Projet dirigé

SCRUM

Saliha Yacoub

Introduction

Plan de séance

- Rappels
  - Objectifs du cours
  - Activités de développent de système ou d'un produit logiciel
- La méthode en cascade
- Le PU
- Les approches agiles
- Étude de besoins :
- Exercice: le Puzzle.

Introduction

Objectifs du cours (Rappels):

- Comprendre les Activités de développement logiciels
- Utiliser une méthodes et outils agiles de gestion de projets
- Travailler en équipe
- Concevoir et développer un produit logiciel
- Développer une stratégie de tests

Introduction

Processus de développement logiciel, c'est quoi et pourquoi ?

- Un processus digne de ce nom doit fournir des directives garantissant le développement efficace de logiciels de qualités.

Qu'est-ce qu'un système logiciel ?

- Un système logiciel se compose de TOUS les artefacts nécessaires à sa représentation sous forme lisible aussi bien pour les machines que pour les humains impliqués dans le développement logiciel.
- Un artefact désigne toute sorte d'information créé, modifiée ou produite ou utilisée par les intervenants dans le processus de développement logiciel (les modèles de BD, les interfaces utilisateur, les cas d'utilisation, les plannings de projet etc …..

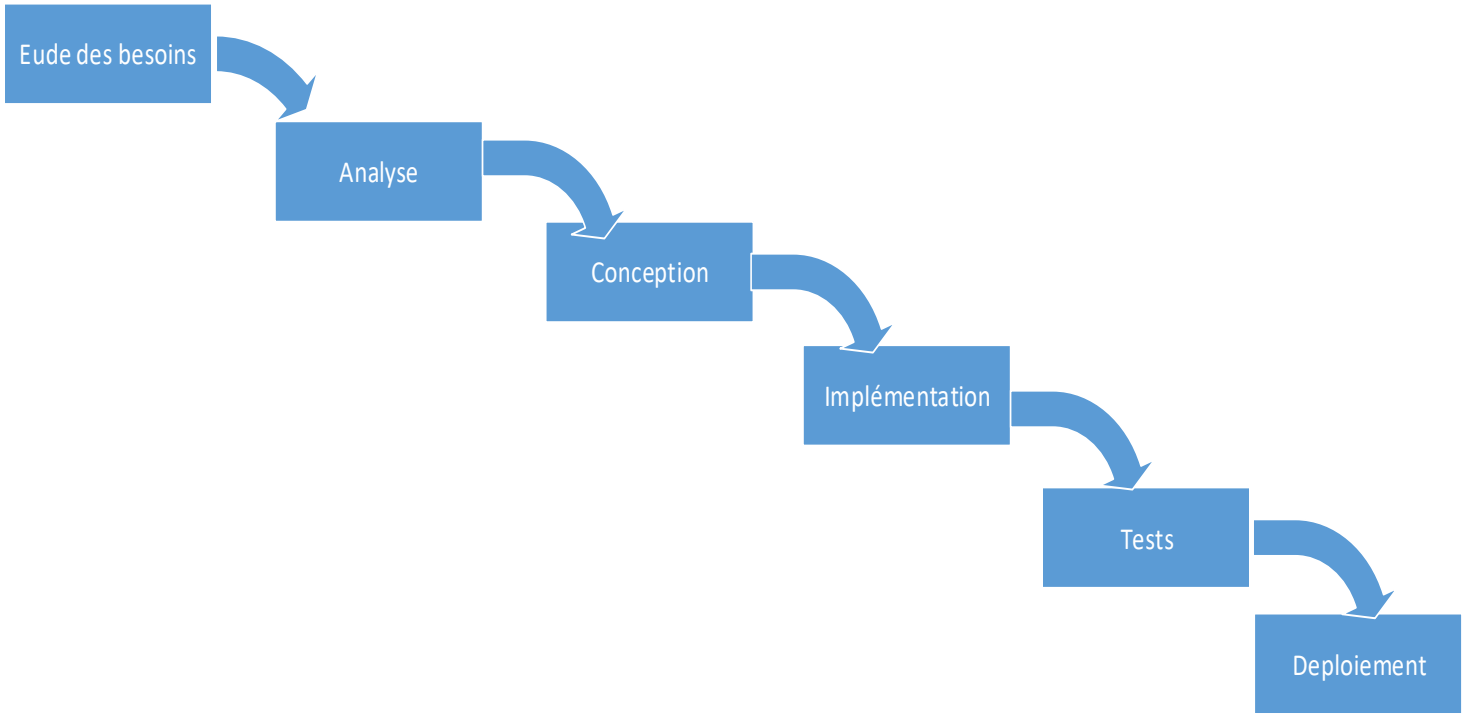
Introduction

Quelles sont les activités de développement d'un produit logiciel? (rappels)

- Étude des besoins
- Analyse
- Conception
- Implémentation et test
- Déploiement
- Évaluation

Introduction

Méthode en cascade



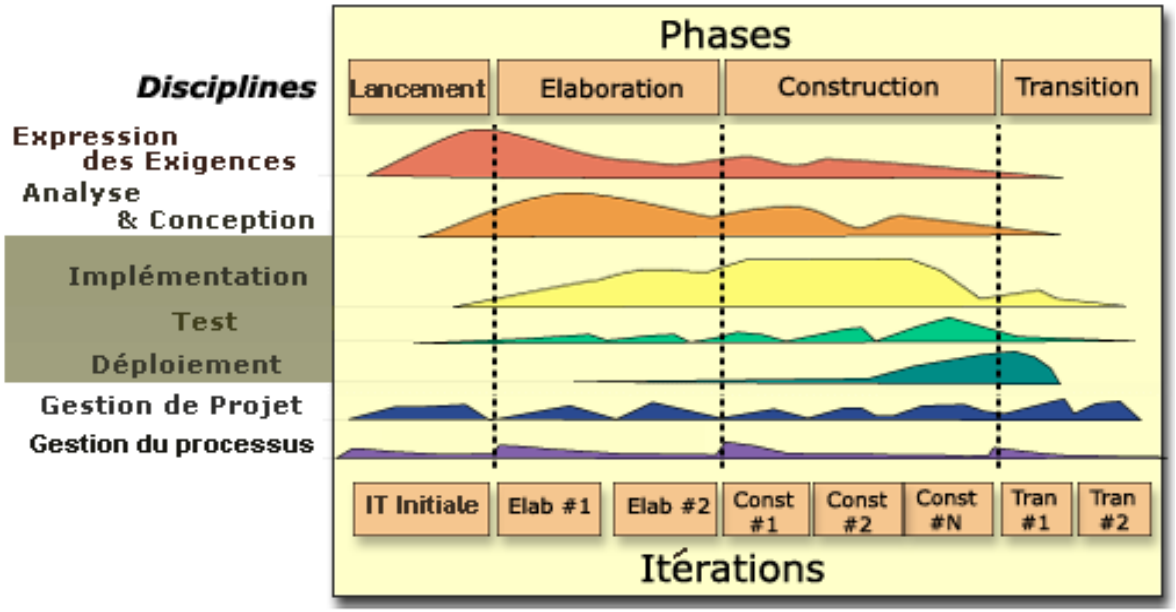
Introduction

Problèmes de la méthode en cascade:

- Implication du client, juste au début
- La propagation des erreurs
- La difficulté à corriger les erreurs
- Satisfaction du client: uniquement vers la fin
- Les besoins évoluent.
- Ce n’est pas vrai que l’on puissent comprendre TOUS les besoins au début. Un peu comme vous pour un travail de session, il faut tout le temps valider avec le prof surtout lorsque l’énoncé n’est pas claire.

Introduction

Le processus Unifié: Le PU



Introduction

Avantages du PU

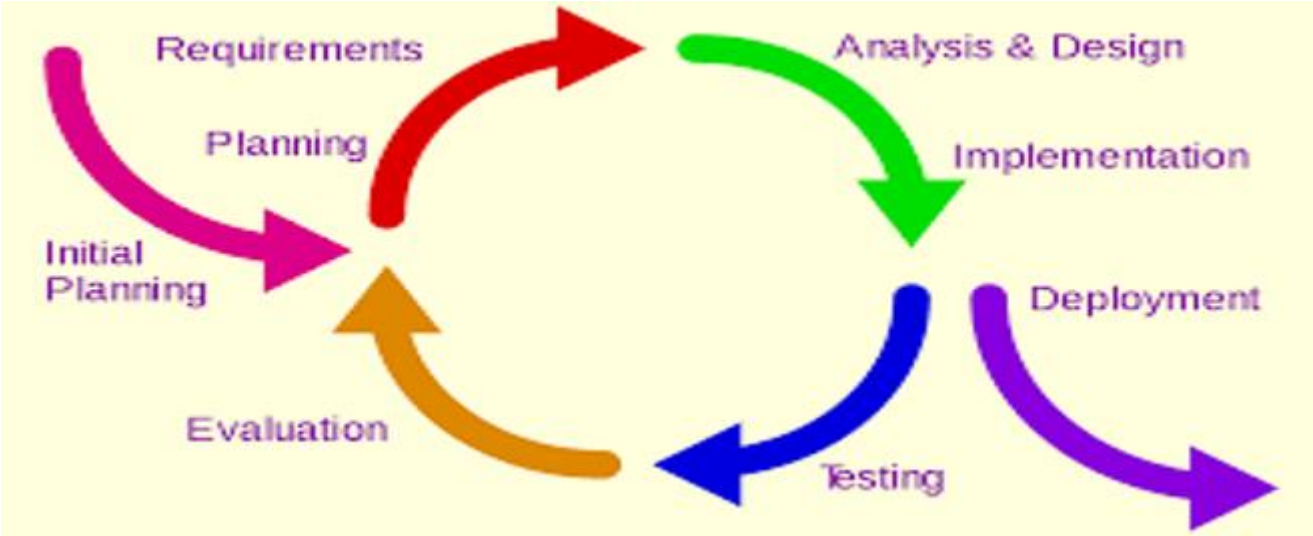
- Itératif et incrémentale
- Repose sur UML pour la modélisation (modélisation objet et unifiée)
- Formalisation des besoins: les uses-case.
- Fusion : Analyse et conception pourrait être vu comme un avantage.

Inconvénients du PU

- Processus trop lourd, surtout pour les petits projets
- Le principe du PU est bon, mais malheureusement sa pratique tend vers le cascade
- La description des uses-case peut être fastidieuse. Documentation lourde

Introduction

L’approche agile



Introduction

Avantages:

- Itératif et incrémental pour vrai pas uniquement dans le principe.
- Livraison tôt et souvent
- Implication du client: les besoins évoluent, feedback tôt.
- Les itérations (sprint) sont courtes.
- La documentation n’est pas lourde.
- Le rythme de travail est constant

Introduction

Manifeste agile: 4 valeurs et 12 principes

Valeurs agiles

1. Les personnes et leurs interactions sont plus importantes que le processus et les outils
2. Un logiciel qui fonctionne prime sur la documentation
3. La collaboration avec les clients est préférable à la négociation contractuelle
4. La réponse au changement passe avant le suivi d'un plan.

Introduction

Les principes :

Le Manifeste annonce douze (12) principes, qui ne définissent pas une méthode agile.

1. Satisfaire le client en livrant tôt et régulièrement des logiciels utiles qui offre une véritable valeur ajoutée.
2. Accepter les changements même tard dans le développement.
3. Livrer fréquemment une application qui fonctionne.
4. Collaborer quotidiennement entre clients et développeurs.
5. Bâtir le projet autour de personnes motivées en leur fournissant environnement et support, et en leur faisant confiance.
6. Communiquer par des conversations face à face.
7. Mesurer la progression avec le logiciel qui fonctionne.
8. Garder un rythme de travail durable.
9. Rechercher l'excellence technique et la qualité de conception
10. Laisser l'équipe s'auto-organiser.
11. Rechercher la simplicité.
12. À intervalle régulier, réfléchir aux moyens de devenir plus efficace.

Introduction

- Si les principes et les valeurs sont universels, la façon de les mettre en œuvre sur des projets varie. Cette application se fait par l'intermédiaire de ce qu'on appelle une pratique.
- Une pratique est une approche concrète et éprouvée qui permet de résoudre un ou plusieurs problèmes courants ou d'améliorer la façon de travailler lors d'un développement.

Parmi les approches nous avons: XP, **SCRUM**, KANBAN etc….

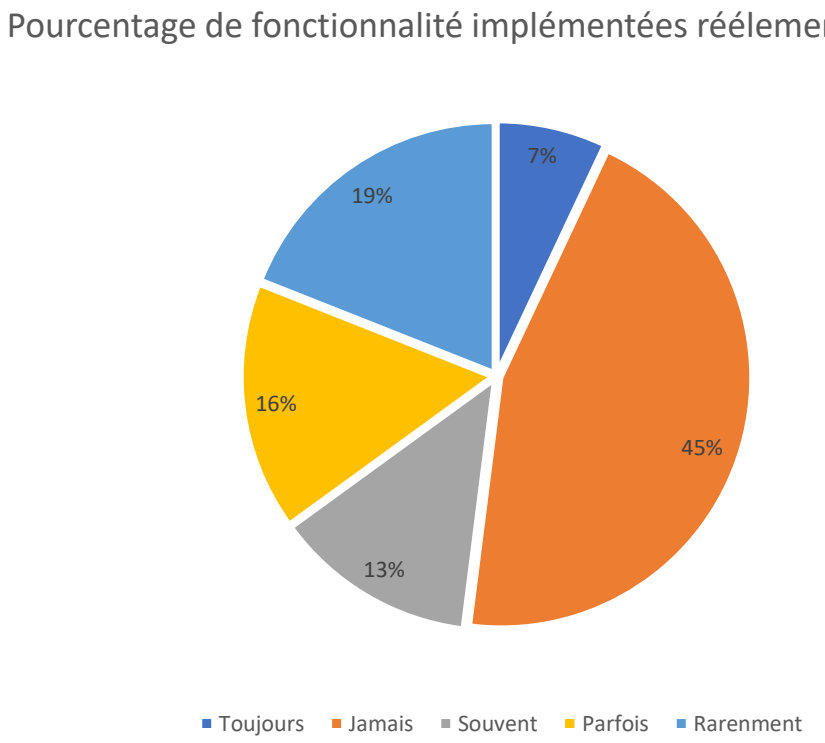
Étude des besoins

COMPRENDRE CE QUE LE CLIENT VEUT

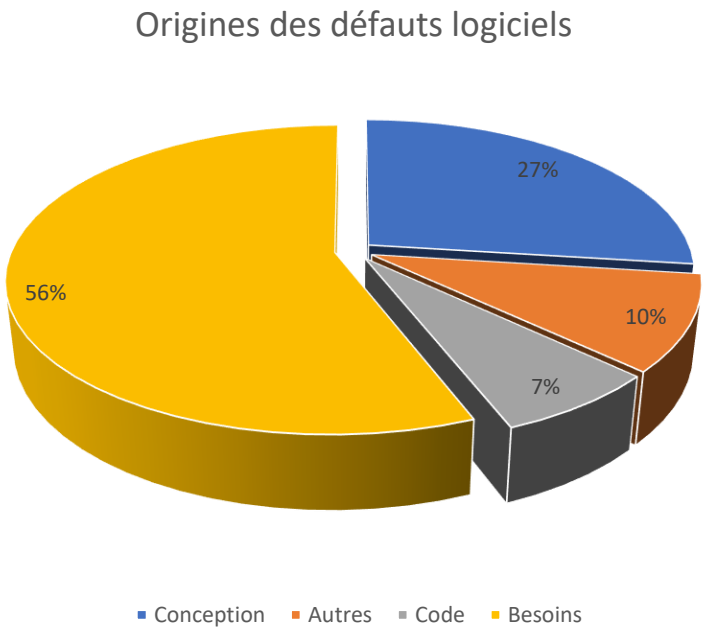
ÉTABLIR LA FAISABILITÉ DU PROJET

FAIRE DES PROPOSITIONS DE SOLUTIONS

Étude des besoins



Étude des besoins



Étude des besoins

- Comment est-il possible de développer des fonctionnalités qui ne seront jamais utilisées ?
- Est-il possible que nous ayons mal communiquer avec le client ?
- Est-il possible que nous ayons mal compris les besoins du client?
- Le client a-t-il bien exprimer ses besoins ?
- Les attentes du client sont-elles clairement exprimées

De la difficulté à recueillir efficacement les besoins

Exercice du Puzzle

Étude des besoins

CONCLUSION

QUESTIONS ??

Étude des besoins

Retour sur la séance précédente

Recueillir les besoins

Formaliser les besoins

Les cas d’utilisation

Exercices

Étude des besoins

Recueillir les besoins : comment ? Pour partager une vision

- Brainstorming
- L'interview
- L'observation
- Le questionnaire
- L'analyse de l'existant

Faire impliquer le client le plus souvent possible car les besoins changent, les besoins évaluent.(exemple: ce qui nous semblait important au début, est finalement peu important ou trop couteux à réaliser)

Ne pas oublier des acteurs dans le système (le puzzle pour lequel il manquerait une pièce)

Étude des besoins

Formaliser les besoins : Pourquoi ?

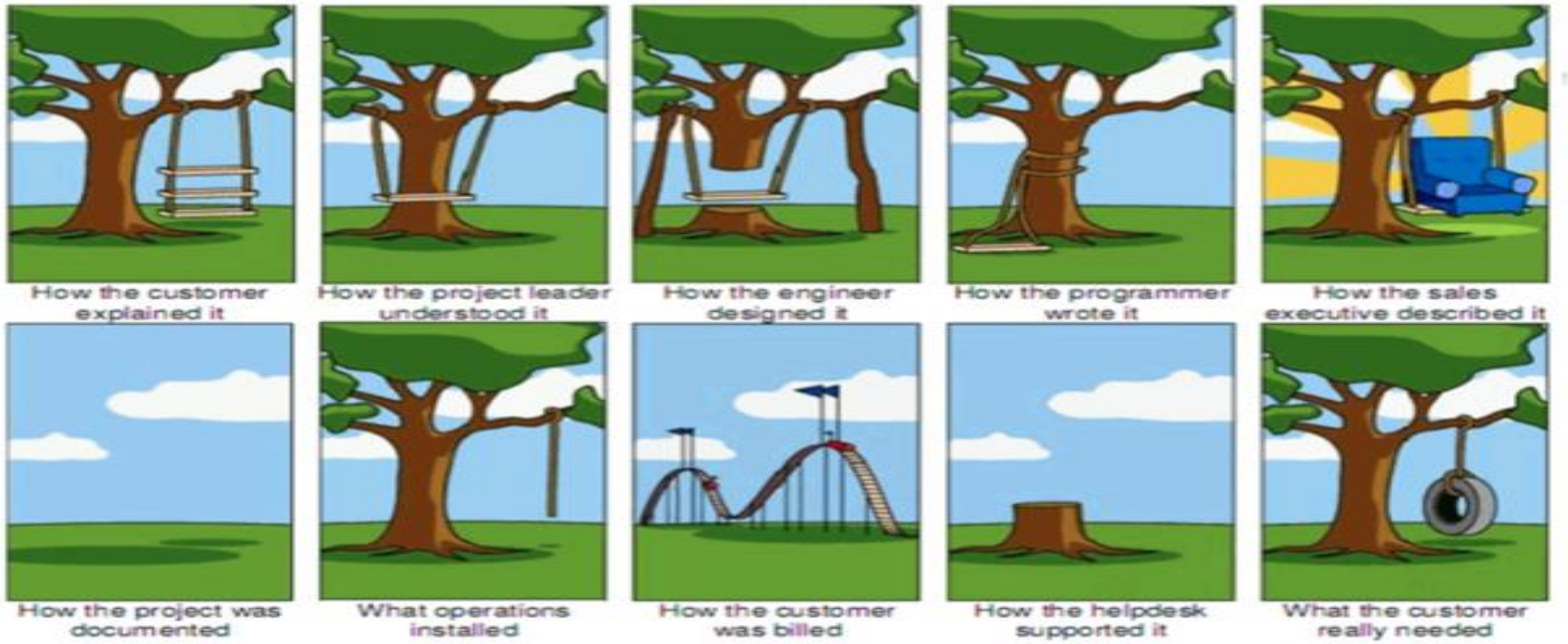
Car on veut retrouver la trace des besoins que nous venons de recueillir. On voudra les consulter, les mettre à jour etc..

Formaliser les besoins : Comment ?

Comment garder un lien entre les besoins initiaux et les besoins des étapes intermédiaires ? Comment s'assurer que tous les besoins sont traités ? En d'autres mots, comment limiter la rupture dans le processus de développement ?

Voir image suivante.

Étude des besoins



Étude des besoins

Formaliser les besoins : Comment ?

- En utilisant la norme IEEE 830.→on parle de SEL(spécification des exigences logiciel)
  - Doit être exprimée de manière claire, concise et cohérente
  - Doit être non ambiguë
  - Doit être valide
  - Doit avoir un bénéfice qui l'emporte sur le coût
  - Doit être vérifiable
  - Doit être identifiable de manière unique
  - Doit être modifiable (car elle évolue)
  - Doit être importante dans la résolution du problème
  - Doit être réaliste en considérant les ressources disponibles.

Étude des besoins

- **En utilisant :Les use-case ou les cas d'utilisation d'UML (Unified Modeling Langage)**

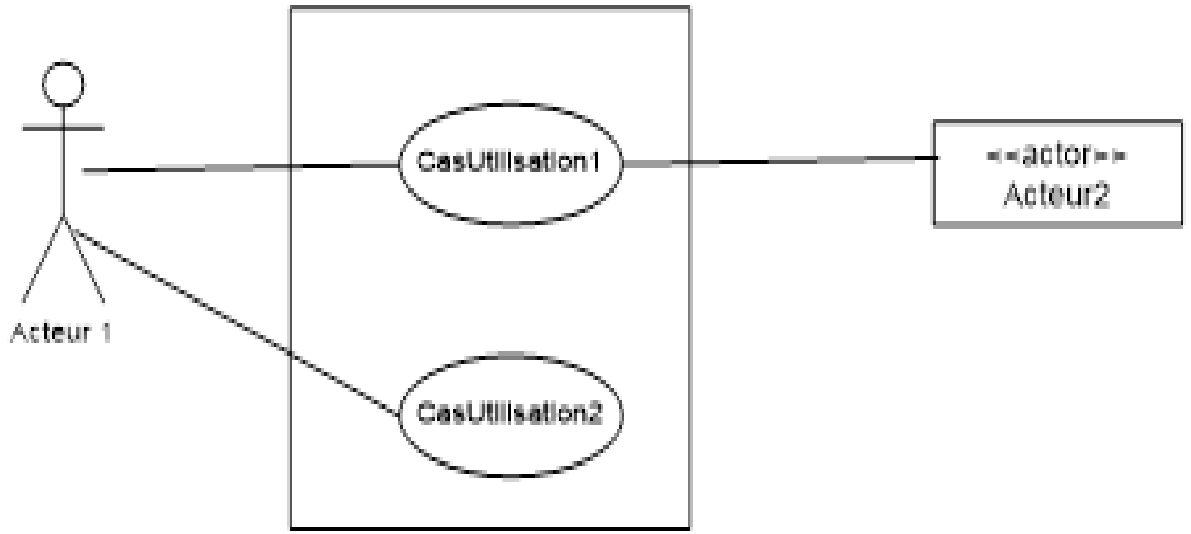
Les cas d'utilisation est une technique de formalisation des besoins utilisée dans le processus unifié. Le diagramme de cas d'utilisation d'UML est utilisé pour représenter les besoins du système ou ce que le système doit faire. Ils décrivent le comportement du système du point de vue de l'utilisateur.

- Permettent de structurer les besoins des utilisateurs et les objectifs correspondants d'un système.
- Ils centrent l'expression des exigences du système sur ses utilisateurs Ils se limitent aux préoccupations "réelles" des utilisateurs ; ils ne présentent pas de solutions d'implémentation et ne forment pas un inventaire fonctionnel du système.
- Ils identifient les utilisateurs du système et leur interaction avec celui-ci

Étude des besoins

- **En utilisant :Les use-case ou les cas d'utilisation d'UML (Unified Modeling Langage)**

Le diagramme des use-case permet, d'un seul coup d'œil, de voir les utilisateurs du système et leur interaction avec lui (système)



Étude des besoins

- **En utilisant :Les use-case ou les cas d'utilisation d'UML (Unified Modeling Langage)**

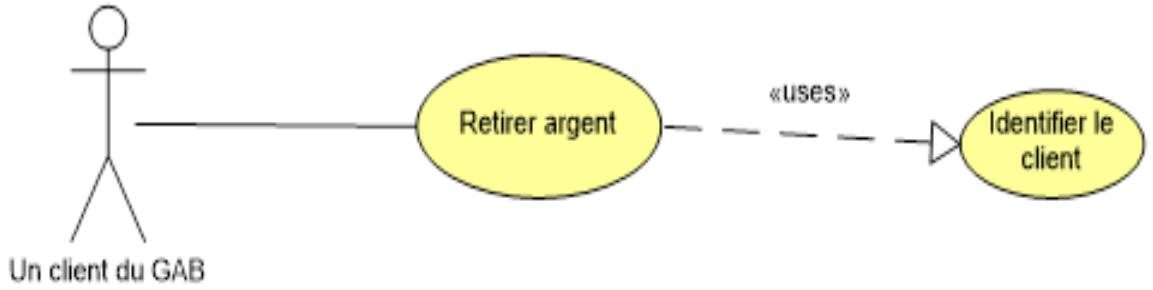
Comment faire un diagramme des use-case ?

- Recenser tous les acteurs du système
- Comprendre ce que les acteurs font dans le système ou avec le système
- Un cas d'utilisation n'est pas une action à faire. Un cas d'utilisation est un ensemble de séquences d'actions qui sont réalisées par le système et qui produisent un résultat observable pour un acteur particulier du système
- Déterminer les cas principaux en premier. Toujours revenir au BUT initial.
- Déterminer les liens entre les cas d'utilisation.
- Si un diagramme de cas d'utilisation est trop gros…(trop de cas) il perd de son objectif.

Étude des besoins

Relation entre cas d'utilisation: La relation « include » ou « uses » (utilise)

Un cas A utilise un cas B si les actions de A ne peuvent s'exécuter
avant l'exécution des actions de B. En d'autres mots, pour exécuter A il faut d'abord exécuter B.
Exemple, un client ne peut retirer de l'argent sans s'identifier

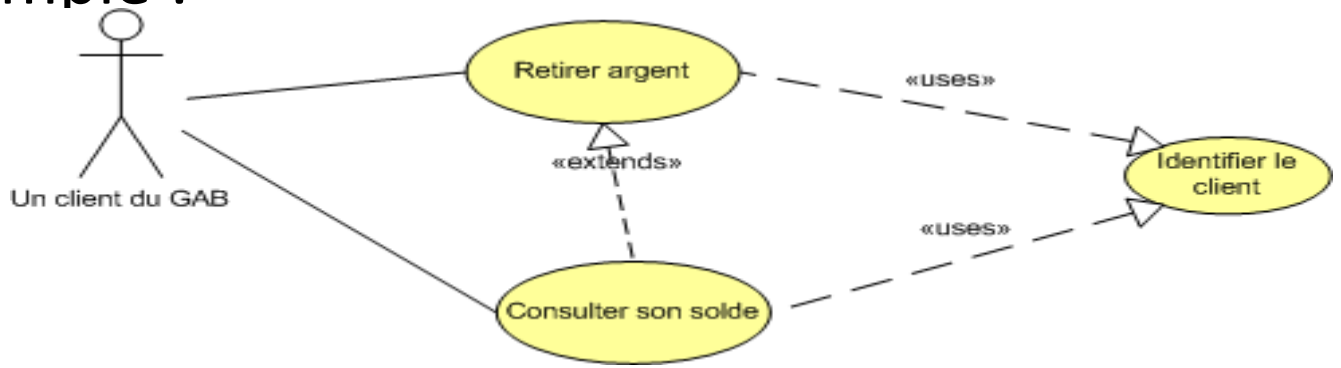


Étude des besoins

Relation entre cas d'utilisation: La relation « extend »

Un cas A étend son action sur un autre cas B si B est la suite logique de A. De plus A et B pourrait s'exécuter de manière indépendante.

Exemple :



Étude des besoins

- Les cas d'utilisation ***Retirer argent*** et *Consulter son solde* sont deux cas indépendants, en ce sens qu'un client du GAB peut consulter son solde sans retirer de l'argent ou qu'il peut retirer de l'argent sans effectuer d'interrogation de solde.

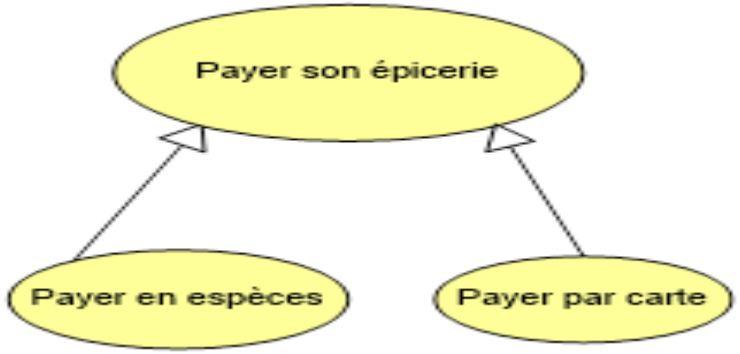
- Cependant le cas ***Consulter son solde*** peut etendre son action au cas *Retirer de l'argent*. Un client après avoir consulter son solde peut décider de retirer de l'argent.

- Quel que soit le cas d'utilisation à exécuter (*Retirer argent* ou *Consulter son solde*), ils doivent faire appel (utilise) au d'utilisation Identifier *le client* (carte débit et nip).

Étude des besoins

Relation entre cas d'utilisation: La relation de généralisation

- Un cas A est une généralisation d'un cas B si B est un cas particulier de B
- Cette relation de généralisation/spécialisation est présente dans la plupart des diagrammes UML et se traduit par le concept d'héritage dans les langages orientés objet.



Étude des besoins

Applications:

- Le GAB
- Le jeux Trivial Pursuit
- Le jeu Puissance 4.

Étude des besoins

CONCLUSION

QUESTIONS ??

Étude des besoins

Plan de séance

- Retour sur la séance précédente:
- Description des cas d’utilisation.
- Application aux exercices précédents.
- Autres exercices.

Étude des besoins

Retour sur la séance précédentes:

- Acteur , c'est quoi ? Pourquoi est-il important de ne pas oublier des acteurs dans le système
- Acteur: Principal et secondaire
  - Exemple, bibliothèque et un prêt: Le client est secondaire, le commis est principal
- Cas d'utilisation: est un ensemble d'actions.
- Diagramme des cas d'utilisation (voir acétate 27):
  - Déterminer les acteurs et comprendre leur interaction avec le système
  - Déterminer les cas d'utilisation sans tomber dans le piège des actions
  - Déterminer le lien entre les cas d'utilisation.

Étude des besoins

- Les cas d'utilisation ont besoin d'être décrits soit textuellement, soit en utilisant un autre diagramme. Deux parties sont importantes lors de la description d'un cas d'utilisation.
- Le sommaire d'identification
  - Le titre
  - Résumé
  - Acteurs
  - Date de création
  - Version
  - Date de mise à jour
  - Responsable

Étude des besoins

- Le scénario nominal.
  - Préconditions
  - Scenario nominal (enchaînement des opérations dans le cas où le cas d'utilisation se déroule normalement.)
  - Postconditions
  - Scénario alternatif
  - Exceptions

Étude des besoins

Avantages de Use-case:

- Permet de voir les acteurs du système ainsi que leur interaction avec le système.
- D’un seul coup d’œil on peut voir les cas d’utilisation principaux du système.
- La description des cas d’utilisation est riche en information pertinente.

Inconvénient

- L’écriture des cas d’utilisation: c’est très long.

Étude des besoins

CONCLUSION

QUESTIONS ??

Étude des besoins

Plan de séance

- Retour sur la séance précédente:
- Description des cas d’utilisation: Le jeu P4, Trivial Pursuit
- Conclusion : Cas d’utilisation
- Autres exercices.

Étude des besoins

CONCLUSION

QUESTIONS ??

Étude des besoins: formaliser les besoins par les user-stories

Plan de séance

- Les user-stories
- La propriété INVEST
- Introduction aux test d’acceptation
- Exercices

Étude des besoins: formaliser les besoins par les user-stories

- Définition

Une **user-story** ou un **scénario** est une exigence du système à développer formulée en une ou deux phrases dans le langage des utilisateurs pour servir un but.

Sa granularité doit permettre à l'équipe de réalisation d'estimer son coût et de la réaliser entièrement à l'intérieur d' une itération.

- Avantage:

Elles facilitent la démarche en deux temps : générales et grossières au début, elles s'enrichissent ensuite d'exemples, de tests d'acceptation. Elles facilitent la communication, l'ajout ou la suppression de détails**.**

Étude des besoins

- Écriture narrative:

  En tant que <rôle>

  Je veux <liste de tâches>

  Afin de <valeur ajoutée ou résultat>

- Exemples:

  En tant qu'étudiant, je veux me connecter à Colnet pour consulter ma note en KBE pour le mini-test1

  En tant que joueur, je veux me connaitre mon solde en écu afin de pouvoir acheter un item.

Étude des besoins: formaliser les besoins par les user-stories

- INVEST: Une story doit respecter la propriété INVEST pour être considérée comme une bonne story:
  - I, Indépendante
  - N, Négociable
  - V, Verticale, Valuable
  - E, Estimable
  - S, Suffisamment petit
  - T, Testable.

Étude des besoins: formaliser les besoins par les user-stories

- Indépendante: lorsque le client peut en toute liberté décider de l'ordre dans lequel les scénarios (story) est implémenté sans qu'interviennent des contraintes techniques
- Négociable: L'équipe de développement n'est pas contrainte par la manière dont sera implémenté le scénario : Elle a la latitude d'imaginer une solution efficace
- **Vertical** : (V pour Valuable , a une valeur ajoutée) un bon scénario ou story décrit une fonctionnalité complète de l'application dont le client apprécie l'intérêt à l'intérieur d'une itération
- Estimable: Connaitre le coût d'implémentation (en points)
- Suffisamment petit: pour pouvoir l'estimer, la story doit être petite. (terminée à l'intérieur d'une itération)
- Testable: la tester pour dire qu'elle est terminer → prévoir des tests d'acceptation.
- Sur le site dur cours, il y a des exemples que vous devez consulter

Étude des besoins: formaliser les besoins par les user-stories

CONCLUSION

QUESTIONS ??

EXERCICES

Étude des besoins: formaliser les besoins par les user-stories

Retour sur les user-storie

Use-case vs user-storie

Exercice bibliothèque

Mini-test2

Étude des besoins, formaliser les besoins: conclusion

User Story	Use case
C'est une brève description d'une fonctionnalité telle que vue par l'utilisateur.	Représente une séquence d'action qu'un système peut accomplir en interagissant avec les acteurs du système
Mode orale et collaboratif	Mode écrit et souvent distant
Grande lisibilité vu sa simplicité	Pourrait souvent manquer de lisibilité (volume)
Format écrit court laissant part à de belles discussions orales	Format écrit très riche en information(postcondition, scénario …) peu de place à l'oral
Utilisée pour spécification des exigences mais surtout pour estimation de la planification	Utilisé seulement en tant que spécification
Émergence rapide au travers des ateliers de collaboration	Long travail d'analyse et de formalisation

Étude des besoins, formaliser les besoins: conclusion

User Story	Use case
Implémentée et testé en une itération seulement	Implémenté et testé en plusieurs opérations
Contient des test d'acceptation (au dos de la carte)	Ne contient pas les cas de test qui en découle.
Difficile à lier les unes aux autres: Absence de vue globale	Liaison et vue globale faciles au travers du diagramme des cas d'utilisation qui lient les use case
Associée au méthodes agiles comme SCRUM, XP	Associée généralement au Processus Unifié
Très facile à maintenir	Plus difficile à maintenir