LES DIFFÉRENTS TYPES DE DIAGRAMMES

Analyse et conception orientée objet

www.elbahihassan.com

elbahihassan@gmail.com

ISTA Oulad Teima

A metaphor for good information design is a map. Hold any diagram against a map and see how it compares.

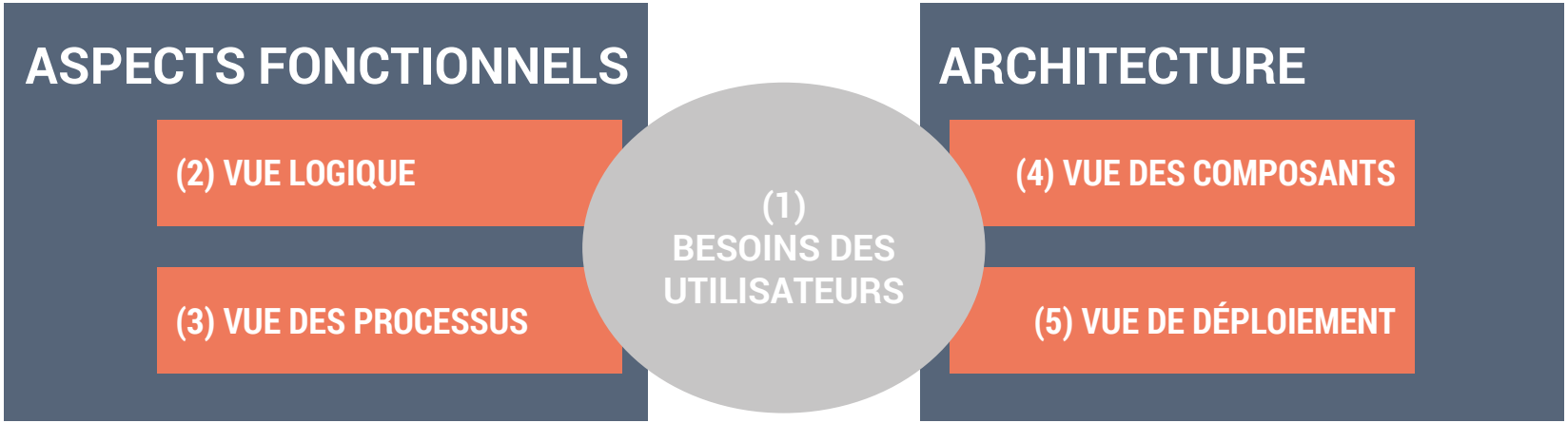
Edward Tufte

Diagrammes

- La réalisation d'une application informatique ou d'un ensemble d'applications est basée sur plusieurs diagrammes.
- À ce jour, il existe **13 diagrammes** « officiels ». Ces diagrammes sont tous 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 ?

Les 4+1 vues d'un système

- La conception d'un logiciel est organisée autour des aspects fonctionnels et d'architecture.
- 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**.



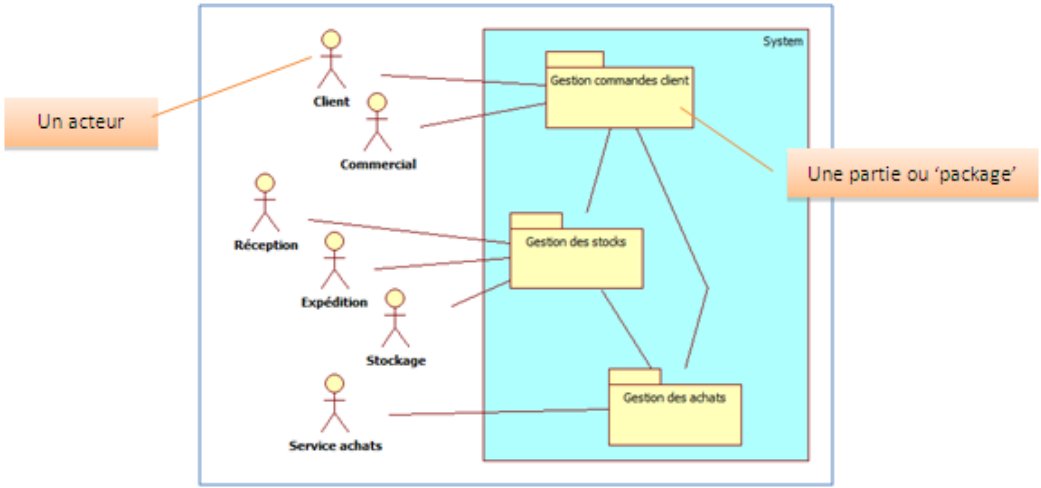
Les besoins des utilisateurs

- Cette partie représente le cœur de l'analyse. Il est composé de cas d'utilisation.
- On y 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
    - Le diagramme de cas d'utilisation

Diagramme de packages

Les besoins des utilisateurs

- 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.



Un exemple de diagramme de package

Diagramme de packages

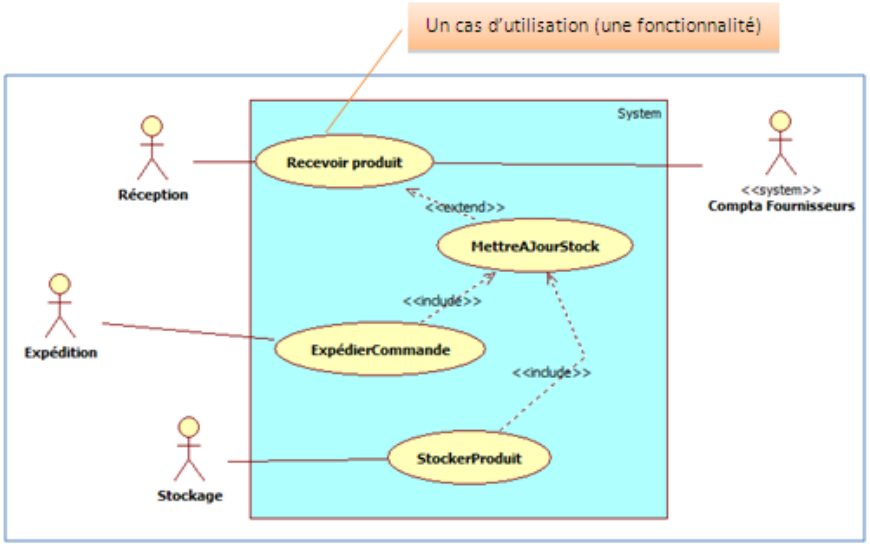
Les besoins des utilisateurs

- Dans l'exemple précédent, nous voyons que le logiciel que nous concevons peut être divisé en trois **parties** (ou packages) observables séparément :
  - La gestion des commandes client
  - La gestion des stocks
  - La gestion des achats
- La boîte qui entoure les packages (la boîte bleue) correspond au **système** (c'est-à-dire le logiciel) qui est analysé.

Diagramme de cas d'utilisation

Les besoins des utilisateurs

- Il 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.



Un exemple de diagramme de cas d'utilisation pour un package (Gestion des stocks)

L’aspect fonctionnel du logiciel

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

- **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.**
  - **Le diagramme d’objets.**

Diagramme de classes

L'aspect fonctionnel du logiciel - La vue logique

- 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.

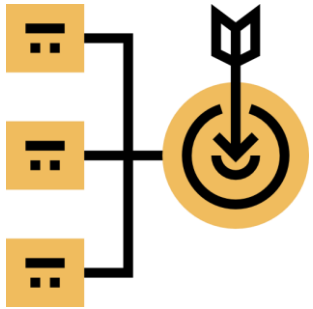
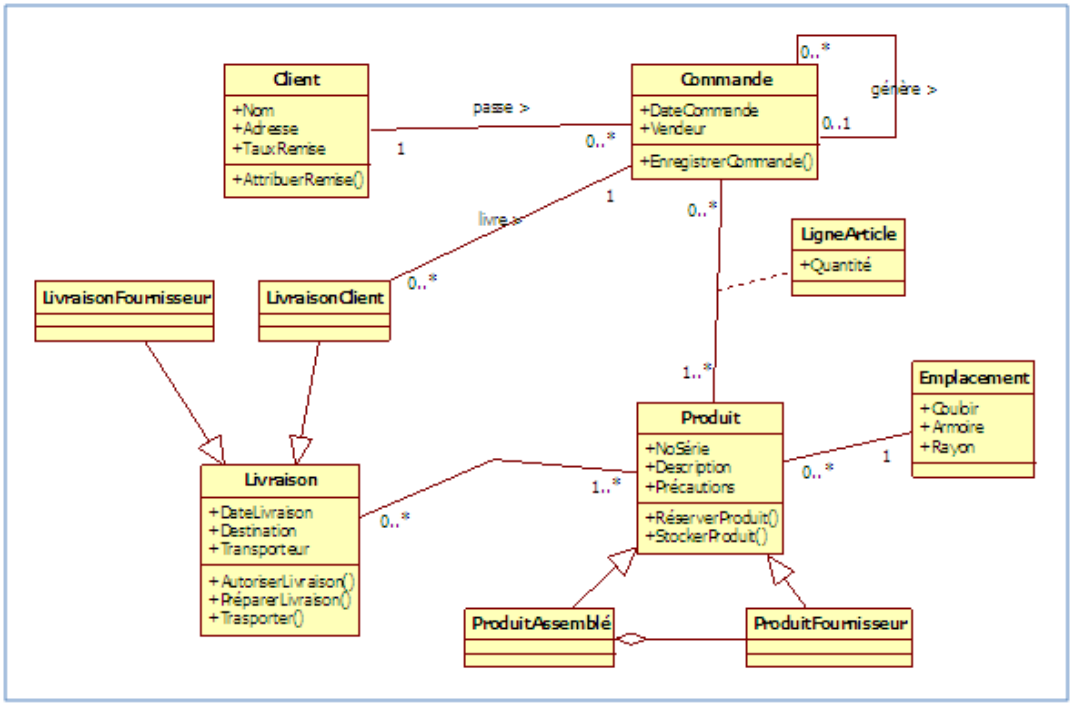


Diagramme de classes

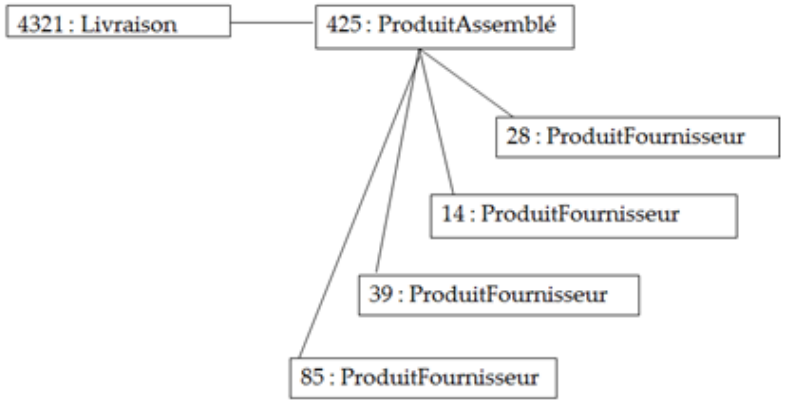
L'aspect fonctionnel du logiciel - La vue logique



Un exemple de diagramme de classes (utilisé en phase d'analyse)

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.



Un exemple de diagramme d'objet

L'aspect fonctionnel du logiciel

La **vue des processus** est la deuxième vue de l'aspect fonctionnel du logiciel. Elle démontre :

- La décomposition du système en processus et actions ;
- Les interactions entre les processus ;
- La synchronisation et la communication des activités parallèles.

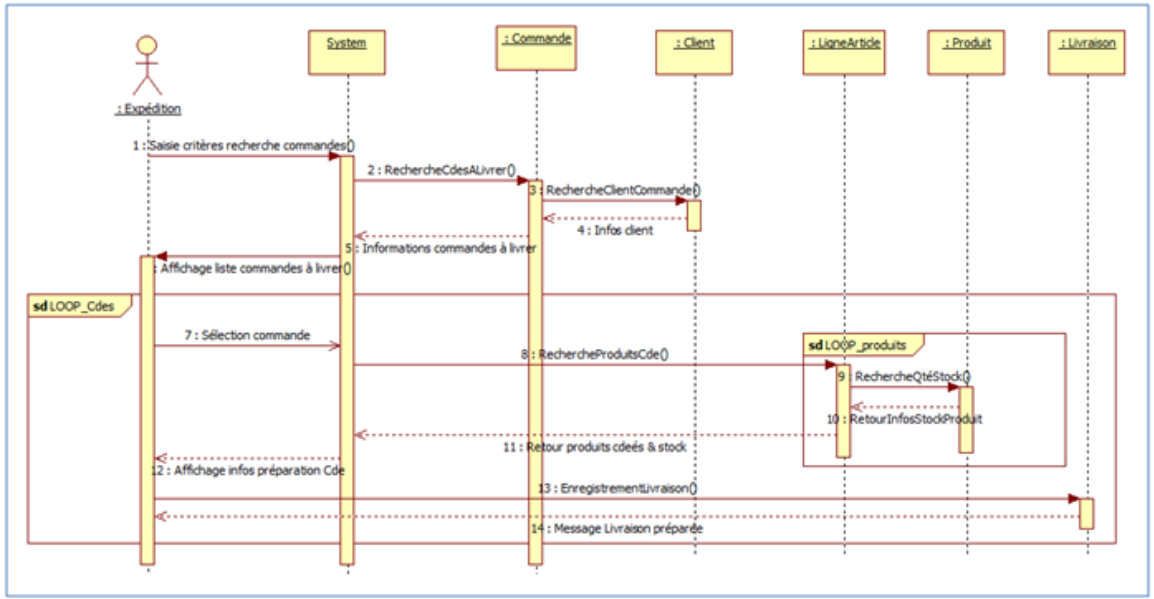
La vue des processus s'appuie sur plusieurs diagrammes.

- Le diagramme de séquence
- Le diagramme d'activité
- Le diagramme de collaboration
- Le diagramme d'état-transition
- Le diagramme global d'interaction
- Le diagramme de temps

Le diagramme de séquence

L'aspect fonctionnel du logiciel - La vue des processus

- Ce diagramme permet de décrire les différents **scénarios** d'utilisation du système.

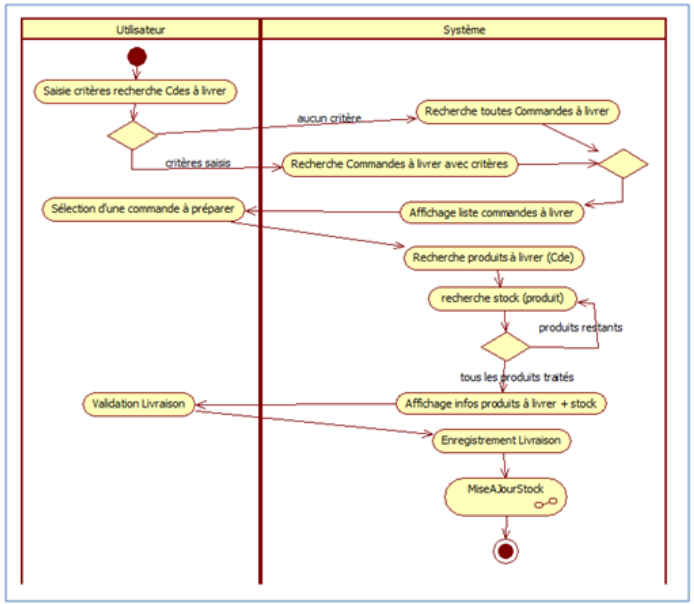


Un exemple de diagramme de séquence

Le diagramme de séquence

L'aspect fonctionnel du logiciel - La vue des processus

- Le diagramme d'activité représente le **déroulement** des actions, sans utiliser les objets.

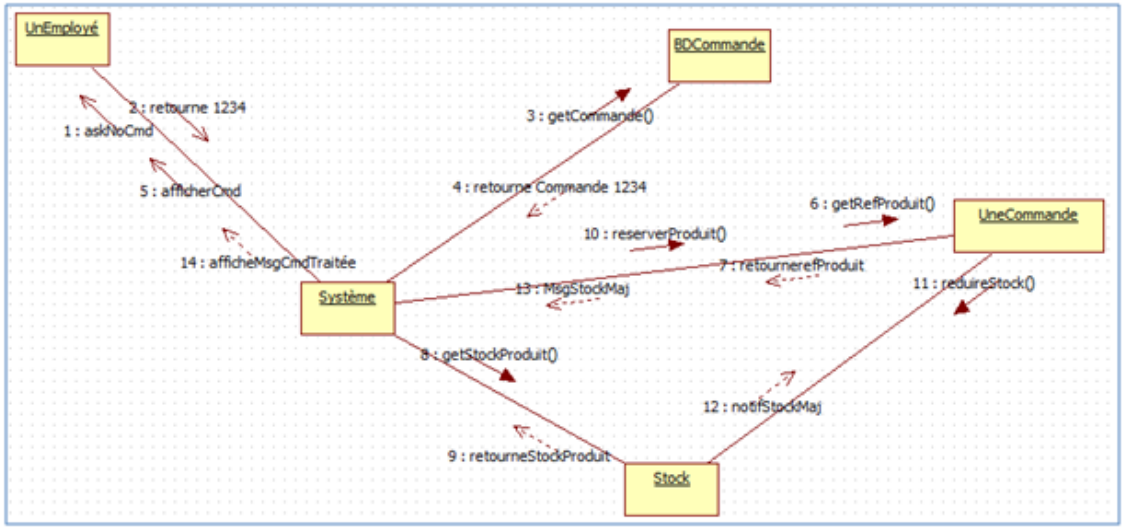


Un exemple de diagramme d'activité

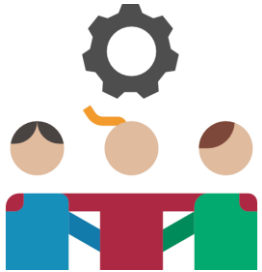
Le diagramme de collaboration

L'aspect fonctionnel du logiciel - La vue des processus

- Il 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.



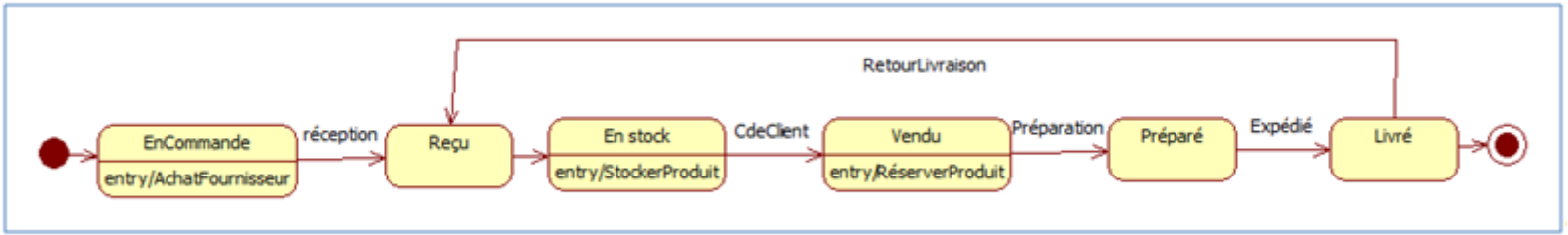
Un exemple de diagramme collaboration



Le diagramme d'état-transition

L'aspect fonctionnel du logiciel - La vue des processus

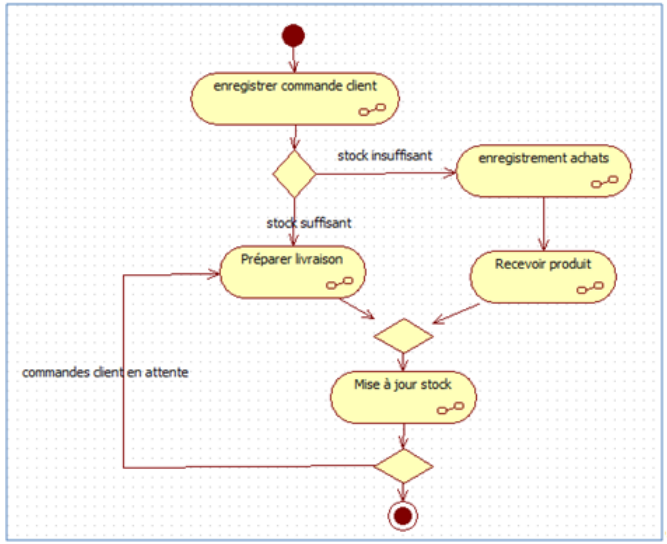
- Le diagramme d'état-transition permet de décrire le **cycle de vie** des objets d'une classe.



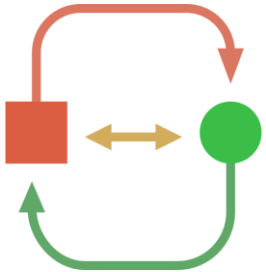
Un exemple de diagramme d'état-transition (objets de la classe produit)

Le diagramme global d'interaction

- Il permet de donner une vue d'ensemble des **interactions** du système. Chaque élément de ce diagramme peut ensuite être détaillé à l'aide d'un diagramme de séquence ou d'un diagramme d'activité.

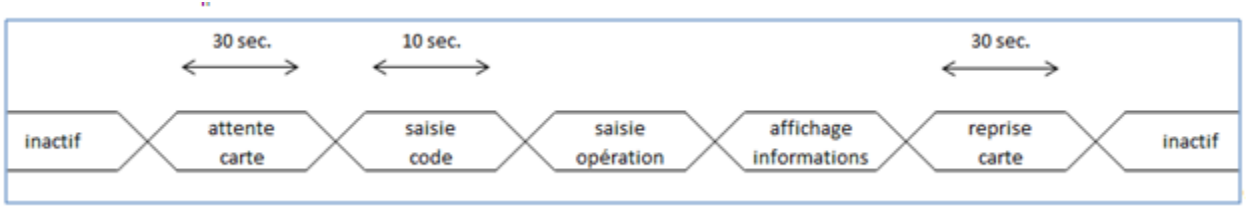


Un exemple de diagramme global d'interaction



Le diagramme de temps

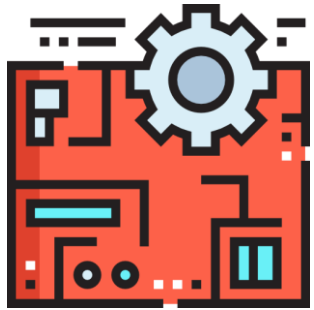
- Il est destiné à l'analyse et la conception de systèmes ayant des contraintes temps-réel.
- Il s'agit là de décrire les interactions entre objets avec des contraintes temporelles fortes.



Un exemple de diagramme de temps avec un seul axe temporel

L’aspect lié à l’architecture du logiciel

- Pour rappel, cette partie du schéma 4+1 vue permet de définir les composantes à utiliser (exécutables, interfaces, base de données, librairies de fonctions, etc.) et les matériels sur lesquels les composants seront déployés.
- Cette aspect contient deux vues :
    - La vue des composants
    - Vue de déploiement



La vue des composants

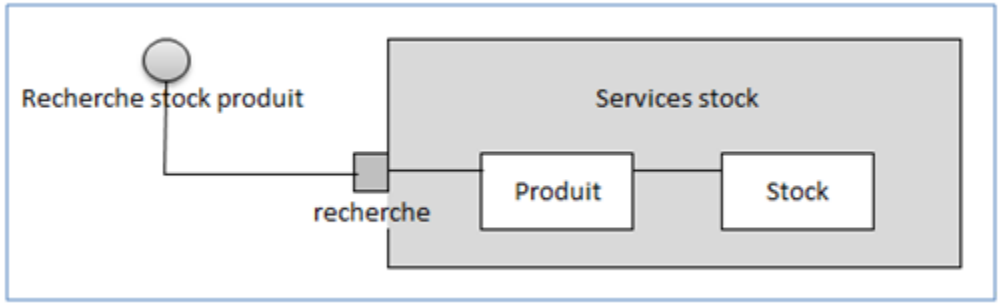
L'aspect lié à l'architecture du logiciel

- 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
  - Le diagramme de composants

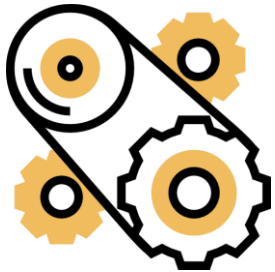
Le diagramme de structure composite

L'aspect lié à l'architecture du logiciel - La vue des composants

- Le diagramme de structure composite décrit un objet complexe lors de son **exécution**.



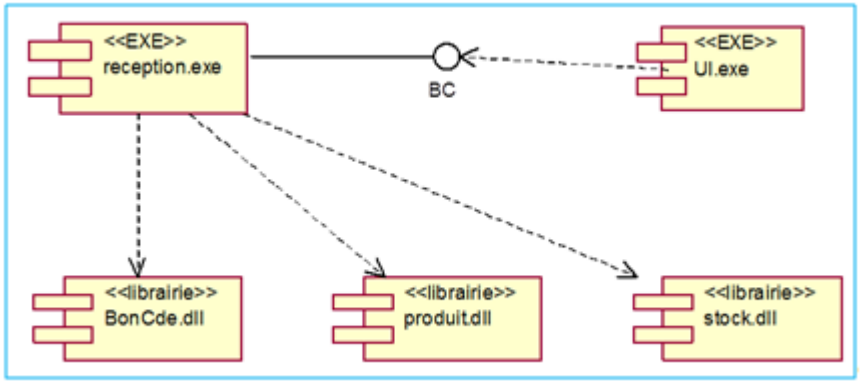
Un exemple de diagramme de structure composite



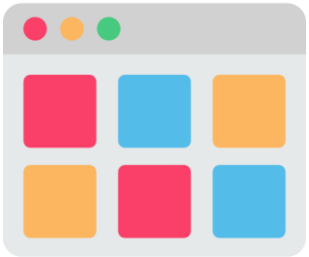
Le diagramme de composants

L'aspect lié à l'architecture du logiciel - La vue des composants

- Le diagramme de composants décrit tous les **composants** utiles à l'exécution du système (applications, librairies, instances de base de données, exécutables, etc.).



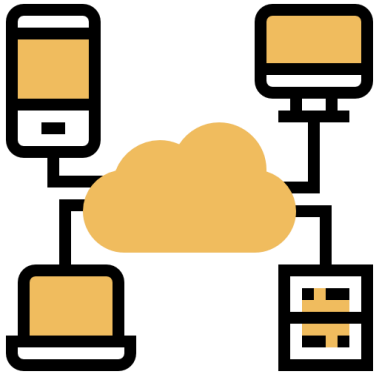
Un exemple de diagramme de composants



La vue des déploiement

L'aspect lié à l'architecture du logiciel

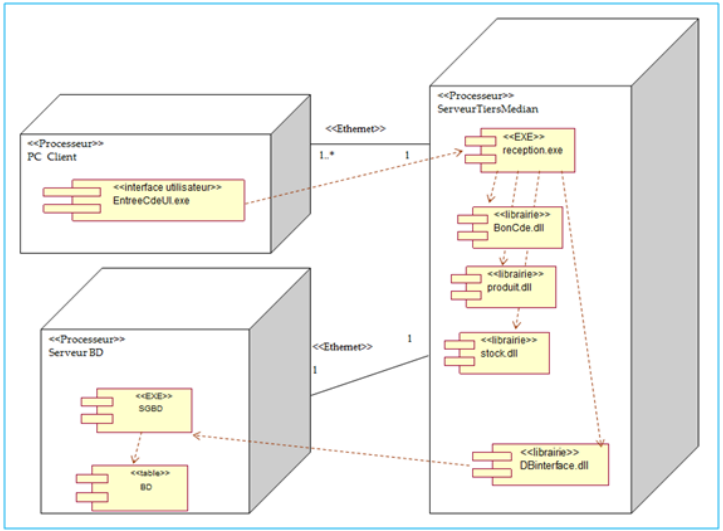
- La vue de déploiement décrit les ressources **matérielles** et la répartition des parties du logiciel sur ces éléments.
- Il contient un diagramme :
    - Le diagramme de déploiement



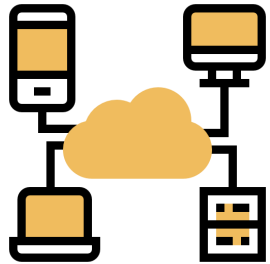
Le diagramme de composants

L'aspect lié à l'architecture du logiciel - La vue des déploiement

- **Ce diagramme** correspond à la description de l'environnement d'exécution du système (matériel, réseau…) et de la façon dont les composants y sont installés.

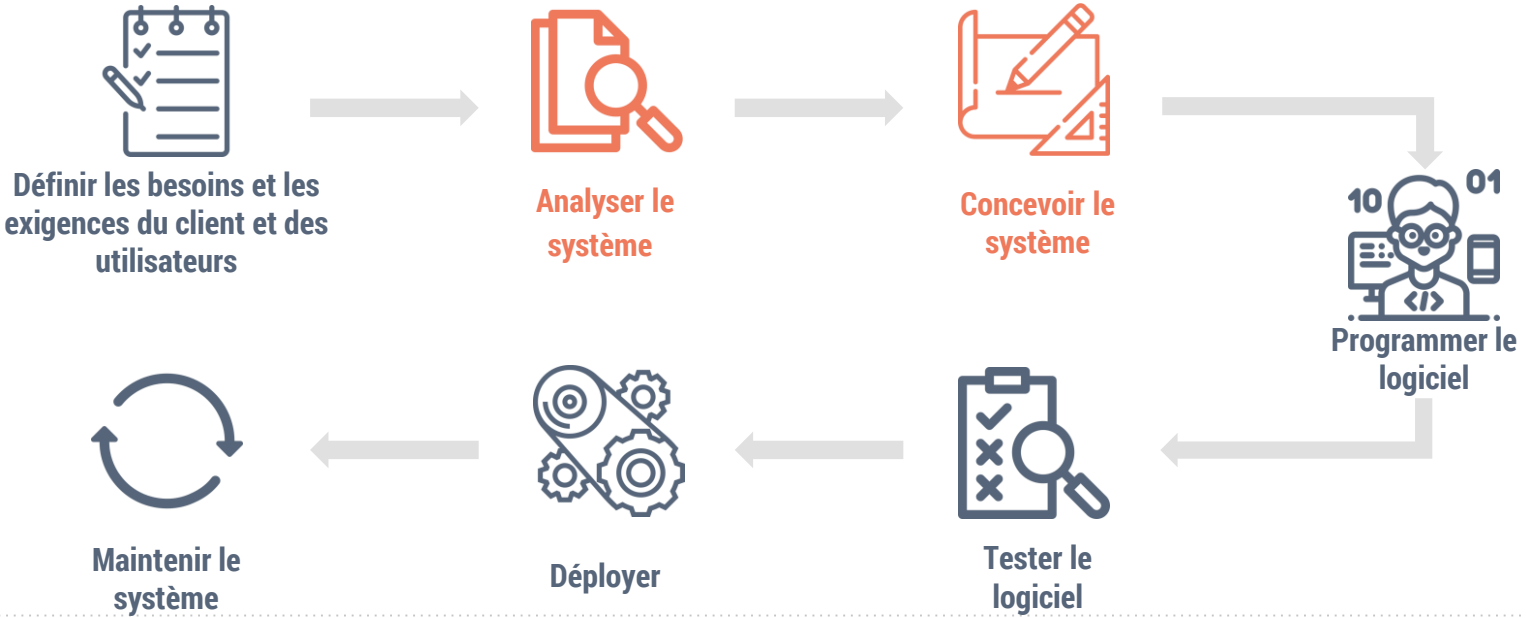


Un exemple de diagramme de déploiement



Diagrammes et étapes de développement logiciel

Lors du développement d'un logiciel, les **diagrammes** UML sont utilisés dans les étapes de l'**analyse** (analyse des besoins, du domaine, applicative) et de la **conception** de la solution.



Analyse

Diagrammes et étapes de développement logiciel

- Pour définir les **besoins** (contexte et système)

QUEL DIAGRAMME?	POURQUOI?
Diagramme de contexte	Pour identifier les **acteurs** qui utiliseront le système
Diagramme de cas d'utilisation	Pour indiquer de quelles **façons** les acteurs utiliseront le système

Analyse

Diagrammes et étapes de développement logiciel

- Analyse de **domaine**

QUEL DIAGRAMME?	POURQUOI?
Diagramme de cas d'utilisation	Pour **détailler les fonctionnalités** en y ajoutant des liens entre cas d'utilisation
Diagramme d'activité	Afin de **décrire le déroulement** des cas d'utilisation
Diagramme de classes	Pour préciser les **informations** nécessaires pour les actions d'un cas d'utilisation

Analyse

- Analyse **applicative** (ou analyse de la solution)

QUEL DIAGRAMME?	POURQUOI?
Diagramme de séquences	Afin de **détailler** le **déroulement** d'un cas d'utilisation tout en indiquant les informations à utiliser
Diagramme d'état-transition	Afin d'**identifier** tous les **états** par lequel un objet peut passer et donc de trouver les actions qui permettent d'obtenir un changement d'état
Diagramme de collaboration (de communication)	Pour **identifier** les **messages échangés** entre objets et trouver de nouvelles actions.

Conception

Diagrammes et étapes de développement logiciel

- Conception de la solution

QUEL DIAGRAMME?	POURQUOI?
Tous les diagrammes précédents	Afin de vérifier la **cohérence** des diagrammes et d'apporter des détails nécessaires à l'implémentation de la solution.
Diagramme de composants	Pour indiquer de quelle façon le logiciel sera **construit**. Un composant pouvant être un exécutable, un fichier, un module, une base de données, etc.
Diagramme de déploiement	Afin de montrer sur quel matériel chacun des composants devra être installé et pour indiquer les éventuels moyens de communication entre les parties.

Résumé

- La définition d'un logiciel peut être divisée en deux étapes majeures : l'**analyse** (analyse des besoins, du domaine et de la solution applicative) et la **conception**.

- L'étape d'analyse comprend 6 diagrammes : diagramme de contexte, diagramme de cas d'utilisation, diagramme de classe, diagramme de séquence, le diagramme d'état-transition et le diagramme de collaboration.

- L'étape de conception apporte des précisions aux diagrammes réalisés lors de l'analyse et comprend 2 diagrammes supplémentaires : le diagramme de composants et le diagramme de déploiement.


YOU KNOW NOTHING JON SNOW !
© Anthony LEPOT - 2015
What matters most in Westeros?
A : Honnor and loyalty
B : Power and strategy
C : UML
D : Hodor !