Introduction

Un système d'information est un ensemble constitué d'éléments unis par des relations, ces éléments et ces relations étant munis de propriétés. La conception d'un système d'information n'est pas évidente car il faut réfléchir à l'ensemble de l'organisation que l'on doit mettre en place pour fidéliser les besoins du client. Une méthode d'analyse et de conception a donc pour objectif de permettre de formaliser les étapes préliminaires du développement d'un système d'information afin de rendre ce développement plus fidèle aux besoins du client. Il existe plusieurs méthodes d'analyse dont MERISE la méthode la plus utilisée en France et celle UML la méthode américaine la plus répandue internationalement.
MERISE (Méthode d'Etude et de Réalisation Informatique pour les Systèmes d'Entreprise) est une méthode d'analyse et de réalisation des systèmes d'information qui est élaborée en plusieurs étapes: schéma directeur, étude préalable, étude détaillée et la réalisation.
Alors que UML (Unified Modeling Langage) est plutôt un langage de modélisation des systèmes standard (et non pas vraiment une méthode), qui utilise des diagrammes pour représenter chaque aspect d'un système c'est-a-dire statique, dynamique,....en s'appuyant sur la notion d'orienté objet qui est un véritable atout pour ce langage.
Mais toutefois leur but étant d'arriver à concevoir un système d'information.
Le projet d'analyse et de conception des systèmes d'information est cependant géré par différents types de méthodes dites méthodes Agiles.
Les méthodes Agiles sont un style de méthode de développement logiciel itératif centré sur les personnes et qui met l'accent sur la satisfaction du client à travers l'intégration continue d'un logiciel entièrement fonctionnel. Elles visent la satisfaction réelle du besoin du client et non les termes d'un contrat de développement. Parmi ces méthodes on trouve la toute première appelée RAD (Rapid Applications Development) créée en 1991, Scrum créée en 1996, XP (Extreme Programming) créée en 1999, ASD (Adaptiv Software Development) créée en 2000…
De toutes les méthodologies Agile, Scrum est unique parce qu'elle introduit le concept de "contrôle empirique des processus". Ce qui signifie que Scrum utilise le progrès réel d'un projet plutôt qu'une estimation ou une prévision mal informée afin de planifier les livrables. Scrum est un processus léger permettant de gérer et de contrôler le développement de produits logiciels. Mettant en œuvre des pratiques itératives et incrémentales, Scrum est, en soi, une méthode efficace. Elle peut néanmoins être associée à d'autres pratiques d'ingénierie, notamment Extreme Programming et d'autres méthodes de développement logiciel itératif. C'est l'une des méthodes agiles les plus employées, car elle permet d'améliorer notablement la productivité tout en accélérant le retour sur investissement.
L'objectif de cette étude est de présenter et de comparer les méthodes d'analyse et de conception des systèmes d'information UML et MERISE ainsi que la méthode Agile de gestion des projets de développement informatique SCRUM.

I. MERISE

1- Définition

Merise est une méthode d'analyse et de conception structurelle du système d'information, très utilisée dans les entreprises françaises. Elle est basée sur la séparation des données et des traitements à effectuer en plusieurs modèles conceptuels et physiques. Son but est juste d'arriver à concevoir un système d'information.

2- Historique

Au début des années 70, les bases de données commencent à se développer. Aux Etats-Unis, Codd propose le formalisme relationnel.
En 1973 le Professeur Jean-Louis Le Moigne invente la notion de système d'information.
En 1974, le CETE d'Aix en Provence et l'Université d'Aix-Marseille s'associent pour présenter un projet de recherche auprès de l'INRIA intitulé : « Méthode, modèles et outils pour la conception de la base de données d'un système d'information ». L'équipe, placée sous la direction scientifique du Professeur Jean-Louis Le Moigne est pilotée par Hubert Tardieu.
Dès 1977, la Mission informatique du Ministère de l'Industrie souhaite établir une méthode nationale dans le domaine de la conception des systèmes d'information.
Merise voit officiellement le jour en 1979, sous la forme d'un premier fascicule publié par Ministère de l'Industrie: « Méthode de définition d'un système d'information ». Le nom de Merise a été trouvé comme la métaphore du merisier qui doit être greffé pour porter des fruits.
Le projet Merise se poursuit donc jusqu'en début 1981 avec la publication de plusieurs documents de référence sur la méthode Merise.
A partir de 1981, certaines grandes SSII (société de services en ingénierie informatique) qui avaient accompagné Merise entament la diffusion de la méthode auprès des grandes entreprises et de l'Administration.
Et en 1983, est publié le premier ouvrage sur Merise, ouvrage qui restera la référence. « La méthode Merise – Tome I : Principes et outils » de H. Tardieu, A. Rochfeld et R. Coletti, qui sera suivi en 85 par : « La méthode Merise – Tome II : Démarches et pratiques » de H. Tardieu, A. Rochfeld, R. Coletti, G. Panet et G. Vahee.
Dès lors, Merise connaît un engouement. De nombreux ouvrages paraissent. Merise est désormais enseignée dans les formations universitaires.
La fin des années 80 dénombrera plus de 15 outils français sur Merise. Quasiment toute grande SSII propose le sien.

3- Etapes et Niveaux

Merise définit une démarche d'analyse par étapes réalisées simultanément :

i- Schéma directeur

Un projet doit s'inscrire dans les objectifs généraux de l'entreprise car il mobilise généralement du personnel pendant une grande période de temps. C'est la raison pour laquelle il est nécessaire pour une organisation, avant même de se lancer dans des projets, de définir ses intentions à moyen terme (un à trois ans). Ce sont ces plans d'actions à moyen terme qui sont appelés schéma directeur.

ii- Étude préalable

Cette étape préliminaire est primordiale pour l'ensemble du projet. Il est prévu qu'elle soit confiée à celui qui est en relation commerciale avec le client et qui aura la charge de la coordination des différents intervenants. Le contact commercial et l'auditeur peuvent néanmoins être deux entités séparées. L'étude préalable a pour but principal de planifier l'opération et de préparer les interventions suivantes. Les éventuelles difficultés techniques devront être aussi intégrées.

iii- Etude détaillée

L'étude détaillée vise l'exhaustivité de la description de la solution. Les spécifications détaillées sont la traduction précise des besoins fonctionnels, techniques et organisationnels exprimés par la maîtrise d'ouvrage ou les utilisateurs, en termes de :

- traitements à proposer aux utilisateurs de l'application,
- données à gérer par l'application,
- interface utilisateurs : écrans, états, enchaînement d'écrans,
- contraintes de sécurité et techniques.

Une fois le dossier d'étude détaillée validé par la maîtrise d'ouvrage, il constitue le cahier des charges pour la production ou la maintenance d'une application. En cas de modifications ou d'évolutions du cahier des charges durant la phase d'élaboration de l'étude détaillée, il est important de veiller à ce qu'il soit maintenu à jour.

iv- Etude technique

L'étude technique est la phase d'adaptation de la conception à l'architecture technique retenue, tout en décrivant et documentant le fonctionnement de chaque unité du logiciel. Son livrable est un dossier d'étude technique.
L'élaboration d'un dossier d'étude technique débute lorsque la conception a été validée par le comité de pilotage (dossier d'étude détaillée, plan de conduite du changement, protocole de réception externe sur sites pilotes, s'il y a lieu). Les objectifs d'un dossier d'étude technique sont de :

- Fournir une description complète et détaillée du logiciel dans l'environnement technique cible de l'application : découpage en composants logiciels, structures des fichiers, des bases de données, des interfaces…

- S'assurer des performances prévisibles et de la sécurité de l'application.

v- Réalisation du logiciel

Il s'agit de la phase opérationnelle de création de du système d'information. Elle est menée par la maîtrise d'œuvre, en relation avec la maîtrise d'ouvrage. Cette phase commence par la réception du cahier des charges fonctionnel et technique et se clôture par l'achèvement du système d'information.

vi- Mise en service

Il s'agit de la mise en production de l'ouvrage, c'est-à-dire s'assurer que l'ouvrage est conforme aux attentes des utilisateurs et faire en sorte que son " installation " et son utilisation se déroule correctement. Dans la mesure où la maîtrise d'œuvre connaît le produit qu'elle a mis au point, il lui revient de l'installer. Cette phase aboutit à un manuel utilisateur.

vii- Maintenance

Elle prend 3 formes qui sont :
La maintenance corrective permet de corriger les erreurs qui n'ont pas été détectées lors des précédentes phases de tests.
La maintenance adaptative doit s'occuper de faire évoluer et d'adapter le logiciel à l'apparition de nouvelles contraintes.
La maintenance perfective a pour objectif l'optimisation des performances du logiciel.

Merise propose ensuite plusieurs niveaux d'abstraction afin de décrire progressivement la complexité :

Niveau conceptuel. A ce niveau 2 modèles sont décrits :
Le modèle conceptuel de données ou MCD est un schéma qui représente la structure du système d'information, du point de vue des données, c'est-à-dire qu'il décrit les objets, les propriétés et les associations entre les objets.
Le Modèle conceptuel des traitements ou MCT permet de traiter la dynamique du système d'information, c'est-à-dire les opérations qui sont réalisées en fonction d'événements. (exemple : Inscription d'un étudiant).

Niveau logique et organisationnel. A ce niveau aussi 2 modèles sont décrits :
Le modèle logique de données ou MLD évolue le plus souvent vers la description d'une structure relationnelle et précise en outre les clés d'accès aux occurrences d'un objet, les relations, les opérateurs (union, jointure) et introduit la notion de table. A ce stade, il est possible de connaître la liste exhaustive des tables qui seront à créer dans une base de données relationnelle.
Le Modèle Logique des Traitements ou MLT précise les acteurs et les moyens qui seront mis en œuvre.

Niveau physique. A ce niveau aussi 2 modèles sont décrits :
Le modèle physique des données ou MPD décrit l'implémentation physique du modèle (structure de la base de données ou des fichiers de données).
le Modèle Opérationnel des Traitements ou MOT permet de spécifier les fonctions telles qu'elles seront ensuite réalisées par le programmeur.

4- Les points forts et faibles de MERISE

i- Les points forts

Merise, méthode la plus utilisée en France dans les domaines de gestion, s'appuie sur une approche systémique. Donc elle permet une appréhension globale et rapide du système d'information à concevoir. Elle est très adaptée à un contexte de création d'application et ses concepts sont peu nombreux et simples pour l'étude du système et elle est assez indépendante vis à vis de la technologie. Néanmoins, certains points faibles ont été notés sur cette méthode lors de l'analyse et de la conception des systèmes d'information.

ii- Les points faibles

Elle ne s'occupe pas de l'interface utilisateur et n'est pas adaptée à un problème de maintenance ou de seconde informatisation. Elle ne permet pas réellement une validation

rapide de la part des utilisateurs et par la suite il est très difficile de valider les traitements par rapport aux données et cela au niveau conceptuel ou organisationnel.

II. UML

1- Définition

Unified Modeling Langage ou encore UML est langage d'analyse et de conception orienté objet défini par l'OMG (Object Management Group) et aussi considéré comme une méthode permettant la mise en schéma syntaxé et structuré de la relation entre divers systèmes (client-serveur, par exemple) dans un cahier des charges et cela permet de découper les différentes parties à coder.

2- Historique

UML est né (1997) de la fusion de 3 langages : OMT : par Rumbaugh (Rational Software), OOD : par Booch (General Electric) et OOSE : par Jacobson (Ericsson).

OMT : Object Modeling Techniques a des vues statiques c'est-à-dire ne modifie pas l'objet, des vues dynamiques c'est-à-dire peut modifier l'objet et des vues fonctionnelles.
OOD : Oriented Object Design a des vues logiques c'est-à-dire basée sur de la décomposition logicielle et des vues physiques basée sur de la décomposition matérielle.
OOSE : Oriented Object Software Engineering a des vues basées sur le cycle vie logiciel (Analyse, Conception, Réalisation/Implémentation, Test/Maintenance).

3- Vues et Diagrammes

UML est un langage cadré sur les analyses d'objets. Il permet de représenter un système selon différentes vues :

- Vue des cas d'utilisation constitue le lien et motive la création de tous les diagrammes d'architecture et correspond aux besoins attendus par chaque acteur (c'est le QUOI et le QUI).
- Vue logique constitue la principale description architecturale d'un système informatique et explique comment peuvent être satisfaits les besoins des acteurs (c'est le COMMENT).
- Vue d'implémentation définit les dépendances entre les modules.
- Vue des processus met en œuvre les notions de tâches concurrentes, stimuli, contrôle, synchronisation, etc.
- Vue de déploiement décrit la position géographique et l'architecture physique de chaque élément du système (c'est le OÙ).

UML comporte différents diagrammes dépendants et hiérarchiques. Ils se complètent de façon à permettre la modélisation d'un projet tout au long de son cycle de vie :

- Diagrammes statiques. Il s'agit de :
    - Le diagramme de classes est utilisé pour représenter l'organisation des données dans les Systèmes d'Information. Mais il ne peut pas mettre en évidence les relations existant à un instant déterminé de la vie du système, entre les diverses instances des classes composant le système. Il ne peut montrer qu'une instance particulière (ayant des valeurs d'attributs bien définies) est nécessaire dans un cas d'utilisation particulier. C'est là le rôle du
    - Diagramme d'objets

- Le diagramme de composants montre les dépendances entre les composants de l'application. Un composant peut être un exécutable, un code source, fichier, bases de données etc. …
- Le diagramme de déploiement est une vue statique qui montre la disposition physique des matériels qui composent le système et la répartition des composants sur ces matériels.
- Le diagramme des paquetages permet de mettre en évidence les dépendances entre paquetages. Un paquetage étant un conteneur logique permettant de regrouper et d'organiser les éléments dans le modèle UML.
- Le diagramme de structure composite est un nouveau diagramme introduit par le standard UML2.0. il joue le même rôle qu'un diagramme de classes mais permet de décrire les relations entre composants d'une classe.

- Diagrammes comportementaux. Il s'agit de :

  - Le diagramme de cas d'utilisation représente les cas d'utilisation, les acteurs et les relations entre les cas d'utilisation et les acteurs, décrit sous la forme d'actions et de réactions, le comportement d'un système du point de vue d'un utilisateur et permet de définir les limites du système et les relations entre un système et l'environnement. Et un cas d'utilisation est une manière spécifique d'utiliser un système. Il est l'image d'une fonctionnalité du système, déclenchée en réponse à la stimulation d'un acteur externe.
  - Le diagramme états-transitions décrit sous forme de machine à états finis le comportement dynamique des objets dans le temps en modélisant les cycles de vie des objets de chaque classe.
  - Le diagramme d'activité est utilisé pour modéliser un workflow (flux d'informations au sein de compagnie) dans un cas d'utilisation ou entre plusieurs cas d'utilisations et pour spécifier une opération (décrire la logique d'une opération).

- Diagrammes d'interaction ou dynamiques. Il s'agit de :

  - Le diagramme de collaboration montre des interactions entre objets (instances de classes et acteurs). Ils permettent de représenter le contexte d'une interaction, car on peut y préciser les états des objets qui interagissent.
  - Le diagramme de séquences permet de représenter des collaborations entre objets selon un point de vue temporel, on y met l'accent sur la chronologie des envois de messages et l'expression des interactions. Contrairement au diagramme de collaboration, on n'y décrit pas le contexte ou l'état des objets, la représentation se concentre sur l'expression des interactions.
  - Le diagramme de communication est un nouveau diagramme introduit par le standard UML2.0. C'est une représentation simplifiée d'un diagramme de séquence qui met l'accent sur l'organisation structurelle des objets qui envoient et reçoivent des messages.
  - Le diagramme d'interaction permet d'établir un lien entre les diagrammes de cas d'utilisation et les diagrammes de classes : ils montrent comment des objets (i.e. des instances de classes) communiquent pour réaliser une certaine fonctionnalité. Ils apportent ainsi un aspect dynamique à la modélisation du système.
  - Le diagramme global d'interaction est aussi un nouveau diagramme introduit par le standard UML2.0 qui permet de décrire les enchaînements possibles entre les scénarios préalablement identifiés sous forme de diagrammes de séquences. C'est une variante du diagramme d'activité.

- Le diagramme de temps est aussi un nouveau diagramme introduit par le standard UML2.0 qui permet de décrire les variations d'une donnée au cours du temps.

4- Les points forts et faibles de UML

i- Les points forts

- UML est un langage formel et normalisé
  - gain de précision
  - gage de stabilité
  - encourage l'utilisation d'outils
- UML est un support de communication performant
  - Il cadre l'analyse.
  - Il facilite la compréhension de représentations abstraites complexes.
  - Son caractère polyvalent et sa souplesse en font un langage universel.

ii- Les points faibles

La mise en pratique d'UML nécessite un apprentissage et passe par une période d'adaptation. UML n'est pas à l'origine des concepts objets, mais en constitue une étape majeure, car il unifie les différentes approches et en donne une définition plus formelle.

Le processus (non couvert par UML) est une autre clé de la réussite d'un projet. Or, l'intégration d'UML dans un processus n'est pas triviale et améliorer un processus est une tâche complexe et longue. Les auteurs d'UML sont tout à fait conscients de l'importance du processus, mais l'acceptabilité industrielle de la modélisation objet passe d'abord par la disponibilité d'un langage d'analyse objet performant et standard.

III. SCRUM

1- Définition

Scrum est une méthode agile pour la gestion de projets basée sur des multiples petites équipes travaillant intensément et indépendamment. Scrum maximise sa productivité facile à adapter à son cycle de développement et nécessite la saisie et le suivi d'une liste de fonctionnalités pour l'ensemble du projet et d'une liste de tâches pour chaque itération. Elle demande également la production de tableaux de bords (c'est-à-dire rapports graphiques). Bien sûr, un simple fichier Excel peut suffire, mais l'accès partagé à un fichier Excel par toute l'équipe, en écriture et en consultation, est toujours problématique.

2- Historique

Le terme Scrum est emprunté au rugby et signifie mêlée. Ce processus s'articule en effet autour d'une équipe soudée, qui cherche à atteindre un but, comme c'est le cas en rugby pour avancer avec le ballon pendant une mêlée.
En 1986, Hirotaka Takeuchi et Ikujiro Nonaka décrivent une nouvelle approche holistique qui augmenterait la vitesse et la flexibilité dans le développement commercial de nouveaux produits. Dans celle-ci les phases se chevauchent fortement et l'ensemble du processus est réalisé par une équipe aux compétences croisées à travers différentes phases. Ils ont comparé cette nouvelle approche au rugby, où l'équipe essaye d'avancer unie, en faisant circuler la balle.
En 1991, DeGrace et Stahl font référence à cette approche comme Scrum (mêlée, en anglais), un terme de rugby mentionné dans l'article de Takeuchi et Nonaka. Dans le début des années 1990, Ken Schwaber a utilisé une approche qui a conduit à Scrum au sein de son entreprise. En même temps, Jeff Sutherland, John Scumniotales et Jeff McKenna développent une approche similaire à Easel Corporation et sont les premiers à appeler cela.
En 1995, Sutherland et Schwaber ont présenté conjointement un document décrivant Scrum à l'OOPSLA de 1995 à Austin. Schwaber et Sutherland ont collaboré au cours des années suivantes pour fusionner les publications, leurs expériences et les meilleures pratiques du secteur en ce qui est maintenant connu comme Scrum. En 2001, Schwaber fait équipe avec Mike Beedle pour décrire la méthode dans le livre "Agile Software Development With Scrum". En France, le premier livre français entièrement consacré à Scrum est publié en février 2010 : "SCRUM : Le guide pratique de la méthode agile la plus populaire".

3- Méthodologie Scrum et Cycle de développement

Scrum utilise le progrès réel d'un projet et les projets sont divisés en parties succinctes de travail, appelées "sprints", qui sont généralement d'une durée de 1 à 4 semaines. À la fin de chaque sprint, les membres de l'équipe ainsi que les autres gens impliqués se rencontrent afin d'évaluer les progrès du projet et planifier les prochaines étapes. Ceci permet d'ajuster ou réorienter un projet selon l'évolution du travail et non des spéculations ou des prédictions.
Chaque Sprint possède un but à atteindre, défini par le Directeur de produit, à partir duquel sont choisies les fonctionnalités à implémenter dans ce sprint, et dure au plus quatre semaines.
Un sprint aboutit toujours sur la livraison d'un produit partiel fonctionnel. Pendant ce temps, le Scrum Master a la charge de réduire au maximum les perturbations extérieures et de résoudre les problèmes non techniques de l'équipe.
Pendant un Sprint, des réunions quotidiennes de moins de 15 minutes (appelées Scrum)

permettent à toute l'équipe de faire le point sur le travail accompli par chacun depuis la dernière réunion Scrum, les obstacles rencontrés, et le travail prévu d'ici la prochaine réunion.
La méthodologie Scrum Agile apporte des rôles, responsabilités et réunions qui assurent le bon déroulement d'un projet.

i- Rôles

- Le propriétaire: Communique la vision du produit aux membres de l'équipe. Représente les intérêts du client via les besoins et la priorisation de ceux-ci. Possède la plus haute autorité des 3 rôles et aussi le plus de responsabilité. Lorsqu'un projet tourne mal, c'est vers lui que l'on pointe.
- Le Scrum Master: Lien entre l'équipe et le propriétaire. Ne gère pas l'équipe mais fait tout en sorte pour enlever les recueilles sur la route de l'équipe de développement afin d'atteindre les buts du sprint. Aide l'équipe à demeurer productive et montre l'avancement au propriétaire. Indique aussi au propriétaire les façons d'optimiser le retour sur investissement pour l'équipe.
- Les membres de l'équipe: Responsables de compléter le travail. L'équipe idéale consiste de 7 membres provenant de disciplines différentes et plus ou moins 2 autre individus. Pour un projet d'application une équipe typique sera composée d'un amalgame d'ingénieurs, architectes, programmeurs, analystes, experts AQ, testeurs et designers d'UI. À chaque sprint, l'équipe est responsable de déterminer comment est accomplira le travail. Ceci offre une grande autonomie mais aussi la responsabilité d'atteindre ces buts pour le sprint.

ii- Réunions

- Réunion de planification du sprint: Durant cette réunion l'équipe et le propriétaire discutent de priorisation et des items en attente de développement. Les membres de l'équipe s'entendent sur le travail à faire et se crée une liste des tâches à effectuer durant le sprint.
- Scrum journalier: À chaque jour durant le sprint, l'équipe se réunie avec le Scrum Master et le propriétaire. Cette réunion est limitée à une durée de 15 minutes. Durant cette réunion, l'équipe discute de ce qui à été fait la journée précédente, ce qui fera aujourd'hui et ce qui pourrait les empêcher d'avancer. Les Scrums journalier permettent à l'équipe de se synchroniser.
- Revue du sprint (à la fin du sprint): Durant cette réunion l'équipe discute des ajouts faits dans le projet durant le sprint. Le but de cette réunion est de colliger les commentaires du propriétaire et toute autre personne impliquée dans l'évaluation ou la revue du produit. Ces commentaires pourraient mener à la modification de nouvelles fonctionnalités fraichement ajouter. Mais il se pourrait que de nouveaux items soient ajoutés/modifiés à la liste des tâches en attentes.
- Rétrospective du sprint (à la fin du sprint): Tous y participent. Permet de revoir le déroulement du sprint et d'apporter des améliorations pour les prochains.

iii- Livrables

- Carnet de commandes du produit: une liste constamment priorisée d'items.
- Carnet de commandes du sprint: une liste des items les plus hautement priorisés du carnet de commandes du produit.

4- Les points forts et faibles de SCRUM

L'avantage des Méthodes Agiles réside avant tout dans le fait qu'elles mettent en avant une démarche plus pragmatique que les démarches traditionnelles, en impliquant au maximum le client, et en privilégiant la réalisation d'un produit véritablement opérationnel, et tout cela à moindre coût.
Répondre au changement plutôt que suivre un plan préétabli, prendre en compte le fait que les conditions et les objectifs d'une entreprise puissent évoluer avec le temps, même pendant que se déroule le projet, correspond tout à fait à la vision de l'innovation, inhérente au développement des Méthodes Agiles.
Un principe fort en Scrum est la participation active du client pour définir les priorités dans les fonctionnalités du logiciel, et pour choisir celles qui seront réalisées dans chaque sprint. Il peut à tout moment compléter ou modifier la liste des fonctionnalités à réaliser, mais jamais celles qui sont en cours de réalisation pendant un sprint.

i- Les points forts

- Il est entièrement développé et testé pour de courtes itérations.
- Ses processus de développement sont très simples.
- Ses règles sont définies clairement.
- Augmentation de productivité.
- Organisation personnelle.
- Chaque équipe a son lot de responsabilité.
- Amélioration de la communication.
- Combinaison possible avec XP.

ii- Les points faibles

- Peu, voire pas, de documentation écrite. Le gros problème réside bien souvent dans l'aspect Documentation du projet. Comme la méthode privilégie le fonctionnel, on se retrouve rapidement avec des projets non-documentés.
- La mise en œuvre du développement n'est pas précisée, seule compte la gestion des ressources humaines.
- L'équipe ne se prête pas au SCRUM. Il est clair que si l'équipe ne prend pas le temps de remplir correctement les indicateurs comme le SCRUM Board et les Burndow Chart, le suivi n'est pas correctement assuré.
- D'autres parts on remarque bien souvent que dans la gestion du SCRUM les manager sont tentés de passer outre les responsabilités hiérarchiques pour aller remonter le problème au sommet. On constate bien souvent ce problème dans des équipes ou un développeur requiert directement le client plutôt que son propre SCRUM master, ou Directeur de Produit.

IV. Comparaison MERISE/UML/SCRUM

1- Approche fonctionnelle

Merise propose une approche descendante où le système réel est décomposé en activités, elles-mêmes déclinées en fonctions.
Les fonctions sont composées de règles de gestion, elles-mêmes regroupées en opérations. Ces règles de gestion au niveau conceptuel génèrent des modules décomposés en modules plus simples et ainsi de suite jusqu'à obtenir des modules élémentaires...
Les limites d'une telle approche résident dans le fait que les modules sont difficilement extensibles et exploitables pour de nouveaux systèmes.

Dans UML les fonctions cèdent la place aux cas d'utilisation qui permettent de situer les besoins de l'utilisateur dans le contexte réel. A chaque scénario correspond des diagrammes d'interaction entre les objets du système et non pas un diagramme de fonction...

Scrum utilise une approche fonctionnelle pour récolter les besoins des utilisateurs. L'objectif est d'établir une liste de fonctionnalités à réaliser, que l'on appelle backlog de produit (NDT : Le terme *backlog* peut être traduit par cahier, liste ou carnet de commandes, qui ne collent pas bien avec l'esprit du terme anglais qui évoque aussi une réserve, un retard accumulé ; aussi ce terme a été gardé tel quel).
À chaque item de backlog sont associés deux attributs : une estimation en points arbitraires (voir Estimation) et une valeur client, qui est définie par le directeur de produit (retour sur investissement par exemple). Ce dernier définit dans quel ordre devront être réalisés ces items. Il peut changer cet ordre en cours de projet et même ajouter, modifier ou supprimer des items dans le backlog.
La somme des points des items du backlog de produit constitue le reste à faire total du projet. Cela permet de produire un release burndown chart, qui montre les points restant à réaliser au fur et à mesure des sprints.

2- Schéma Entité/Association

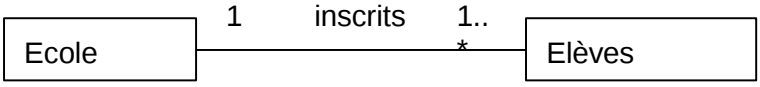
Le schéma entité/association est un diagramme pour des descriptions de haut niveau de modèles conceptuels de données ou MCD.
L'entité est une association d'objets traduisant un réseau de relations de dépendance.
Et une association est définie comme un lien sémantique de même nature reliant les occurrences de plusieurs entités donc elle définit des liens entre des types d'entité. Les associations sont caractérisées par des cardinalités. Le choix des cardinalités est essentiel et est aussi parfois discutable, et constitue donc l'aspect le plus délicat de la modélisation. Les cardinalités sont notées par deux chiffres. Le chiffre de droite est la cardinalité maximale, et celui de gauche est la cardinalité minimale. Mais cette dernière est moins importante que la cardinalités maximales.

- La notation « * » est équivalente à « 0..* ».
- La notation « 1 » est équivalente à « 1..1 ».

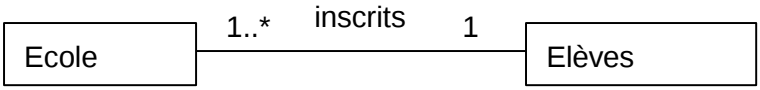
Dans la notation UML, on indique les cardinalités aux deux extrémités d'un lien d'association entre deux entités E1 et E2. Les cardinalités pour E1 sont placées à l'extrémité du lien allant d'E1 vers E2 et les cardinalités pour E2 sont l'extrémité du lien allant d'E1 vers E2. Alors que dans la notation Merise c'est le contraire. Les cardinalités pour E1 sont l'extrémité du

lien allant d'E1 vers E2 et les cardinalités pour E2 sont placées à l'extrémité du lien allant d'E1 vers E2.

- Notation UML



- Notation Merise



Dans les 2 cas, les schémas ci-dessus veulent dire que dans une Ecole sont inscrits un à plusieurs (1..*) Elèves mais les Elèves sont inscrits dans une et une seule Ecole (1..1 ou 1).

3- Méthodologie

Merise est une méthode d'analyse de système d'information qui est élaborée en plusieurs étapes.
Alors que UML est un métalangage (et pas une méthode à proprement parler) de description d'application se basant sur l'hypothèse que le système d'information développé est orienté objet.
Et Scrum est un processus agile permettant de gérer et de contrôler le travail de développement et peut être utilisé pour intégrer des pratiques de génie logiciel existantes.

Ils travaillent tous trois sur des concepts différents. Merise travaille sur un concept relationnel tandis qu'UML travaille sur un concept objet. Et Scrum travaille sur le concept de la qualité de l'environnement de travail de l'équipe. Cela inclut :

- Pas de changements imposés pendant un sprint.
- Toute l'équipe dans une même pièce.
- Un bon outil de suivi du projet
- Prévenir des interventions extérieures (téléphone, irruption dans la pièce, etc)
- Un tableau blanc et/ou en liège

Merise = Franco-français utilisée pour le relationnel
UML = International et méthode américaine utilisée pour la modélisation objet.
Scrum = Méthode agile pour la gestion des projets.
En plus UML est beaucoup plus vaste que son modèle de données. Donc pour l'analyse et la conception des systèmes d'information, Scrum est une méthode à utiliser pour gérer le projet tandis que Merise et UML sont des méthodes à utiliser pour analyser et concevoir le projet.

La méthode Merise ressemble à la méthode UML pour la phase de modélisation de la base de données. La différence principale est que Merise est une méthode d'analyse, et UML un langage de modélisation de données. Par la conception, Scrum aide l'équipe à détecter et à éliminer les obstacles entravant le développement et la livraison des produits et améliore la communication. Le développement de produits logiciels fait naître un chaos considérable qui se traduit par une grande incertitude et des comportements imprévisibles. Donc Scrum optimise la coopération et permet de contrôler le chaos dérivant d'intérêts et de besoins divergents.

4- Cycle de développement

La différence de MERISE, UML ne propose pas de cycle précis : les organisations sont libres de choisir le cycle qui leur convient.
UML fonctionne sur un principe d'itérations qui ne s'oppose pas aux phases définies dans MERISE. MERISE découpe plus au travers de ses phases l'analyse métier et l'architecture logicielle. Dans UML, l'architecture logicielle a une place prépondérante et est intégrée très en amont dans l'élaboration du système d'information.
Dans UML, l'avancement du projet est mesuré par le nombre de cas d'utilisation, de classes... réellement implantées et non par la documentation produite ce qui est le cas dans Merise. Les itérations servent en outre à répartir l'intégration et les tests tout au long du processus d'élaboration du système d'information.

Scrum repose sur une théorie moderne du contrôle des processus et permet d'élaborer de façon systématique les meilleurs logiciels possibles à partir des ressources disponibles, à un niveau de qualité acceptable et dans le respect des délais de livraison. Le cycle de vie Scrum s'articule en brèves itérations de développement appelées « Sprints ». Scrum synchronise étroitement les exigences logicielles avec toute une série de prototypes itératifs. Et ses projets sont contrôlés par la mise en place, l'actualisation et le suivi de paramètres de contrôle essentiels, qui forment l'ossature du processus Scrum.

5- Les acteurs

Les acteurs d'un système sont les entités externes à ce système qui interagissent (saisie de données, réception d'information, …) avec lui. Les acteurs sont donc à l'extérieur du système et dialoguent avec lui. Ces acteurs permettent de cerner l'interface que le système va devoir offrir à son environnement. Oublier des acteurs ou en identifier de faux conduit donc nécessairement à se tromper sur l'interface et donc la définition du système à produire.

Trois acteurs sont essentiels à un projet SCRUM :

- Le product owner représente le client, disponible pour orienter l'équipe, il ordonne le travail en mettant à jour le product backlog.
- Le Scrum master protège de toutes perturbations la dynamique de l'équipe en solutionnant ses problèmes non techniques.
- L'équipe s'autogère sans hiérarchie interne

Il faut faire attention à ne pas confondre acteurs et utilisateurs (utilisateur avec le sens de la personne physique qui va appuyer sur un bouton) d'un système. D'une part parce que les acteurs inclus les utilisateurs humains mais aussi les autres systèmes informatiques ou hardware qui vont communiquer avec le système. D'autre part parce qu'un acteur englobe tout une classe d'utilisateur. Ainsi, plusieurs utilisateurs peuvent avoir le même rôle, et donc correspondre à un même acteur, et une même personne physique peut jouer des rôles différents vis-à-vis du système, et donc correspondre à plusieurs acteurs.
En UML on distingue 2 types d'acteurs :

- Acteurs principaux : Ceux qui vont réaliser le cas d'utilisation (la relation avec le cas d'utilisation est illustrée par le trait liant le cas d'utilisation et l'acteur dans un diagramme de cas d'utilisation).
- Acteurs secondaires : Ceux qui ne font que recevoir des informations à l'issue de la réalisation du cas d'utilisation.

Ils sont représentés dans le diagramme des cas d'utilisation.

En Merise on distingue aussi 2 types d'acteurs :

- Acteurs externes : Eléments externes avec lesquels le système échange des flux d'information. Ex : clients, fournisseurs...
- Acteurs internes : Acteurs faisant partie du système d'information étudié. Ex : guichet, service informatique...

Ils sont représentés dans le MCC (Modèle Conceptuel de Communication) et dans le MCT (Modèle Conceptuel de Traitement).

6- La démarche

UML est un métalangage et n'est pas une méthode donc elle ne présente aucune démarche. Cependant, les auteurs d'UML proposent d'utiliser une démarche pour favoriser la réussite du projet :

- Une démarche itérative et incrémentale. Pour modéliser (comprendre et représenter) un système complexe, il vaut mieux s'y prendre en plusieurs fois, en affinant son analyse par étapes.
- Une démarche guidée par les besoins des utilisateurs du système. Avec UML, ce sont les utilisateurs qui guident la définition des modèles :
  - Le périmètre du système à modéliser est défini par les besoins des utilisateurs (les utilisateurs définissent ce que doit être le système).
  - Le but du système à modéliser est de répondre aux besoins de ses utilisateurs (les utilisateurs sont les clients du système).

  Les besoins des utilisateurs sont indispensables tout au long de la démarche itérative et incrémentale :
  - A chaque itération de la phase d'analyse, on clarifie, affine et valide les besoins des utilisateurs.
  - A chaque itération de la phase de conception et de réalisation, on veille à la prise en compte des besoins des utilisateurs.
  - A chaque itération de la phase de test, on vérifie que les besoins des utilisateurs sont satisfaits.
- Une démarche centrée sur l'architecture. Une architecture adaptée est la clé de voûte du succès d'un développement. Car elle décrit des choix stratégiques qui déterminent en grande partie les qualités du logiciel (adaptabilité, performances, fiabilité...).

Contrairement à UML, Merise qui en propose une. La démarche de développement d'un système d'information de Merise doit être conduite suivant trois axes appelés cycles dont le degré de prise en compte par les différentes méthodes oriente le choix de l'une d'entre elles en fonction des objectifs de l'étude :

- Cycle de vie. Il se situe sur une échelle de temps qui nous mène du point de départ à l'exploitation du système, en passant par sa création, sa maturité et sa maintenance.
- Cycle de décision. Il représente l'ensemble des choix qui doivent être fait durant le déroulement du cycle de vie.
- Cycle d'abstraction. C'est le découpage en ensembles homogènes de préoccupations :
  - Le niveau conceptuel qui détermine les choix de gestion.
  - Le niveau organisationnel qui détermine les choix d'organisation.
  - Le niveau technique qui détermine les contraintes techniques.

La réalisation d'un système d'information est conduite au travers d'un projet décomposé en étapes s'appuyant sur les trois cycles définis précédemment :

- Le schéma directeur.
  - Définition des domaines d'études
  - planification du développement de chaque domaine
  - évaluation des moyens en personnel et matériel

    - mise en œuvre de la méthode.
- L’étude préalable. Détermination du système de l’étude afin de donner aux responsables les moyens de prendre des décisions pertinentes sur la globalité du projet.
- L’étude détaillée. Détermination des spécifications fonctionnelles.
- Réalisation.
    - Etude technique
    - Production des programmes
    - Rédaction des consignes d’utilisation.
- Maintenance.
    - Corrections
    - Adaptation aux évolutions de l’entreprise.

Les trois premières étapes correspondent à la partie conception du cycle de vie et les suivantes concernent la réalisation du système et son lancement. La méthodologie Merise s’intéresse plus particulièrement à cette partie conception.

En Scrum, les projets sont divisés en parties succinctes de travail, appelées "sprints", qui sont généralement d'une durée de 1 à 4 semaines. À la fin de chaque sprint, les membres de l'équipe ainsi que les autres gens impliqués se rencontrent afin d'évaluer les progrès du projet et planifier les prochaines étapes. Ceci permet d'ajuster ou réorienter un projet selon l'évolution du travail et non des spéculations ou des prédictions.

Ci-dessous les étapes à suivre durant un Scrum :

- Création d'un carnet de commandes du produit: Le propriétaire et l'équipe se rencontre afin de prioriser les items du carnet de commandes du produit. Le propriétaire doit être en mesure de formuler sa vision du produit.
- Réunion de planification de sprint.
- Création de la planification du sprint. Séparation et distribution des items du carnet entre les membres de l'équipe selon leur disponibilité.
- Début du sprint pour une durée de 1 à 4 semaines. Aucune autre tâche ne peut être ajoutée.
- Scrum journalier.
- Revue du sprint.
- Redémarrage du cycle.

Conclusion

En définitive, nous avons vu comment modéliser un développement de systemes d'information à l'aide des méthodes telles que Merise et UML bien qu'UML est plus un langage qu'une méthode. Merise et UML ont des caractéristiques voisines au niveau de la modélisation des bases de données mais également des points de divergence.
En effet, la méthode MERISE nécessite une démarche par étape qui favorise la qualité de chaque modèle avec ses différents niveaux de validations. Alors que le langage UML n'impose pas de méthode de travail particulière. Ces deux méthodes peuvent être intégrées à n'importe quel processus de développement logiciel dont Scrum.
Scrum, processus de développement logiciel, introduit dont des règles pour suivre un processus itératif empirique permettant d'obtenir un produit très proche de besoins qui évoluent et ainsi de maximiser la valeur pour les clients. Mais Scrum est un processus idéal pour de petites équipes et tout le monde dans l'équipe surveillée par un Scrum Master se doit non seulement d'interagir avec le projet mais aussi d'être réellement investie dans sa bonne réalisation.