**LA CRISE LOGICIELLE**

**Les symptômes de La crise du logicielle :**

* Le logiciel est hors de délais, coût ou de qualité
* Le logiciel ne conforme pas aux spécifications
* Difficultés des entretiens

**Les causes de la crise logicielle**

* Des causes matérielles, une spécifications fausses, erreurs de programmation, manque de tests, , ou mauvaise utilisation de l’IHM (Interface Homme Machine).

**Les 3 types de tests :**

* **Test unitaires :** vérifier qu’une entité individuelle (classe, module, composant) se comporte correctement (Junit, xUnit).
* **Test d’intégration :** vérifier qu’un ensemble des entités se comporte correctement entre eux(Jenkins).
* **Tests de non-régression :** vérifier qu’un changement n’a pas d’effets négatif sur le fonctionnement global du système.

**Quatre types de maintenance :**

* **Corrective :** (correction des bugs)
* **Évolutive :** (modification des spécifications initiales)
* **Adaptative :** (nouvel environnement)
* **Perfective**: (améliorer les performance).

**GÉNIE LOGICIEL**

**Définition :** un ensemble de méthodologie, de techniques, et d’outils.

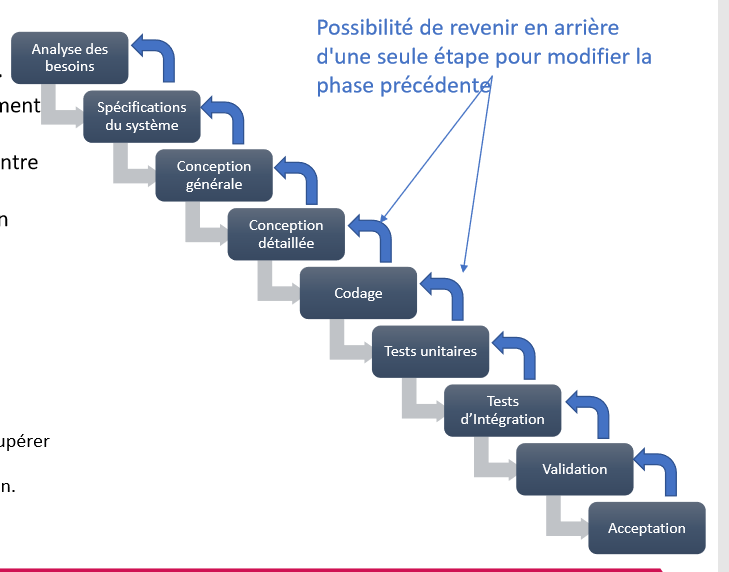
**Objectifs :** regagner la confiance des utilisateurs, augmenter la productivité d’équipe de développement, et de garantir la livraison en respectent le coût, le délai et la qualité.

**Activités du cycle de vie logiciel :**

1. **Définition des objectifs :** définir la finalité et la stratégie globale du projet.
2. **Analyse des besoins et faisabilité** : la collection et la formalisation (cahier de charge) des besoins du client ainsi que toutes les contraintes, et également l’estimation de la faisabilité de ces besoins.
3. **Spécification fonctionnelle:** l’ensemble des fonctions rendu par le système exprimer par un diagramme d’use case.
4. **Spécification non fonctionnelle:** Facteurs de qualité, condition de mis en œuvre, de déploiement, de maintenance et d’exploitation, ainsi que l’aspect juridique et financiers.
5. **Conception générale :** traduction des besoins dans un langage de programmation, ainsi que la description générale du système, et de chacun de ces modules (paquet, composants, déploiement).
6. **Conception détaillée :** Concevoir et documenté le code, traduction des spécifications en terme d’artefacts (classe, interactions, activité …)
7. **Codage :** Traduction de la conception en code
8. **Test Unitaires**.
9. **Test d’intégration**.
10. **Validation :** Test du logiciel final par rapport aux spécification.
11. **Recette :** acceptation du système final, tester que le logiciel conforme aux spécifications dans le site de production.
12. **Démarrage de production :** le déploiement du logiciel sur le site de production
13. **Maintenance :** action correctives et action évolutives sur l’application
14. **Documentation**: les information nécessaires á l’utilisation du logiciel et á son développement ultérieur.

**Cycle de vie en cascade :** chaque phase se termine á une date précise et elle permet la production d’un certains document ou logiciel, on ne passe pas á la phase suivante jusqu’á ce que le résultat de la phase précédente sera jugé satisfaisant.

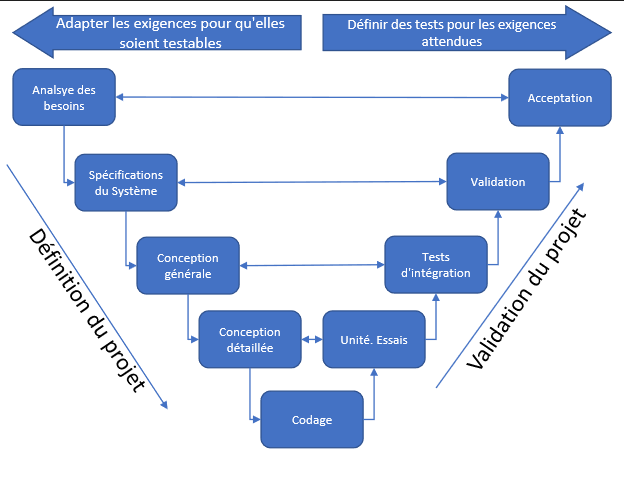
**Problème :** Vérification trop tardive du comportement du logiciel, sensible á l’arrivé des nouveaux changements, facile á planifier mais difficile á respecter le plan.



**Modèle en V :** éviter d’énoncer une spécification qui ne peut pas être vérifier dans la mise en œuvre, permet d’anticiper les anomalies lors des phases et de test et de validation, chaque phase de réalisation accompagné par une phase de test qui permet de la validé.

Si je trouve qu’une spécification est non testable je la remplace par une autre qui est testable.

**Problème :** Vérification améliorer mais trop tardive, manque de réactivité vis-à-vis des nouveaux changements.



**Cycle de vie en spirale :** atteindre les objectifs par la construction successive et progressive du produit :

* Détermination des **objectifs** du cycle,
* **Analyse des risques**, évaluation des **alternatives** et, si nécessaire, conception ;
* **Développement** et vérification de la solution retenue, en cascade ou en V ;
* **Examen des résultats** et **planification du prochain cycle**

**Avantages :** capacité de gérer les risques et les grands projets ainsi que l’intégration du client et l’évolution des spécifications.

**Inconvénients :** cycle dure de 6 mois á 2 ans, nécessite une équipe experte dans la gestion des risques, les changements ne seront prise en charge que jusqu’à le cycle suivant.

**Modèle par incréments :** Le logiciel est décomposé en modules développés séparément et ils sont intégrés à la fin du processus.

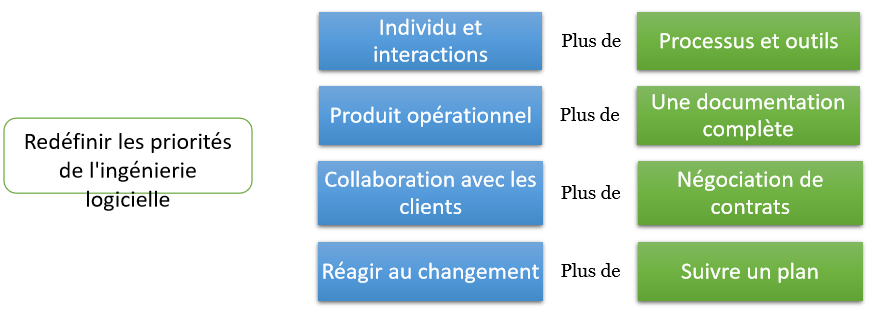
**Modèle par itération :** le logiciel est développé version après version.

**Modèle adaptatif :** le logiciel accepte l’arrivé des nouveaux changements.

**Manifeste Agile**

**Définition :** Texte rédigé en février 2001 aux USA, Réunion de 17 spécialistes pour unifier les méthodes « Agile » → « Notre plus haute priorité est de satisfaire le client à travers des livraisons tôt et continues de logiciels de grande valeurs ».

**Les priorités agiles :**



**Les 12 principes agiles :**

1. **Client :**

* Satisfaction des clients
* Accepter le changement à tout moment
* Livraison fréquente
* Travail client-développeur

1. **Équipe :**

* Motivation de l'équipe
* Conversation face à face
* Logiciel opérationnel
* Rythme durable

1. **Produit / Procédé :**

* Excellence technique
* Simplicité
* Des équipes auto-organisées
* Amélioration continue

**SCRUM**

**Définition:** Processus de gestion de projet, itératif et incrémental qui s'adapte à la réalité des projets et à l'expérience du domaine du développement logiciel (il est á la fois incrémental, itératif, et adaptatif).

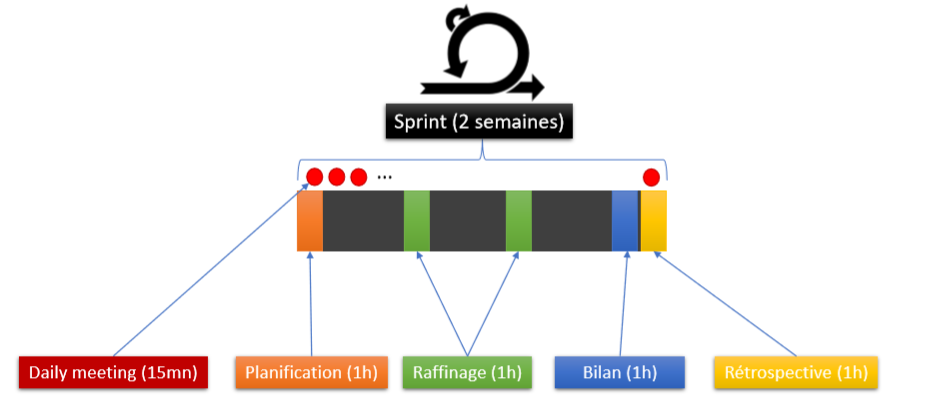
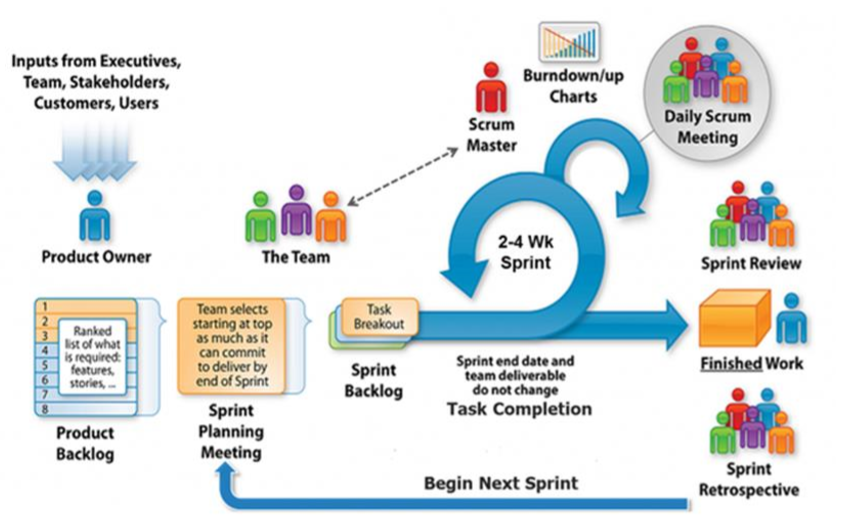
**Principe de scrum :**

* **Transparence :** les progrès et les risques doivent être visible par une communication honnête entre les membres de l’équipe.
* **Adaptation :** l’équipe doit s’adapter au changement á tout moment.
* **Inspection :** l’équipe repense á tout moment sur ses méthodes de travail.

**Valeur de scrum (OFCRC):**

* **OPENNESS (ouverture) :** visualisez votre travail, vos échéances, vos problèmes
* **FOCUS (concentration) :** Se concentrer sur l'essentiel, l’équipe travaillent dans la même direction.
* **COMMITMENT (engagement) :** Respect les étapes d'une mission tel qu’ils ont été convenu.
* **RESPECT (respect) :** Les membres de l’équipe se respecte mutuellement, et respecte l’indépendance de chacun…
* **COURAGE (courage) :** considérer l’échec comme une occasion de s’améliorer.

**Sprint :** bloc de temps non extensible de 2 á 4 semaines donne lieu á la création d’un incrément (le résultat d’un sprint s’appelle incrément qui peut être module ou livrable).



**Parties prenantes :** parties externe impliquées dans le développement du logiciel (client, sponsors, …).

**Product Owner PO :** Porteur de la vision globale du produit, définit les spécifications fonctionnelles et liste des priorités qu'ils faut développés. PO Accepte ou Rejette les livrables.

**Scrum Master :** gardien des pratiques SCRUM, facilitateur et coach qui élimine les obstacles et qui veille sur le bon fonctionne de l’équipe.

**Équipe de Scrum :** de 5 á 9 personne qui doivent avoir les compétences nécessaires, le sens de responsabilité, le sens de partage de l’information, l’aptitude de l’auto-formation, et l’esprit de travail collaboratif. Les décisions sont prises collectivement entre les membres de l’équipe.

**Sprint Planning**: au début du sprint ou le PO présente les spécifications du logiciel qui souhaite inclure dans l’itération courante, á la fin de cette réunion on sorte avec le sprint backlog.

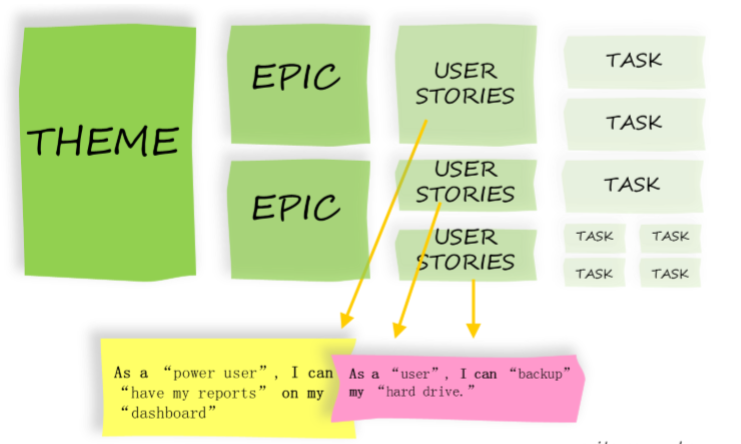
**Affinement du Product backlog :** une fois par semaine, alimenter le backlog du départ par des histoires toutes prêtes, décomposer les histoires et épiques, réorganiser les priorités, supprimer les éléments obsolètes.

**Product Backlog (carnet de commande pour le produit) :** Géré par le Product Owner (peut ajouter, modifier ou effacer une spécification à n’importe quel moment du développement) → Liste de tout ce qui causera du travail pour l’équipe

**Sprint backlog :** Liste avec les engagements de l’équipe pendant un sprint, où il y’a l’estimation des heures de travail, et l’estimation de la complexité des spécifications par des story points.

**Elément du backlog :**

* **Thème :** service ou fonctionnalité á produire, elle est réalisée par le biais de plusieurs histoires.
* **Épique :** team story grande ou pas encore connu qui reste dans le backlog jusqu’à son décomposition (affinage).
* **User story :** de petits fonctionnalité du produit qui peuvent être développés pendant un sprint.
* **Tache :** des petites taches du user story assez petit qui ne dépassent pas 8 heures.



**Daily meeting :** 15 minutes chaque jour, trois questions á se poser 🡪 qu’avez-vous faire hier ?, que ferez-vous aujourd’hui?, quels sont vos problèmes?

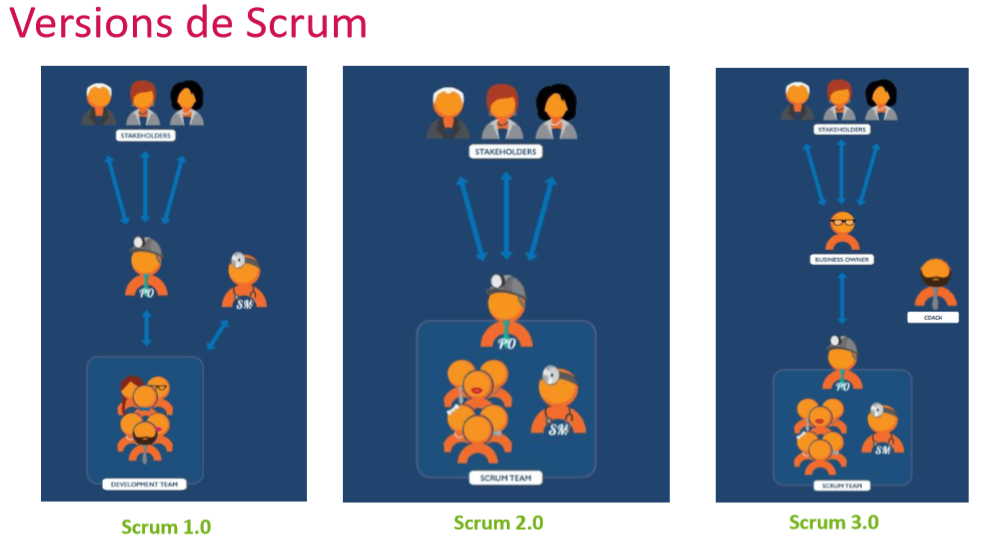
**Sprint meeting review :** réunion amicale et informel á la fin du sprint qui regroupe PO, SM, équipe et la partie prenantes pour les raisons suivantes :

* Accepter ou non le livrable par le PO.
* Avoir le feedback de la partie prenante.

**Sprint retrospective :** se passe après la sprint meeting review en regroupant seulement PO, SM, et équipe pour l’observation de ce qui se n’est pas bien passé dans l’organisation et de décider les actions á entreprendre.

**Impact du SCRUM sur les structures d’équipe d’entreprise :**

* Scrum élimine le rôle de chef de projet qui est finalement répartie entre le PO et le reste de l’équipe.
* Eviter les couches des fonctionnaires improductifs.



**SAFE :** ensemble de modèles d’organisation et de flux de travail permettant la collaboration d’un grand nombre d’équipe agiles (AGILE RELEASE TRAIN : ART) pour le développement d’un même grand projet.

**Incrément de produit (SAFE) :** période dans laquelle un ART fournit une valeur incrémentielle (logiciel ...).

**Planification de l’incrément de produit (SAFE) :** á chaque 5 sprints pendant 2 jours où tout le monde est impliqué face á face pour avoir des objectifs du PI ( tableau de programme).

**Tableau de programme (PI) :** comporte une vue globale du travail á effectuer par toutes les équipes sans entrer dans les détails.

**Programmation extrême XP**

**Objectifs :** respecter le délai, comprendre les besoins du client, accroitre la qualité du logiciel et la possibilité d’accepter les changements afin d’éviter l’annulation du projet.

**Valeurs XP (SFCRC):**

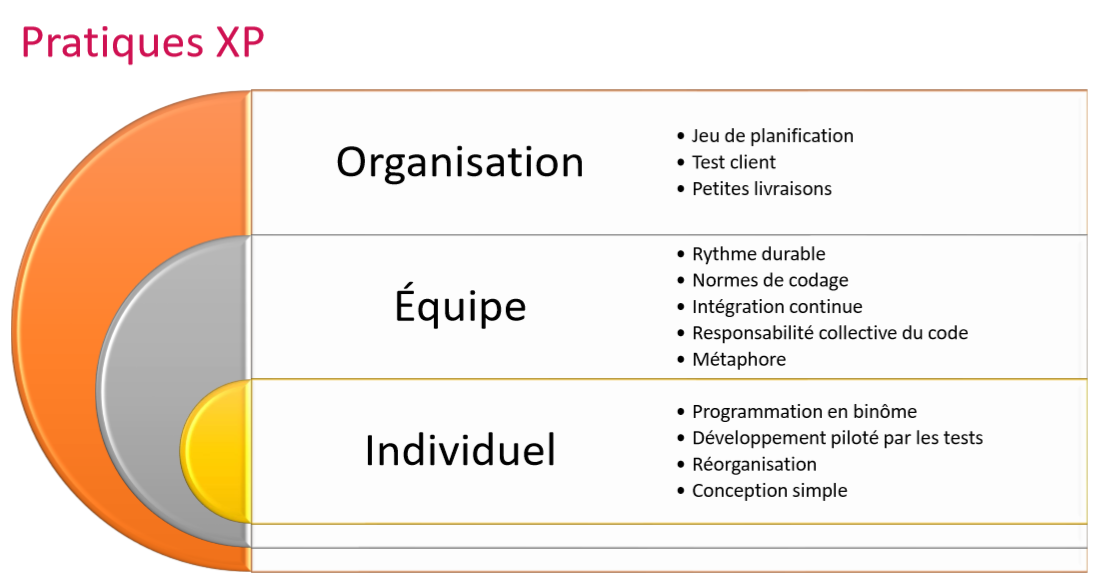
* **Simplicité :** faites les choses les plus simple qui puissent marcher.
* **Feedback :** retour d’information aide les développeurs á améliorer et réviser leur pratique.
* **Communication :** discussions en face á face.
* **Respect :** les membres de l’équipe doivent se respecter mutuellement.
* **Courage :** Les équipes doivent avoir le courage de soulever de leurs problèmes organisationnels

**Principes XP :**

* Le client (MOA) gère lui-même le projet,
* Les cycles itératifs sont extrêmement courts (1 ou 2 semaines).
* L'équipe livre une première version très tôt dans le projet.
* Les livraisons se succèdent pour obtenir un retour d'information maximal
* L'équipe s'organise pour atteindre ses objectifs
* L'équipe met en place des tests automatiques pour chaque fonctionnalité développée
* Les développeurs améliorent constamment la structure interne du logiciel.

**Les acteur XP (RÔLES) :**

* **Programmeur :** rôle centrale, en même temps codeur, testeur, concepteur et analyste, il doit avoir la compétence de l’apprentissage.
* **Client :** représentant des utilisateurs finaux, il décrit les fonctionnalités et l’apporteur des changements des besoins.
* **Testeur :** il définit et automatise les tests d’acceptation, comme il conseille le client sur la testabilité d’une fonctionnalité.
* **Tracker :** suivre les taches, et le respect du temps de la réalisation, ainsi qu’il doit découvrir les anomalies avant qu’il ne soit trop tard.
* **Directeur :** Superviseur des programmeurs, qui exige la responsabilité et il prend soin de l’équipe sur le plan matériel.
* **Coach**: garant du processus qui vérifie que chaque rôle est bien respecté.



**Pratique XP individuel :**

* **Développement piloté par les tests :** les tests unitaires sont automatisables, et que les tests de recette sont spécifiés par le client, et finalement les tests fonctionnels sont écrits par le testeur.
* **Programmation en binôme :** deux développeurs devant la machine pour finir les taches plus rapidement et de s’amuser pendant le développement.
* **Refactoring**: améliorer la structure interne d’un logiciel sans modifier son comportement externe, et transformation du code.
* **Conception simple :** mettre en œuvre le strict nécessaire.

**Pratique XP de l’équipe :**

* **Rythme durable (Équipe) :** semaines de 40 heures, journée de 8 heures, savoir travailler mais aussi savoir se reposer.
* **Propriété collective du code :** code appartient au projet pas aux développeurs, et que le développeur travail sur n’importe quel code
* **Métaphore :** une description générale sur le fonctionnement du système avec l’utilisation du terme métier pour obtenir une vision commune et claire.
* **Règle du codage :** tout le code doit écrit avec la même manière (désigne pattern, règle de nommage …)
* **Intégration continue :** éviter les longue période d’intégration, gestion des version ….

**Pratique XP de l’organisation :**

* **Client sur place (organisation) :** client intégré á l’équipe de développement, pour valider les tests ou pour fournir une expertise commerciale ou retenir la gestion de projet.
* **Livraisons fréquentes**: laisse le client fixe la date du projet et la date entre 2 livraisons (max 3 mois), les livraisons doivent être le plus proche possible.
* **Jeu de planification :** livraison (Scénario client) se compose de 2 ou 3 itérations (tache á effectuer par le développeur) et que chaque itération se compose de 3 phase
  + **Exploration :** identification du travail et estimation de son coût.
  + **Engagement :** sélection les travaux pour le cycle en cours.
  + **Pilotage :** contrôle de la réalisation de ce qui requis.

**Les différent classes d’UML :**

* **Diagramme de classes (class diagram) :** représentation des classes intervenant dans le système.
* **Diagramme d'objets (object diagram) :** représentation des instances de classes (objets) utilisées dans le système.
* **Diagramme de composants (component diagram) :** représentation des composants du système d'un point de vue physique, tels qu'ils sont mis en œuvre (fichiers, bibliothèques, bases de données…)
* **Diagramme de déploiement (deployment diagram) :** représentation des éléments matériels (ordinateurs, périphériques, réseaux, systèmes de stockage…) et la manière dont les composants du système sont répartis sur ces éléments matériels et interagissent entre eux.
* **Diagramme des paquets (package diagram) :** représentation des dépendances entre les paquets (un paquet étant un conteneur logique permettant de regrouper et d'organiser les éléments dans le modèle UML), c'est-à-dire entre les ensembles de définitions.
* **Diagramme de structure composite (composite structure diagram) :** représentation sous forme de boîte blanche des relations entre composants d'une classe (depuis UML 2.x).
* **Diagramme de profils (profile diagram) :** spécialisation et personnalisation pour un domaine particulier d'un méta-modèle de référence d'UML (depuis UML 2.2).

**Les diagrammes de comportement (behavior diagrams) rassemblent :**

* **Diagramme des cas d'utilisation (use-case diagram) :** représentation des possibilités d'interaction entre le système et les acteurs (intervenants extérieurs au système), c'est-à-dire de toutes les fonctionnalités que doit fournir le système.
* **Diagramme états-transitions (state machine diagram) :** représentation sous forme de machine à états finis du comportement du système ou de ses composants.
* **Diagramme d'activité (activity diagram) :** représentation sous forme de flux ou d'enchaînement d'activités du comportement du système ou de ses composants.

**Les diagrammes d'interaction ou diagrammes dynamiques**

* **Diagramme de séquence (sequence diagram) :** représentation de façon séquentielle du déroulement des traitements et des interactions entre les éléments du système et/ou de ses acteurs.
* **Diagramme de communication (communication diagram) :** représentation de façon simplifiée d'un diagramme de séquence se concentrant sur les échanges de messages entre les objets (depuis UML 2.x).
* **Diagramme global d'interaction (interaction overview diagram) :** représentation des enchaînements possibles entre les scénarios préalablement identifiés sous forme de diagrammes de séquences (variante du diagramme d'activité) (depuis UML 2.x).
* **Diagramme de temps (timing diagram) :** représentation des variations d'une donnée au cours du temps (depuis UML 2.3).

