L'analyse et la modélisation, suivies de l'implantation par SPRINTs successifs, sont des étapes cruciales dans le processus de développement logiciel agile. Elles permettent de transformer les besoins et les idées en un logiciel fonctionnel et de haute qualité. Voici une explication détaillée de chaque étape :

1. **Obtention d'un mandat** :
   * À cette étape, vous obtenez un mandat qui spécifie ce que le logiciel doit accomplir.
   * Le mandat peut provenir d'un client externe (une entreprise ou un utilisateur final), d'un client interne (un autre département ou une équipe de votre entreprise) ou même de vous-même si vous initiez le projet.
   * Le mandat clarifie les attentes en termes de livrables, ce que le logiciel doit réaliser pour répondre aux besoins et aux objectifs.
2. **Analyse textuelle** :
   * L'analyse textuelle consiste à examiner en détail le mandat pour comprendre pleinement les besoins et les objectifs.
   * Il s'agit d'effectuer une lecture formelle du document du mandat, de l'analyser ligne par ligne pour s'assurer que rien n'est négligé.
   * Cette étape est essentielle pour éviter les malentendus et les erreurs de compréhension.
3. **Identification des cas d'usage et des rôles** :
   * Vous identifiez les cas d'usage, c'est-à-dire les différentes manières dont le logiciel sera utilisé par les acteurs (utilisateurs, systèmes externes, etc.).
   * Les rôles définissent qui sont les acteurs impliqués dans chaque cas d'usage.
   * Cette étape aide à établir une compréhension claire des fonctionnalités attendues du logiciel du point de vue des utilisateurs.
4. **Scénarios d'utilisation** :
   * Les scénarios d'utilisation décrivent séquentiellement les étapes nécessaires pour accomplir chaque cas d'usage.
   * Ils mettent en lumière le flux de travail du logiciel, montrant comment les utilisateurs interagiront avec le système pour atteindre leurs objectifs.
   * Cela permet de visualiser le comportement attendu du logiciel.
   * L’explication de la manipulation humaine dans l’interface et les opérations effectuées par l’ordinateur
5. **Maquettes graphiques** :
   * À cette étape, vous créez des maquettes graphiques qui représentent le design visuel du logiciel, y compris les pages et les écrans que les utilisateurs verront.
   * Ces maquettes servent de base pour le design de l'interface utilisateur (UI) et permettent de visualiser à quoi ressemblera le logiciel une fois développé.
6. **Définition des classes et des méthodes** : CRC
   * Vous identifiez les classes (objets) et les méthodes (opérations) qui seront nécessaires pour implémenter les fonctionnalités du logiciel.
   * Cette étape concerne la conception orientée objet et aide à définir la structure du code source.
7. **Modélisation de données** :
   * Vous définissez la structure des tables de données ou d'autres modèles de données nécessaires pour stocker les informations du logiciel.
   * Cela garantit que la gestion des données est correctement planifiée, ce qui est essentiel pour assurer la cohérence des données. CSV
8. **Planification globale** :
   * La planification globale consiste à établir un cadre temporel pour le projet.
   * Vous déterminez l'ordre de réalisation des différentes parties du logiciel et découpez le projet en itérations appelées SPRINTs.
9. **Implantation par SPRINTs successifs** :
   * Après la planification globale, le développement réel commence avec les SPRINTs.
   * Chaque SPRINT est une période courte (généralement de 2 à 4 semaines) au cours de laquelle une partie spécifique du logiciel est développée, testée et livrée.
10. **SCRUMs quotidiens** :
    * Les SCRUMs sont des réunions quotidiennes de l'équipe de développement.
    * Chaque membre de l'équipe partage les tâches qu'il a complétées, celles en cours et signale les problèmes rencontrés.
11. **Debriefing/Ajustement** :
    * À la fin de chaque SPRINT, une réunion de débriefing est organisée pour évaluer l'avancement du projet.
    * Des ajustements sont faits au plan en fonction des retours d'expérience et des problèmes rencontrés.
12. **Livraison partielle** :
    * Le logiciel est livré partiellement au fur et à mesure de l'avancement des SPRINTs.
    * Cela permet au client de commencer à utiliser des parties fonctionnelles du logiciel dès les premières itérations, ce qui est bénéfique pour obtenir des retours rapidement.

En résumé, ce processus permet de développer un logiciel de manière itérative, en se concentrant sur les besoins du client, en planifiant et en ajustant continuellement le projet pour assurer la livraison de fonctionnalités de haute qualité.

Un cas d'utilisation, également appelé bloc fonctionnel ou cas d'usage en génie logiciel et ingénierie des systèmes, est un concept qui définit comment un système doit être utilisé pour apporter de la valeur ou de l'utilité aux acteurs impliqués. Il s'agit d'une manière de décrire comment un système interagit avec ses utilisateurs pour atteindre un objectif précis. Les cas d'utilisation sont une partie importante de la collecte des exigences d'un système, et ils adoptent le langage et le point de vue des utilisateurs finaux.

Historiquement, les cas d'utilisation ont été introduits par Ivar Jacobson en 1987 et ont évolué au fil des années pour devenir une méthode puissante de capture des besoins et de modélisation dans le développement de logiciels et d'ingénierie des systèmes. Ils ont été initialement utilisés dans le cadre de méthodes orientées objet, mais leur utilisation s'est étendue à d'autres paradigmes de développement.

Principes clés des cas d'utilisation :

1. **Finalité** : Chaque cas d'utilisation est associé à une finalité ou un objectif spécifique pour un acteur du système. L'acteur primaire est généralement un utilisateur humain ou un autre système, et il peut y avoir des acteurs secondaires.
2. **Scénarios** : Les cas d'utilisation sont décrits à travers des scénarios qui détaillent comment le système interagit avec les utilisateurs pour accomplir l'objectif. Il y a souvent un scénario principal, mais il peut également y avoir des variantes et des exceptions.
3. **Langage utilisateur** : Les cas d'utilisation sont exprimés dans le langage de l'utilisateur final ou de l'expert du domaine, évitant ainsi le jargon technique.
4. **Types de cas d'utilisation** : Il existe plusieurs types de cas d'utilisation, notamment les cas d'utilisation concrets (décrivant en détail les interactions), les cas d'utilisation paramétrés (regroupant des cas similaires avec des paramètres), les cas d'utilisation essentiels (décrivant les intentions de l'utilisateur sans préjuger de l'implémentation), et les cas d'utilisation métier (dans le contexte des modèles d'affaires).

Les cas d'utilisation sont utilisés pour définir les besoins fonctionnels d'un système, guider le développement logiciel, et servir de base pour la conception et les tests. Ils sont également adaptés à l'approche agile, où ils sont découpés en tranches pour une mise en œuvre itérative. Les cas d'utilisation jouent un rôle essentiel dans la communication entre les parties prenantes du projet et les équipes de développement, assurant ainsi que le système réponde aux besoins réels des utilisateurs.