Système d'information et leur développement:

1 Introduction

Aujourd'hui, grâce aux nouvelles technologies, les Systèmes d'information représentent des outils extrêmement puissants en matière de gestion d'entreprise dans tous les métiers. Ils jouent un rôle principal dans l'amélioration de ces performances et sa réactivité en permettant de gérer les différents flux d'information présents dans toute entité.

1.1 Système d'information

Le système d'information (SI) est l'ensemble des méthodes, techniques et outils pour la mise en place et l'exploitation de la technologie informatique nécessaire aux utilisateurs et à la stratégie de l'entreprise.(2)

1.1.1Les objectifs du système d'information

Le système d'information à la particularité d'être une fonction ressource pour l'entreprise dans son ensemble. Dans cette optique systémique, nous pouvons représenter le système d'information par cinq objectifs majeurs au service du système opérationnel, de gestion et décisionnel d'une entreprise, comme l'illustre la figure 1.(2)

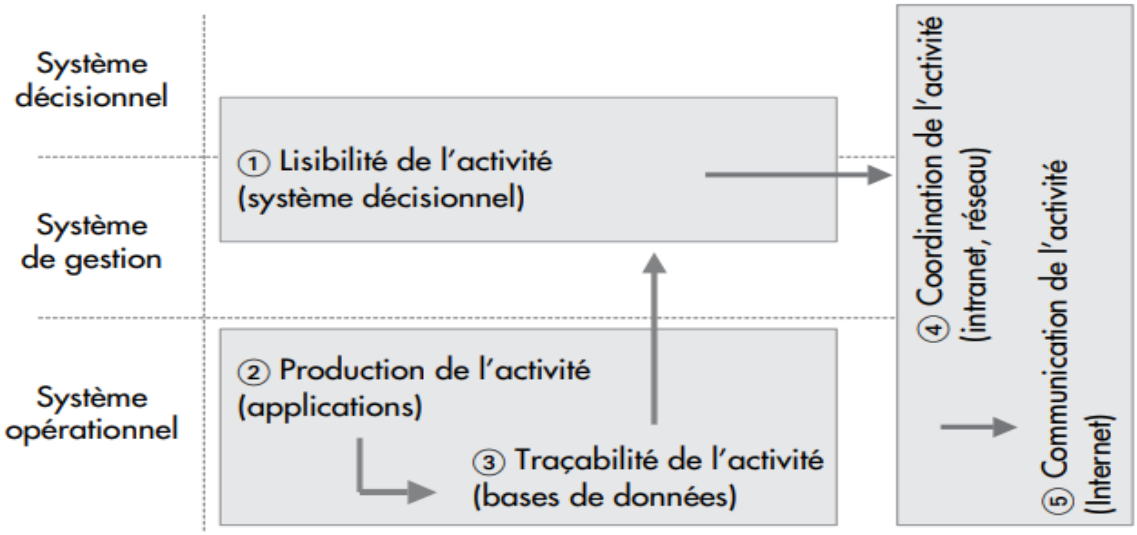

Figure 1-1 : Les cinq objectifs du SI (2)

1.1.2Les principales sources de système information

Données :sous des formes variées (chiffre, texte, images, sons …) ses ressources essentiellement matérialiser l'information détenue par l'organisation. Elles sont la matière première sur laquelle le système d'information agit, elles sont traitées à l'aide de model qui expriment des connaissances, et permet de déduire un résultat ou une action.(3)

Logiciel et procédure : il constitue la description formelle des opérations effectuée(les programmes : les système d'exploitations, traitement des textes, feuille de paie. Les procédures : saisies, correction d'erreur distribution des chèques de paie). (3)

Matériel : le système d'information repose des technologies numériques de l'information (réseaux ordinateur, unité périphériques, station de travail, papier…). (3)

Personne : il y a pas du système d'information sans personne, sans acteur : se sont soit des utilisateurs de système, employés, cadre qui pour la réalisation de leurs taches, utilisent l'information produit par le système et ces possibilités d'automatisation ou qui alimente le système.(3)

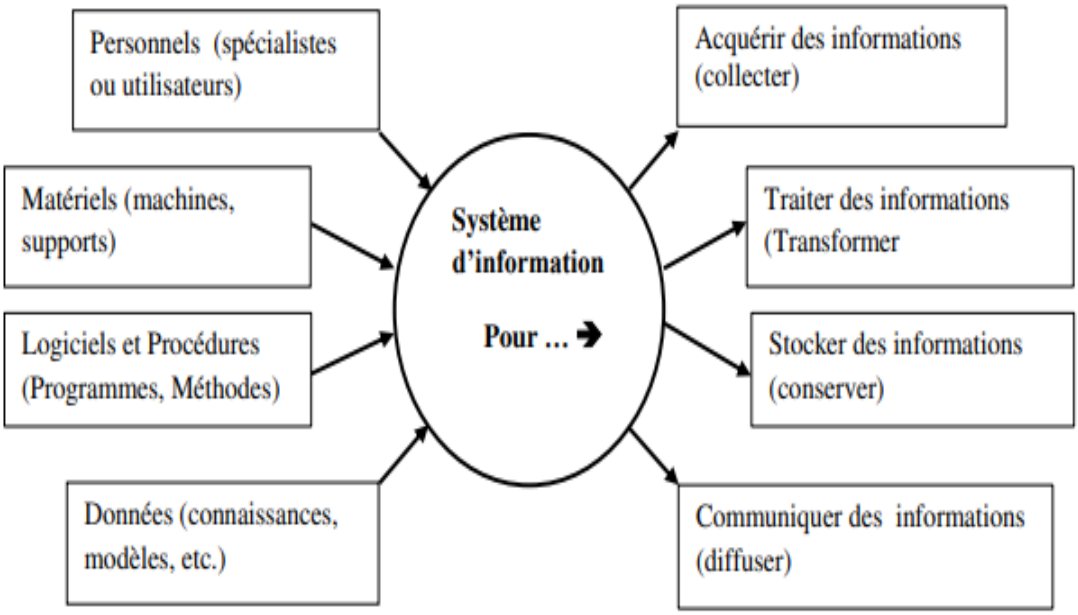

Figure 1-2 : source de système d'information(3)

1.1.3Les fonctions d'un système d'information

Dans un système d'information, il y'a trois activités principales qui aident à la production de l'information à l'organisation : l'entrée, le traitement et la sortie. (4)

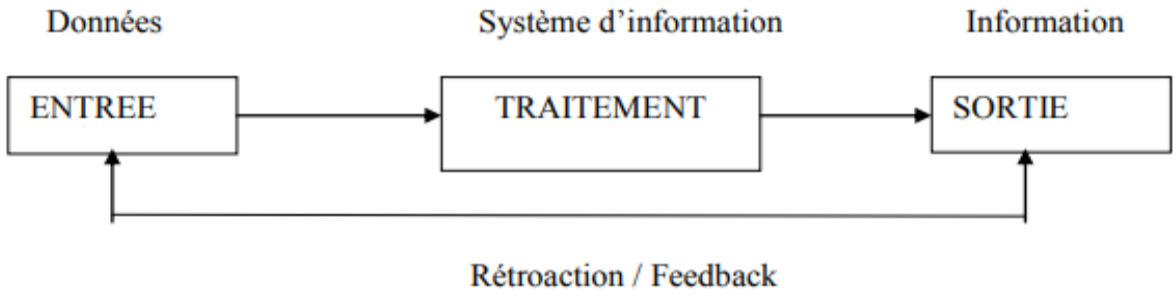

Figure 1-3 : Les fonctions d'un système d'information(4)

- L'Entrée, est le processus au cours duquel les données brutes son fournies au système. Ce processus peut prendre des formes –différenciées.(4)
- Le Traitement, est le processus qui transforme les données brutes pour leur donner un sens. (4)
- La Sortie, est le processus de diffusion de l'information traitée aux utilisateurs qui ont besoin.(4)

Remarque : Un système d'information se fonde également sur la rétroaction ou le feedback, c'est-à-dire sur le processus de transmission des informations de sortie aux utilisateurs appropriées pour les aider à évaluer l'étape antérieure et à y intervenir à nouveau si besoin (pour mise à jour, par exemple).(4)

1.1.4Le rôle de système d'information

Le système d'information est une aide pour la prise de décision : Le Système d'Information permet aux responsables d'obtenir les informations qui leurs sont nécessaires pour les prises de décision .Ils vont pouvoir étudier plus facilement les conséquences possibles de leur décision .le Système d'Information va aussi permettre d'automatiser certain décisions. (3)

Le Système d'Information est un outil de contrôle de l'évolution d'organisation : Le Système d'Information va permettre de détecter des dysfonctionnements interne ou des situations anormal pour que cet outil soit opérationnel ; le Système d'Information doit être la « mémoire collective » de l'organisation cela en gardant constamment une trace de chaque information. (3)

Le Système d'Information est un outil de coordination des déférentes activités de l'entreprise : Le Système d'Information va aussi fournir des informations sur le présent, elles seront les mêmes pour l'ensemble des services et seront mises à jour régulièrement .tout le monde est informé de mêmes manières selon son accès aux informations. (3)

1.1.5Les systèmes d'information au défi de la mobilité

La mobilité est sur le point de transformer radicalement le paysage des technologies de l'information et de la communication dans l'entreprise. Une révolution qui ouvre de réelles opportunités, mais pose aussi des questions de sécurité. D'ores et déjà, les premiers enseignements émergent pour mieux gérer les risques et les coûts associés à ce défi organisationnel.

Les directeurs des systèmes d'information (DSI) sont accoutumés à vivre avec le changement permanent. Systèmes centralisés, ordinateurs personnels et Internet ont successivement transformé les technologies de l'information, aussi bien pour les entreprises elles-mêmes que pour leurs fournisseurs de matériel et de services. La prochaine rupture, celle de la mobilité, arrive à grands pas – et les DSI l'ont désormais tous en ligne de mire.

Après des débuts modestes, la mobilité connaît un essor sans précédent, portée par des smartphones, tablettes et autres appareils toujours plus performants, grâce aux réseaux 3G et 4G ainsi qu'à l'explosion des applications innovantes. Dans les entreprises, les technologies de l'information et la communication sont à la veille d'une révolution qui promet de démultiplier la productivité en étendant le champ des fonctionnalités par-delà les limites physiques de bureaux jusqu'à présent circonscrits à leur espace physique. Toutefois, si les opportunités sont légion, il faudra également composer avec les défis en matière de coûts, de gouvernance et de sécurité. (5)

1.1.6Les avantages de la mobilité pour les entreprises

La mobilité peut servir les entreprises dans quatre domaines principaux.

- ***La communication des salariés***. Un accès amélioré aux e-mails et aux calendriers, ainsi que des applications vocales, vidéo et de messagerie vont renforcer la communication interne. A titre d'exemple, on pourra voir des vidéoconférences mobiles s'organiser spontanément.
- ***La productivité pendant les déplacements***. L'accès à distance aux contenus et aux applications permettra aux salariés de mettre pleinement leur temps à profit lorsqu'ils seront en déplacement. En fournissant un accès mobile au CRM, à un progiciel de gestion intégrée (PGI/ERP) et à des tableaux de bord, par exemple, on améliore la productivité des salariés dans le cœur de métier. Pour les collaborateurs dont le temps travaillé se fait par définition hors des bureaux (par exemple, la force de vente et les employés sur le terrain), les technologies mobiles améliorent leur productivité en amenant sur le terrain des informations et des ressources avec une facilité inédite, cela vaut également pour les tâches administratives.
- ***Les réseaux de capteurs*.** Des capteurs intelligents peuvent automatiser ou contrôler les processus et les systèmes, les rendant ainsi plus efficaces. Les capteurs peuvent aussi conférer aux produits de nouvelles capacités et susciter de nouveaux modèles d'affaires. En matière de soins de santé par exemple, des capteurs utilisés ou portés par les patients rapportent en permanence des variations des paramètres vitaux aux médecins, qui peuvent ajuster les traitements ou, le cas échéant, intervenir auprès du patient de manière proactive.
- ***Un nouveau vecteur pour atteindre les clients*.** Les TIC mobiles ne servent pas qu'à améliorer la productivité : en développant quantitativement et qualitativement les points de contact, les innovations de mobilité peuvent permettre aux entreprises d'entrer en contact avec leurs clients d'une manière tout à fait nouvelle.(5)

1.2 Le génie logiciel

Les logiciels, suivant leur taille, peuvent être développés par une seule personne, une petite équipe, ou un ensemble d'équipes coordonnées. Le développement de grands logiciels par de grandes équipes pose d'importants problèmes de conception et de coordination. Or, le développement d'un logiciel est une phase absolument cruciale qui monopolise l'essentiel du coût d'un produit et conditionne sa réussite et sa pérennité. (6)

Le génie logiciel est un domaine de recherche qui a été défini en 1968, à Garmisch-Partenkirchen, sous le parrainage de l'OTAN. Il a pour objectif de répondre à un problème qui s'énonçait en deux constatations : d'une part le logiciel n'était pas fiable, d'autre part, il était incroyablement difficile de réaliser dans des délais prévus des logiciels satisfaisant leur cahier des charges.(6)

1.2.1Objectif du génie logiciel

L'objectif du génie logiciel est d'optimiser le coût de développement du logiciel. L'importance d'une approche méthodologique s'est imposée à la suite de la crise de l'industrie du logiciel à la fin des années 1970. Cette crise de l'industrie du logiciel était principalement due à :

- l'augmentation des coûts ;
- les difficultés de maintenance et d'évolution ;
- la non-fiabilité ;
- le non-respect des spécifications ;
- le non-respect des délais.(6)

1.2.2Notion de qualité pour un logiciel

En génie logiciel, divers travaux ont mené à la définition de la qualité du logiciel en termes de facteurs, qui dépendent, entre autres, du domaine de l'application et des outils utilisés. Parmi ces derniers nous pouvons citer : (6)

Validité : aptitude d'un produit logiciel à remplir exactement ses fonctions, définies par le cahier des charges et les spécifications.

Fiabilité ou robustesse : aptitude d'un produit logiciel à fonctionner dans des conditions anormales.

Extensibilité (maintenance) : facilité avec laquelle un logiciel se prête à sa maintenance, c'est-à-dire à une modification ou à une extension des fonctions qui lui sont demandées.

Réutilisabilité : aptitude d'un logiciel à être réutilisé, en tout ou en partie, dans de nouvelles applications.

Compatibilité : facilité avec laquelle un logiciel peut être combiné avec d'autres logiciels.

Efficacité : Utilisation optimale des ressources matérielles.

Portabilité : facilité avec laquelle un logiciel peut être transféré sous différents environnements matériels et logiciels.

Vérifiabilité : facilité de préparation des procédures de test.

Intégrité : aptitude d'un logiciel à protéger son code et ses données contre des accès non autorisés.

Facilité d'emploi : facilité d'apprentissage, d'utilisation, de préparation des données, d'interprétation des erreurs et de rattrapage en cas d'erreur d'utilisation.

1.3 Cycle de vie d'un logiciel

Le cycle de vie d'un logiciel (en anglais software lifecycle), désigne toutes les étapes du développement d'un logiciel, de sa conception à sa disparition. L'objectif d'un tel découpage est de permettre de définir des jalons intermédiaires permettant la validation du développement logiciel, c'est-à-dire la conformité du logiciel avec les besoins exprimés, et la vérification du processus de développement, c'est-à-dire l'adéquation des méthodes mises en œuvre.L'origine de ce découpage provient du constat que les erreurs ont un coût d'autant plus élevé qu'elles sont détectées tardivement dans le processus de réalisation. Le cycle de vie permet de détecter les erreurs au plus tôt et ainsi de maîtriser la qualité du logiciel, les délais de sa réalisation et les coûts associés.(6)

1.3.1Etapes d'un cycle de vie

Le cycle de vie du logiciel comprend généralement au minimum les étapes suivantes : (6)

Analyse des besoins et faisabilité : c'est-à-dire l'expression, le recueil et la formalisation des besoins du demandeur (le client) et de l'ensemble des contraintes, puis l'estimation de la faisabilité de ces besoins.

Spécifications ou conception générale : il s'agit de l'élaboration des spécifications de l'architecture générale du logiciel.

Conception détaillée : cette étape consiste à définir précisément chaque sous-ensemble du logiciel.

Codage (Implémentation ou programmation) : c'est la traduction dans un langage de programmation des fonctionnalités définies lors de phases de conception.

Tests unitaires : ils permettent de vérifier individuellement que chaque sous-ensemble du logiciel est implémenté conformément aux spécifications.

Intégration : l'objectif est de s'assurer de l'interfaçage des différents éléments (modules) du logiciel. Elle fait l'objet de tests d'intégration consignés dans un document.

Qualification (ou recette) : c'est-à-dire la vérification de la conformité du logiciel aux spécifications initiales.

Documentation : elle vise à produire les informations nécessaires pour l'utilisation du logiciel et pour des développements ultérieurs.

Mise en production : c'est le déploiement sur site du logiciel.

Maintenance : elle comprend toutes les actions correctives (maintenance corrective) et évolutives (maintenance évolutive) sur le logiciel.

La séquence et la présence de chacune de ces activités dans le cycle de vie dépendent du choix d'un modèle de cycle de vie entre le client et l'équipe de développement. Le cycle de vie permet de prendre en compte, en plus des aspects techniques, l'organisation et les aspects humains. (6)

1.3.2 Modèle de cycle de vie d'un logiciel

1.3.2.1 Modèle du cycle de vie en cascade :

Le modèle de cycle de vie en cascade (cf. figure 5) a été mis au point dès 1966, puis formalisé aux alentours de 1970. Dans ce modèle le principe est très simple : chaque phase se termine à

une date précise par la production de certains documents ou logiciels. Les résultats sont définis sur la base des interactions entre étapes, ils sont soumis à une revue approfondie et on ne passe à la phase suivante que s'ils sont jugés satisfaisants.(6)

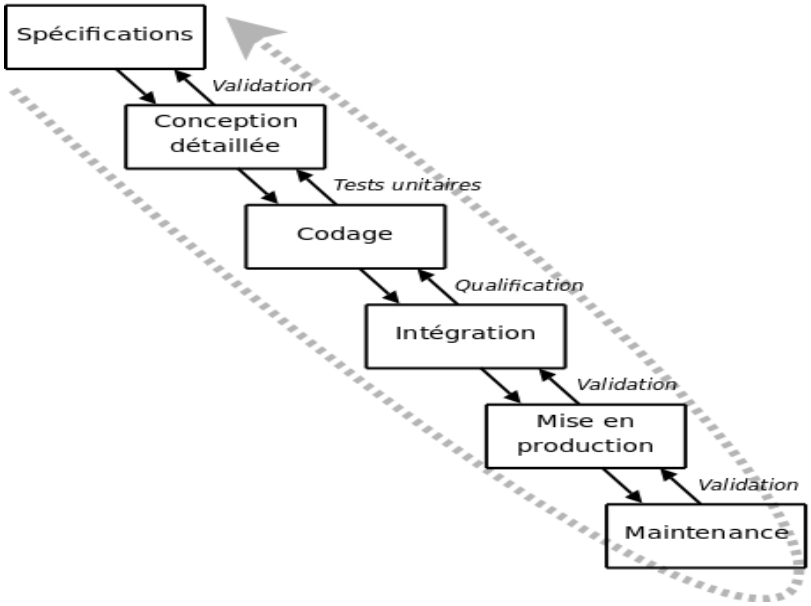

Figure 1-4 : Modèle du cycle de vie en cascade(6)

1.3.2.2 Modèle du cycle de vie en V:

Le modèle en V (cf. figure 6) demeure le cycle de vie le plus connu et certainement le plus utilisé. Il s'agit d'un modèle en cascade dans lequel le développement des tests et du logiciel sont effectués de manière synchrone. Le principe de ce modèle est qu'avec toute décomposition doit être décrite la recomposition et que toute description d'un composant est accompagnée de tests qui permettront de s'assurer qu'il correspond à sa description.(6)

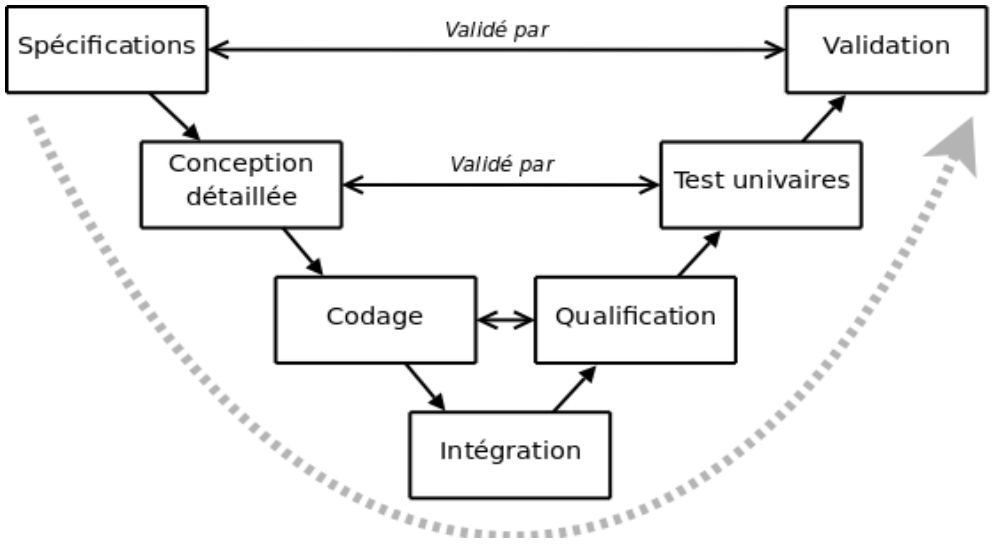

Figure 1-5 : Modèle du cycle de vie en V(6)

1.3.2.3 Modèle du cycle de vie spirale :

Proposé par B. Boehm en 1988, ce modèle est beaucoup plus général que le précédent. Il met l'accent sur l'activité d'analyse des risques : chaque cycle de la spirale se déroule en quatre phases (6) :

1. Détermination, à partir des résultats des cycles précédents, ou de l'analyse préliminaire des besoins, des objectifs du cycle, des alternatives pour les atteindre et des contraintes.
2. Analyse des risques, évaluation des alternatives et, éventuellement maquettage.
3. Développement et vérification de la solution retenue, un modèle « classique » (cascade ou en V) peut être utilisé ici.
4. Revue des résultats et vérification du cycle suivant.

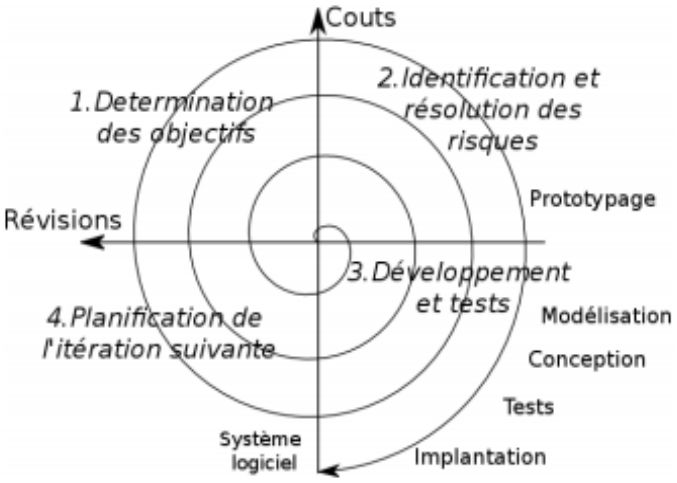

Figure 1-6 : Modèle du cycle de vie en spirale(7)

1.3.2.4 Modèle Prototype :

Le modèle de prototypage est une méthode de développement de système dans laquelle un prototype est créé, testé et ensuite reconstruit si nécessaire jusqu'à ce qu'un résultat approprié soit atteint par lequel développer le système ou produit complet. Sur la base des exigences, la conception est créé et le prototype pour une conception particulière est modélisé et livré aux utilisateurs, puis sur la base du formulaire de retour de l'utilisateur, les modifications appropriées ont été apportées.(8)

Figure 1-7 Modèle de Prototypage(9)

Phases du modèle de prototypage :

Communication: C'est la phase où le développeur et le client organisent la réunion et discutent des objectifs à atteindre pour le logiciel.(8)

Conception: La conception a été réalisée rapidement car les exigences sont éliminées des deux côtés de la fourniture et de la réception. (8)

Il est utilisé pour construire le prototype. Il comprend les aspects importants du logiciel qui sont des entrées et des sorties, mais principalement axés sur les aspects visibles que les activités planifiées.(8)

Modélisation: elle donne une meilleure idée de la nécessité de développer le logiciel et une meilleure compréhension du produit logiciel.(8)

Déploiement: avant le déploiement, le client évalue le logiciel et si le client n'est pas satisfait, il est affiné en fonction des besoins du client. Ce processus se poursuit jusqu'à ce que les exigences du client ponctuel ne soient pas satisfaites. Une fois le client satisfait du produit, leproduit se déploie enfin dans l'environnement de production. Il est soigneusement évalué et testé, et la maintenance est effectuée régulièrement.(8)

Comment fonctionne le modèle prototype?

- Détermination des objectifs.
- Développez le code.
- Communication et raffinement.
- Démontrer.
- Tester.
- Mettre en place.

1.3.2.5 Modèle RAD (Rapid Application Development) :

Le modèle RAD est un développement parallèle de fonctions et une intégration ultérieure, où chaque composant ou fonction est développé en parallèle comme s'il s'agissait de mini projets. Le développement des composants est minuté, livré, puis assemblé en prototype fonctionnel. Cette méthodologie ou ce modèle permet un changement et un développement rapides des composants ou du produit, encourage les retours clients actifs en offrant une visibilité précoce du produit à un client.(10)

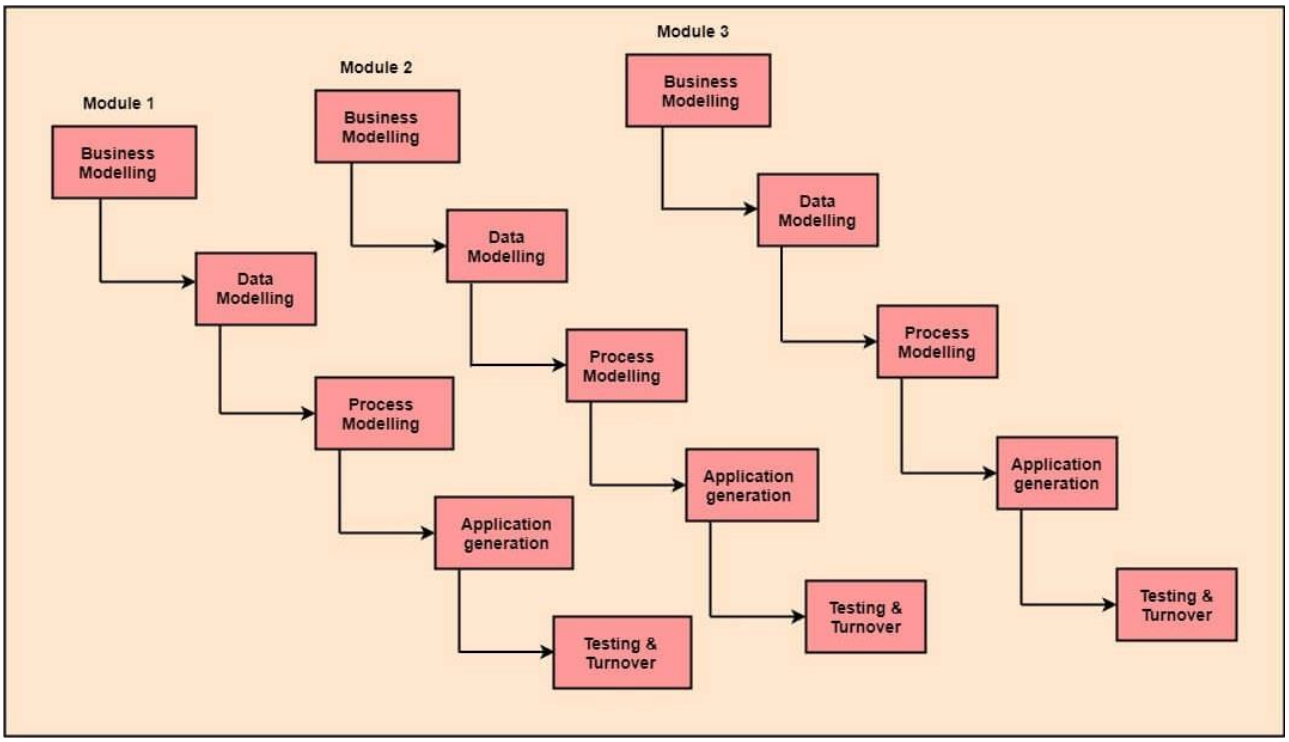

Figure 1-8 Modèle RAD(11)

Phases de modèle RAD :

Planification et analyse des besoins : Cette étape est l'une des étapes les plus cruciales. Ici, les exigences initiales sont rassemblées et analysées correctement.

N'oubliez pas qu'une bonne compréhension des exigences est absolument nécessaire pour que le produit final fabriqué réponde aux attentes.(10)

Conception de l'architecture de projet : Une fois les exigences rassemblées, le prochain objectif est le développement de l'architecture de projet. Une architecture de projet doit être suffisamment flexible pour s'adapter facilement au nouvel ajout de fichiers et de dossiers.(10)

Développement et programmation : Une fois l'architecture conçue, la prochaine tâche majeure est de développer le projet. Cette étape consiste à écrire des piles de code afin d'obtenir l'état irréalisable du produit.(10)

Test : La phase de test consiste à tester le produit développé. Une équipe est impliquée pour tester correctement le produit développé.(10)

Déploiement et maintenance : Une fois les tests terminés, le produit peut être déployé sur le serveur. Un projet déployé nécessite généralement une maintenance et peut-être l'ajout de quelques fonctionnalités supplémentaires.(10)

1.3.3Les méthodes de développement agiles

1.3.3.1 L'agilité

En gestion de projet, la méthode en Cascade c'est l'une des premières méthodes utilisées en raison de sa simplicité et de son caractère séquentiel et linéaire. Cette méthode convient plus aux projets dont les besoins et exigences sont bien définis au début du projet. Les demandes de changements ne sont pas souvent les bienvenues. Les projets de développement logiciel sont souvent caractérisés par des besoins non totalement définis et des demandes de changements qui surviennent à tout moment dans le projet. Cette complexité a conduit progressivement à l'adoption des méthodes agiles dans le domaine de l'ingénierie logicielle. Les méthodes agiles visent fondamentalement la grande satisfaction des parties prenantes du projet (les clients, les utilisateurs, les membres d'équipe, la direction, etc.). Suivant le «manifeste agile», on retient la définition suivante : « **Une méthode agile** est une approche itérative et incrémentale pour le développement de logiciel, réalisée de manière très collaborative par des équipes responsabilisées, appliquant un cérémonial minimal, qui produisent, dans un délai contraint, un logiciel de grande qualité répondant aux besoins changeants des utilisateurs ».(12)Selon le manifeste agile, une méthode agile possède quatre valeurs et 12 principes :

Valeurs

• **Les individus et les échanges** plus que processus et des outils ;
• **Le produit** plus que de la documentation excessive ;
• **La collaboration du client** plus que la négociation ;
• **La réactivité** plus que le suivi d'un plan

Principes

• Satisfaire le client en livrant tôt et régulièrement des logiciels utiles, qui offrent une véritable valeur ajoutée ;
• Accepter les changements, même tard dans le développement ;
• Livrer fréquemment une application qui fonctionne ;
• Collaborer quotidiennement entre clients et développeurs ;
• Bâtir le projet autour de personnes motivées en leur fournissant environnement et support, et en leur faisant confiance ;
• Communiquer par des conversations en face à face ;
• Mesurer la progression avec le logiciel qui fonctionne ;

• Garder un rythme de travail durable ;
• Rechercher l'excellence technique et la qualité de la conception ;
• Laisser l'équipe s'auto organiser ;
• Rechercher la simplicité ;
• À intervalles réguliers, réfléchir aux moyens de devenir plus efficace.

1.3.3.2 Les méthodes agiles en développement logiciel

Cette partie présente les méthodes agiles de développement logiciel les plus usuelles et les plus répandues.

1.3.3.2.1 eXtreme Programming (XP)

XP se définit comme un ensemble de pratiques qui vise à se consacrer à la réalisation elle-même. Ces pratiques sont essentiellement axées la programmation permettant d'améliorer continuellement la conception et code. Voici quelques pratiques d'ingénierie logicielle comme:

• **Le développement piloté par les tests** : les tests unitaires sont développés avant le développement des fonctionnalités ;
• **La programmation en binôme** : XP encourage le développement à 2 afin de faciliter la détection les anomalies et erreurs éventuelles.

1.3.3.2.2 Dynamic Systems Development Method (DSDM)

La méthode DSDM a été créée en 1994 pour répondre aux besoins de qualité et aux principes de RAD (Rapid Development Application). Elle permet donc la livraison rapide et efficace des résultats (13). La méthode AgilePF DSDM est conçue pour permettre l'intégration facile des autres approches agiles comme Scrum et XP. Elle est utilisée pour faciliter l'accessibilité des projets par les parties prenantes. Les phases de développement de la méthode AgilePF sont :

• ***Pre-Project*** **:** pour s'assurer que le projet est aligné avec les stratégies de l'entreprise et répond aux objectifs d'affaires.

• ***Feasability*****:** pour vérifier si le projet est réalisable techniquement et s'il est rentable du point de vue commercial.

• ***Foundations*****:** elle permet de s'assurer de la compréhension de la portée du projet et détermine le cycle de vie du projet ainsi que les processus du modèle à utiliser. Elle peut durer plusieurs semaines pour les gros projets.

• ***EvolutionaryDevelopment*:** utilisation d'une approche de développement incrémentale et itérative.

• ***Deployment*:** phase finale d'assemblage des différents livrables. Livraison et déploiement du produit.

• ***Post-project***: pour s'assurer que les objectifs commerciaux ont été atteints.

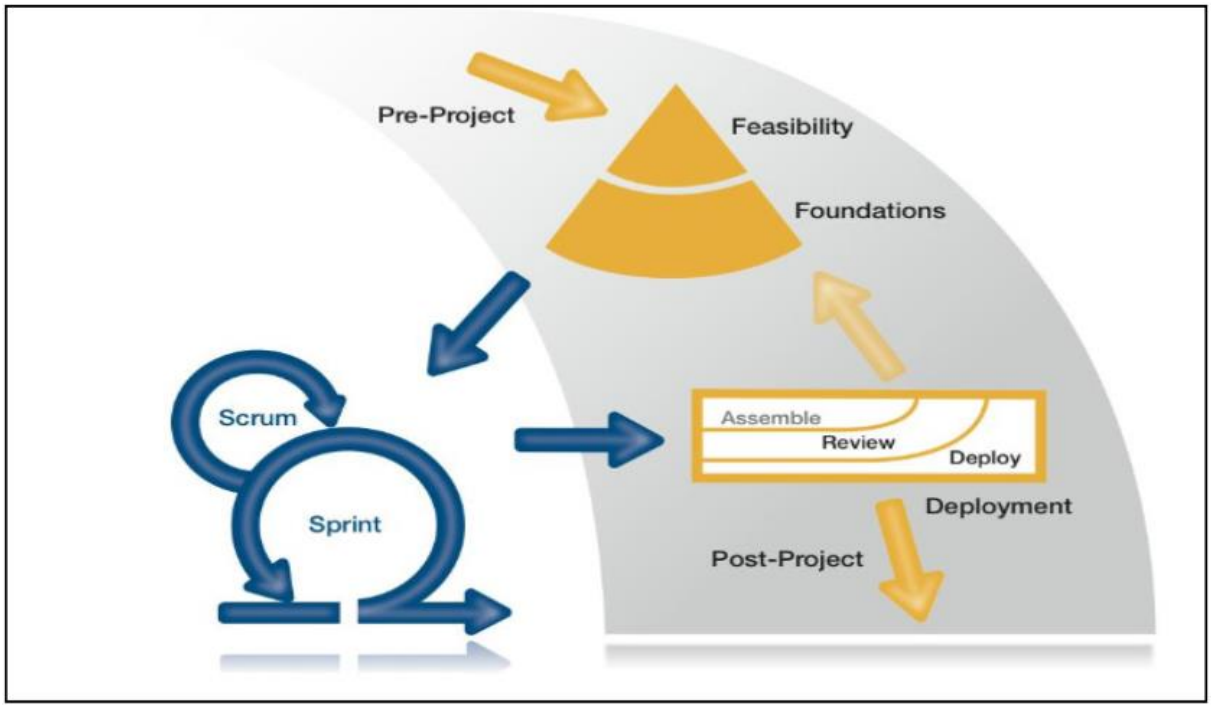

Figure 1-9 Les Phases de développement de Agile(13)

Chaque projet a un cycle de vie qui commence par l'identification d'un besoin potentiel et se termine à un moment où ce besoin est soit terminé a été rejeté (13). Le rejet peut se produire à tout moment si le projet devient non viable d'un point de vue commercial. La phase post-project permet à AgilePF de tenir compte de l'après-projet, ce que ne fait pas habituellement la méthode Scrum. AgilePF tient donc compte de 2 éléments que sont **la gestion du projet** et **la livraison du produit.**

La méthode DSDM est plus formelle avec un nombre assez important de rôles, de processus et d'artéfacts. Elle est aussi caractérisée par la méthode de classification et de priorisation des exigences suivant la méthode **MoSCoW**(13)(14) :

• **M** (Must have) : fonctionnalités obligatoires ;
• **S** (Should have) : fonctionnalités importantes à faires si possibles ;

• **C** (Could have) **:** fonctionnalités possibles, mais non indispensables ;

• **W** (Won’t have) : fonctionnalités qui peuvent attendre la prochaine fois.

1.3.3.2.3 Scrum

La méthode Scrum est la méthode agile la plus populaire. Elle est définie par son fondateur Ken Schwaber comme un cadre (framework) qui présente les éléments qui feront partie du processus appliqué pour la réalisation d'un produit. Scrum impose des itérations de courtes durées appelées ***Sprint*** pour le développement du produit. Le schéma ci-dessous montre le déroulement d'un ***Sprint.***(15)

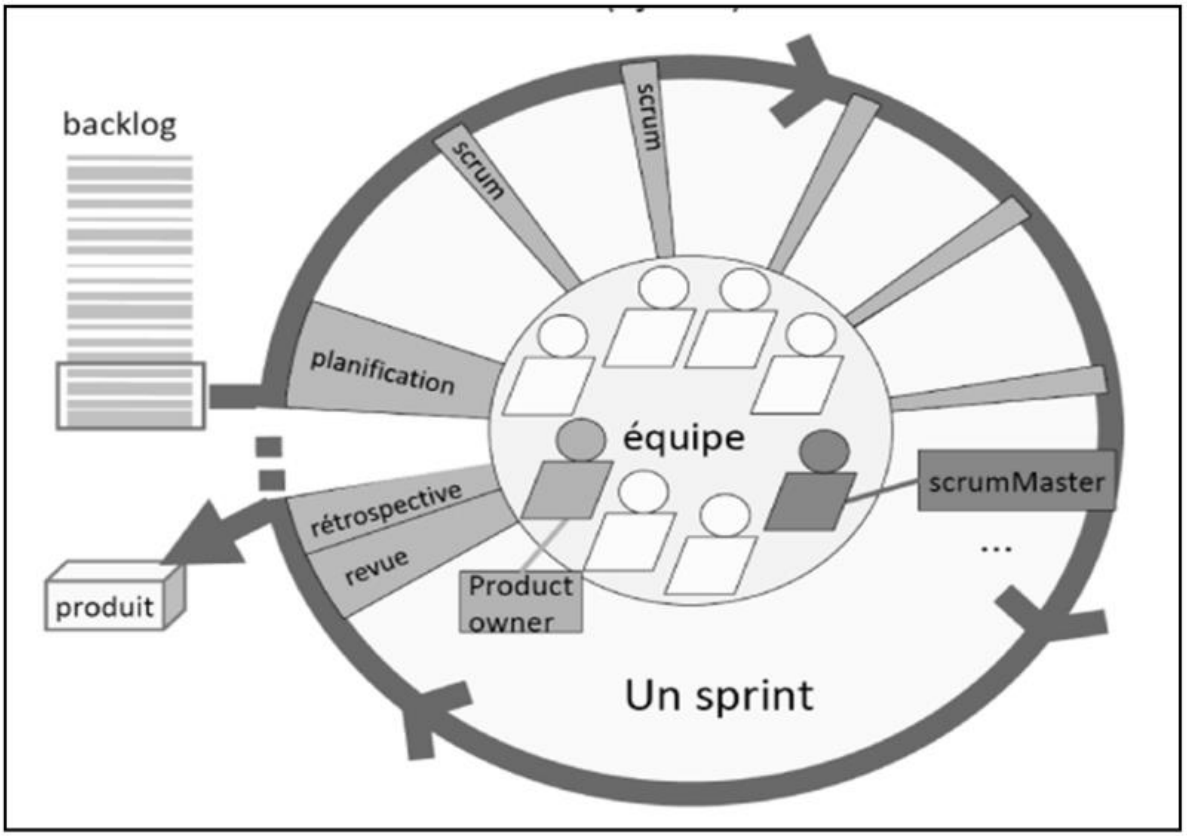

Figure 1-10 Schéma de déroulement d'un Sprint(15)

Les caractéristiques

• Les fonctionnalités à réaliser sont définies et regroupées par ordre de priorité dans le ***Backlog***de produit par le ***Product Owner***;

• À chaque itération ou ***Sprint*** de durée courte et fixe et définie à l'avance, des fonctionnalités sont développées pour créer une version du produit appelée ***Release.*** Le contenu de chaque itération est défini par le ***Product Owner***;

• Au cours des ***Sprints*** des rencontres quotidiennes appelées **mêlées** sont organisées par le ***Scrum Master*** pour l'application des principes de Scrum et suivre l'avancement par rapport aux engagements afin d'assurer le succès du ***Sprint*** ;

• À la fin de chaque ***Sprint,*** un produit partiel est livré. Son évaluation et les rétroactions permettent d'ajuster le ***Backlog***pour le prochain ***Sprint.*** À cette étape, l'équipe de projet identifie les anomalies afin d'améliorer le processus lors des prochains ***Sprints.*** Cette étape est très importante pour le succès du projet. Elle implique l'intervention du ***Product Owner***, du client ou des utilisateurs finaux.

Les composantes

Les composantes nécessaires à l'utilisation de la méthode Srcum sont présentées ci-dessous :

• **Rôles :** Définissent les acteurs clés de la méthode Scrum

• **Cérémonial** : Définissent les blocs de temps nécessaire pour créer la régularité dans les ***Sprints***

• **Artefacts :** Définissent les différents documents et outils. Le plus important est le ***Backlog***. Il faut noter que Scrum impose très peu d'artefacts.

Les phases de développement

En ingénierie logicielle, quatre phases sont habituellement utilisées pour le développement des produits :

• **S**pécification fonctionnelle (définition des exigences fonctionnelles);
• **A**rchitecture (conception);
• **C**odage (et test unitaire);
• **Te**st (d'intégration et de recette)

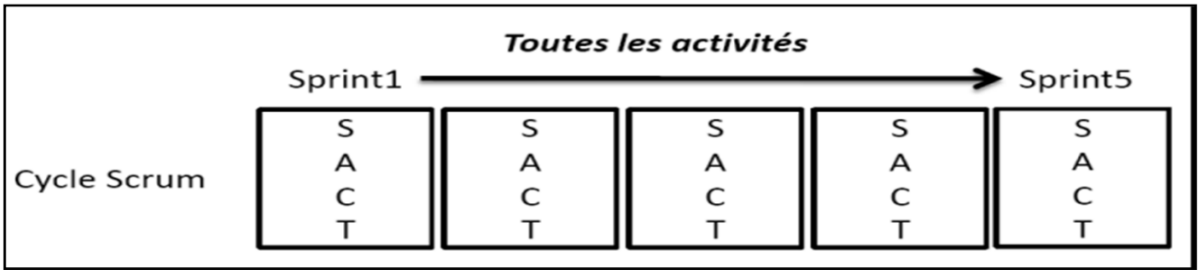

Figure 1-11 Cycle Scrum basé sur 5 Sprints – Itérations(15)

Figure 1-12 Cycle séquentiel (Classique)(15)

Les 2 figures ci-dessus montrent la différence fondamentale entre la méthode classique comme cascade et celle basée sur les principes de la méthode agile. Dans la méthode classique, les différentes activités sont séquentielles. Ce qui implique que les tests sont effectués à la fin du projet. En cas d'erreurs ou de problèmes liés à la conception ou à la définition des exigences, il est difficile de faire des changements. Les demandes de changement ne sont pas souvent acceptées. Ce qui entraîne des délais complémentaires pour la livraison du produit et l'insatisfaction du client dans certains projets. Avec les méthodes agiles comme Scrum, toutes les activités formelles ou évènements sont exécutés durant chaque itération. Ce qui facilite la détection et la correction des erreurs et la prise en compte des demandes de changements.

1.4 UML

UML n'est pas une méthode (*i.e.* une description normative des étapes de la modélisation) : ses auteurs ont en effet estimé qu'il n'était pas opportun de définir une méthode en raison de la diversité des cas particuliers. Ils ont préféré se borner à définir un langage graphique qui permet de représenter et de communiquer les divers aspects d'un système d'information. Aux graphiques sont bien sûr associés des textes qui expliquent leur contenu. UML est donc un métalangage, car il fournit les éléments permettant de construire le modèle qui, lui, sera le langage du projet. (6)

Il est impossible de donner une représentation graphique complète d'un logiciel, ou de tout autre système complexe, de même qu'il est impossible de représenter entièrement une statue (à trois dimensions) par des photographies (à deux dimensions). Mais il est possible de donner sur un tel système des *vues* partielles, analogues chacune à une photographie d'une statue, et dont la conjonction donnera une idée utilisable en pratique sans risque d'erreur grave. (6)

UML 2.0 comporte ainsi treize types de diagrammes représentant autant de *vues* distinctes pour représenter des concepts particuliers du système d'information. Ils se répartissent en deux grands groupes : (6)

1.4.1 Diagrammes structurels ou diagrammes statiques (*UML Structure*)

- diagramme de classes (*Class diagram*)
- diagramme d'objets (*Object diagram*)
- diagramme de composants (*Component diagram*)
- diagramme de déploiement (*Deploymentdiagram*)
- diagramme de paquetages (*Package diagram*)
- diagramme de structures composites (*Composite structure diagram*)

1.4.2 Diagrammes comportementaux ou diagrammes dynamiques (*UML Behavior*)

- diagramme de cas d'utilisation (*Use case diagram*)
- diagramme d'activités (*Activity diagram*)
- diagramme d'états-transitions (*State machine diagram*)
- **Diagrammes d'interaction (*Interaction diagram*)**
  - diagramme de séquence (*Sequencediagram*)
  - diagramme de communication (*Communication diagram*)
  - diagramme global d'interaction (*Interaction overviewdiagram*)
  - diagramme de temps (*Timing diagram*)

Ces diagrammes, d'une utilité variable selon les cas, ne sont pas nécessairement tous produits à l'occasion d'une modélisation. Les plus utiles pour la maîtrise d'ouvrage sont les diagrammes d'activités, de cas d'utilisation, de classes, d'objets, de séquence et d'états-transitions. Les diagrammes de composants, de déploiement et de communication sont surtout utiles pour la maîtrise d'œuvre à qui ils permettent de formaliser les contraintes de la réalisation et la solution technique. (6)

Diagramme de cas d'utilisation : Le diagramme de cas représente la structure des grandes fonctionnalités nécessaires aux utilisateurs du système. C'est le premier diagramme du modèle UML, celui où s'assure la relation entre l'utilisateur et les objets que le système met en œuvre.(6)

Diagramme de Classes : Le diagramme de classes est généralement considéré comme le plus important dans un développement orienté objet. Il représente l'architecture conceptuelle du système : il décrit les classes que le système utilise, ainsi que leurs liens, que ceux-ci représentent un emboîtage conceptuel (héritage) ou une relation organique (agrégation).(6)

Diagramme d'objets : Le diagramme d'objets permet d'éclairer un diagramme de classes en l'illustrant par des exemples. Il est, par exemple, utilisé pour vérifier l'adéquation d'un diagramme de classes à différents cas possibles.(6)

Diagramme d'états-transitions : Le diagramme d'états-transitions représente la façon dont évoluent les objets appartenant à une même classe. La modélisation du cycle de vie est essentielle pour représenter et mettre en forme la dynamique du système.(6)

Diagramme d'activités : Le diagramme d'activités n'est autre que la transcription dans UML de la représentation du processus telle qu'elle a été élaborée lors du travail qui a préparé la modélisation : il montre l'enchaînement des activités qui concourent au processus. (6)

Diagramme de séquence : Le diagramme de séquence représente la succession chronologique des opérations réalisées par un acteur. Il indique les objets que l'acteur va manipuler et les opérations qui font passer d'un objet à l'autre. On peut représenter les mêmes opérations par un diagramme de communication graphe dont les nœuds sont des objets et les arcs (numérotés selon la chronologie) les échanges entre objets. En fait, diagramme de séquence et diagramme de communication sont deux vues différentes, mais logiquement équivalentes (on peut construire l'une à partir de l'autre) d'une même chronologie. Ce sont des diagrammes d'interaction.(6)

Conclusion

Le système d'information est désormais l'une des principales valeurs dans la matière de transmission d'informations et de développement des connaissances. Dans ce chapitre nous avons parlé de système d'information et de son importance dans une entreprise et de génie logiciel, et la modélisation des systèmes d'informations avec UML.