingénierie et ne seront plus exclusivement guidées par les métiers représentés par le Product Owner.

Cette intervention peut être ponctuelle, pour accompagner les équipes ou pour trancher des sujets de conceptions. En effet, impliquer que chaque projet doit être composé d'un architecte dédié est lourd à mettre en place pour les organisations. Ainsi, dans le modèle que nous proposons, l'architecte désigné peut accompagner plusieurs projets en parallèles. Or, la difficulté qui est rencontrée par ces interventions ponctuelles, et surtout lorsqu'on n'évolue pas dans l'équipe projet, est la complexité d'appréhender toutes les composantes métiers du projet avant de pouvoir produire une solution technique. Cela se traduit généralement par d'innombrables ateliers et réunions pour mettre à niveau la connaissance de l'intervenant relatives aux contraintes du domaine et des besoins attendus.

Pour répondre à cette problématique nous préconisions une catégorisation de ce rôle d'architecte par domaine. Ainsi, un architecte expert du domaine aura sous sa responsabilité les projets du domaine. En étant expert du domaine, l'architecte sera déjà sensibilisé aux enjeux de son domaine d'expertise et sera dans une position favorable vis-à-vis des métiers pour proposer des solutions qui prennent en compte les besoins du SI et les besoins des métiers.

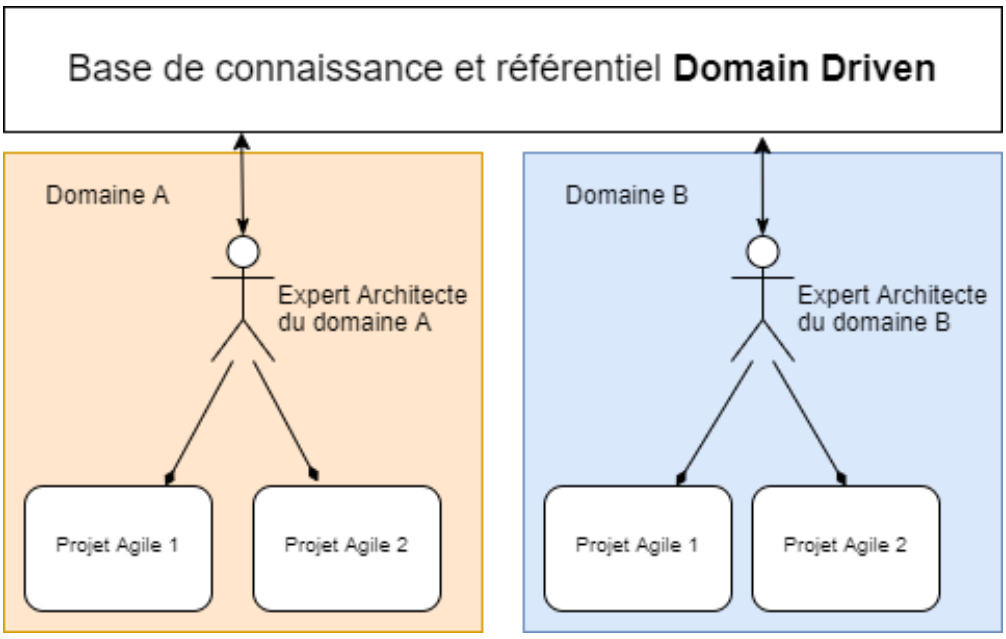

Figure 24: Articulation de l'expert Architecte dans le modèle proposé

Cela contribuera au maintien de l'alignement stratégique métier et IT en conservant une cohérence entre les différents projets du domaine qui s'inscrivent parfaitement dans le concept du **Domain Driven Design (DDD).**

Le concept de Domain Driven Design propose de répondre à la complexité des systèmes modernes en proposant une approche holistique pour le développement

logiciel. Dans son livre « Domain-Driven Design : Tackling Complexity in the Heart of Software », Eric Evans [Evans 2003] fait état de la difficulté des équipes à développer des solutions en toute autonomie. Il arrive à identifier un facteur récurrent et générateur de complexité dans les projets : le langage métier. Afin de résoudre ce problème, Eric Evans a créé le concept de « Bounded Context », un pattern qui consiste à découpler les briques applicatives en définissant les frontières des modèles avec un langage métier cohérent, récurent « Ubiquitous language ». En étendant ce concept à l'organisation et plus spécifiquement à l'intervention des architectes dans les projets, nous nous assurons de maintenir une cohérence et une autonomie entre les différents domaines métiers.

4.2 Organisation Agile et ingénierie du SI

Comme présenté dans les précédents chapitres, l'Agile a bouleversé les organisations en proposant une nouvelle forme d'organisation à l'échelle qui remplace le modèle hiérarchique, structuré par une forme communautaire avec plus de transversalité. Or, cette refonte de l'organisation a réduit le champ de contribution des architectes dans la production de valeur. **Cela a pour conséquence direct d'augmenter l'entropie de la résultante des différents projets et ainsi rendre la convergence et la rationalisation du SI chaotique à maintenir.**

L'autonomie des individus et des équipes proposées par les modèles d'Agile à l'échelle apporte une vraie réponse au problème d'inertie rencontré par les grandes structures. Les avantages sont multiples : capacités de répondre rapidement aux changement, responsabilisation des individus et donc renforcement des initiatives pour innover, etc. Cependant, cette autonomie peut rapidement se transformer en chaos sans alignement. Ainsi, on retrouve dans les modèles à l'échelle des notions de communautés qui proposent de répondre à la problématique. Ces communautés permettent d'assurer le lien entre les individus et de maintenir une organisation cohérente. Néanmoins, leur simple présence ne permet pas d'assurer automatiquement l'alignement du SI.

Dans ce contexte nous proposons un modèle d'organisation des communautés qui repose sur le modèle **Domain Driven Design** afin d'apporter une réponse au besoin d'alignement du SI.

Les piliers du modèle sont :

- **Center of Excellence (CoE)** : composé d'experts reconnus dans une pratique. Ils ont un rôle fédérateur par rapport à la pratique qu'ils représentent. En effet,

ils doivent être en mesure d'identifier les nouvelles techniques (pratiques, stratégies, principes.) qui pourraient améliorer la productivité, capturer ces techniques en fonction du contexte et les formaliser (dans la base de connaissance notamment) en normes pour les partager et les instancier à l'ensemble de l'organisation

- **Design Authority :** entité composée de leaders experts dans leurs domaines, ils sont les garde-fous du respect des normes, c'est-à-dire qu'ils s'assurent que les livrables des équipes projets répondent aux normes formalisées par le CoE. Ils sont en échange directe avec la CoE et les équipes études. Ainsi, ils articulent les échanges entre ceux qui mettent en place les normes et ceux qui les pratiquent dans un processus d'amélioration continue.

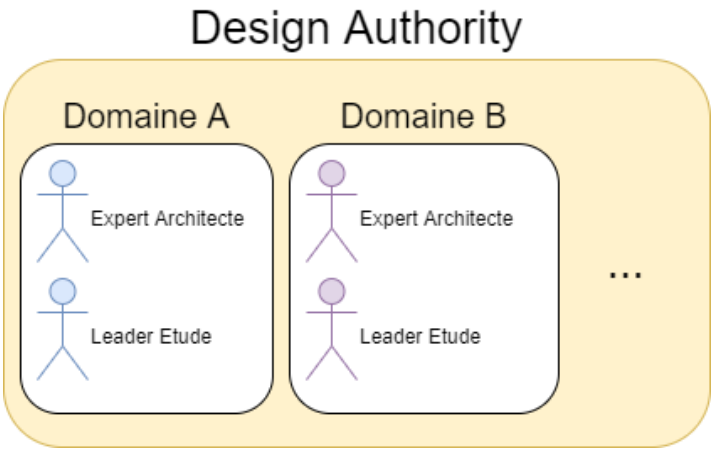

Figure 25 : Composition de la Design Authority

- **Community of Practice (CoP)** : cette communauté évolue dans la pratique des techniques et s'alimente pas les échanges entre ses membres et les autres entités et par la base de connaissance formalisée. Ainsi l'objectif est de promouvoir un apprentissage continu par la pratique et d'une pollinisation croisée des idées.
- **La base de connaissance** : un référentiel commun entre l'ensemble des entêtée/communautés qui repose sur les ontologies et qui contribue à améliorer la circulation des flux de connaissance entre l'ensemble des acteurs.

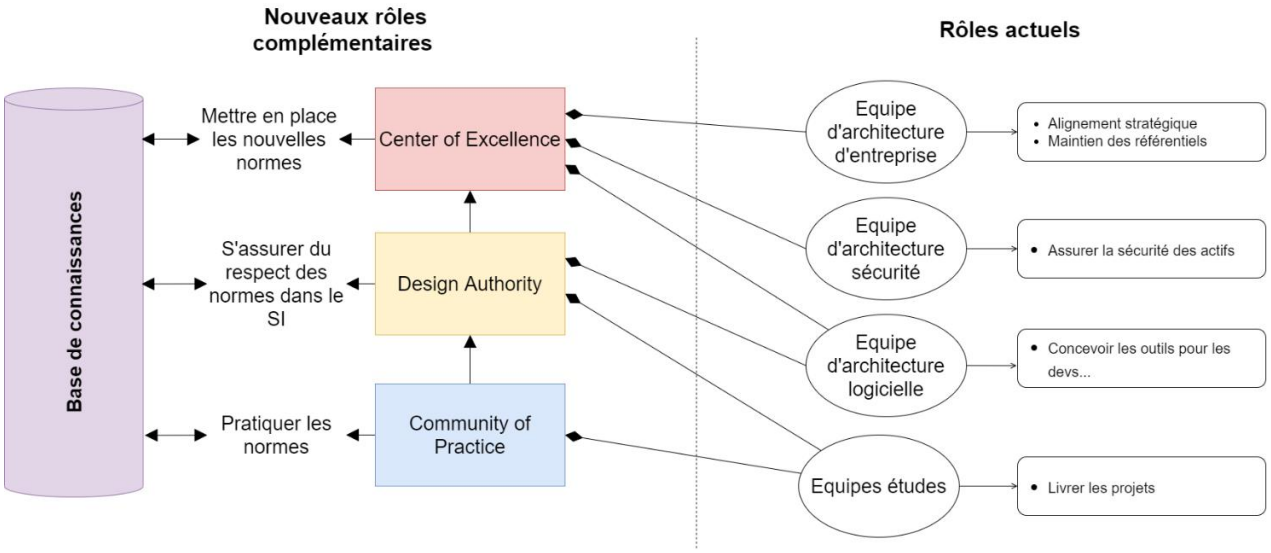

Figure 26 : Proposition d'un modèle d'organisation

4.2.1 Autonomie et alignement avec l'ingénierie du SI

Un des avantages de ce modèle organisationnel est qu'il n'est pas disruptif. En effet, il s'applique de manière complémentaire à l'organisation. Ainsi, les collaborateurs continuent à évoluer dans leurs équipes pour accomplir leurs missions respectives, mais ils se voient attribuer des rôles complémentaires et une appartenance à des communautés qui d'une part les valorise en les mettant en avant et d'autre part déploie leurs expertises à l'ensemble de l'organisation en n'étant plus cotonnés aux frontières de leur équipe.

Ce modèle préserve le paradigme Agile avec toutes ses composantes : autonomie des individus, rôle déterminent du client dans la conduite des projets etc, mais l'accompagne en proposant une forme d'organisation autour des communautés afin d'amener un équilibrage avec la composante ingénierie du SI.

En effet, l'organisation proposée est centrée sur les besoins du SI. Qu'il s'agit du CoE, de la Design Authority ou de la CoP, ces organes ont une réflexion autour des pratiques « indépendamment » des projets. L'idée est de prendre du recul sur notre manière de produire et de réfléchir de manière globale et systémique sur l'impact des choix conceptuels sur le SI de demain, le tout sous le prisme de l'Agile. Cette organisation inclusive, amène l'ensemble des collaborateurs à réfléchir sur les questions de performance et rationalisation du SI et les responsabilise par rapport à ces sujets. *In fine*, l'objectif est de créer un cercle vertueux entre ceux qui formalisent les normes et ceux qui les pratiquent pour tendre à ce que l'écart constaté entre les cibles architecturales et les livrables soient les plus faibles possible.

Pour répondre aux besoins et exigences du SI, le modèle propose de :

- Construire des référentiels de pratique du SI (CoE), s'assurer de leurs bonnes mises en œuvre (Design Autority) et faire infuser ces pratiques au sein de l'organisation (CoP) ;
- Structurer son organisation autour des principes du Domain Driven Design. En effet, la Design Authority repose sur des membres qui sont des experts leaders dans leur domaine métier. Ainsi, une cohérence des pratiques du SI est maintenue par domaine métier. Cette cohérence s'inscrit dans une cohérence globale ;
- Etablir un référentiel commun détenu par l'ensemble des collaborateurs afin de faciliter son alimentation et son exploitation ;
- Créer des regroupements d'experts qui généralement n'ont pas l'occasion de travailler ensemble (architectes d'entreprises, architectes data, architectes sécurité, architectes logiciels, développeurs concepteurs…) dans le but de proposer des solutions pour le SI.

4.2.2 Analyse des flux de connaissance

Il est intéressant de porter un regard sur le modèle que nous proposons sous le prisme des flux de connaissances. L'objectif est d'identifier le rôle des acteurs du modèle dans le cycle de vie de la connaissance et d'évaluer la couverture du modèle du spectre d'état de la connaissance comme étudié dans les chapitres précédents.

Ainsi, si nous reprenons les étapes de cycles de vie des connaissances que nous avons déjà définies, nous retrouvons les trois étapes ci-dessous :

- La création de la connaissance : qui se compose de la génération des connaissances et de leur stockage ;
- Le partage de la connaissance ;
- La réutilisation de la connaissance.

Ces trois étapes se matérialisent dans notre modèle grâce aux articulations que nous représentons dans le schéma ci-dessous :

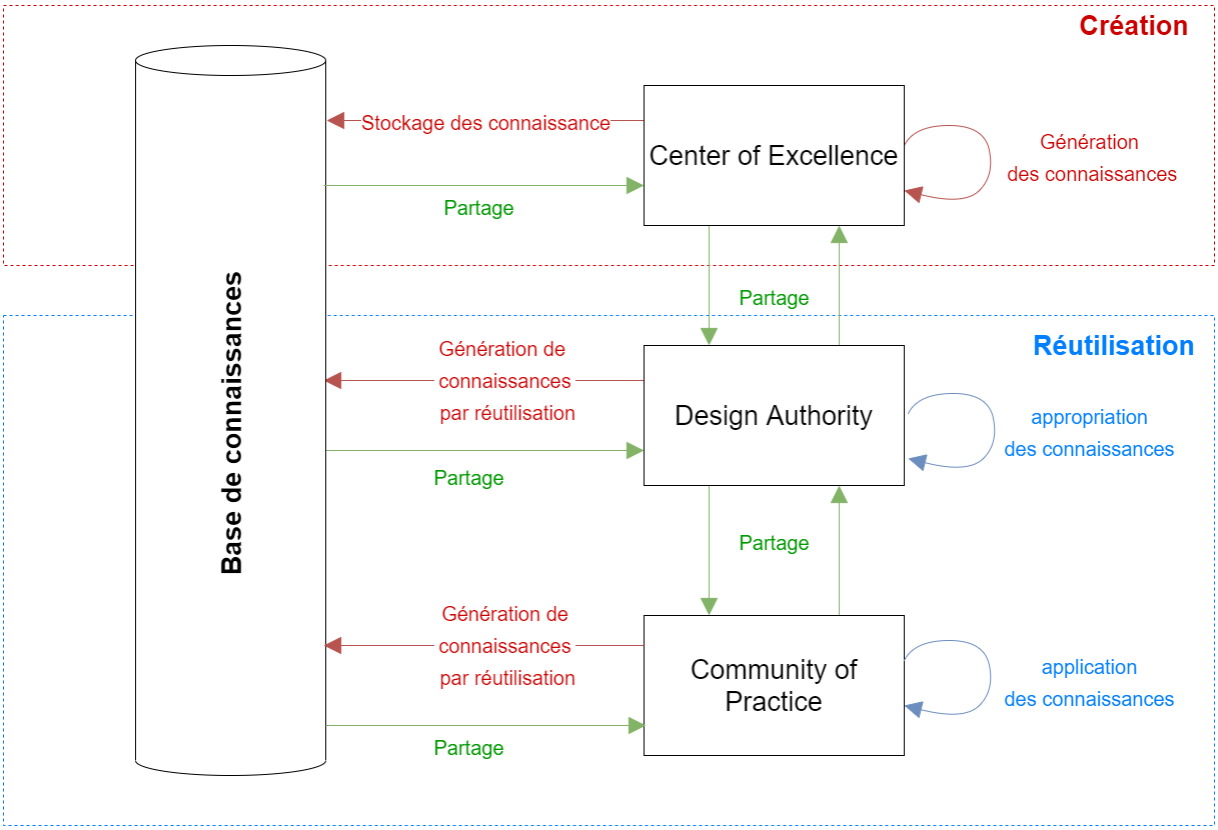

Figure 27 : Cycle de vie de la connaissance dans le modèle proposé

Nous pouvons constater que l'ensemble des composantes du cycles de vie de la connaissance sont représentées dans notre modèle. Cela permet de s'assurer qu'il ne manque pas une composante essentielle et, par conséquent, d'assurer une bonne gestion de la connaissance au sein du modèle.

Au même titre que la représentation du cycle de vie de la connaissance dans le modèle que nous proposons, la caractérisation de la connaissance en tant que connaissance tacite / explicite et collective/ individuelle est importante à déceler afin de comprendre l'ensemble des interactions et la place de chaque acteur dans le modèle. Ces interactions sont présentées dans le schéma ci-dessous :

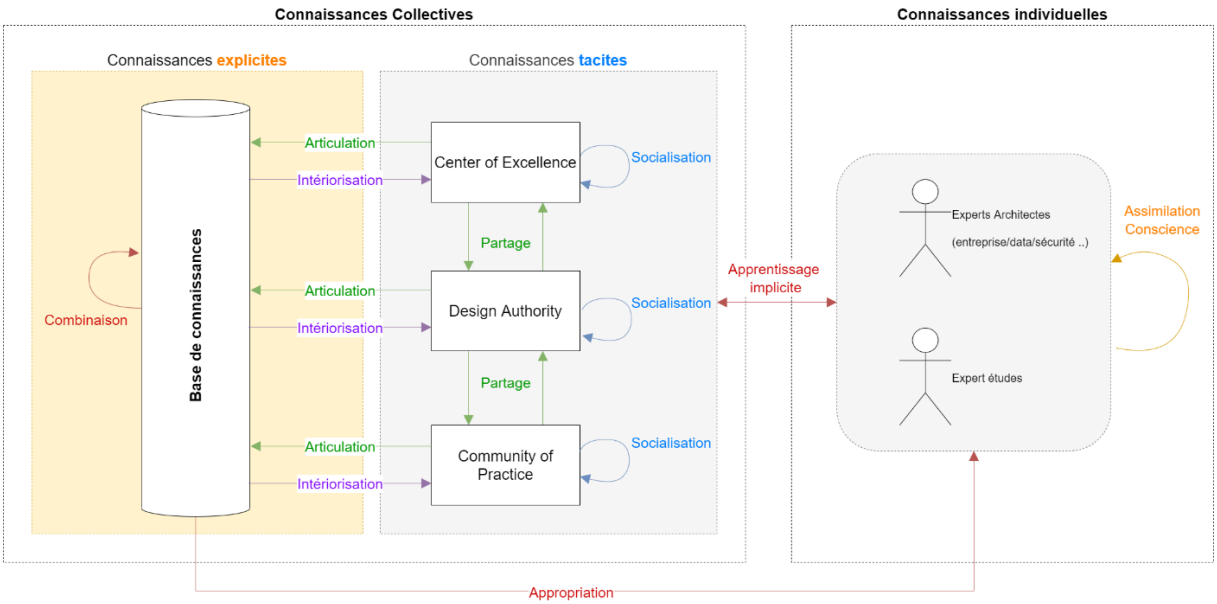

Figure 28: Flux de connaissances

Dans l'ensemble, nous constatons que le modèle est compatible avec les différentes modélisations des systèmes de management de la connaissance rencontrées dans la littérature.

4.3 Culture d'entreprise et ingénierie du SI

La culture d'entreprise se construit grâce à l'accumulation des connaissances et le partage de valeurs et de comportements. Le paradigme Agile a largement participé à transcender cette culture au sein des organisations en appliquant un changement profond au niveau des structures sociales et en mettant en avant l'action des acteurs de l'organisation. Nous avons constaté, grâce au parallèle fait avec la théorie de la structuration, qu'il est indispensable d'amener un changement à tous les niveaux de l'organisation afin de disposer d'une convergence de pensée et de créer un cercle vertueux dans la pratique. Or, le système d'information en tant qu'organe de l'organisation n'a pas été directement concerné par ce mouvement. Nous pouvons faire ce constat en observant que les architectes qui contribuent à l'alignement du SI ne sont quasiment pas concernés par les méthodes Agile. En effet, leur rôle se limite à observer les équipes études évoluer dans cette culture sans y être réellement acteurs. Cela peut conduire à une fracture entre pratiquants et observateurs de la pratique Agile.

Afin de pallier cette problématique, une des promesses du modèle que nous proposons est d'inclure l'ensemble des experts du SI dans le mouvement Agile. La pratique de l'Agile est étendue aux architectes, aux collaborateurs responsables du maintien en condition opérationnelle, etc. L'Agile n'est plus dédié aux développements

logiciels ou aux projets, mais s'applique pour répondre aux besoins du SI, à son alignement et à sa rationalisation.

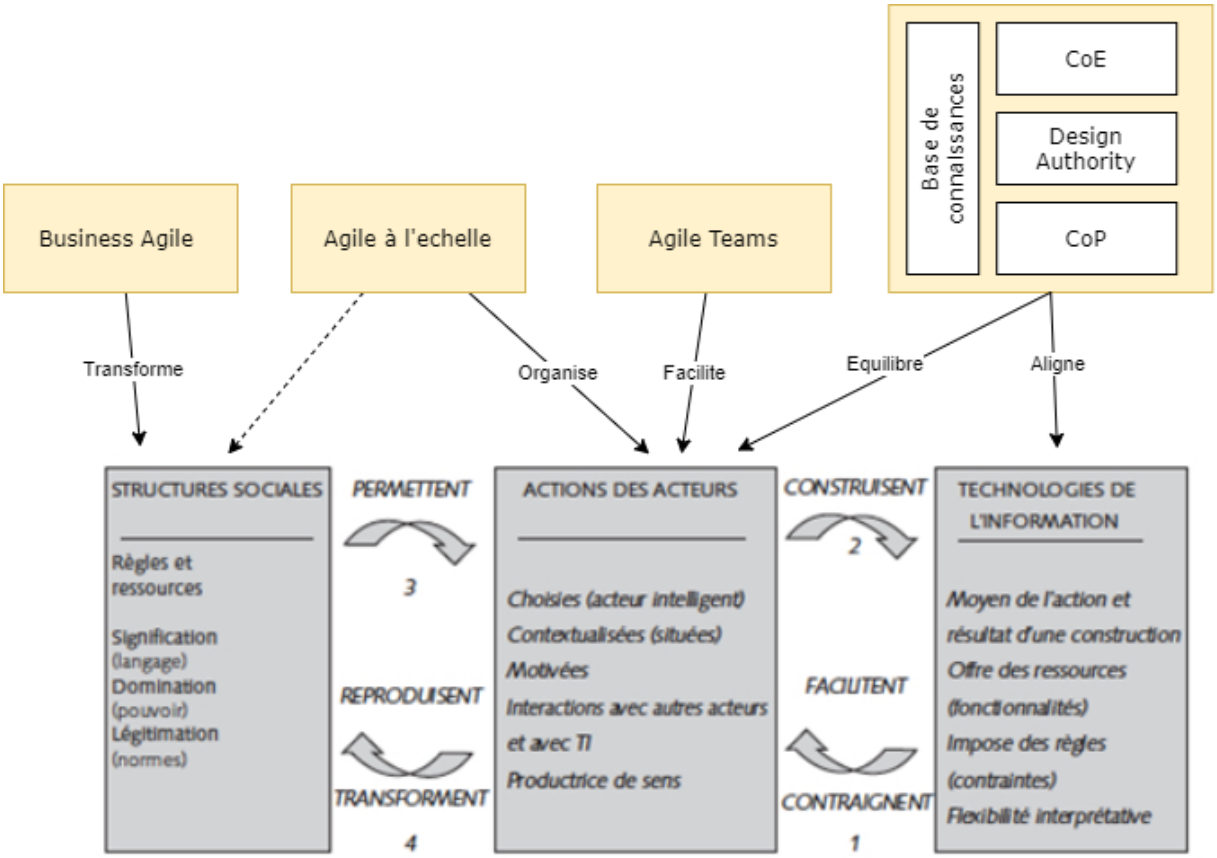

Figure 29 : Théorie de la structuration appliquée à la proposition

En introduisant cette nouvelle composante au modèle de la théorie de la structuration instanciée aux systèmes d'information, l'agilité est appliquée de manière systémique à l'organisation. Les structures sociales, les actions des acteurs et les technologies de l'information sont redéfinies autour du paradigme Agile tout en assurant un équilibre entre autonomie et alignement. Cette application holistique de l'agilité induit de nouvelles connaissances, valeurs et comportements qui finissent par redéfinir la culture de l'entreprise.

4.3.1 Valorisation de l'expertise

On observe aujourd'hui que certains experts se sont tournés vers le management en raison du manque de valorisation des filiales d'expertises et du manque d'opportunités d'évolution. En effet, on a eu tendance à associer le mangement comme l'unique parcours d'évolution. Cela s'est traduit par l'orientation de plusieurs experts techniques vers le rôle de manager sans que cela soit forcément en adéquation avec leurs

compétences ou leur volonté première d'évolution. Notre modèle propose de répondre à ce problème structurel des organisations en proposant un parcours de valorisation des expertises accessibles à l'ensemble des collaborateurs désireux de développer leurs compétences et d'être reconnus à ce niveau-là.

Ce parcours est structuré en 3 étapes :

- Faire ses armes en pratiquant la technique dans une communauté de pratique ;
- Intégrer une Design Authority après avoir maitrisé les nomes et contribuer ainsi à leurs bonnes mises en œuvre au sein de l'organisation ;
- Intégrer le Centre d'excellence et être ainsi reconnu comme un expert incontestable, capable de capturer les nouvelles pratiques et de les instancier à l'organisation.

L'aspect réellement disruptif de ce parcours est qu'il ne fait pas intervenir le management dans le choix d'intégration des collaborateurs au sein des différentes communautés. En effet, cette gestion des membres est déléguée aux communautés qui sont les plus aptes à évaluer le niveau de maturité des individus dans leur pratique.

4.4 Conclusion

Après avoir constaté un déséquilibre dans la mise en œuvre actuelle de l'agilité au sein des organisations, plus précisément entre l'attention apportée aux besoins clients et celle apportée aux besoins du SI, mais aussi entre l'autonomie des individus et les mécanismes d'alignements intrinsèque à l'Agile, nous avons introduit un nouveau modèle qui vise à compléter cette orientation Agile en amenant encore plus d'agilité au niveau des systèmes d'information. Cette proposition s'inscrit dans le paradigme Agile et repose sur un modèle communautaire. Ce modèle communautaire maintient l'autonomie des individus tout en la mettant au service de la convergence des systèmes d'information. Notre proposition s'applique à tous les niveaux de l'Agile. En effet, nous nous sommes intéressé dans un premier lieu à la méthode Agile en proposant d'introduire un nouveau rôle d'expert architecte par domaine d'étude. Dans un second temps, nous avons traité l'Agilité à l'échelle en introduisant un nouveau modèle d'organisation opérationnel qui repose sur les communautés et les articule de façon à assurer l'alignement des systèmes d'information. Nous avons clôturé notre proposition en mettant en perspective notre modèle avec la culture Agile et en démontrant que notre modèle ne fait que renfoncer cette culture en apportant encore plus d'agilité à l'ensemble du système.

Afin d'appuyer notre raisonnement et notre proposition, nous avons appliqué des modèles conceptuels théoriques afin de s'assurer que notre modèle est cohérent. Nous avons notamment instancié la méthodologie de formalisation de la connaissance afin de montrer que nous couvrons l'ensemble du spectre de la transformation de la connaissance. Nous avons aussi repris la théorie de la structuration de Giddens afin de mettre en lumière toutes les interactions qui sont apportées par l'agilité et particulièrement par notre nouveau modèle.

Conclusion générale et perspectives

Dans un monde ultra-compétitif qui allie les géants de la finance avec des startups en pleine croissance, un des enjeux majeurs des DSI des organisations historiques consiste à réduire le coefficient d'exploitation en rationnalisant le SI tout en maintenant un rythme soutenu d'évolution afin de demeurer compétitif vis-à-vis des startups qui ne sont pas contraints par un lourd actif historique.

Dans ce contexte de plus en plus dynamique et imprévisible, la capacité d'une entreprise à naviguer dans cette complexité est devenue un prérequis indispensable pour assurer sa compétitivité et sa pérennité sur le temps long. Afin de répondre à ces nouvelles exigences, le paradigme Agile s'est imposé comme une pratique privilégiée par les organisations pour évoluer vers un système de pensée adaptatif, en rupture avec un cadre de pensée prédictif.

Les travaux conduits dans le présent mémoire, développés et appliqués de manière concrète au sein d'une entreprise financière du CAC40, disposant d'un système d'information complexe et historique, proposent un nouveau modèle qui vise à assurer un alignement de l'ingénierie avec les métiers en amenant encore plus d'agilité à l'Agile.

L'Agile, qui s'est démocratisé à l'ensemble des niveaux hiérarchiques, en méthode de projet, puis en organisation et qui s'installe comme une culture, est néanmoins à ses débuts et n'apporte pas une réponse pleinement satisfaisante à l'ensemble des enjeux inhérents aux systèmes d'information, notamment en termes d'alignement des besoins des systèmes d'information avec les besoins métiers.

La dissonance des objectifs (rationalisation *vs* évolution continue) fixés par les DSI maintien une forte pression sur le SI et génère de la tension au sein des organisations entre les équipes qui œuvrent pour développer des nouveaux outils pour le client et ceux qui œuvrent pour rationaliser le SI.

Ce mémoire a également pour ambition de proposer un nouveau modèle centré sur l'Agile afin de concilier ces deux objectifs en intégrant l'alignement du SI au cœur même de la méthode de développement et ainsi inclure l'ensemble de l'organisation dans cet effort qui vise à pérenniser le SI sur le long terme tout en maintenant une cadence élevée dans la livraison de nouveaux actifs pour le client.

Sur la base d'une étude bibliographique relative au domaine Agile, nous avons pu retracer la naissance de ce mouvement et son évolution qui l'a hissé au sommet des organisations modernes. Cette étude relève trois vagues du mouvement, que sont, l'agilité dans les méthodes de gestion de projet, l'agilité dans l'organisation et l'agilité au service de la cuture d'entreprise. Nous avons démontré comment ce paradigme est disruptif par rapport au système de pensée prédictif. En appliquant la théorie de la structuration sociale de Giddens aux systèmes d'information dans une organisation Agile, nous avons identifié les nouvelles articulations qui s'appliquent aux organes du système et mis en évidence le besoin de maintenir une cohérence holistique du modèle.

L'étude bibliographique des différentes formes d'alignement du SI / métiers a montré que celles-ci se préoccupent principalement à répondre aux besoins des métiers et relèguent les exigences propres du SI au second plan. Or, l'avènement de l'agilité a dilué le rôle des architectes et leur impact sur la consistance des actifs produits. Ce postulat nous a amené à étudier les modèles de gestion de connaissance et de là, à proposer une base de connaissance transverse pour remettre au premier plan la connaissance inhérente au SI à l'ensemble des acteurs. Afin de construire cette base de connaissance, nous nous sommes intéressé aux langages formels et aux ontologies pour disposer d'une connaissance explicite au service d'un consensus pour l'ensemble des acteurs du SI.

En mettant en perspective l'ensemble des concepts étudiés, nous avons proposé un nouveau modèle organisationnel / opérationnel qui vise à recentrer les besoins du SI au cœur de la méthode Agile au même titre que sa capacité à répondre aux besoins des clients. Afin de s'assurer de la complétude de notre proposition, nous avons transposé les modèles conceptuels théoriques sur notre modèle. Nous avons pu constater la transition des flux de connaissance ainsi que l'impact de notre modèle sur l'ensemble des structures du système.

Comme perspectives de nos travaux, il serait intéressant d'intégrer la notion de système cyber-physique sociale. Ce système relie le monde computationnel virtuel au monde physique et recentre l'humain au cœur du système. Cette dimension pourrait compléter notre base de connaissance en y intégrant l'interaction sociale afin de disposer d'un système analytique complexe qui permettrait de simuler l'état opérationnel futur de notre système d'information et de ses acteurs.

Références Bibliographiques

[Ait-Cheik-Bihi 2012] Ait-Cheik-Bihi W. : " Approche orientée modèles pour la vérification et l'évaluation des performances de l'interopérabilité et l'interaction des services", Thèse en informatique de l'université de technologie de Belfort-Montbeliard, soutenue le 21 juin 2012.

[Allen et Garlan 1997] Allen R., Garlan D.: "A formal basis for architectural connection". ACM Transactions on Software Engineering and Methodology, 6(3) :213–249, 1997.

[Appelo 2011] Jurgen Appelo : «Management 3.0 Leading Agile Developers, Developing Agile Leaders»

[Arduin et al 2012] Arduin P-E., Doan Q-M., Grigori D., Grim M., Grundstein M., Negre E., Rosenthal-Sabroux C. et Thion V. : " Evaluation d'un système d'information et de connaissance - De l'importance de la prise en compte de la connaissance ", INFORSID 2012,

[Baizet 2004] Baizet, Y. (2004) : "Knowledge Management in design: Application to the computational mechanics at Renault-BED". Ph. D Thesis University of Grenoble.

[Bass et al 1998] Bass L., Clements P., Kazman R. : "Software architecture in practice". Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1998.

[Beck et al 2001] Beck, K., et al. : «The Agile Manifesto» . Agile Alliance. http://agilemanifesto.org/

[Beek 2007] Beek M. H, Bucchiarone A., and Gnesi S. : "Formal methods for service composition". Annals of Mathematics Computing Teleinformatics, 1(5) :1–14, 2007.

[Ben Amira 2015] Ben Amira A., : "Modélisation agile pour un système de fabrication complexe et dynamique", Thèse en Génie Industriel de l'Ecole Nationale des Mines de Saint-Étienne soutenue à Gardanne, le 14 octobre 2015.

[Bernus et Nemes 1996] Peter Bernus et Laszlo Nemes : A framework to define a generic enterprise reference architecture and methodology. Computer Integrated Manufacturing Systems, Volume 9, pp. 179-191.

[Berry et Gonthier 1992] Gérard B. ; Gonthier G. : "The Esterel synchronous programming language : Design, semantics, implementation". In : Science of computer programming 19 (1992), Nr. 2, p. 87–152

[Bleistein et al 2006] Bleistein, S. J., Cox, K., Verner, J. & Phalp, K. T., 2006. : "B-SCP: A requirements analysis framework for validating strategic alignment of

organizational IT based on strategy, context, and process". Information and Software Technology, 48(9), pp. 846-868.

[Bolognesi et Brinksma 1987] Bolognesi T. ; Brinksma E. : "Introduction to the ISO specification language LOTOS ". In : Computer Networks and ISDN systems 14 (1987), Nr. 1, p. 25–59.

[Bouzid 2017] Bouzid M. : "Intégration des informations et connaissances en Usine Numérique", thèse de doctorat en informatique de l'Ecole Nationale des Ingénieurs de Sfax et de l'Université Paris-Saclay préparée à CentraleSupelec et soutenue le 3 mars 2017.

[Cigref 2003] Cigref : "Accroître l'Agilité du système d'information - Urbanisme : des concepts au projet", Rapport publié en septembre 2003 site web : www.cigref.fr

[Clark et al 1990] Clark J., Modgil .C, Modgil: Concensus and controversy. In : Anthony Giddens : concensus and controversy. London ; New-York: Falmer Press, 1990, 352 p. (Anthony Giddens : concensus and controversy).

[Drucker et Fontaine 1993] Drucker, P. F., & Fontaine, J. (1993) : "Au-delà du capitalisme: la métamorphose de cette fin de siècle". – Dunod, Paris 1993

[Dumez 2010] Dumez Ch. : " Approche dirigée par les modèles pour la spécification, la vérification formelle et la mise en œuvre de services Web composés". PhD thesis, UTBM, Août 2010.

[Evans 2003] Evens E. : «Domain-Driven Design: Tackling Complexity in the Heart of Software»

[ForeSight 2020] ForeSight 2020 – Economic, industry and corporate trends (http://www.polia-consulting.com/Foresight-2020.html, p. 93 et p. 87

[Fuxman 2001] Fuxman, A. D. : "Formal analysis of early requirements specifications", Ph.D. thesis, 2001.

[Gandon 2008] Gandon, F., 2008. : "Graphes RDF et leur Manipulation pour la Gestion de Connaissances" (Habilitation à Diriger des Recherches).

[Giddens 1984] Giddens A. The constitution of society : Outline of the theory of structuration, Cambridge, Polity Press, 1984

[Grundstein 2004] Grundstein, M. (2004). : "De la capitalisation des connaissances au management des connaissances dans l'entreprise", les fondamentaux du knowledge management. 25-54.

[Grundstein et al 2003] Grundstein, M., Rosenthal-Sabroux, C., & Pachulski, A. (2003). "Reinforcing decision aid by capitalizing on company's knowledge: Future prospect". European Journal of Operational Research, 145(2), 256–272.

[Grüber 1995] Grüber, T.R., 1995. "Toward principles for the design of ontologies used for knowledge sharing", International Journal of Human-Computer Studies 43, 907–928.

[Guarino 1998] Guarino, N., 1998 : "Formal Ontology and Information Systems", in: Proceedings of FOIS'98. IOS Press, Trento, Italy, pp. 3–15.

[Hadj Kacem 2008] Hadj Kacem M., : "Modélisation des architectures distribuées à architecture dynamique : Conception et Validation", Doctorat de l'Université de Toulouse et de l'Université de Sfax, soutenue le 13 novembre 2008.

[Hirshein et al 1995] R. Hirschein, H. Klein, K. Lyytinen : «Information systems development and data modeling: conceptual and philosophical foundation» Cambridge University Press, 1995

[Kniberg 2014] Henrik Kniberg : «Spotify engineering culture» https://engineering.atspotify.com/2014/03/27/spotify-engineering-culture-part-1/

[Lalouette 2014] Lalouette, C. (2013). " Gestion des connaissances et fiabilité organisationnelle : état de l'art et illustration dans l'aéronautique ". Les Cahiers De La Sécurité Industrielle.

[Lamsweerde 2000] Lamsweerde, A. : "Formal specification : a roadmap". In : Proceedings of the Conference on the Future of Software Engineering ACM (event), 2000, p. 147–159.

[Le Goaer 2009] Le Goaer O. : "Styles d'évolution dans les architectures logicielles", thèse de doctorat de l'université de Nantes, soutenue le 9 octobre 2009.

[Le Moigne 1997] Jean-Louis Le Moigne : "La théorie du système général: théorie de la modélisation" 3ème édition, 1990 éd. s.l.:Presses Universitaires de France.

[Longépé 09] Longépé C. : " Le projet d'urbanisation du SI- Cas concret d'architecture d'entreprise" - Dunod 2009 (4ème édition)

[Longépé 2001] Longépé C. : " Le projet d'urbanisation du SI", Dunod, Paris 2001

[Luckham 1996] David C Luckham. : "Rapide : A language and toolset for simulation of distributed systems by partial orderings of events". IEEE Transactions on Software Engineering, 1996.

[Magee et al 1993] Magee J., Dulay N., and Kramer J. : "Structuring parallel and distributed programs". IEEE Software Engineering Journal, 8(2) :73–82, Mars 1993.

[Magee et al 1994] Magee J., Dulay N., and Kramer J. : "A constructive development environment for parallel and distributed programs". In IWCCS'94 : Proceedings of the IEEE Workshop on Configurable Distributed Systems, North Falmouth, Massachusetts, USA, Mars 1994.

[Marina 2004] Marian S. : «Les Applications de la Théorie de la Structuration aux Technologies de l'information et de la Communication» pp. 24-25 DESSID | Lyon1/ENSSIB | Recherche Bibliographique | 2004

[Medvidovic et Taylor 2000] Medvidovic N., Taylor R. : "A classification and Comparaison Framework for Software Architecture Description Languagess" IEEE Transactions on Software Engineering, Vol. 26, N°1, Junuary 2000.

[Mesli 2017] Mesli-Kesraoui S. : "Intégration des techniques de vérification formelle dans une approche de conception des systèmes de contrôle-commande : application aux architectures SCADA", thèse de doctorat de l'université Bretagne –Loire, soutenue le 11 Mai 2017.

[Mizoguchi et Bourdeau 2000] Mizoguchi, R., Bourdeau, J., 2000 : "Using Ontological Engineering to Overcome Common AI-ED Problems". Journal of Artificial Intelligence and Education 107–121.

[Morley et al 2005] Morley C., Hugues J., Leblanc B., Hugues O. : "Processus Métiers et systèmes d'information: Evaluation, modélisation, mise en oeuvre". s.l.:Dunod.

[Nonaka et Takeuchi 1995] Nonaka, I. et Takeuchi, H. (1995) : "The Knowledge-Creating Company". New York : Oxford University Press.

[Object Management Group 2008] Object Management Group : "Business process modeling notation", v1.1 omg available specification, January 2008.

[Papazoglou et Heuvel 2000] Papazoglou, M. P. & Heuvel, W., 2000 : "Configurable Business Objects for Building Evolving Enterprise Models and Applications". Dans: Berlin: J.D. W.M.P. van der Aalst, & A. Oberweis (Ed.), pp. 328-344.

[Paulin et Suneson 2015] Paulin, D., & Suneson, K. (2015). : "Knowledge transfer, knowledge sharing and knowledge barriers–three blurry terms in KM". Leading Issues in Knowledge Management, 2(2), 73.

[Prax 2007] Prax, J. Y. (2007) : "Le Manuel du Knowledge Management – mettre en réseau les hommes et les savoirs pour créer de la valeur" (2ème édition). Paris : Dunod.

[Reix et al 2016] Reix R. , Fallery B. , Kalika M., Rowe F. : «Systèmes d'information et management» pp .320

[Reix et Rowe 2002] Robert Reix et Frantz Rowe : «Faire de la recherche en systèmes d'information» s.l.:Paris: Editions Vuibert

[Rosenthal et Grunddstein 2009] Rosenthal-Sabroux C., Grundstein M. : "Management et gouvernance des SI", chapitre 5, p. 85-127, Collection Informatique, Hermes Science Publications, 2009.

[Rudd 2016] Charlie Rudd : «The Third Wave of Agile» https://www.solutionsiq.com/resource/white-paper/the-third-wave-of-agile-2/

[Sarasvathy 2008] Saras D. Sarasvathy : «Effectuation: Elements of Entrepreneurial Expertise»

[Scheer et Nüttgens 2000] Scheer, A. W. & Nüttgens, M., 2000 : "ARIS Architecture and Reference Models for Business Process Management". Berlin: J.D. W.M.P. van der Aalst, & A. Oberweis.

[Smith 2012] Smith G. : "The Object-Z specification language". Volume 1. Springer Science & Business Media, 2012

[Smith 2001] Smith, E. A. (2001) : "The role of tacit and explicit knowledge in the workplace". Journal of Knowledge Management, *5*(4), 311-321.

[Studer et al 1998] Studer, R., Benjamins, V.R., Fensel, D., 1998 : "Knowledge engineering: Principles and methods ". Data & Knowledge Engineering 25, 161–197.

[Theocharis et al 2015] Theocharis G., Kuhrmann M., Münch J., Diebold P. : «Is Water-Scrum-Fall Reality? On the Use of Agile and Traditional Development Practices»

[Thomas 2014] Dave Thomas : «Agile is Dead (Long Live Agility)» https://pragdave.me/blog/2014/03/04/time-to-kill-agile.html

[Tsuchiya 1993] Tsuchiya, S. (1993) : "Improving knowledge creation ability through organizational learning". ISMICK'93 Proceedings (pp. 87-95). International Symposium on the Management of Industrial and Corporate Knowledge.

[Uschold et Gruninger 1996] Uschold, M., Gruninger, M., 1996. : "Ontologies: Principles, methods and applications". Knowledge Engineering Review 11, 93–136.

[Van Heijst et al 1997] Van Heijst, G., Schreiber, A., Wielinga, B., 1997. : "Using explicit ontologies in KBS development". International Journal of Human-Computer Studies 46, 183–292.

[Vandecsteele 2012] Vandecasteele A. : " Modélisation ontologique des connaissances expertes pour l'analyse de comportements à risque : application à la surveillance maritime ", thèse de Doctorat ParisTexch – Ecole Nationale Supérieure des Mines de Paris, soutenue le 30 octobre 2012.

[Wang et al 2014] Wang, S., Noe, R. A., & Wang, Z. M. (2014) : "Motivating knowledge sharing in knowledge management systems a quasi–field experiment". Journal of Management, 40(4), 978-1009.

[Wegmann et al 2002] Wegmann, A. et al., 2002 : "Business and IT Alignment with SEAM for Enterprise Architecture". s.l., Proc. of the 11th IEEE International - EDOC, 111-121.

[White 2004] White S. A. : "Process modeling notations and workflow patterns". On BPMN website, 2004.

[Wieringa et al 2003] Wieringa, R. J., Blanken, H. M. & Fokkinga, M. M., 2003 : "Aligning application architecture to the business context". Klagenfurt/Velden, Austria, Conference on Advanced Information System Engineering - 209-225.