UML

Table des matières

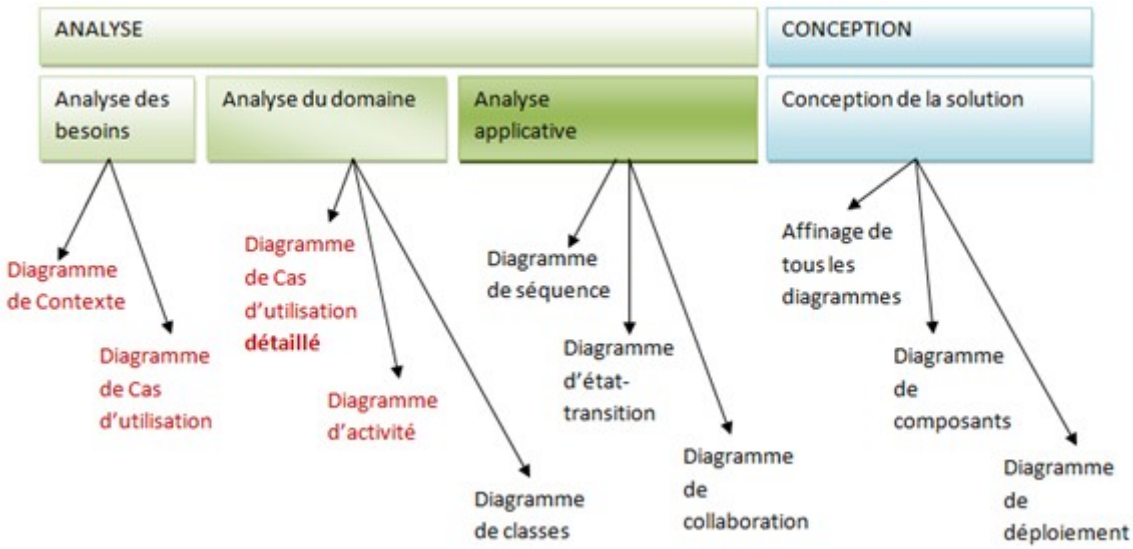
UML est un langage de modélisation graphique à base de pictogrammes conçu pour fournir une méthode normalisée pour visualiser la conception d'un système. Il est couramment utilisé en développement logiciel et en conception orientée objet.

1. Introduction

La notation UML[1] est un langage visuel constitué d'un ensemble de schémas, appelés des diagrammes, qui donnent chacun une vision différente du projet à traiter. UML fournit donc des diagrammes pour représenter le logiciel à développer : son fonctionnement, sa mise en route, les actions susceptibles d'être effectuées par le logiciel, etc.

Comme n'importe quel type de projet, un projet informatique nécessite une phase d'analyse, suivi d'une étape de conception.

- Dans la phase d'analyse, on cherche d'abord à comprendre et à décrire de façon précise les besoins des utilisateurs. Que souhaitent-ils faire avec le logiciel ? Quelles fonctionnalités veulent-ils ? Pour quel usage ? Comment l'action devrait-elle fonctionner ? C'est ce qu'on appelle « l'analyse des besoins ». Après validation de notre compréhension du besoin, nous imaginons la solution. C'est la partie analyse de la solution.
- Dans la phase de conception, on apporte plus de détails à la solution et on cherche à clarifier des aspects techniques, tels que l'installation des différentes parties logicielles à installer sur du matériel.



2. ArgoUML

ArgoUML est un logiciel libre de création de diagrammes UML. Il supporte sept types de diagramme : cas d'utilisation, classes, séquence, état, collaboration, activité et déploiement.

La génération de code à partir de diagrammes de classes est supportée dans les langages suivants : Java, C++, PHP, C# et SQL.

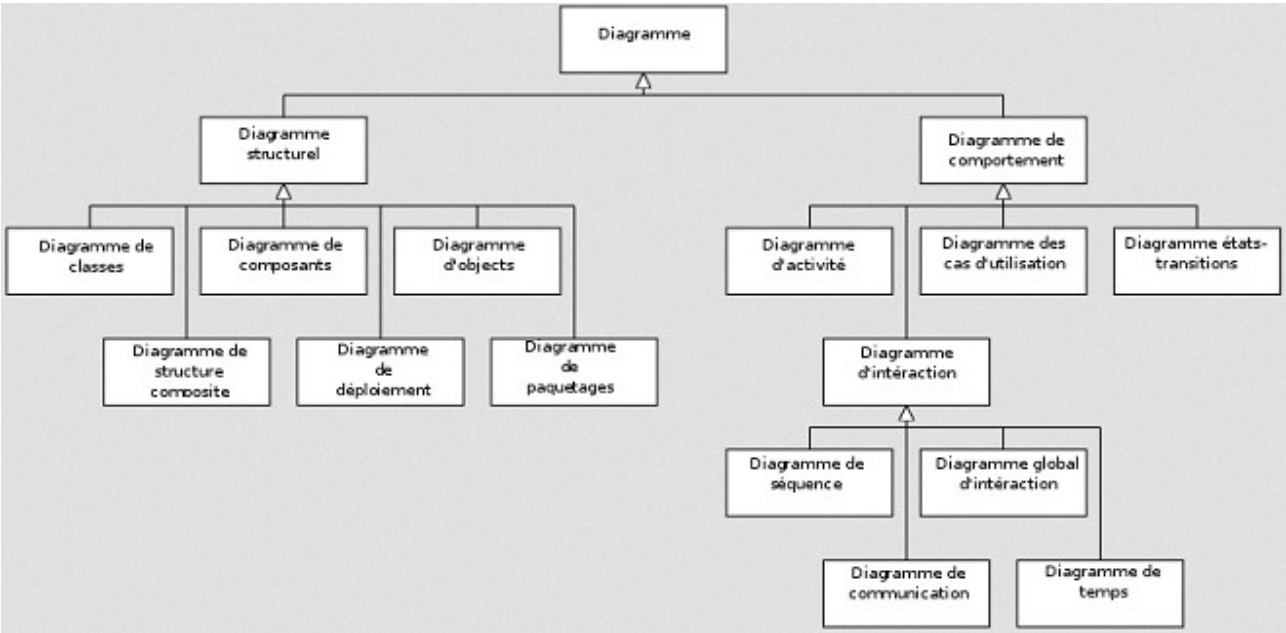
Vous pouvez télécharger ArgoUML à cette adresse.

3. Les différents types de diagrammes

UML se décompose en plusieurs sous-ensembles :

1 Unified Modeling Language : Langage de modélisation unifié

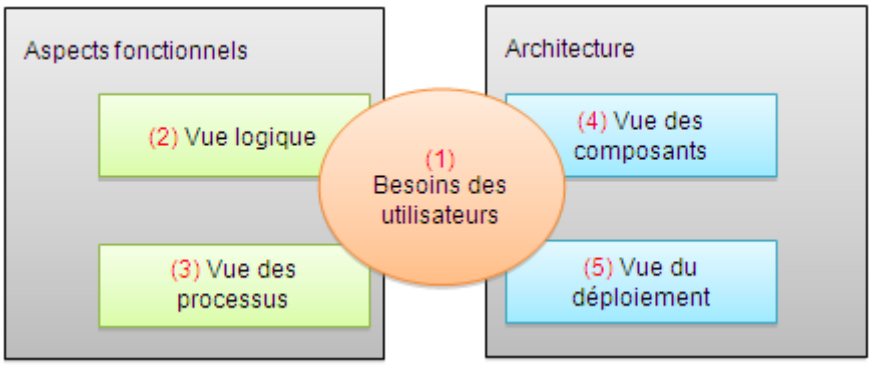
- Les vues : les vues sont les observables du système. Elles décrivent le système d'un point de vue donné, qui peut être organisationnel, dynamique, temporel, architectural, géographique, logique, etc. En combinant toutes ces vues, il est possible de définir (ou retrouver) le système complet.
- Les diagrammes : les diagrammes sont des éléments graphiques. Ceux-ci décrivent le contenu des vues, qui sont des notions abstraites. Les diagrammes peuvent faire partie de plusieurs vues.
- Les modèles d'élément : les modèles d'élément sont les briques des diagrammes UML, ces modèles sont utilisés dans plusieurs types de diagrammes.



Le langage UML est constitué de 13 diagrammes réalisés à partir du besoin des utilisateurs et peuvent être regroupés selon les deux aspects suivants :

- Les aspects fonctionnels : Qui utilisera le logiciel et pour quoi faire ? Comment les actions devront-elles se dérouler ? Quelles informations seront utilisées pour cela ?
- Les aspects liés à l'architecture : Quels seront les différents composants logiciels à utiliser (base de données, librairies, interfaces, etc.) ? Sur quel matériel chacun des composants sera installé ?

Ces deux aspects sont représentés par le schéma de 4 vues, axées sur les besoins des utilisateurs (parfois intitulé des cas d'utilisation), appelé 4+1 vues.



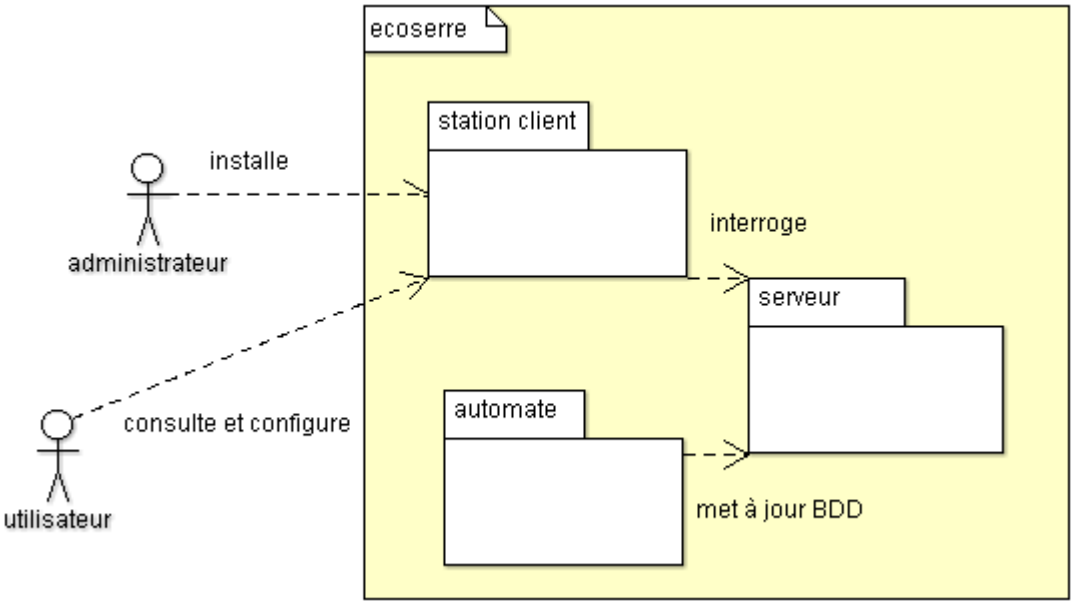
3.1. Besoins des utilisateurs

Cette partie représente le cœur de l'analyse. Il est composé de cas d'utilisation qui décrit le contexte, les acteurs ou utilisateurs du projet logiciel, les fonctionnalités du logiciel mais aussi les interactions entre ces acteurs et ces fonctionnalités.

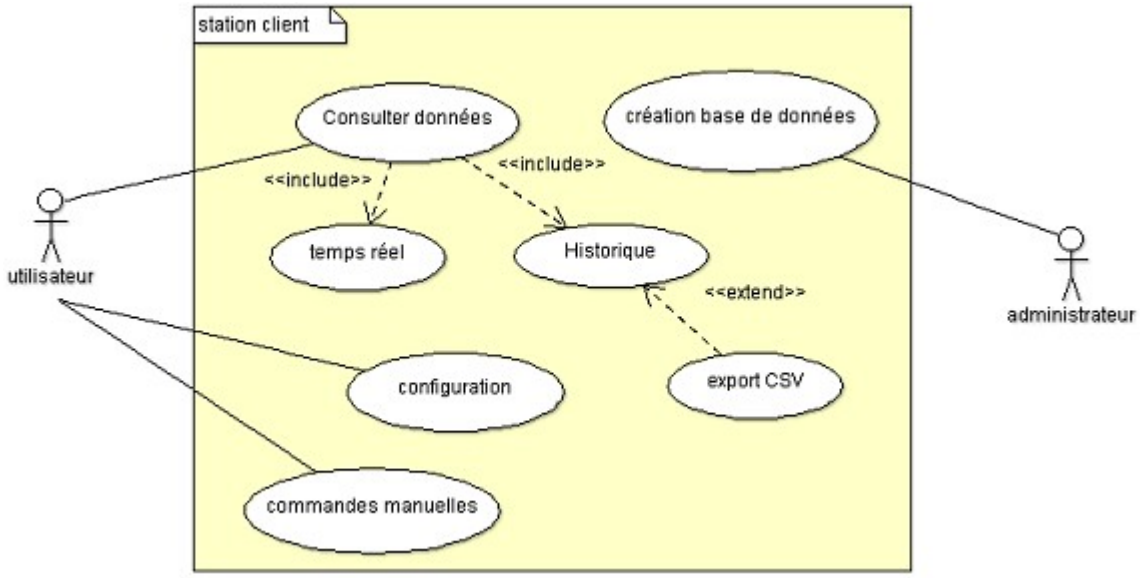
Le besoin des utilisateurs peut être décrit à l'aide de deux diagrammes

- Le diagramme de packages permet de décomposer le système en catégories ou parties plus facilement observables, appelés « packages ». Cela permet également d'indiquer les acteurs qui interviennent dans chacun des packages.

  La boîte qui entoure les packages correspond au système (c'est-à-dire le logiciel) qui est analysé.



- Le diagramme de cas d'utilisation représente les fonctionnalités (ou dit cas d'utilisation) nécessaires aux utilisateurs. On peut faire un diagramme de cas d'utilisation pour le logiciel entier ou pour chaque package.



Plusieurs types de relations sont prises en charge :

- Les inclusions : dans ce type d'interaction, le premier cas d'utilisation inclus le second et son issue dépend souvent de la résolution du second. Elle est représentée par une flèche en pointillée et le terme include.
- Les extensions : elles représentent des prolongements logiques de certaines tâches sous certaines conditions. Autrement dit un cas d'utilisation A étend un cas d'utilisation B lorsque le cas d'utilisation A peut être appelé au cours de l'exécution du cas d'utilisation B. Elle est représentée par une flèche en pointillée avec le terme Extend.

3.2. Aspects fonctionnels

Cette partie du schéma 4+1 vues permet de définir qui utilisera le logiciel et pour quoi faire, comment les fonctionnalités vont se déroulement, etc.

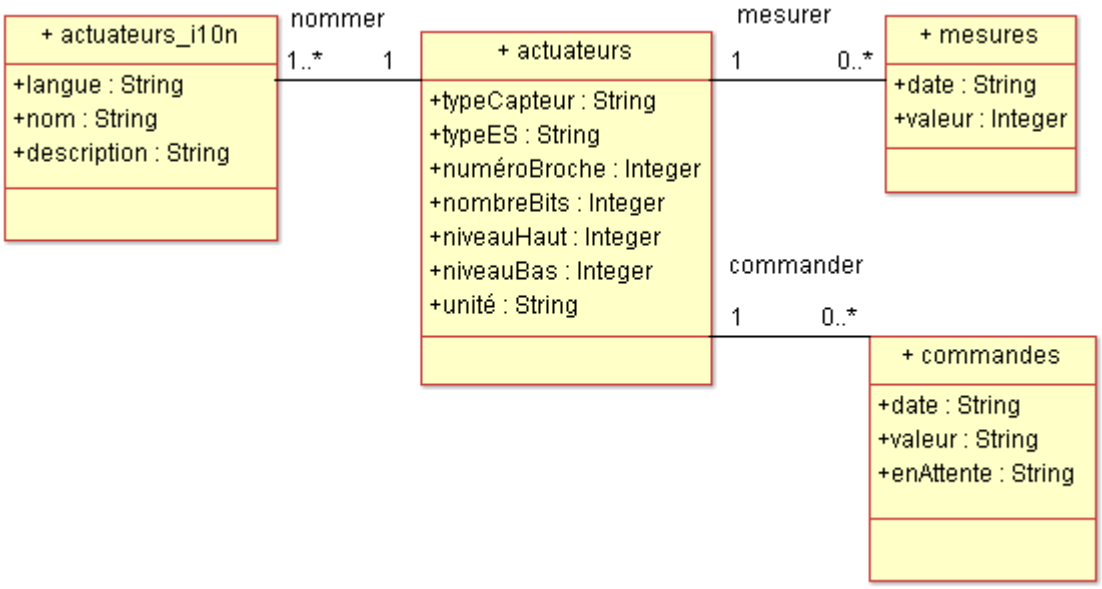
3.2.1. Vue logique

La vue logique a pour but d'identifier les éléments du domaine, les relations et interactions entre ces éléments. Cette vue organise les éléments du domaine en « catégories ». Deux diagrammes peuvent être utilisés pour cette vue.

- Le diagramme de classes

  Dans la phase d'analyse, ce diagramme représente les entités (des informations) manipulées par les utilisateurs.

  Dans la phase de conception, il représente la structure objet d'un développement orienté objet.



Une classe est représentée par un rectangle séparée en trois parties :

1. la première partie contient le nom de la classe
2. la seconde contient les attributs de la classe
3. la dernière contient les méthodes de la classe

La seconde et la dernière représentent le comportement de la classe.

- Le diagramme d'objets sert à illustrer les classes complexes en utilisant des exemples d'instances.

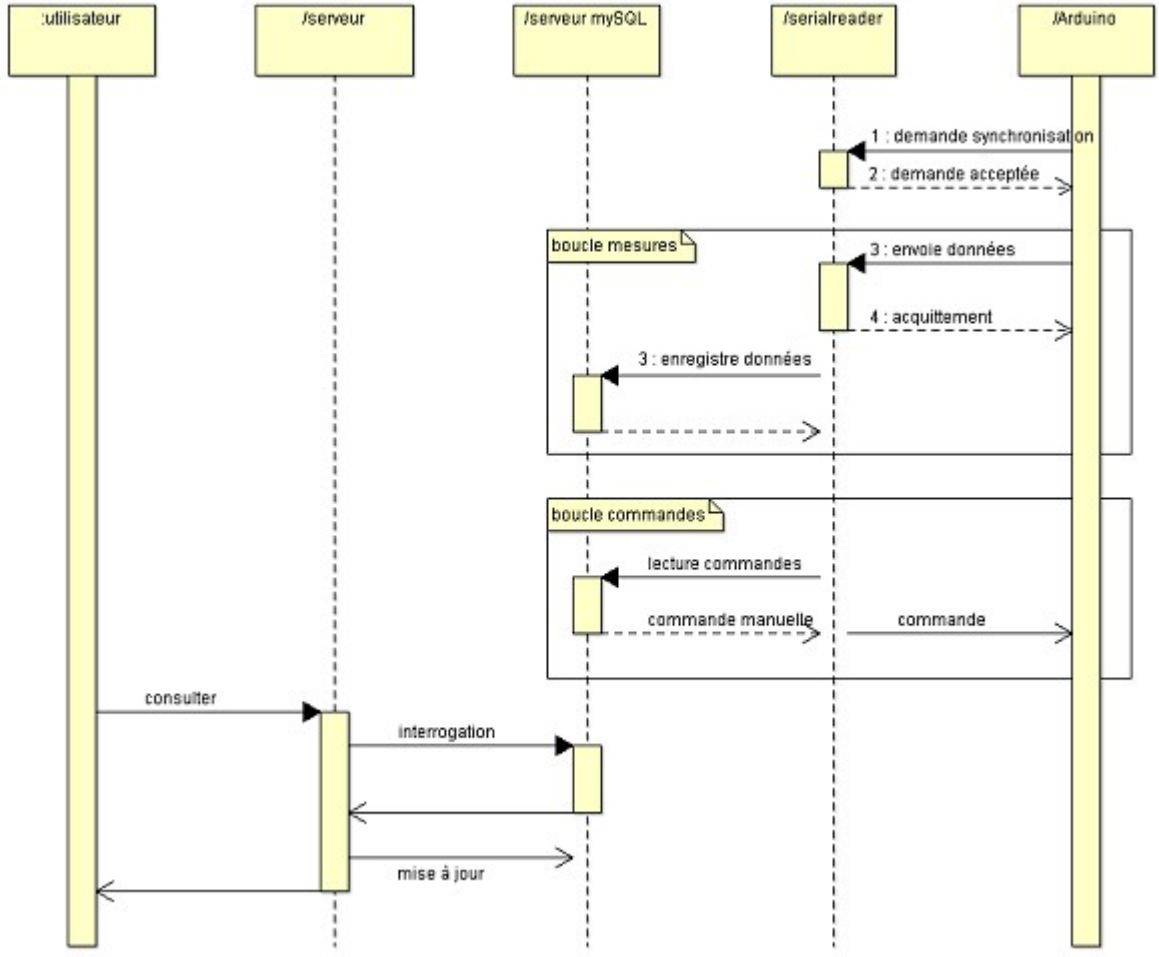
  Une instance est un exemple concret de contenu d'une classe. En illustrant une partie des classes avec des exemples (grâce à un diagramme d'objets), on arrive à voir un peu plus clairement les liens nécessaires.

3.2.2. Vue des processus

La vue des processus démontre la décomposition du système en processus et actions, les interactions entre les processus et la synchronisation et la communication des activités parallèles.

La vue des processus s'appuie sur plusieurs diagrammes.

- Le diagramme de séquence permet de décrire les différents scénarios d'utilisation du système.



La dimension verticale du diagramme représente le temps, permettant de visualiser l'enchaînement des actions dans le temps, et de spécifier la naissance et la mort d'objets. Les périodes d'activité des objets sont symbolisées par des rectangles, et ces objets dialoguent par le biais de messages.

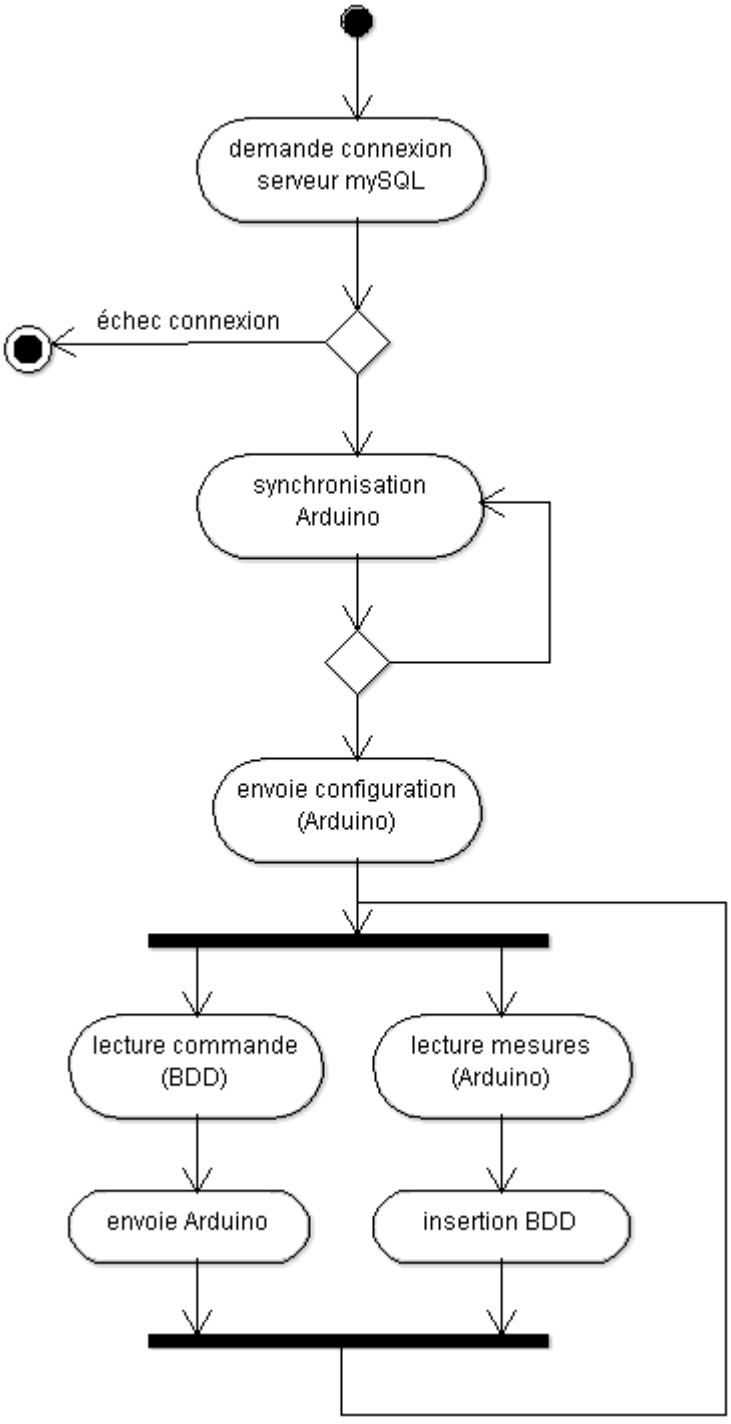
Plusieurs types de messages (actions) peuvent transiter entre les acteurs et objets :

- message simple : le message n'a pas de spécificité particulière d'envoi et de réception.
- message avec durée de vie : l'expéditeur attend une réponse du récepteur pendant un

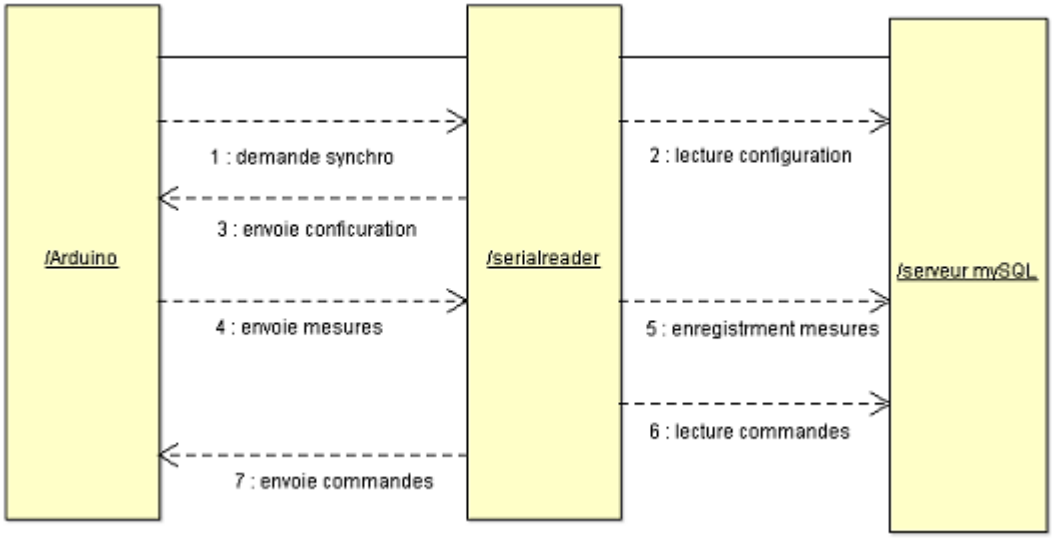
certain temps et reprend ses activités si aucune réponse n'a lieu dans un délai prévu.

- message synchrone : l'expéditeur est bloqué jusqu'au signal de prise en compte par le destinataire. Les messages synchrones sont symbolisés par des flèches barrées.
- message asynchrone : le message est envoyé, l'expéditeur continue son activité que le message soit parvenu ou pris en compte ou non. Les messages asynchrones sont symbolisés par des demi-flèches.

- Le diagramme d'activité représente le déroulement des actions, sans utiliser les objets. En phase d'analyse, il est utilisé pour consolider les spécifications d'un cas d'utilisation.



- Le diagramme de collaboration (appelé également diagramme de communication) permet de mettre en évidence les échanges de messages entre objets. Cela nous aide à voir clair dans les actions qui sont nécessaires pour produire ces échanges de messages. Et donc de compléter, si besoin, les diagrammes de séquence et de classes.



- Le diagramme d'état-transition permet de décrire le cycle de vie des objets d'une classe.
- Le diagramme global d'interaction permet de donner une vue d'ensemble des interactions du système. Il est réalisé avec le même graphisme que le diagramme d'activité. Chaque élément du diagramme peut ensuite être détaillé à l'aide d'un diagramme de séquence ou d'un diagramme d'activité.
- Le diagramme de temps est destiné à l'analyse et la conception de systèmes ayant des contraintes temps-réel. Il s'agit là de décrire les interactions entre objets avec des contraintes temporelles fortes.

3.2.3. Vue des composants

La vue des composants (vue de réalisation) met en évidence les différentes parties qui composeront le futur système (fichiers sources, bibliothèques, bases de données, exécutables, etc.). Cette vue comprend deux diagrammes.

- Le diagramme de structure composite décrit un objet complexe lors de son exécution.
- Le diagramme de composants décrit tous les composants utiles à l'exécution du système (applications, librairies, instances de base de données, exécutables, etc.).

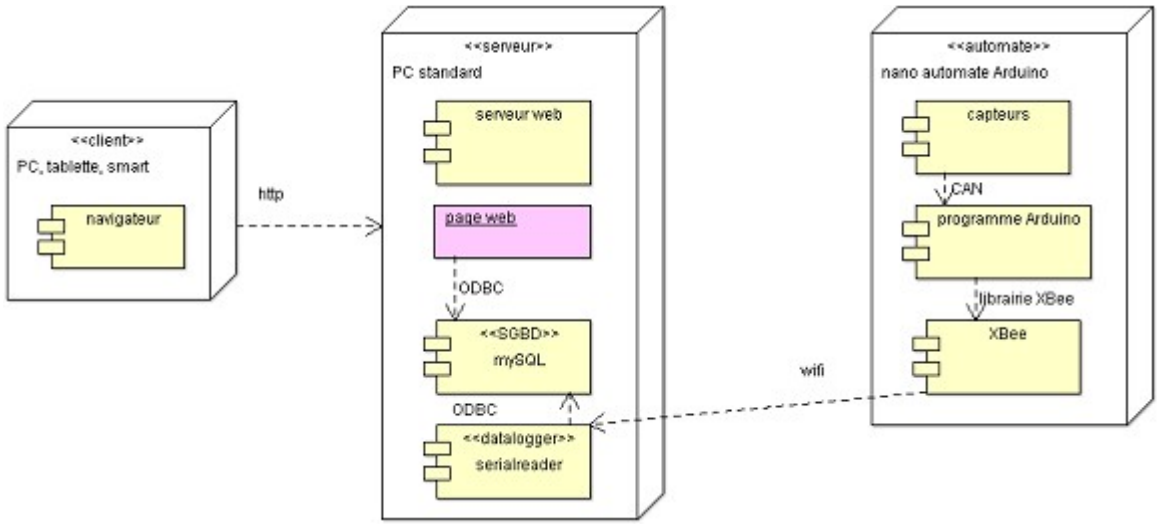
3.2.4. Vue de déploiement

La vue de déploiement décrit les ressources matérielles et la répartition des parties du logiciel sur ces éléments. Il contient un diagramme :

Le diagramme de déploiement correspond à la description de l'environnement d'exécution du système (matériel, réseau…) et de la façon dont les composants y sont installés. Les caractéristiques des ressources matérielles physiques et des supports de communication peuvent être précisées par stéréotype.

- **Nœuds** : Les nœuds (nodes), représentés par des cubes en trois dimensions, sont des composants mécaniques de l'infrastructure tel un routeur, un ordinateur, un assistant personnel, ... Ceux-ci peuvent comprendre d'autre nœuds ou artefacts. Les nœuds internes indiquent l'environnement d'exécution plutôt que l'infrastucture physique.

- **Composants** : Les composants, représentés par des boites rectangulaires avec deux rectangles sortant du côté gauche, sont les différentes parties du système étudié.
- **Connexions** : Les connexions sont principalement de deux types : associations ou dépendances.
  - Associations : Les associations, représentées par de simples lignes sont des liens de communication, s'établissent entre les différents composants du système.
  - Dépendances : Les dépendances, représentées par des flèches vides.
- **Artefacts** : Dans ce contexte, un artefact est une manière de définir un fichier, un programme, une bibliothèque ou une base de données construite ou modifiée dans un projet.



4. La démarche

4.1. Analyse du besoin

Analyser les besoins, c'est découvrir des éléments de plus en plus précis.

4.1.1. Étape 1

On commence par décrire le contexte du logiciel à créer. Il s'agit là de décrire QUI devra utiliser le logiciel.

Pour décrire le contexte, on réalise un diagramme de contexte dans lequel on indiquera quel acteur interagit avec le logiciel.

Nous distinguons :

- les acteurs principaux agissent directement sur le système. Il s'agit d'entités qui ont des besoins d'utilisation du système.
- les acteurs secondaires n'ont pas de besoin direct d'utilisation. Ils peuvent être soit consultés par le système à développer, soit récepteur d'informations de la part du système. Cela est généralement un autre système (logiciel) avec lequel il doit échanger des informations.

4.1.2. Étape 2

On décompose ensuite le logiciel en plusieurs parties distinctes : la décomposition en packages. La décomposition peut se faire en réfléchissant à des « familles » de fonctionnalités qui seraient nécessaires.

- Quelles sont les grandes parties qui doivent composer le logiciel ?
- Pour une partie précise, qui, parmi les acteurs identifiés l'utilisera ?

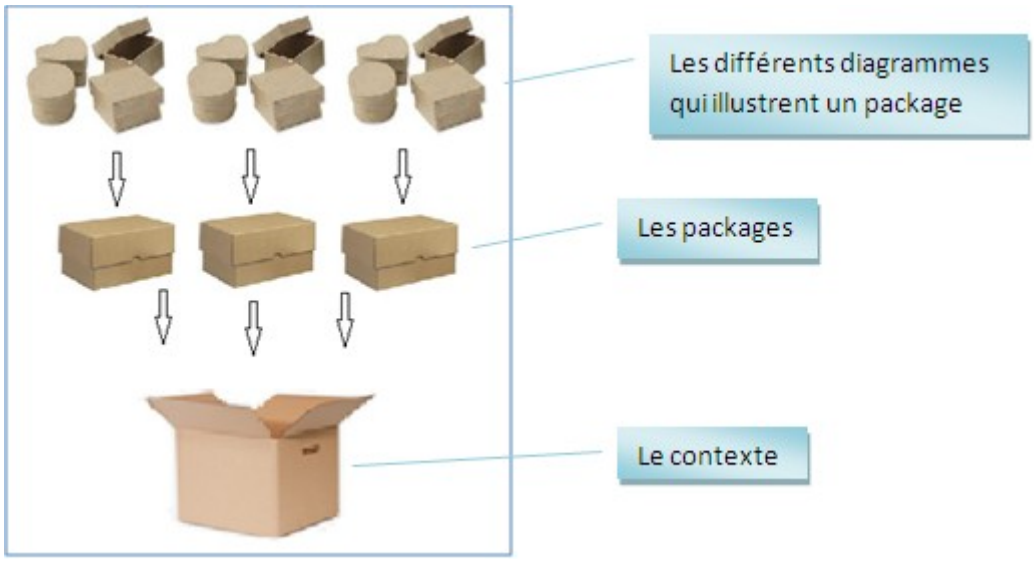
Pour expliquer la décomposition du logiciel en parties distinctes, on se sert d'un diagramme de packages. Celui-ci permet d'indiquer qui aura besoin de chacune des parties, en mettant en évidence les acteurs qui interviennent dans chacun de ces packages.

4.1.3. Étape 3

Nous avons identifié les acteurs, et les grandes familles de fonctionnalités (packages). Maintenant, il s'agit de définir plus en détail les besoins de chaque acteur dans ces deux packages. C'est ce qu'on appelle définir les cas d'utilisation.

On utilise un diagramme des cas d'utilisation qui met en évidence de quelle façon les acteurs utiliseront le logiciel : QUI doit pouvoir faire QUOI ?

Chaque package est donc une boîte qu'il faudra ouvrir pour en découvrir le contenu. Le contenu d'un package est illustré par différents diagrammes. Un de ces diagrammes représente les cas d'utilisation, c'est-à-dire les fonctionnalités ou lots d'actions que devront réaliser les acteurs.



4.2. Analyse du domaine

Il s'agit ici de détailler les cas d'utilisation principaux afin de comprendre tous les lots d'actions qu'ils impliquent. On parle donc les fonctionnalités principales du logiciel à développer.

4.2.1. Cas d'utilisation interne

Chacun de ces cas d'utilisation nécessitera un certain nombre d'actions. Il arrive que l'on doive regrouper certaines actions dans un ou plusieurs cas d'utilisation complémentaires qui ne sont pas directement liés à un acteur. On parle alors d'un cas d'utilisation interne.

Il s'agit essentiellement :

- de cas d'utilisation interne qui peuvent être nécessaires à plusieurs autres cas

d'utilisation (qu'ils soient des cas d'utilisation principaux, ou même d'autres cas d'utilisation interne) ;

- de cas d'utilisation qui regroupent un certain nombre d'actions qui forment un tout homogène et pouvant être décrit séparément.

Pour trouver les cas d'utilisation internes, il faut réfléchir à chaque cas d'utilisation déjà indiqué dans le diagramme.

Les liens (ou relations) entre cas d'utilisation peuvent être divisés en deux groupes :

- La spécialisation
- Les relations stéréotypées

4.2.2. La spécialisation

La spécialisation peut être indiquée à deux niveaux :

- au niveau des acteurs ;
- au niveau des cas d'utilisation principaux.

Cela se justifie si plusieurs acteurs ont besoin d'un ou de plusieurs cas d'utilisation communs et qu'ils ont également besoin de cas d'utilisation spécifiques.

On utilise alors :

- un acteur générique qui est lié aux cas d'utilisations communs ;
- des acteurs spécialisés qui sont liés à des cas d'utilisation spécifiques.

On indique qu'un acteur est la spécialisation d'un autre en dessinant une flèche à pointe fermée vers l'acteur principal.

4.2.3. Les relations stéréotypées

Les cas d'utilisation internes sont toujours reliés, par une flèche, au cas d'utilisation initial qui en besoin. Le cas d'utilisation initial peut être un cas d'utilisation principal (donc directement relié à un acteur) ou à un autre cas d'utilisation interne. Ces relations sont appelées des relations stéréotypées car ils définissent le type de lien entre les cas d'utilisation, qui peuvent être de deux sortes :

- les relations « include » ;

  Une relation « include » est utilisée pour indiquer que le cas d'utilisation source (départ de la flèche) contient TOUJOURS le cas d'utilisation inclus. Pour représenter cette relation, on dessine donc une flèche en pointillé partant du cas d'utilisation principal vers le cas d'utilisation interne. Puis, on indique le stéréotype « include » sur la flèche.

- les relations « extend ».

  Cette relation est utilisée pour indiquer que le cas d'utilisation source (à l'origine de la flèche) n'est pas toujours nécessaire au cas d'utilisation principal, mais qu'il peut l'être dans certaines situations.

  L'ajout de cette relation se fait en dessinant une flèche en pointillé partant du cas d'utilisation interne vers le cas d'utilisation principal. Puis, on indique le stéréotype « extend » sur la flèche. Dès lors qu'il y a une relation « extend », il faudra toujours définir la condition, c'est-à-dire : à quelle condition cette relation peut avoir lieu ? Pour indiquer cela, il faut ajouter une ligne « extension points » et définir la condition « EXT ».

4.2.4. Le diagramme d'activité

Le diagramme d'activité est une représentation visuelle des descriptions détaillées des cas

d'utilisation.

Le diagramme est composé d'un point de démarrage, d'un point arrêt et d'action, qui sont représentés par des cercles rouges.

Il est organisé en actions réalisées soit par un acteur, soit par le système, relié par une flèche indiquant l'enchaînement des actions.

Si une action du cas d'utilisation correspond à l'appel d'un cas d'utilisation interne (lié par une relation de type « include » ou « extend ») ; elle est représentée par une action contenant un signe spécial : deux cercles reliés par un trait.

L'alternative permet d'indiquer les différents scénarios du cas d'utilisation dans un même diagramme.

La synchronisation indique qu'il faut avoir réalisé deux actions pour pouvoir réaliser la troisième en-dessous.

Les couloirs permettent d'indiquer qui (de l'utilisateur ou du système) réalise les actions.