Ingénierie logicielle

L’Ingénierie logicielle repose sur l’art de concevoir des logiciels et d’améliorer la façon de les concevoir.

Un peu d’histoire : Des Calculis au Smarphone, de la note G d’Ada Lovelace à Rust.

Le triptyque Coûts/Délais/Qualité.

Sommaire

Qu’est-ce qu’un logiciel :
- Produit ;
- Licences ;
- Propriété ;
- Normes ;
- Communautés.

Les méthodes :
- La méthode Rache ;
- Le cycle en V ;
- Les méthodes agiles ;
- Les méthodes orientées objet.

L’ingénierie :
- Spécification ;
- Conception ;
- Implémentation ;
- Intégration ;
- Documentation ;
- Vérification ;
- Validation ;
- Déploiement ;
- Maintenance.

L’élaboration de la conception.
- La Systémique.
- QQOQCCP (Qui ? Quoi ? Où ? Quand ? Comment ? Pourquoi ? Combien ?).
- CRUD (Create, Read, Update, Delete)
- KISS (Keep It Simple and Stupid).
- Itérations et Réification exemple du RUP

La notation UML.

Produire les documents de conception :
- Markdown et PlantUML versus les ateliers orientés objets tel Papyrus ;
- Le cartouche d’un document ;
- Le Besoin vs l’Envie (Élicitation) ;
- Analyse du besoin ;

- Glossaire des concepts ;
- Glossaire des termes techniques ;
- Cas d'utilisations ;
- Scénarios Système ;
- Scénario détaillé ;
- Diagrammes de classes des scénarios ;
- Diagramme de classes de l'application.

Ingénierie dirigée par les modèles :
- Eclipse et Papyrus.
- Méta modèle et transformation de Modèle.
- Méta Méta Modèle et circularité.
- Génération de code.

Qu'est ce qu'un logiciel ?

L'Ingénierie logicielle repose sur l'art de concevoir des logiciels et d'améliorer la façon de les concevoir (l'ingénieur cherche à optimiser ses processus de travail).
Depuis près de 10 000 ans (c.f mégalithes de Gobekli Tepe) les très grands projets témoignent d'un sens de l'organisation, d'abord pour la construction des bâtiments et des navires (voir les découvertes de 2013 de l'équipe de Pierre Tallet décrite dans « Les papyrus de la Mer Rouge I, le journal de Merer (papyrus Jarf A et B) »), et maintenant pour les œuvres immatérielles comme les films ou les logiciels . Ces activités soulignent ce que l'on sait depuis fort longtemps :
Il n'y a pas de grand projet sans une organisation rigoureuse des processus et des équipes.

Pourtant une majorité des projets informatiques sont en échec à savoir que soit les budgets sont dépassés, soit ce sont les délais, et très souvent les deux.
L'une des premières causes est la mauvaise compréhension du besoin.
Le « client » vient avec une envie et il est du devoir de l'ingénieur de détecter le besoin pour d'abord satisfaire celui-ci.
On peut illustrer cette obligation par la métaphore suivante :
Une personne rentre dans une concession automobile, le commercial qui l'aborde lui demande s'il a besoin d'une information. La personne perçue comme un prospect aux yeux du commercial annonce qu'il souhaiterait s'offrir la dernière sportive de la marque. Il énonce ici une envie. Le commercial lui demande alors de lui décrire l'usage qu'il en fera, et lui fait formuler différents scénarios. En l'interrogeant, il arrive à détermine son besoin, qui dans cet exemple sont de se transporter de chez lui à son travail lors des heures de pointe et de temps à autre de sortir le weekend à la plage avec sa famille de deux enfants, d'ailleurs sa femme prendra le véhicule au moins trois fois par semaine, car ils n'auront qu'une voiture. Après avoir sollicité habilement son prospect, le commercial dirige son presque client vers une familiale hybride, automatique et élégante de façon à satisfaire le besoin et ménager l'envie. Il y gagne un client qui reviendra, car celui-ci aura l'impression d'avoir été écouté, puisque son besoin aura été satisfait tout autant que son envie. Si le commercial n'avait su identifier le besoin, la vente se serait conclue, mais peu de temps après le client aurait été insatisfait et ne serait jamais revenu.

Il est du rôle de l'ingénieur d'interroger ses clients pour trouver le besoin derrière l'envie, mais également de rapidement comprendre le contexte et pour cela d'interroger un échantillon représentatif des différents rôles tenus par les futurs utilisateurs, car comme dans le film « Répétition d'orchestre » de Fellini, chaque protagoniste ne voit que son travail et n'a pas la vue d'ensemble permettant de comprendre le processus que viendra aider le logiciel développé.

L'élicitation des besoins est l'art de faire dire au client toutes les informations nécessaires au projet.

Qu’est-ce qu’un logiciel :

- Produit ;
- Licences ;
- Propriété ;
- Normes ;
- Communautés.

Les méthodes :

- La méthode Rache ;
- Le cycle en V ;
- Les méthodes agiles ;
- Les méthodes orientées objet.

L’ingénierie :

- Spécification ;
- Conception ;
- Implémentation ;
- Intégration ;
- Documentation ;
- Vérification ;
- Validation ;
- Déploiement ;
- Maintenance.

L’élaboration de la conception.

- La Systémique.
- QQOQCCP (Qui ? Quoi ? Où ? Quand ? Comment ? Pourquoi ? Combien ?).
- CRUD (Create, Read, Update, Delete)
- KISS (Keep It Simple and Stupid).
- Itérations et Réification exemple du RUP

La notation UML.

Produire les documents de conception :

- Markdown et PlantUML versus les ateliers orientés objets tel Papyrus ;
- Le cartouche d’un document ;
- Le Besoin vs l’Envie (Élicitation) ;
- Analyse du besoin ;
- Glossaire des concepts ;
- Glossaire des termes techniques ;
- Cas d’utilisations ;
- Scénarios Système ;
- Scénario détaillé ;
- Diagrammes de classes des scénarios ;
- Diagramme de classes de l’application.

Ingénierie dirigée par les modèles :

- Eclipse et Papyrus.
- Méta modèle et transformation de Modèle.
- Méta Méta Modèle et circularité.
- Génération de code.

Projet exemple

L'association CultureDiffusion souhaite réaliser une bibliothèque numérique à gestion décentralisée.
Le principe est de permettre à chaque membre de numériser les œuvres et aux bibliothécaires de les proposer à l'emprunt selon deux modalités.
Toutes les œuvres du domaine public sont accessibles gratuitement, toutes les œuvres sous droits sont proposées en location pour une période de 2 semaines.
Lorsqu'une œuvre passe dans le domaine public, elle devient accessible gratuitement et est diffusée aux membres ayant accordé à l'application suffisamment d'espace disque.
Chaque membre de la bibliothèque peut emprunter ou louer une œuvre.
Chaque membre peut proposer une œuvre et ainsi enrichir la bibliothèque.
Chaque œuvre est constituée d'un fichier contenant l'œuvre et d'un fichier au format json contenant les informations sur l'œuvre.
L'application bibliothèque possède un index des œuvres qui est mis à jour à chaque ajout ou suppression d'œuvre.
L'application possède quatre rubriques proposées sous forme de répertoires.
Un répertoire "fond_commun" avec une partie des œuvres libres de droits de l'association.
Un répertoire "emprunts" avec les œuvres sous droit empruntées par le membre qui sont chiffrées avec sa clé.
Un répertoire "séquestre" avec une partie des œuvres sous droit gérée par l'association, tous les fichiers y sont chiffrés, l'index n'y est pas accessible de façon directe.
Un répertoire "à modérer" avec les œuvres que le membre a proposées.
Lorsqu'une œuvre est proposée par un membre, elle est soumise à modération.
Les bibliothécaires voient les œuvres soumises, vérifient et complètent les données telles que les auteurs, éditeur, langue, pays d'origine, date de publication, droits, catégorie de l'œuvre, format du support, puis ils décident du statut de l'œuvre et donc si elle est dans le domaine public ou non, ou si elle est rejetée.
Selon le statut de l'œuvre modérée, l'application range dans la bonne rubrique et gère les droits d'accès.
Si l'œuvre est dans le domaine public il peut y en avoir autant de copie qu'il y a de membre.
Si l'œuvre est protégée alors il ne peut y avoir que 3 fois plus de copies que de licence d'exploitation, et chaque œuvre est chiffrée par une clé différente ayant une date de validité.
À la fin de la validité d'un emprunt, l'application bibliothèque supprime automatiquement l'œuvre du répertoire du membre.
Les œuvres sont classées selon les types et sous-types suivants :
Article
Livre :
- BD
- Enfants
- Romans
- Livre technique
- Education
- Loisir
- Culture
- Sante
- etc

Musique :
- Classique
- Jazz/Funk/Soul
- Pop
- Metal

etc
Vidéos :
SF
Histoire
Série
Documentaire
etc

Certaines œuvres appartiennent à plusieurs catégories en même temps.

On utilisera PlantUML et Papyrus (Eclipse).
Réalisez le glossaire.
Réalisez le diagramme de cas d'utilisations.
Listez les scénarios de l'application, ajouter ceux fondamentaux n'apparaissant pas dans le cahier des charges.
Triez les scénarios par ordre d'importance.
Réalisez les 4 diagrammes de séquence système les plus importants.
Réalisez les 5 diagrammes de classes des scénarios les plus importants.
Faites un choix d'architecture et justifiez le.
Quels design pattern utilisez vous et pourquoi ?
Réalisez le diagramme de classe détaillant les associations et leur cardinalité.

Analyse et conception

Comme tout cahier des charges, une première étape dite de spécification permet de préciser ce qui va être réalisé.

Le but est de lever les imprécisions de l'expression des besoins et donc de permettre la conception.

Pour cela, nous réalisons des scénarios d'utilisations du futur développement.

Ceci a deux intérêts :

1. Reformuler permet de vérifier que nous avons bien compris ;

2. Les détails fonctionnels précisent ce qui va être livré.

Une fois écrits, les scénarios sont soumis au client qui les commente. Le prestataire tient compte des remarques et lui soumet à nouveau les scénarios corrigés, jusqu'à ce qu'ils soient stables.

Les navettes entre le prestataire et le client sont inévitables, car le sujet est abstrait, et le client perçoit son besoin d'un point de vue métier contrairement au prestataire qui le voit d'un point de vue technique.

Toutefois, et par expérience, cette étape n'est pas suffisante, car souvent le client a besoin d'être en face du système pour « préciser » son besoin. Il faut alors développer des parties de la solution et la soumettre au client le plus tôt possible.

Il devient alors inutile de vouloir détailler les scénarios dès le début puisque le besoin change tout au long de la réalisation.

La seule façon de gérer ces projets et d'utiliser une méthode agile comme « scrum », où les scénarios sont détaillés au fur et à mesure de leur implémentation.

Organisation du projet

Nous utiliserons git pour travailler à plusieurs.
La majorité de la documentation sera faite en markdown.
La première étape est donc de nommer son projet.
De lui créer un dépôt soit sur github, soit sur gitlab
Le dépôt doit contenir
À la racine :
README.md
Titre du projet
Objectif du projet en 3 lignes
Description en 10 lignes
Date de dernière modification
Listes des auteurs
Version
Liens vers les readme.md des sous répertoires suivants

CahierDePaillasse.md
Fichier qui contient les réflexions, observations, questions en cours, et autre remarques.

Suivi.md
Fichier qui contient un tableau qui réparti le travail en tâches et pour chacune donne une définition, un temps estimé puis un temps réellement consommé.

Répertoire Analyse
Un fichier libreoffice ou word permettant de surligner les mots clés du cahier des charges en fonction de leur nature

Répertoire Glossaire
Un README.md référençant le glossaire métier et le glossaire technique
GlossaireMetier.md
Liste des termes avec leur définition.
GlossaireTechnique.md
Liste des termes techniques (noms des types de données comme les classes, la liste des rôles des acteurs etc).

Répertoire Scénarios
README.md (contient la liste des liens vers les scénarios)
Un fichier markdown par scénario dont le contenu est précisé au chapitre Liste des scénarios.
Un diagramme de séquence par scénario faisant intervenir le système à réaliser comme une boite noire.
Pour chacun de ses diagrammes on crée un ou plusieurs diagrammes de séquences où le système devient une boite blanche comprenant les classes que l'on crée pour réaliser les fonctions identifié du systèmes. Pour cela on utilise et la méthode QQOQCP et le CRUD. On itère à plusieurs reprise en passant de ces diagrammes de séquences détaillés au diagrammes de classes associés à chacun de ses diagramme. C'est ce que l'on appelle réification et c'est un processus itératif.
Jusqu'à obtenir une conception stable qui de toute façon bougera avec l'implémentation.

Répertoire DiagrammesDeClasses

README.md référençant tous les diagrammes

Un diagramme de classe générale contenant toute les classes du projet et les associations entre classes.

Autant de diagrammes de classe qu'il y a de scénarios. Chaque diagramme détaille uniquement les classes (attributs, méthodes et associations) fournissant les services, méthodes et les données apparaissant dans le scénario ayant servi à établir ce diagramme. Si une classe est utilisée dans plusieurs scénarios les méthodes est attributs d'un diagramme de classe correspondent uniquement au scénario associé à son scénario.

Répertoire DiagrammesDEtatsTransitions

README.md

Un diagramme qui donne les états pour chaque type de donnée identifié.

Répertoire DActivités

README.md

Analyse du cahier des charges

Repérage des mots clés

Légende des couleurs

Concept : Mot portant une signification pour le projet ;
Nom propre : Identifiant unique d'une chose ou d'une personne ;
Action : transformation de l'état ;
Propriété : qualité, durée etc.

Repérage

L'association CultureDiffusion souhaite réaliser une bibliothèque numérique à gestion décentralisée.
Le principe est de permettre à chaque membre de numériser les œuvres et aux bibliothécaires de les proposer à l'emprunt selon deux modalités :
Toutes les œuvres du domaine public sont accessibles gratuitement, toutes les œuvres sous droits sont proposées en location pour une période de 2 semaines.
Lorsqu'une œuvre passe dans le domaine public, elle devient accessible gratuitement et est diffusée aux membres ayant accordé à l'application suffisamment d'espace disque.
Chaque membre de la bibliothèque peut emprunter ou louer une œuvre.
Chaque membre peut proposer une œuvre et ainsi enrichir la bibliothèque.
Chaque œuvre est constituée d'un fichier contenant l'œuvre et d'un fichier au format json contenant les informations sur l'œuvre.
L'application bibliothèque possède un index des œuvres qui est mis à jour à chaque ajout ou suppression d'œuvre.
L'application possède quatre rubriques proposées sous forme de répertoires.
Un répertoire "fond_commun" avec une partie des œuvres libres de droits de l'association.
Un répertoire "emprunts" avec les œuvres sous droit empruntées par le membre qui sont chiffrées avec sa clé.
Un répertoire "séquestre" avec une partie des œuvres sous droit gérée par l'association, tous les fichiers y sont chiffrés, l'index n'y est pas accessible de façon directe.
Un répertoire "à modérer" avec les œuvres que le membre a proposées.
Lorsqu'une œuvre est proposée par un membre, elle est soumise à modération.
Les bibliothécaires voient les œuvres soumises, vérifient et complètent les données telles que les auteurs, éditeur, langue, pays d'origine, date de publication, droits, catégorie de l'œuvre, format du support, puis ils décident du statut de l'œuvre et donc si elle est dans le domaine public ou non, ou si elle est rejetée.
Selon le statut de l'œuvre modérée, l'application range dans la bonne rubrique et gère les droits d'accès.
Si l'œuvre est dans le domaine public il peut y en avoir autant de copie qu'il y a de membre.
Si l'œuvre est protégée alors il ne peut y avoir que 3 fois plus de copies que de licence d'exploitation, et chaque œuvre est chiffrée par une clé différente ayant une date de validité.
À la fin de la validité d'un emprunt, l'application bibliothèque supprime automatiquement l'œuvre du répertoire du membre.
Les œuvres sont classées selon les types et sous-types suivants :
Article
Livre :
- BD
- Enfants
- Romans
- Livre technique

- Education
- Loisir
- Culture
- Sante
- etc

Musique :

- Classique
- Jazz/Funk/Soul
- Pop
- Metal
- etc

Vidéos :

- SF
- Histoire
- Série
- Documentaire
- etc

Certaines œuvres appartiennent à plusieurs catégories en même temps.

Glossaire métier

1er étape identifier les concepts, mots clés et phrases clés et créer une entrée dans un tableau ;
2me étape trouver une définitions pour chaque entrées créée ;
3me étape classer les entrées alphabétiquement.

Exemple à l'étape deux nous avons :

Association :Personne morale au sens loi 1901.
CultureDiffusion:Nom de l'application.
Bibliothèque numérique : Application informatique qui fonctionne métaphoriquement comme une bibliothèque/médiathèque physique.
Gestion décentralisée : Les administrateurs ainsi que les bibliothécaires accèdes à l'application depuis des lieux et des moments différents, en plus après discussion avec le client, celui-ci veut que les œuvres numériques soient stockées sur les terminaux des membres et seulement sauvegardées sur un serveur du client ainsi il souhaite utiliser des protocoles de partage pair à pair et de contrôle de version.
Membre : Toute personne ayant un compte sur l'application.
Numériser : Consiste à créer ne projection de l'œuvre dans un fichier, par exemple à scanner un livre.
Œuvres : Création ou réalisation humaine ayant généralement un intérêt artistique .
Bibliothécaires : Membre de l'application en charge de classer les œuvres et de vérifier leur nature.
Proposer à l'emprunt : Action des bibliothécaires permettant de partager les œuvres.
Œuvres du domaine public : Création ou réalisation dont les droits moraux permettent leur usage par tous gratuitement généralement par ce que l'auteur est mort depuis plus de 70 ans.
Accessibles gratuitement : L'accès à l'œuvre est permis au membre sans contrepartie financière.
Œuvres sous droits: œuvres pour lesquelles ils existent encore des ayant droits et qui imposent de respecter des règles strictes de mise à disposition.
Location d'une œuvre
Délai de deux semaines de location : Contrainte sur la durée d'utilisation d'une œuvre
Œuvre passant dans le domaine public
Œuvre accessible gratuitement
Œuvre diffusée aux membres
Membre accordant de l'espace disque au partage
Application
Teste et validation de l'espace disque
Emprunter une œuvre libre de droits
Œuvre Libre de droits : synonyme de « Œuvres du domaine public »,
Œuvre orpheline : œuvre sous droits mais dont on ne connaît pas l'auter
Œuvre sous « creative common » : certaine permettent le partage d'autre non selon la licence.
Louer une œuvre sous droits
Enrichir la bibliothèque.
Fichier
Format json
Informations sur une œuvre
Index
Index mis à jour
Mise à jour de l'index
Ajout d'une œuvre
Suppression d'une œuvre
Rubriques

Répertoires
Fond_commun
Fichier chiffré
Clé
Répertoire "séquestre"
Fichiers chiffrés
Index
Accessible de façon directe
Répertoire "à modérer"
Soumise à modération
Voire les œuvres soumises
Données :
- Auteurs : ,
- éditeur : ,
- langue : ,
- pays d'origine : ,
- date de publication : ,
- droits : ,
- catégorie de l'œuvre : ,
- format du support :

Statut de l'œuvre : Ensemble des informations permettant de savoir si l'œuvre est dans le domaine public ou non.
Fichier rejeté : fichier dont le partage a été refusé par les bibliothécaires.
Œuvre modérée
Range dans la bonne rubrique :
Gère les droits d'accès
Domaine public
Copie :
Œuvre protégée :
3 fois plus de copies :
Licence d'exploitation :
Date de validité :

Glossaire technique et concepts introduits par les scénarios :

Anonyme : Utilisateur non encore authentifié ou n'ayant pas de compte.
Téléchargement de l'application :
Installation de l'application :
Utilisateur : Anonyme ou Membre ou Bibliothécaire utilisant l'application.

Média : Support permettant d'exécuté l'application de la bibliothèque décentralisée.

Membre authentifié:Membre dont la connexion s'est faite avec FranceConnect.

Numéro de transaction : Identification unique de toute opération réalisée.
Fichier journal local : Fichier contenant un historique de toutes les opérations ayant été effectuées par le membre.
Filtre : Permet de cacher des éléments d'une liste lorsque ces éléments ne sont pas voulus.
Tri : Permet d'ordonner les éléments d'une liste selon les critères souhaités.

Liste des scénarios

La transcription du cahier des charges en scénario donne lieu à des échanges avec le client à la fois pour préciser les hypothèses, mais également par ce qu'elle fait apparaître de nouveau concept et décrire dans le glossaire.
Pour chaque scénario on donne un nom, une description, la liste des acteurs, les préconditions, puis la liste des étapes du cas nominal sous forme d'une liste de phrase au présent construite avec un sujet un verbe et un complément. On peut fournir les scénarios alternatifs et ceux d'erreurs.

Il faut détailler en premier les étapes des scénarios les plus importants.

Installer l'application

Description : Décrit le processus d'installation de l'application sur différents médias
Acteurs : L'utilisateur, le serveur de mis à disposition de l'application.
Précondition : Le téléchargement de l'application doit être possible depuis le média (ordinateur ou smartphone) de l'utilisateur
Étapes :

1. Le futur utilisateur ouvre son navigateur ou le market store de son système d'exploitation.
2. Le futur utilisateur saisit le nom de l'application.
3. La page de sélection affiche la description de l'application et ses informations légales, notamment celles correspondant aux partages des œuvres et à leurs droits d'auteurs.
4. Le futur utilisateur sélectionne l'application et l'installe.

Devenir Membre

Description : Un anonyme
Acteurs : un utilisateur Anonyme, FranceConnect, les différents mandataires de FranceConnecte.
Prérequis : le scénario « Installer l'application » a été exécuté sans erreur.
Étapes :

1. L'utilisateur anonyme lance l'application.
2. L'application détecte que c'est son premier lancement sur le média.
3. L'application affiche une page d'information.
4. L'application propose à l'utilisateur de se connecter via FranceConnect.
5. L'application propose de créer un compte ne pouvant déposer ou louer des œuvres.
6. L'utilisateur choisit de se connecter via FranceConnect.
7. L'application propose les différents sites d'identification.
8. L'utilisateur choisit l'un des sites.
9. L'utilisateur s'authentifie.
10. L'application reçoit les informations de connexion.
11. L'application affiche toutes les actions, dont celles de dépôt.

Scénario alternatif:
branchement à l'étape 6

1. L'utilisateur choisit de créer un nouvel identifiant.
2. L'application demande le nom , prénom et date de naissance de l'utilisateur.
3. L'utilisateur saisit les informations.
4. À partir de ces informations, l'application crée
    - un identifiant unique garantissant l'anonymat,

- une paire de clés, publique et privée, pour enregistrer les opérations qui seront faites.

5. L'application transmet l'identifiant et la clé publique au serveur de l'association.
6. L'application affiche les actions permises.
7. L'application affiche la liste des œuvres dans le domaine public.

Documents :

Page d'information affichée : pour expliquer l'objectif de l'application, ses possibilités et précise que l'utilisateur devra utiliser un compte FranceConnect s'il veut pouvoir louer des œuvres sous droits ou déposer des œuvres.

Devenir Bibliothécaire

Description : Un membre demande à devenir bibliothécaire.

Acteurs : Membre, Bibliothécaire

Prérequis : Il existe au moins un Bibliothécaire actif.

Étapes :

1. Un membre demande à l'application à devenir bibliothécaires.
2. L'application enregistre la demande et soumet la demande aux bibliothécaires.
3. L'application sur le média d'un autre bibliothécaire transmet à son utilisateur la demande pour modération.
4. L'application demande au bibliothécaire s'il :
   - accepte,
   - rejette,
   - n'a pas d'avis,
   - ou souhaite ignorer la candidature.
5. Le bibliothécaire indique à l'application quel est son choix.
6. L'application partage de façon anonyme et unique ce choix avec les autres applications sur les médias des autres Bibliothécaires.
7. Le Bibliothécaire peut modifier son choix tant que le délai imparti n'est pas écoulé.
8. Une fois le délai écoulé, les applications des bibliothécaires propagent aux autres applications la décision automatique prise ainsi :
   - Si la majorité des bibliothécaires ont accepté la candidature du membre alors celui-ci devient bibliothécaire.
9. Le membre consulte la décision depuis l'application sur son média.
10. Si l'application du futur bibliothécaire constate qu'il est promu bibliothécaire alors il reçoit les droits lui permettant d'accéder à l'ensemble des contenus et de pouvoir accepter ou refuser les modérations.
11. Sinon l'application indique le refus de sa promotion.

Scénarios alternatifs :

La majorité des bibliothécaires refuse la promotion.

Scénarios erreurs :

Données, documents, écrans :

Se connecter

@TODO

Accéder à la liste des œuvres

Description : L'utilisateur demande à voir la liste des œuvres

Acteurs : Utilisateur

Prérequis :

Étapes :

Scénarios alternatifs :

Scénarios erreurs :

Données, documents, écrans :

Déposer une œuvre numérisée

Description : Un membre a numérisé une œuvre et souhaite la partager avec la bibliothèque pour enrichir son fond.

Acteurs : Membre authentifié

Prérequis : Le membre est authentifié sur l'application par FranceConnect

Étapes :

1. Le membre authentifié demande à l'application à partager une œuvre.
2. L'application affiche un formulaire de saisie d'information concernant l'œuvre.
3. Le membre authentifié saisit les informations, et joint le fichier de l'œuvre numérisée.
4. L'application demande confirmation de l'envoi.
5. Le membre authentifié confirme l'envoi.
6. L'application enregistre le fichier dans le répertoire « à modérer ».
7. L'application crée un numéro de transaction et l'enregistre dans le fichier journal local.

8. L’application transmet le fichier, ses informations et les numéros de transaction au dépôt sur le serveur de l’association.
9. Les serveurs de l’association notifient les bibliothécaires qu’une nouvelle œuvre est en attente de modération.
10. L’application indique au membre que son partage est en attente de modération.

Scénarios alternatifs :

Scénarios erreurs :

Erreur de connexion avec le serveur.

Données, documents, écrans :

@TODO

Modérer une œuvre numérisée

Description : Les bibliothécaires sont avertis des numérisations d’œuvres à modérer.

Acteurs : Bibliothécaire, serveur de la Bibliothèque Nationale de France

Prérequis :Le fichier d’une œuvre numérisée doit avoir été déposé et soumis en modération

Étapes :

1. L’un des bibliothécaires se connecte à l’application.
2. L’application lui affiche la liste des fichiers d’œuvres numérisées soumises.
3. Le bibliothécaire positionne ses filtres pour ne voir que les œuvres susceptibles de l’intéresser.
4. L’application n’affiche que les résultats correspondant aux filtres.
5. Le bibliothécaire positionne les tris pour les afficher dans l’ordre voulu.
6. L’application affiche les résultats dans l’ordre souhaité.
7. Le bibliothécaire sélectionne un fichier d’une œuvre.
8. L’application affiche les informations saisies par le membre ayant soumis l’œuvre.
9. L’application affiche un lecteur spécifique au type de fichier.
10. Le bibliothécaire parcourt l’œuvre.
11. Le bibliothécaire complète les informations sur l’œuvre.
12. Le bibliothécaire accepte l’œuvre en précisant sa nature.
13. @TODO

Scénarios alternatifs :

Scénarios erreurs :

Données, documents, écrans :

Consulter une œuvre du domaine public

@TODO

Remplir les informations concernant une œuvre

@TODO

Consulter les informations concernant une œuvre

@TODO

Modifier les informations concernant une œuvre

@TODO

Louer une œuvre sous droits d’auteur

@TODO

Passage d’une œuvre dans le domaine public

@TODO

Diffusion d’une œuvre libre de droits

@TODO

Mise à jour de l’index des œuvres

@TODO

Consultation des rubriques

@TODO

Consultation de la rubrique « Fond commun »

@TODO

Qu’est ce qu’un modèle

(Je reprends ici mon article publié sur https://www.linkedin.com/pulse/lutopie-de-la-mod%C3%A9lisation-premi%C3%A8re-partie-quest-ce-quun-launay/)

Nos sociétés ont toujours cherché à expliquer et prédire ce qui leur arrive.

Depuis le siècle des Lumières, les mathématiques sont devenues l'outil le plus efficace de prédiction et les « scientifiques » ont remplacé les augures.

L'une des conséquences les plus importantes a été l'usage des statistiques qui permirent aux politiques de prendre "de bonnes" décisions. Il est ainsi devenu possible de passer de l'expérience individuelle à celle du groupe et de remplacer la diseuse de bonnes aventures par le pourcentage. La place sociale de la mort a été profondément modifiée et nous sommes passés du fait divin imprévisible et incompréhensible à la certitude qu'un certain pourcentage des individus vivront et auront des enfants et qu'il est possible de prévoir au plus juste les infrastructures, et les dépenses nécessaires.

Le bon souverain devint alors un bon gestionnaire. Puis les états utilisèrent des modèles de leur fonctionnement et pûrent ainsi assurer leur pérennité. Bref, nous étions passés d'un modèle purement « historique », où la connaissance des faits et actions permettait de se projeter dans un monde constant, à un modèle statistique de notre société permettant de prévoir les évolutions « linéaires », de les anticiper voire de les favoriser ou non.

Pour la balistique, nos « scientifiques » ont développé les systèmes d'équations différentielles linéaires qui par résolution permettaient de prévoir la bonne trajectoire de nos projectiles, mais nécessitaient un calcul rapide. La guerre devint une science gourmande en technique.

Les puissances de calcul ont alors augmenté en même temps que les besoins, et la technologie a créé de nouveaux besoins avec à la clé une croissance exponentielle de nos savoirs et de nos économies. Mais fin 19ème, il fallait plus de 15 ans pour passer du recensement d'un grand pays aux chiffres permettant les projections utiles aux décisions politiques. IBM naquit des premières machines mécanographiques de Hermann Hollerith, qui en utilisant des cartes perforées réalisées pendant les recensements permettaient d'avoir les statistiques nécessaires aux décisions politiques en moins de trois ans.

Cela a si bien fonctionné que ces outils ont été utilisés par les dictatures du 20ème siècle pour mettre en place l'élimination de masse. Mais il faut comprendre que la rationalisation et la création de modèles avaient enfanté du pire, et le nazisme n'est pas seulement la pensée monstrueuse de certains, mais bien le produit d'une intense réflexion autour de modèles de société, je vous invite à ce propos à voir les conférences de Johann Chapoutot sur le sujet.

Depuis les modèles sont partout et touchent tous les domaines d'activité humaine, ils sont par nature descriptifs et prédictifs et sont plus simples à appréhender que la réalité, mais ne sont pas toujours complets et ont généralement des limites d'applications.

Un bon modèle est simple et précis, manipulable et permet d'avoir des prévisions suffisamment proches de la réalité pour présenter un intérêt.

Le langage de description d'un modèle dépend de son domaine d'application, il peut prendre la forme d'équations mathématiques linéaires, stochastiques, de systèmes bouclés dit à rétroaction, d'algorithmes sophistiqués permettant de modéliser des comportements discontinus et hybrides, ou celle de descriptions littéraires qui généralement finissent par être traduite en équation.

On trouve des modélisations numériques de cellule vivante jusqu'aux descriptions textuelles et factuelles d'une bataille historique. Il est d'ailleurs étonnant de voir comment un même modèle peut être utilisé pour des domaines différents, par exemple les boucles de rétroaction des modèles en physique de systèmes oscillants de l'électronique peuvent être utilisées en science du vivant pour

élaborer des relations entre lapins et carottes ou entre hommes et consommation des ressources primaires. La grande difficulté et non des moindres et alors d'identifier les similitudes et de savoir faire les transformations d'un modèle vers un autre et ainsi de remonter dans une science les trouvailles faites par une autre science.

Mais le plus beau est que notre cerveau crée des modèles sans nécessairement intellectualiser la démarche, le résultat est ce que nous nommons avoir de l'expérience. Le cerveau le fait en permanence en réarrangeant continuellement nos connexions synaptiques. La modélisation de ce fonctionnement est à l'origine des réseaux de neurones utilisés en informatique, qui a poursuivi son propre développement pour aboutir à l'apprentissage profond (deep learning) où comme pour l'humain, c'est le nombre, la qualité et la variabilité des exemples utilisés lors de l'apprentissage qui font la qualité de la cognition (je vous invite à suivre la prochaine session du Mooc deep learning présenté par le CNAM).

Toutefois, notre cerveau n'hésite pas à compléter les informations perçues, il comble les trous de notre perception visuelle, prend des raccourcis pour traiter plus rapidement l'information reçue. Ces optimisations sont efficaces pour la majorité des cas, mais créer des biais cognitifs exploitables, comme ceux liés à la perception du mouvement très utilisés par les magiciens, ou ceux sociaux comme les biais de confiance ou de renforcement (voir l'excellente série "Crétin de cerveau" de la chaîne ScienceEtonnante), etc.

Mais en fait l'élaboration de modèle tout comme l'apprentissage ne sont que quêtes incessantes. Chaque fois que quelqu'un affirme avoir « enfin » un modèle complet vient une mesure ou expérience entrant en contradiction avec les prédictions faites, et l'on s'aperçoit alors qu'il faut travailler à compléter voire redéfinir le modèle.

Le modèle initial perd alors de sa simplicité, comme dans le cas des lois de Newton qui à cause de la planète Mercure ont été "complétées" par les équations d'Einstein (voir "Einstein et la pensée de Newton").

Mais il arrive qu'on doive renoncer à atteindre un modèle complet. Ainsi, comme en mathématiques avec le théorème d'incomplétude de Gohdel, il arrive qu'on rencontre des problèmes qui nécessitent de modifier le système de représentation du modèle, car il n'y a pas de concept dans le langage de modélisation permettant de le décrire et que le modèle porte des concepts modifiant le langage de description du modèle. Bref il apparaît des problèmes indécidables nécessitant une extension profonde du langage utilisé pour la modélisation, à ne pas confondre avec les DSL (Domain Specific Language) qui reviennent à regrouper l'assemblage de plusieurs concepts sous le nom d'un nouveau concept afin "d'aérer" le modèle.

On peut parler de classe de modèle, c'est-à-dire que les modèles peuvent être rangés par catégorie selon leur structure. Dans une classe les modèles sont substituables alors qu'ils ne traitent pas du même domaine, on parle de généralisation ou « pattern » de modèle. En informatique, il a ainsi été identifié des motifs récurrents dans les conceptions logicielles qui fonctionnent et ces motifs sont devenus des patrons de conceptions.

À force d'abstraction, on finit par perdre « le principe de réalité », ainsi :

La plupart des modèles des sciences « pures » sont indépendants de la réalité, bref si on modifie le modèle, la réalité ne change pas. La démarche scientifique consiste à élaborer des modèles que l'on teste pour prédire la réalité et que l'on modifie itérativement jusqu'à converger vers un modèle le

plus simple possible et le plus précis possible, généralement jusqu'à ce que notre connaissance globale ait suffisamment augmenté pour pouvoir le compléter. À titre d'exemple les médecins grecs d'Alexandrie dont Èrasistrate avait établi un modèle "pneumatique" du corps, notamment pour l'activation des muscles. Ils ne comprenaient pas l'électricité et son rôle dans le contrôle des muscles qui furent découverts par Luigi Galvani. Ils ne pouvaient donc imaginer d'autres interactions et il aura fallu attendre 20 siècles que l'ensemble de la société ait absorbé le niveau scientifique suffisant pour qu'apparaisse la notion d'influx électrique comme déclencheur de l'action musculaire. Les découvertes faites à une époque sont corrélées avec le niveau culturel de la population, plus il y a de sachants, plus la chance pour que l'on trouve et que cela se diffuse est importante. Par exemple l'explication du rôle des artères qui commence à l'époque des grandes dissections de Léonard de Vinci et André Vésale, mettra près de 300 ans à s'imposer.

Ainsi, et encore aujourd'hui, même si l'information circule à la vitesse de la lumière, tout "nouveau" modèle demande du temps pour être testé (principe de réfutabilité). Une fois validé, et son contexte d'application déterminé, le modèle aura besoin de temps pour imprégner notre culture, et dans certains cas changer nos éléments de langage. Par exemple en 1905, Einstein met à terre notre conception d'un temps absolu et l'on peut constater que malgré les preuves expérimentales accumulées démontrant un temps local, nous parlons toujours d'un temps qui coule, unique à tous (voir les conférences d'Étienne Klein sur le sujet). Bref, nous continuons à raisonner avec nos vieux réflexes. Pire, ces biais culturels nous freinent dans la compréhension de nouveau modèle.

Dans d'autres sciences, les modèles ont un pouvoir de transformation du réel, proportionnel à la croyance dans le modèle choisi, ainsi en économie le modèle prédominant oriente les politiques et modifie la réalité, quitte à devoir appliquer une énergie très importante pour tenter de conformer la réalité mesurée au modèle. Nous pouvons dire qu'au plus le comportement d'un agent s'écarte du modèle, au plus le groupe appliquera une force de rappel importante pour ramener l'égaré vers le chemin tracé par le modèle.

On retrouve aussi cela en politique, mais là la vertu du marketing et de la communication transforme le modèle en conviction marketée pour être facilement ingérable par de nouveaux "croyants". Les conséquences frôlent parfois la catastrophe, comme dans le cas de la société d'investissements en bourse LTCM dont le modèle établi par trois prix Nobel a failli conduire à une catastrophe majeure au système bancaire.

Modélisation UML

Objectifs

Connaître les bases de la notation UML.

Savoir

- Les différents diagrammes,
- La représentation des classes, attributs, méthodes, associations,
- La représentation des objets, des composants, des collaborations, des flux d'informations.

Pour réaliser l'analyse et générer les composants nécessaires à notre projet nous allons utiliser l'outil PlantUML et Papyrus (projet Eclipse).

Nous réaliserons dans un premier temps un diagramme de classes, qui permettra de décrire le plan documentaire de notre application, puis nous associerons à nos documents des "workflows" sous forme de diagramme d'états transitions.

Mais avant cela nous allons réaliser un bref rappel des notions UML utilisées dans la suite.

Qu'est ce qu'UML

UML (Unify Modeling Langage) est un langage de modélisation graphique des applications informatiques, c'est aussi une notation graphique qui permet de traduire sous forme de vues l'architecture d'un logiciel ou d'une application orientée objet.

UML est la fusion de différentes notations issues des méthodes de génie logiciel orienté objet antérieures aux années 1995. La version 1.0 a officiellement été publiée en 1997 par l'OMG (Object Management Group) qui est l'organisme responsable de sa standardisation. Depuis 2007, la version officielle est la 2.5 qui comporte 13 diagrammes, mais il n'est pas nécessaire d'utiliser ces 13 diagrammes pour réaliser une cartographie complète d'un système informatique.

Historiquement, son usage intervenait dans la phase de conception pour permettre de "visualiser" ce que sera et fera le logiciel. Comme les feuillets des plans d'un architecte en bâtiment, différents diagrammes composent une vue spécifique de la solution apportée au problème.

Le lecteur doit parcourir plusieurs vues et donc lire plusieurs diagrammes pour se créer son image mentale du logiciel.

Cette abstraction n'est malheureusement pas évitable, puisque un logiciel est composé à la fois de parties statiques, telle que le code exécutable, de données dynamiques telles que les entrées sorties, et de comportements.

Il permet de créer différents diagrammes de l'architecture d'un logiciel selon différents points de vue. Et comme pour une carte routière c'est au lecteur de faire l'effort de traduction du modèle vers le monde réel en appliquant la légende aux symboles regardés.
UML se rapproche des plans d'une bâtisse où plusieurs types de plan co-existent (plan de masse, de câblage, de chauffage) qu'il faut superposer mentalement pour créer sa représentation mentale compète de l'architecture.
Bref la compréhension et la réalisation des différents diagrammes UML nécessite de comprendre l'objectif et la légende des différents types de diagrammes.
L'objectif d'UML est de fournir un support à la modélisation des architectures informatiques orientées objets, c'est le fruit du croisement de plusieurs méthodes de conception orientée objets.

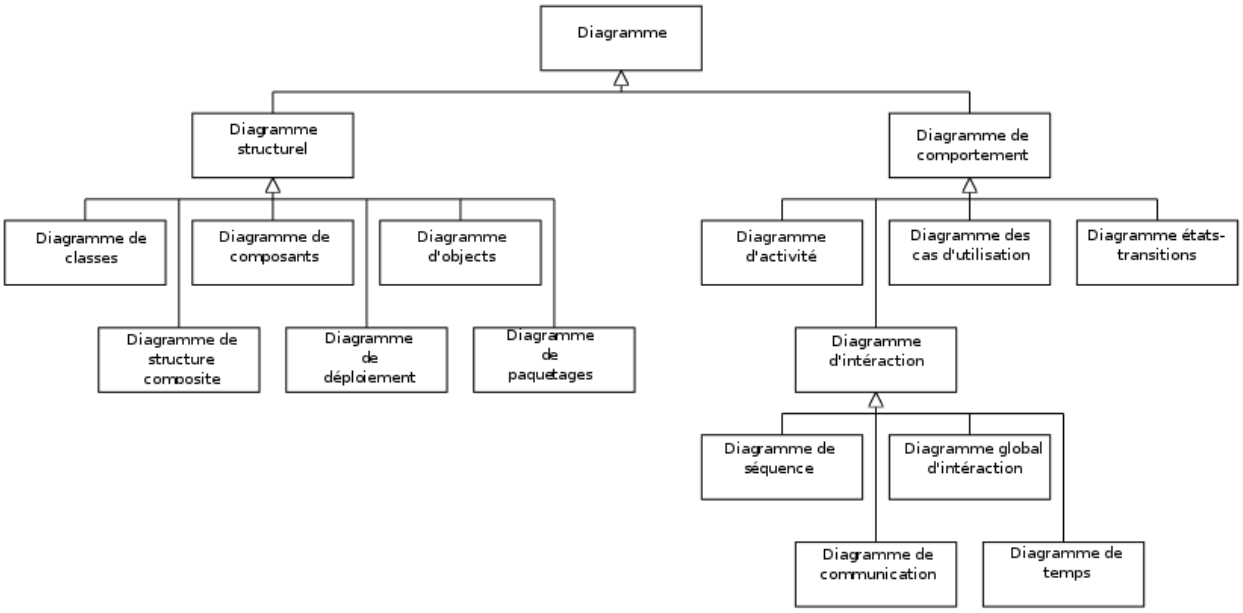


Figure 1: Les différents diagrammes d'UML 2.5 (Source wikipedia)

Les vues sont des regroupements abstraits des diagrammes qui sont eux les "dessins" représentant les interactions entre les éléments constitutifs du logiciel. Ces diagrammes permettent de visualiser l'interaction entre les éléments de l'architecture qui sont regroupés dans un "modèles d'éléments" et constitués des cas d'utilisations, classes, associations, attributs, etc.

Actuellement, les courants de l'IDM (Ingénierie Dirigée par les Modèles), font que son usage est présent tout au long du cycle de vie du logiciel, et l'on met à jour le logiciel en modifiant le modèle puis en générant à nouveau l'application. Les outils de génération se chargent de fusionner l'existant avec les nouvelles fonctionnalités.

Initialement, la conception orientée objet cherchait à résoudre un problème en le découpant non pas en un ensemble de fonctionnalités dont le tout forme un arbre ayant pour racine une fonction "main", mais en boites travaillant ensemble par l'échange de messages et dont le comportement global réalise la tâche attendue.

La description de ces boites forme un tout homogène de données et de fonctions appelée interface, c'est une abstraction.

La définition d'un type de boite respectant une interface donnée est nommée "classe" si elle est simple, et composant si pour respecter le contrat de l'interface elle encapsule la collaboration de plusieurs classes et peut à elle seule être considérée comme une application.

L'exécution d'une classe est appelée instance, elle possède alors ses propres valeurs d'attributs à commencer par son occupation mémoire ce qui la distingue des autres instances de la même classe.

Mais revenons à UML.

UML se base sur trois types de briques pour permettre la modélisation d'une architecture :

- Les éléments, qui sont un peu le vocabulaire de UML ;
- Les relations, qui représentent la syntaxe du projet ;
- Les diagrammes, qui contiennent la sémantique du projet.

Les éléments

Nous n'allons détailler ici que les éléments d'UML que nous utiliserons.

Les éléments structurels

La classe

C'est un types d'éléments qui partagent les mêmes propriétés (attributs, méthodes, relations, sémantique).
Si les propriétés définies par la classe sont immuables et ne dépendent pas des éléments, on parle d'attributs ou de variables de classe et de méthodes de classes, inversement si leur contenu dépend de l'élément on parle d'attribut ou de variable d'instances et de méthodes.

Les propriétés d'une classe ont une visibilité, c'est-à-dire que l'accès aux propriétés d'une classe par une autre est contrôlé.

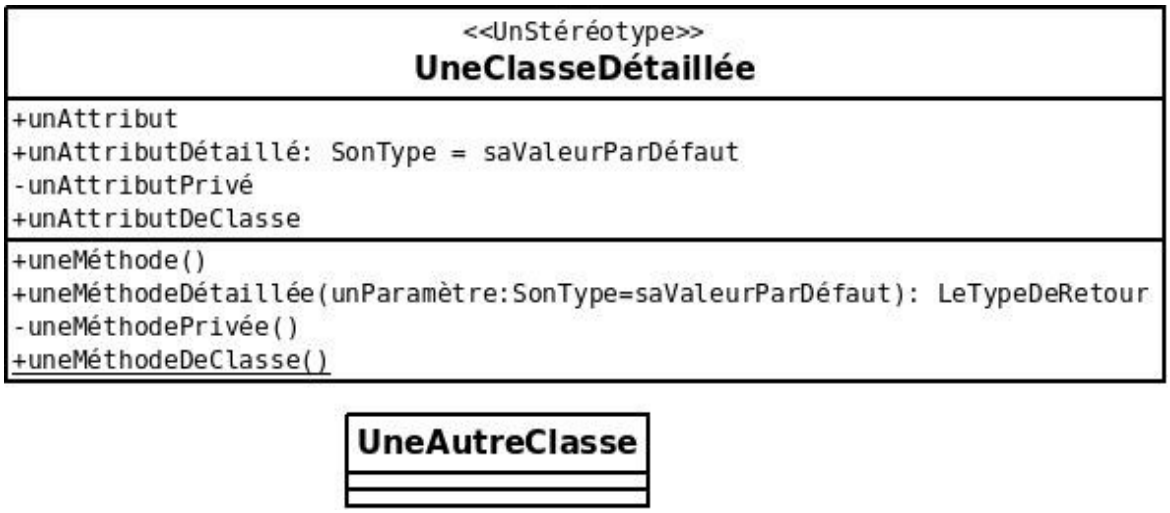


Figure 1: Représentation des classes

L'interface

C'est un ensemble de méthodes qui définissent le comportement attendu d'une classe en laissant aux classes issues de cette interface de choisir l'implémentation.

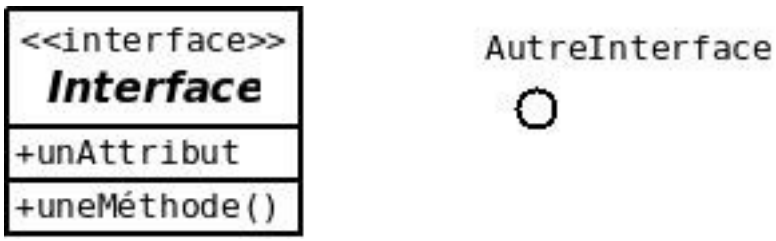


Figure 2: Représentation des interfaces

L'objet

C'est un élément concret actif ou passif appartenant à une classe. Si la classe de l'objet est connue, on parle alors d'instance de cette classe. L'objet possède toutes les propriétés de sa classe et définie les valeurs des attributs d'instances.

Figure 3: Représentation des objets

L'acteur

C'est une classe externe au système agissant ou réagissant au système.
Ainsi tous les intervenants sur le système sont des acteurs.

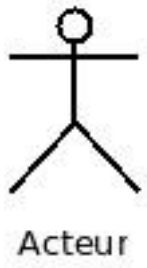


Figure 4: Représentation des acteurs

Le composant

C'est une classe formant un tout cohérent et suffisant permettant de réaliser un ensemble de fonctionnalités, c'est la boite noire par excellence.

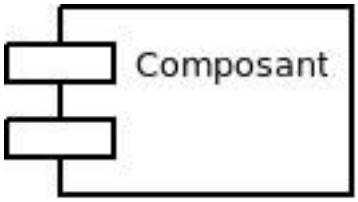


Figure 5: Représentation des composants

La collaboration

Lorsque plusieurs instances de classes réalisent un comportement global par échange de messages, on dit qu'ils forment une collaboration et que les objets y remplissent un rôle.
La collaboration permet donc de visualiser la réalisation des exigences fonctionnelles.

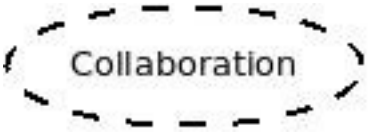


Figure 6: Représentation des collaborations

Le cas d'utilisation

C'est un comportement du système dont est témoin un acteur, il est réalisé par des collaborations.

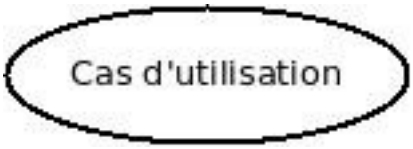


Figure 7: Représentation des cas d'utilisations

Les éléments comportementaux

Les interactions

Elles sont constituées par l'envoi de messages ou d'événements provoquant des actions chez le récepteur.
L'appel d'une méthode de classe correspond à l'envoi d'un message dont le nom est suivi de parenthèses.

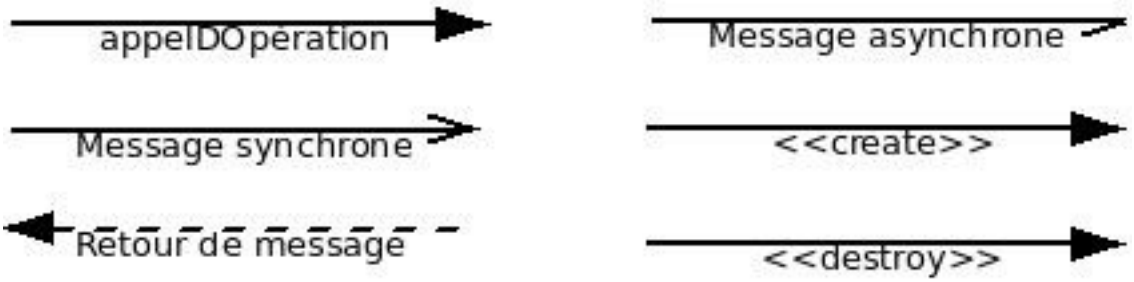


Figure 8: Représentation des messages

Les états

Ils permettent de constituer des automates décrivant le comportement d'une classe ou d'une méthode.

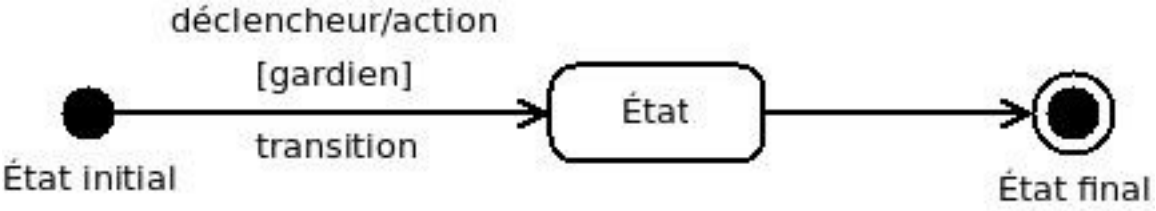


Figure 9: Représentation des états dans un diagramme d'états

Les activités

Les activités sont des comportements exécutables séquentiellement ou parallèlement.

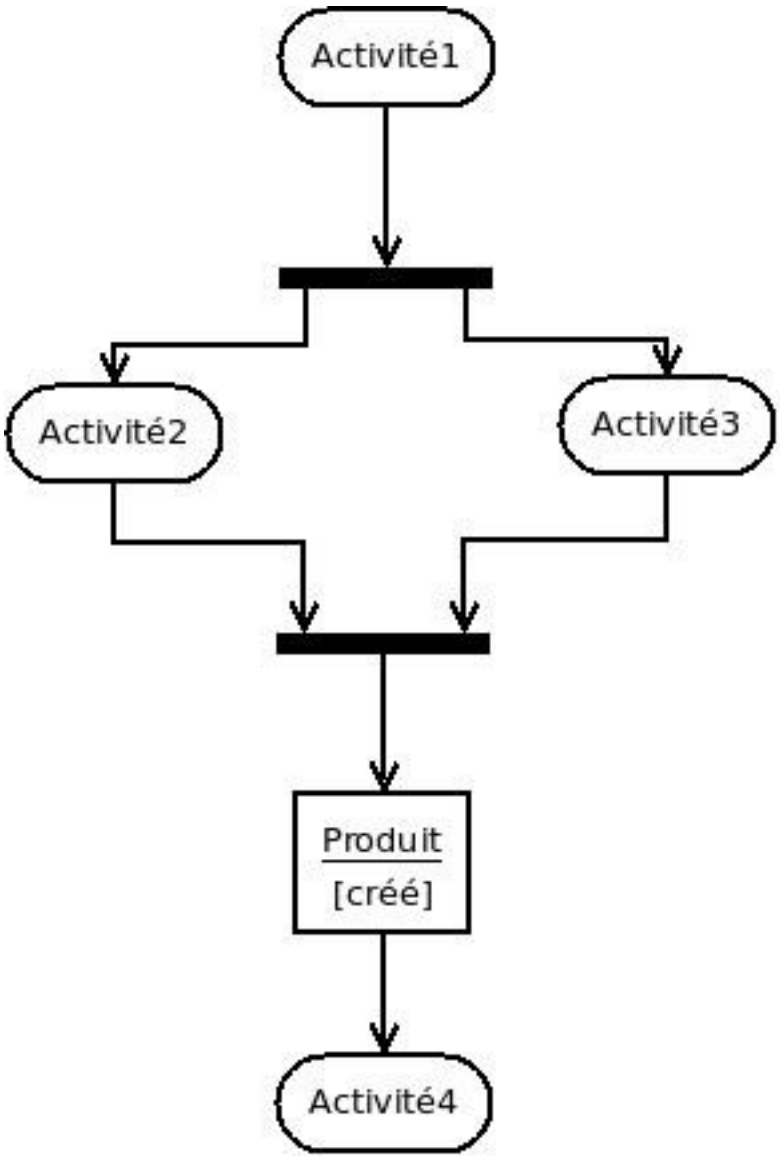


Figure 10: Représentation des activités dans un diagramme d'activités

Les éléments de regroupement

Les paquetages

Ils permettent de ranger les classes et diagrammes de façon à rendre plus clair le modèle.

Figure 11: Représentation des paquetages

Les éléments d'annotation

Les notes permettent de donner des informations.

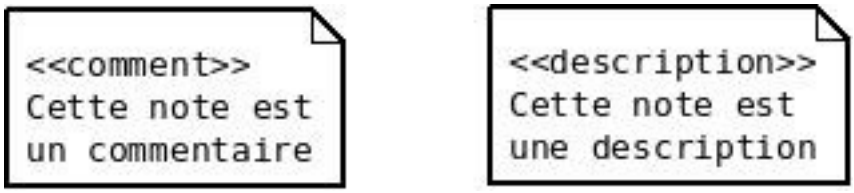


Figure 12: Représentation des notes

Les relations

La dépendance

C'est une relation sémantique indiquant que tout changement de l'élément indépendant peut affecter l'élément dépendant.

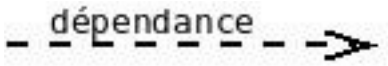


Figure 13: Représentation des dépendances

L'association

Elle permet d'indiquer une relation entre deux éléments.
La nature de la relation est donnée par le texte situé dessus. Un lien peut être précisé aux extrémités par le nombre et le rôle de chacun des éléments.

Un lien de contenance est symbolisé par l'agrégation.

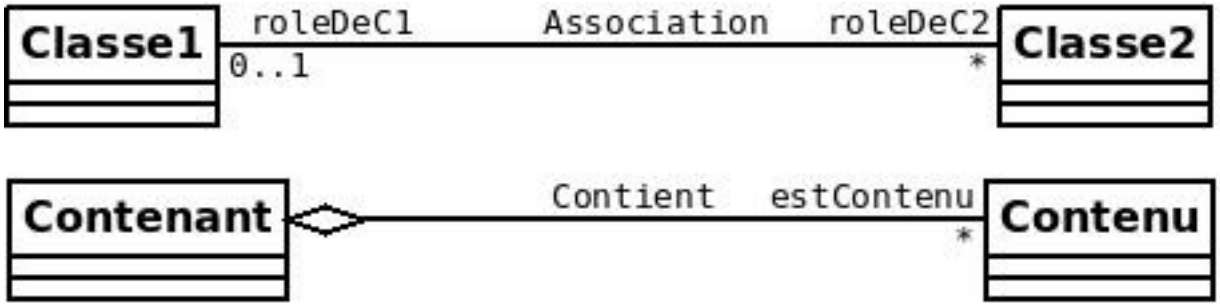


Figure 14: Représentation des associations

La généralisation

Cette relation permet de définir des niveaux d'abstraction entre les classes, le but est de permettre de manipuler de façon homogène des ensembles d'objets qui partagent les mêmes propriétés. Cette factorisation des traitements s'appelle le polymorphisme.

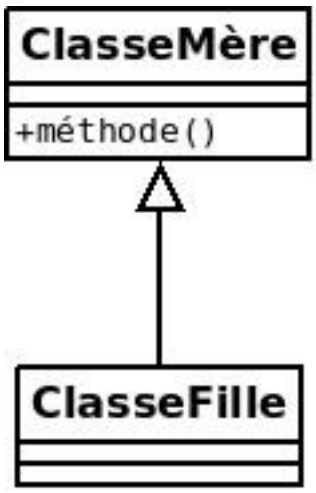


Figure 15: Représentation des généralisations

La réalisation

Cette relation indique que la classe implémente les comportements définis dans l'interface qu'elle réalise.

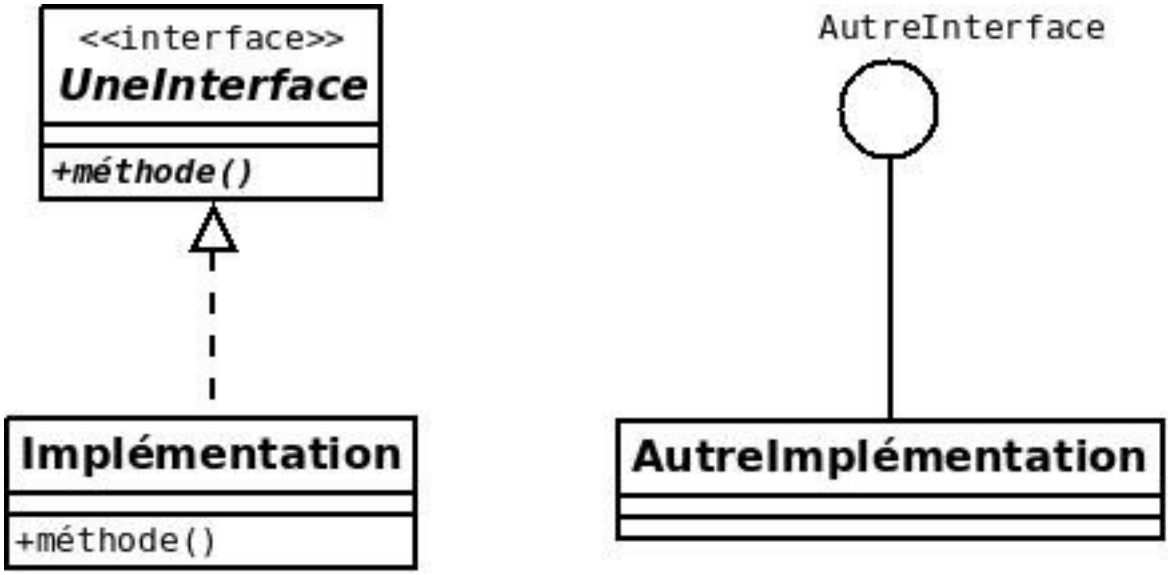


Figure 16: Représentation des réalisations

La transition

Cette relation joint deux états et permet d'expliciter les conditions à respecter et les traitements à réaliser lors des changements d'états.

Les diagrammes que nous utiliserons

Alors que UML permet de créer 13 types de diagrammes nous n'allons n'en utiliser que trois pour la génération de notre produit. Les autres diagrammes servirons à la documentation de notre architecture.
Les diagrammes qui servirons à la génération sont :

- Le diagramme de classes
- Le diagramme d'états transitions

- Le diagramme d'activités

Le diagramme de classes

Ce diagramme permet de mettre en évidence la structure des classes c'est-à-dire les attributs et méthodes présents dans les classes, les relations entre classes.
Ainsi, il fait apparaitre les hiérarchies de classes (héritage, abstraction, interface), les compositions et agrégations (un instance d'une classe contenant des instances d'autres classes), les dépendances, les "packages".

Par convention on limite le nombre de classes visibles dans un diagramme de classes. On va alors réaliser différents diagrammes de classes qui chacun doit montrer un aspect du problème en n'affichant que les attributs, méthodes et associations concernés par cet aspect.

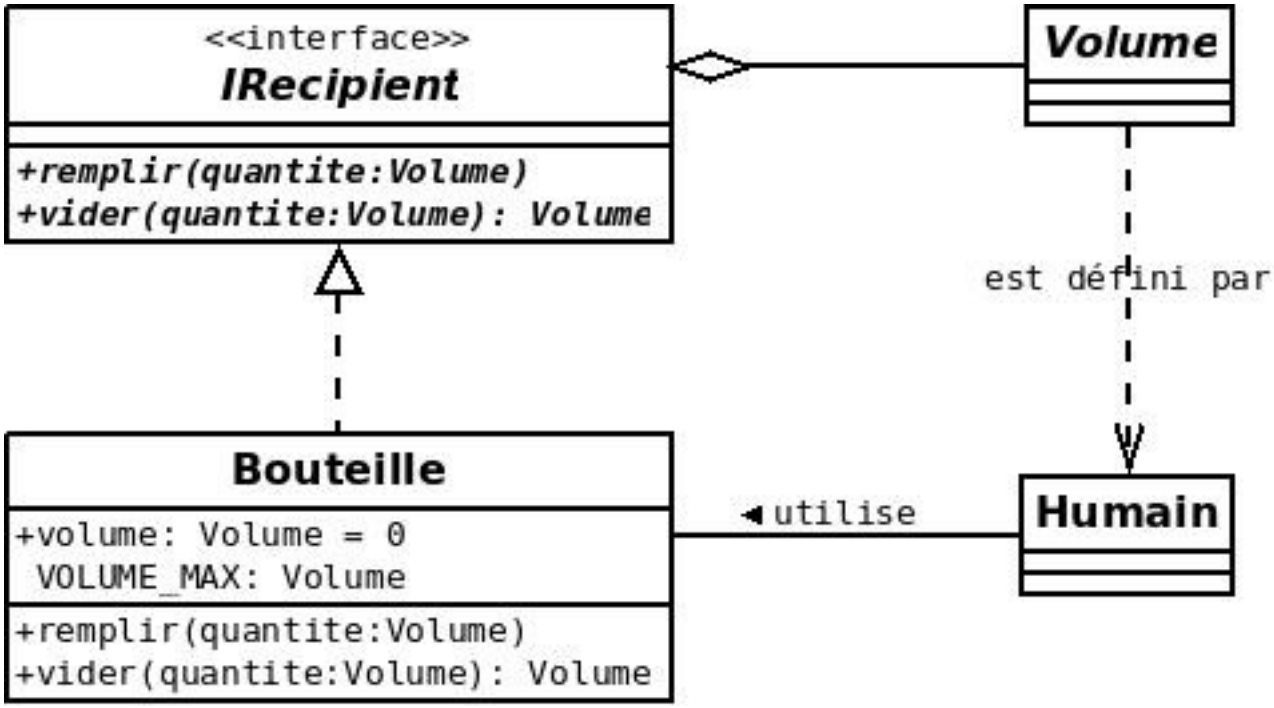


Figure 17: Diagramme de classes

Le diagramme d'objets

Il regroupe les objets et permet de les représenter avec leur liens à un moment donné.

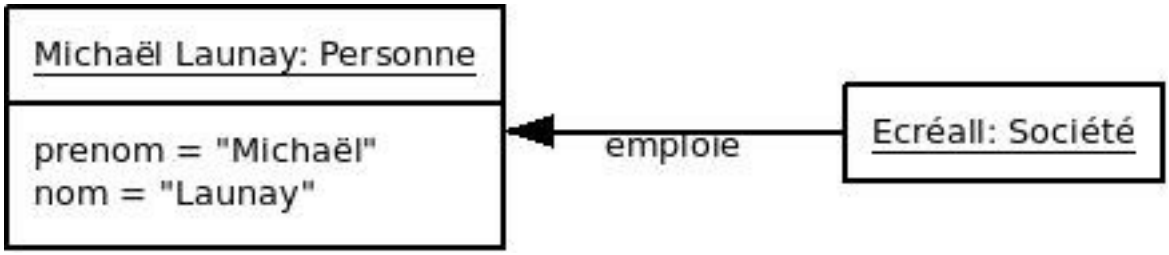


Figure 18: Diagramme d'objets

Le diagramme de cas d'utilisation

Il regroupe les cas d'utilisations et les acteurs concernés.

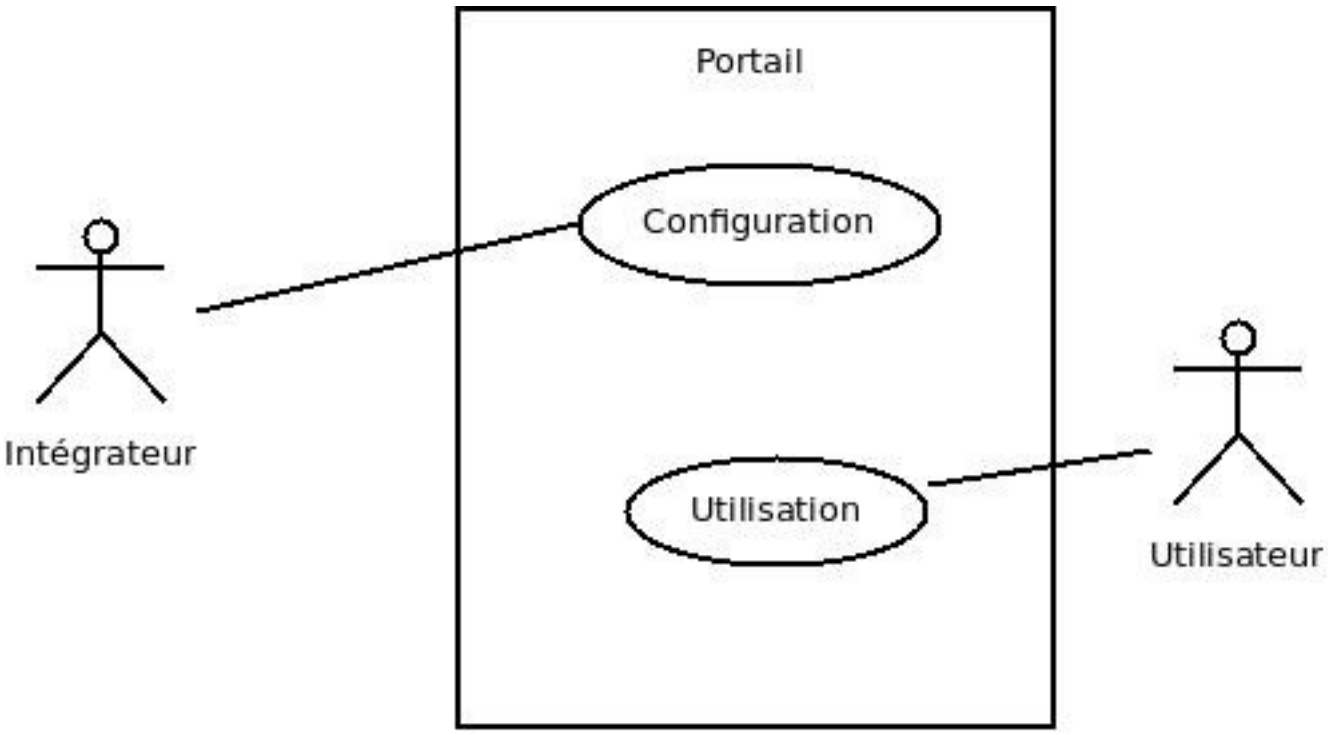


Figure 19: Diagramme de cas d'utilisations

Le diagramme de séquence

Il permet de visualiser l’enchaînement des messages entre objets. L'ordre de lecture est de haut en bas.

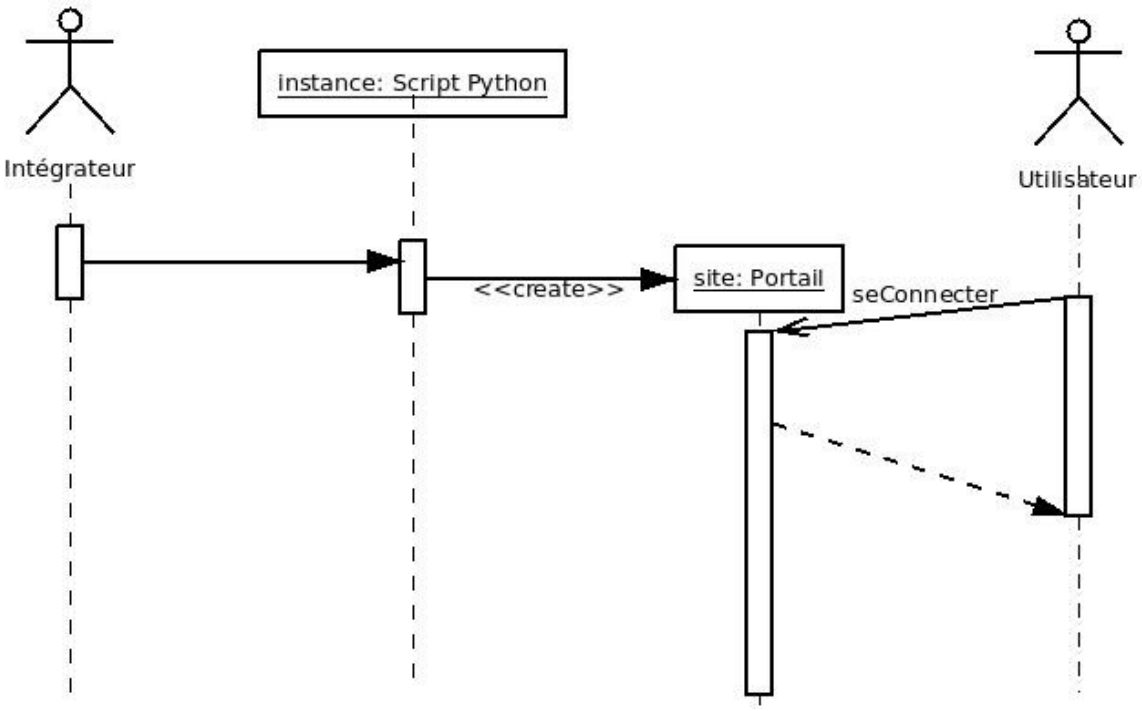


Figure 20: Diagramme de séquences

Le diagramme de collaboration

C'est une représentation symétrique du diagramme de séquence.

Le diagramme d'états transitions

Il est utilisé pour modéliser les changements d'états d'un élément.
Nous l'utiliserons pour modéliser les workflows de nos types de contenu.

Le diagramme d'activités

Il permet de voir l'enchainement des tâches du système ou des traitements déclenchés par les acteurs.

Le diagramme de composants

Il montre les liens entre composants.

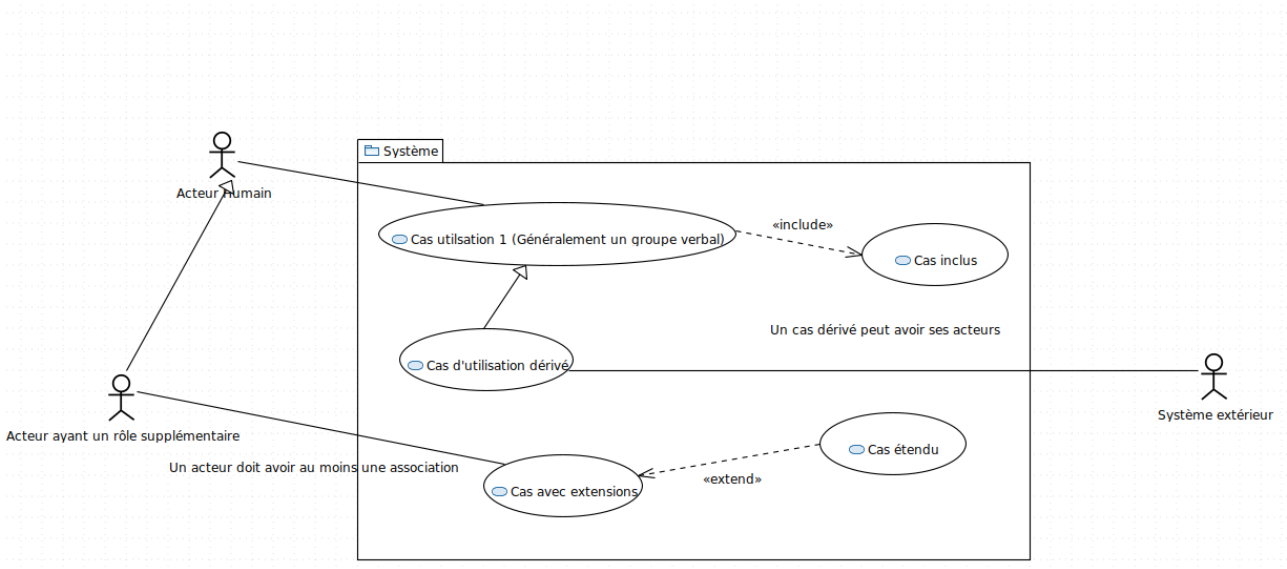
Le diagramme de déploiement

Il permet de voir l'intégration.

Présentation des diagrammes :

Diagramme de cas d’utilisation

Le système est vu comme une boite noire comprenant plusieurs grandes fonctions auxquelles participent des acteurs.
Acteur : Humain ou système extérieur qui échangent avec le système analysé.
Cas d’utilisation : Ensemble cohérent de scénarios décrivant une grande fonctionnalité du système.
Association : Lien entre un acteur et un cas d’utilisation.



Inclusion : Le cas d’utilisation de base inclut obligatoirement un autre cas à une étape donnée.
Extension : Le cas d’utilisation de base inclut optionnellement un autre cas d’utilisation selon le résultat d’une étape.
Généralisation : Le descendant enrichit le cas parent.

Détail sur le Diagramme de classes

Pour rappel il contient la structure du logiciel sous forme de classes, d’attributs, de méthodes, d’associations, de généralisations, d’interfaces.

Classe : Type des objets partageant les mêmes propriétés et comportement.

Les classes ont soit des relations d'héritage, soit des associations.

Dans les pages qui suivent nous réutiliseront plusieurs diapos de l'excellent cours « Introduction à UML 2 » de Eric Cariou de l'Université de Pau et des Pays de l'Adour (Eric.Cariou@univ-pau.fr), lui même basé sur le cours Laurence Duchien : http://www.lifl.fr/~duchien/

Diagramme de classes

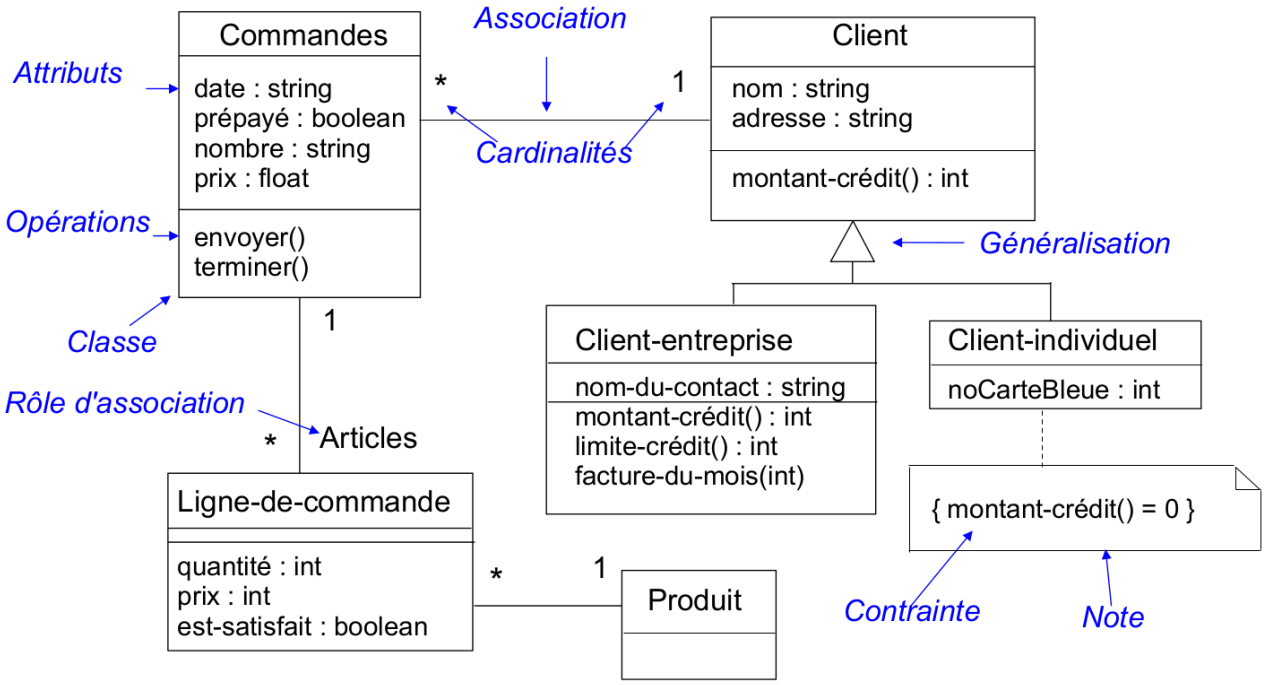


Diagramme de classes

- Attributs
  - Élément caractérisant une partie de l'état d'un objet
- Syntaxe UML pour la définition d'un attribut :

```
visibilité nom [multiplicité] : type = init
{propriétés}
```

  - `visibilité` : + (public), # (protégé) ou – (privé)
  - `nom` : nom de l'attribut
  - `multiplicité` : nombre d'attributs de ce type (tableau : [1..5])
  - `type` : type de l'attribut
  - `init` : valeur initiale de l'attribut
  - `propriétés` : propriétés, contraintes associées à l'attribut

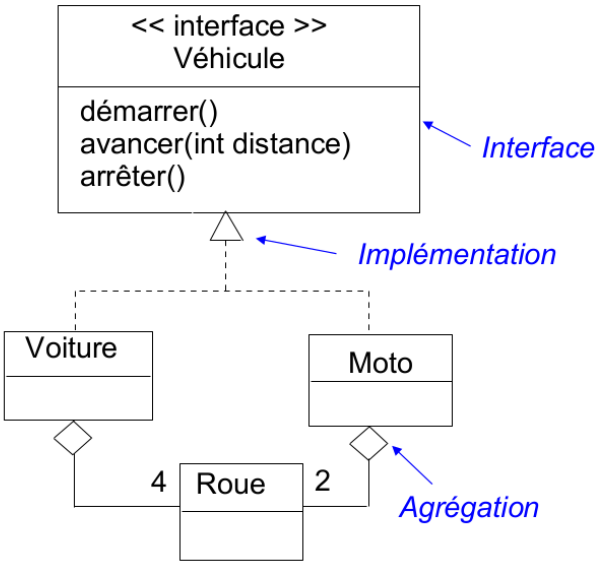
Diagramme de classes

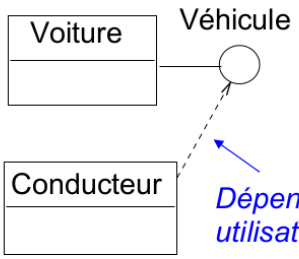
- Opérations
  - Processus/fonction qu'une classe sait exécuter
  - Appelées également méthodes dans les langages objets
- Syntaxe UML pour la définition d'une opération :

```
visibilité nom(paramètres) : typeRetourné
{propriétés}
```

  - `visibilité` : + (public), # (protégé) ou – (privé)
  - `nom` : nom de l'opération
  - `paramètres` : liste des paramètres de l'opération
  - `typeRetourné` : type de la valeur retournée par l'opération (si elle retourne une valeur)

Diagramme de classes





Agrégation : les éléments existent toujours quand l'association est détruite

Composition : les éléments disparaissent quand l'association est détruite

- Variante avec classe abstraite
  - Nom en italique
  - Ne peut être instanciée

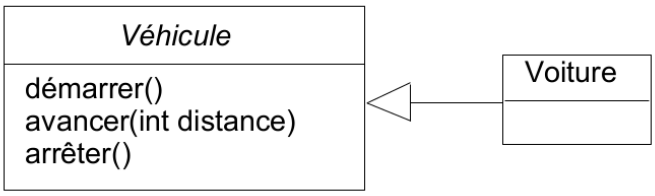
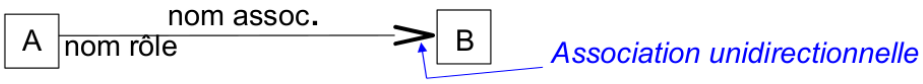


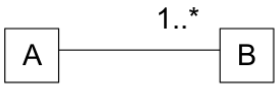
Diagramme de classes

- Détails sur associations

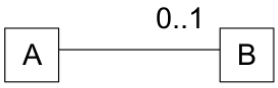


- Exemples de cardinalités d'associations

Une instance de la classe A est toujours associée avec une instance de la classe B

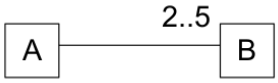


Une instance de la classe A est toujours associée avec une ou plusieurs instances de la classe B



Une instance de la classe A est associée avec zéro ou une instance de la classe B

Une instance de la classe A est associée avec zéro, une ou plusieurs instances de la classe B



Une instance de la classe A est associée avec entre deux et cinq instances de la classe B

Diagramme de classes

- Enumeration
  - Liste de valeurs manipulées comme un type
- Classe d'association
  - A chaque couple des éléments de l'association, une instance d'une autre classe est associée
    - Ici, à chaque employé d'une entreprise sont associées les informations sur son poste

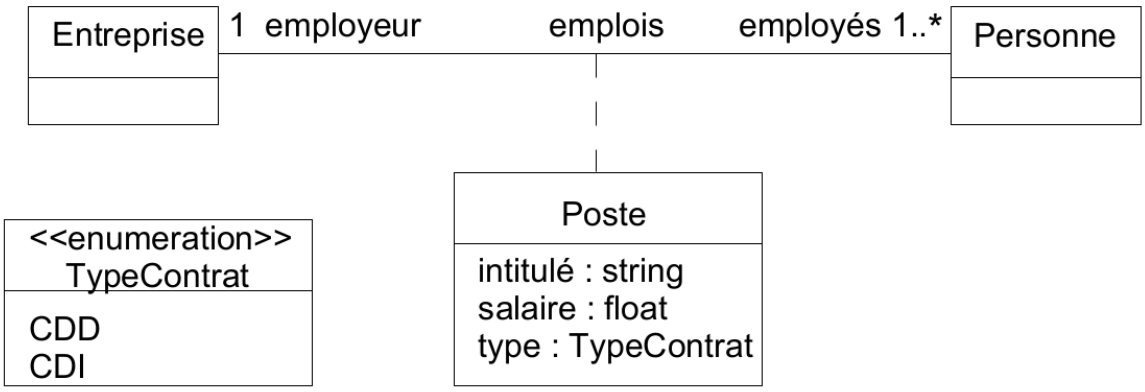
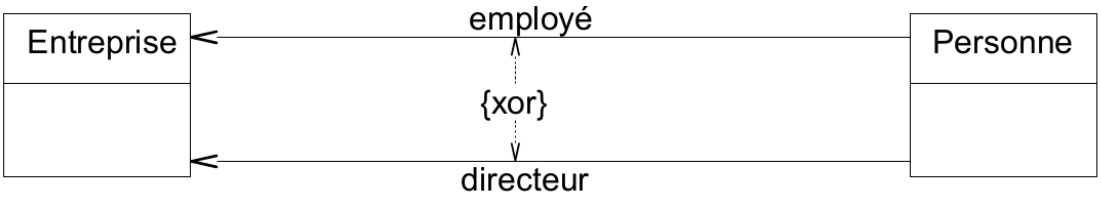
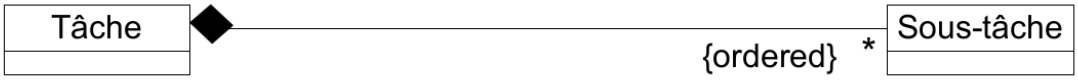


Diagramme de classes

- Contraintes sur les associations (en plus des cardinalités)
  - Relation d'exclusion entre deux associations : soit l'une soit l'autre mais pas les deux à la fois



  - Les éléments d'une association peuvent être ordonnés



  - Une association peut être le sous-ensemble d'une autre

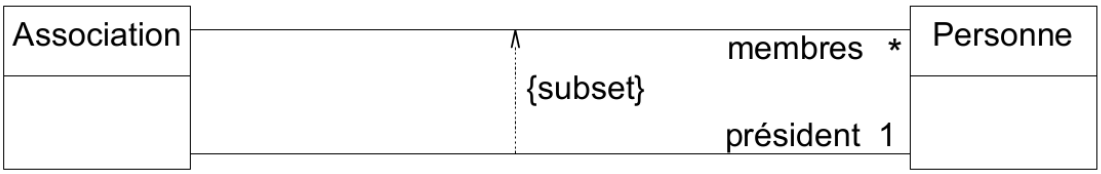
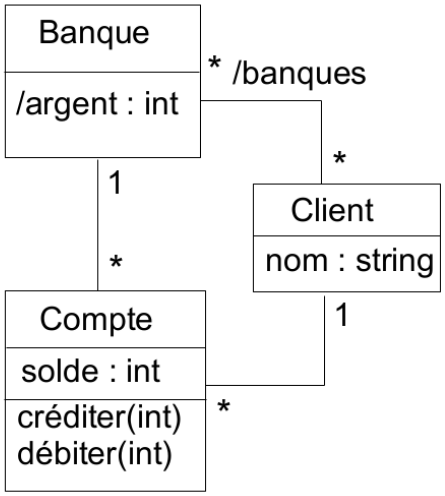


Diagramme de classes



- Eléments dérivés
  - Principalement pour attributs et associations
  - Se déduisent d'autres parties du diagramme
  - Nom de l'élément commence par /
- Exemples
  - L'ensemble des banques dont on est client se déduit de ses comptes bancaires
  - L'argent géré par une banque est la somme des soldes de ses comptes
- Ces éléments dérivés peuvent formellement être définis en OCL

Diagramme de classes

- Contraintes
  - Associations, attributs et généralisations spécifient des contraintes importantes (relations, cardinalités), mais ils ne permettent pas de définir toutes les contraintes
- UML permet d'ajouter des contraintes sur des éléments (classe, attribut, association, ...)
  - Soit des prédéfinies
    - Exemple : `{ordered}` et `{xor}` pour les associations
  - Soit des spécifiques définies par le concepteur
    - Pas de syntaxe précise préconisée, uniquement l'utilisation de `{ ... }`
    - En pratique, pour être précis, on exprimera ces contraintes en OCL
    - Exemple de contrainte explicite : on indique qu'un client individuel n'a pas de droit de crédit

Exemple de génie logiciel dirigé par les modèles.

L’exemple d’une Société spécialisée dans la modélisation es processus métier

Nous nous inspirerons de la SARL Ecréall créée en 2005 par Michaël Launay, auteur de cette formation. En 2009 la Société obtient le statut de Jeune entreprise innovante et a levé des fond en 2012, n’ayant pas eu le développement escompté, la société n’a aujourd’hui plus de collaborateur autre que son fondateur.

Ecréall constitue un cas d’école puisqu’elle avait basé son modèle économique sur l’Ingénierie dirigée par les modèles avec toute fois une très forte connotation Logiciel Libre.

Mais autopsions la Société pour comprendre la démarche IDM appliquée à une Entreprise.

Dans la suite du document nous utiliserons le présent pour narrer la méthode de cette Société en la dépersonnalisant pour des raisons de neutralité pédagogique. La méthode développée est toujours d’actualités et survivra à son inspiratrice (ce document est basé sur son Plan d’Affaires de 2020).

Un objectif

Les intranets, extranets et portails collaboratifs offrent la possibilité d'accélérer et sécuriser les échanges de l'information entre les collaborateurs des organisations que ce soient des entreprises, des associations ou des administrations.

La Société développe et intègre des intranets, extranets et portails collaboratifs full web, souples, évolutifs, faits sur mesure répondant aux besoins spécifiques de chacun de ses clients.

Ces portails collaboratifs permettent à ses clients de faire travailler ensemble plusieurs collaborateurs sur des lieux différents, à n'importe quel moment, en toute sécurité et selon les processus métiers de l'entreprise.

Un métier

La Société proposait des solutions et des services en logiciels libres[1] pour réaliser les intranets, extranets et portails collaboratifs adaptés aux besoins.

Elle était spécialiste en solutions open source et libres GNU/Linux, Python, Zope, Plone, Pyramid, React.

Ses services comprenaient : l'analyse des besoins, la conception, le développement, l'intégration, la maintenance et la formation.

1 Logiciels libres: Logiciels que l'on peut installer gratuitement, fournis avec les codes sources, que l'on peut modifier, que l'on peut diffuser, avec pour obligation d'en respecter la licence.

Une philosophie

Les logiciels utilisés, et ceux produits étaient sous licences libres, ce qui permettaient à ses clients :

- d'analyser leur code source,
- de modifier leur fonctionnement,
- de les exécuter sans avoir à acquitter de frais de licence,
- de les distribuer.

Plone et GNU/Linux ne sont pas de simples logiciels libres, ils sont gérés par des communautés regroupées au sein de fondations. Ces fondations statuent démocratiquement des questions d'avenir et de stratégie des logiciels qui en conséquence, ne sont pas imposés par la volonté d'un éditeur unique.

Pour l'instant (C.f. débat européen sur la brevetabilité des logiciels), les logiciels libres sont régis par les droits d'auteurs conformément au Code de la Propriété Intellectuelle (C.f. Article L.112-2 de la loi n°94-361 du 10 mai 1994)

Une méthode

Sa méthode se résume aux étapes suivantes :

1. Elle doit découvrir les processus métiers de son client.
2. Créer un modèle de son métier.
3. Générer un portail collaboratif qui automatise et sécurise les processus modélisés.
4. Intégrer ce portail au système d'information du client.
5. Le maintenir et le faire évoluer.

Pour adapter ses portails collaboratifs à ses clients, elle réalise une analyse de leurs processus métiers et modélise les processus identifiés soit avec Plantuml soit avec Papyrus qui est un outil dédié. Enfin, elle génére les composants qui permettent d'adapter le gestionnaire de contenu web Plone au métier de son client.

Lorsque le client a suivi une démarche qualité, la tâche d'analyse est facilitée, car une partie des processus métiers ont fait l'objet d'une description dans le référentiel qualité.

Pour accompagner ses clients dans la modélisation de leurs processus métiers, la Société réalise un audit qui consiste à identifier :

1) les documents produits,
2) les acteurs concernés par ces documents,
3) les changements d'état de ces documents,
4) d'établir les flux de documents au sein de l'entreprise,
5) de mettre à plat le plan documentaire utilisé,
6) d'identifier les processus d'entreprise.

L'ensemble des informations recueillies est rassemblé dans un modèle des processus, à partir duquel la Société génére le système d'information sous forme d'un portail collaboratif.

Ce modèle est composé de différents diagrammes qui comme les plans d'une maison donnent un aperçu du métier du client et détermine l'architecture de la solution collaborative.

À titre d'exemple, la modélisation du processus métier de notification des anomalies logicielles d'une société de services agile est représentée par le diagramme suivant.

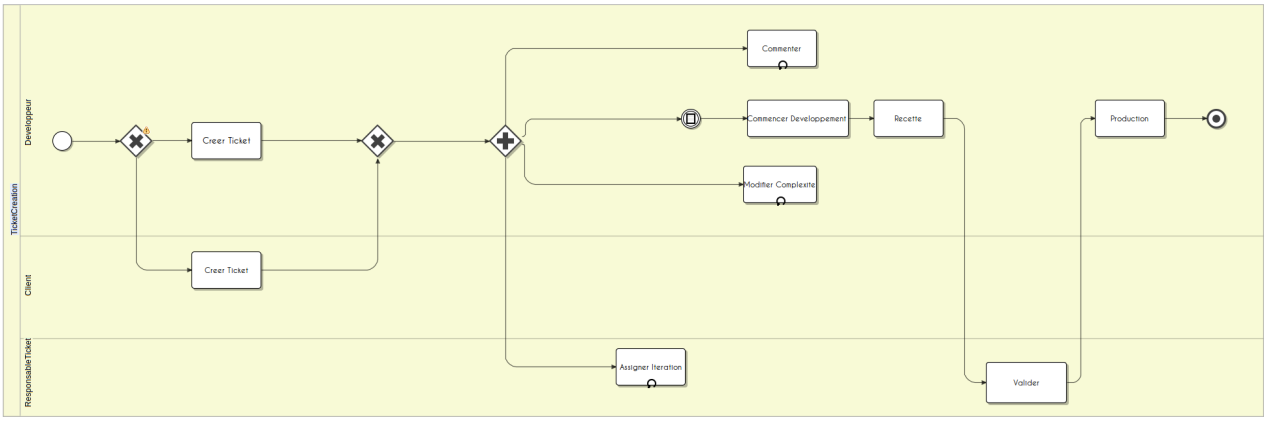


Illustration 21: Diagramme BPMN du processus Ticket d'Erreur

La réalisation des modèles repose sur des normes, des outils de modélisation et de génération de code qui existent sous forme de logiciels libres ou que la Société développe.

La grande différence avec la construction d'une maison est qu'avec une méthode comme la sienne, on peut modifier les plans et mettre à jour l'architecture des applications collaboratives alors que celles-ci sont déjà en fonctionnement. Ces changements de plan peuvent accompagner l'évolution de l'organisation du client et suivre la volonté de rapprocher le modèle de la réalité.

Le système d'information collaboratif peut être hébergé :

- chez le client par un serveur relié au réseau de l'entreprise,
- sur internet dans un *datacenter*[2],
- dans une ou plusieurs machines virtuelles (vps) fournissant le service depuis une solution de *cloud-computing*,
- *dans des images docker.*

Chaque employé de la société partage et synchronise son travail avec celui des autres, cela par le simple usage de son navigateur Internet qu'il connecte au serveur du portail.

Sa solution collaborative ne nécessite ni de changer les postes informatiques, ni d'y installer de nouveaux logiciels du moment qu'ils disposent d'un navigateur internet moderne.

Des notions de droits d'accès et de validation des documents créés ou enregistrés sur le portail permettent de gérer la confidentialité et le respect de la hiérarchie de l'entreprise.

Puisque le système d'information est un portail collaboratif, il peut être rendu accessible depuis Internet en toute sécurité en utilisant des solutions de chiffrement classique telle SSL.

La solution collaborative est progressive : dans un premier temps, ne sont informatisés que les processus du client qui apportent un gain immédiat.

2 Offre de type serveur dédié.

La prestation de développement et d'intégration réalisée par la Société est constituée d'une succession d'itérations.

À la fin de chaque itération, nous avons simultanément :

- un modèle des procédures du client de plus en plus complet,
- une nouvelle version du portail collaboratif automatisant encore un peu plus les procédures du client.

L’offre de la Société :

Les développements au forfait ou en régie

La Société réalise des « contrats agiles »[3], ses projets sont réalisés par itérations courtes de deux à trois semaines impliquant fortement le client.

À chaque itération, le client et la Société décident des fonctionnalités devant être livrées. Les fonctionnalités apportant le plus de valeur à l'application sont réalisées en premier. Ainsi, **le projet couvre rapidement les fonctionnalités importantes et une relation de confiance s'établit**.

Le projet est facturé à l'itération. Le client peut décider d'arrêter le projet à la fin de chaque itération.

L'ajout de nouvelles fonctionnalités à couvrir est donc supporté de façon claire et non ambiguë, soit ces fonctionnalités décalent la réalisation d'autres moins importantes soit on ajoute une itération au projet dont le coût financier est clairement déterminé et supporté par le client.

Cette méthode est basée sur la méthodologie agile adaptée aux besoins des clients.

La maintenance

Si le client a choisi de suspendre les développements principaux, il aura probablement besoin de **conserver une garantie de service** au-delà de la garantie initiale**, et d'intégrer régulièrement de nouveaux développements et mises à jour**.

La Société lui propose un contrat de Tierce Maintenance Applicative (TMA). Il permet de garantir, via un système d'abonnement, que toute demande de résolution de panne, de correctif logiciel, ou d'évolution ponctuelle, sera réalisée et livrée dans des courts délais.

Dans le cadre de son offre de maintenance, la Société s'engage fortement sur sa rapidité de réaction et de livraison. Des pénalités supérieures aux standards sont prévues en cas de retard.

L’application du client reste donc vivante et réactive aux nouveaux usages, et ce, en dehors des périodes de développements.

La formation

La Société est organisme formateur et dispense régulièrement des formations à Python, Zope, Plone, et Linux. **Toute offre de développement pourra être accompagnée d'une formation spécifique** à la plateforme livrée, et ce, pour les différents profils d'utilisateur : utilisateur, administrateur, et même développeur.

Conformément à l'esprit du logiciel libre, la Société forme ses clients de sorte qu'ils soient les plus autonomes possible avec leur application, et qu'ils puissent la faire évoluer eux-mêmes suivant leurs besoins.

---

3 Voir méthode agile http://fr.wikipedia.org/wiki/Méthode_agile et http://alistair.cockburn.us/Earned-value+and+burn+charts

L'assistance technique

La Société propose enfin de mettre à disposition une expertise technique dans le cadre d'audits (de sécurité, de qualité, de performance), d'accompagnement technique.

La Société propose différents types d'audit :

- **Audit de sécurité** : elle vérifie la résistance de la plateforme développée aux attaques logicielles, et la conformité de son paramétrage aux contraintes de confidentialité du client ;
- **Audit de qualité** : la Société évalue la pérennité de la plateforme par un audit de qualité du code, de respect des standards, d'emploi de technologies actuelles;
- **Audit de performances** : la Société recherche les améliorations permettant d'optimiser les temps de réponse de l'application, elle évalue sa capacité à répondre à la charge attendue.

La Société propose également des accompagnements techniques pour aider ses clients (utilisateurs ou développeurs) à faire évoluer leur plateforme sans s'engager dans un processus en mode projet.

Les outils

Modélisation d'un portail collaboratif

La phase de modélisation permet de capturer les besoins du clients en synthétisant sa façon de travailler dans des processus métier.
L'objectif est de montrer comment le portail collaboratif va soutenir les processus métier, interagir avec les collaborateurs ou les systèmes extérieurs et comment ce nouvel outil va s'insérer dans les processus, quel sont les événements et documents qu'il va recevoir, traiter, créer, modifier, ou transmettre.
Les processus métier sont décrits avec des diagrammes BPMN (Business Process Management Notation). Ces processus métier permettent d'identifier les documents produits et échangés.
Les documents et leurs cycles de vie sont à leur tour décrit par des diagramme UML.

L'ensemble des diagramme sont réalisés avec des outils et modeleur/générateur comme Papyrus ou Plantuml.
La Société utilise le diagramme BPMN pour modéliser les processus métier et trois diagrammes UML pour modéliser ses portails (le diagramme de classes, d'états-transitions et d'activités).
L'ensemble des documents disponible sur un portail est modélisé par un **diagramme de classes**, dans lequel une classe représente un document donné.
C'est la **partie statique** de la modélisation.

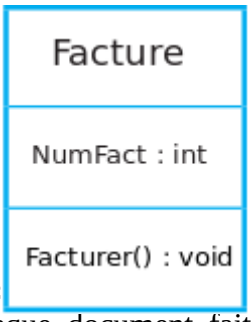


Illustration 22 : la classe Facture

Les **parties dynamiques** sont modéli ımme BPMN pour l'orchestration des processus, leur synchronisation, et chaque document fait l'objet d'un ***workflow*** (cycle de vie) documentaire modélisé par un **diagramme d'états-transitions** pour décrire les différents états que peut avoir un document.

Prenons l'exemple d'une facture, vous avez sa classe ci-dessus, et son *workflow* associé ci-dessous. La facture lorsqu'elle est créée est dans l'état *due*, elle peut ensuite évoluer vers l'état *payée* lorsque le client l'aura effectivement payée.

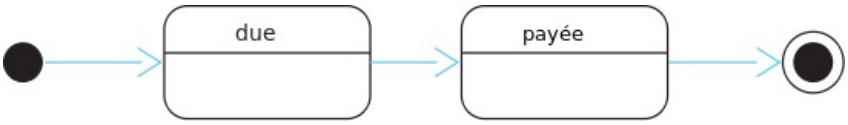


Illustration 23 : le workflow associé à la facture

Jusqu'en 2011, la Société utilisait le logiciel libre **ArchgenXML** pour transformer la modélisation en des produits Python (sorte de modules) pour le portail **Zope/Plone**.
Depuis 2010 elle a basculé progressivement sur les outils Papyrus et Acceleo basés sur les standards EMF, Ecore, GMF et MOF, QVTO et XText proposés par l'EDI Eclipse (Voire cours Cédric Dumoulin https://www.fil.univ-lille1.fr/portail/index.php?dipl=MInfo&sem=GL&ue=IDM&label=Pr%C3%A9sentation
).

Modélisation d'un processus métier

Un processus métier désigne « un ensemble d'activités reliées les unes aux autres pour atteindre un objectif, généralement dans un contexte organisationnel qui définit des rôles et des relations ».
Une modélisation de processus peur se représenter par un diagramme d'activités.
Voici un exemple concret tiré du livre *Le guide de l'utilisateur UML*.

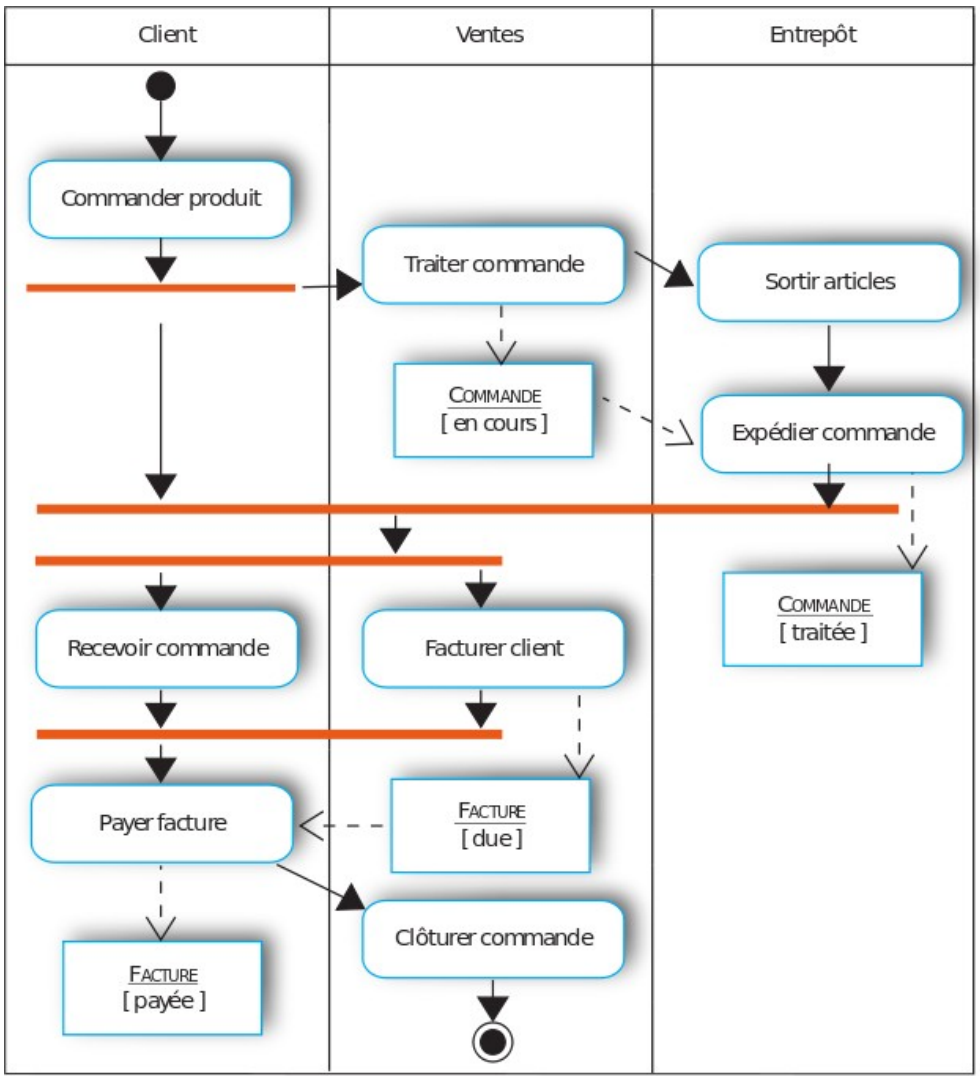


Illustration 24 : exemple de processus

Ce diagramme s'ajoute aux diagrammes de classes et d'états-transitions. Comme nous pouvons le deviner avec ce diagramme d'activités, nous avons la classe Facture, avec un workflow associé (états due et payée) et la classe Commande avec son workflow (états en cours et traitée).
À un des états du diagramme d'activités, nous pouvons associer un objet, préciser le type et l'état dans lequel il se trouve.
Sur le diagramme, l'action « Traiter commande » produit un objet Commande dont l'état est en cours, ce même objet est passé à l'action « Expédier commande » qui modifie son état en traitée (l'état de l'objet est indiqué entre crochets sur le diagramme).
Pour générer du code à partir de cette modélisation de processus, il faut tout d'abord le transformer en un autre modèle dépendant de la plateforme. Ensuite ce dernier serait transformé en code Python pour le portail.

Depuis 2009 et conséquence de ses travaux de recherche, la Société modélise les **processus métiers** et génère leur portail à partir d'un métamodèle appelé MACOP qui est la fusion du BPM et d'UML. Le processus métier est alors modélisé dans un diagramme qui suit en partie la notation Business Process Management Notation.

Développement par les modèles : approche MDA

Les développements réalisés par la Société ont ceci d'original qu'un portail est réalisé presque sans écrire une seule ligne de code. Ce mode de développement repose sur des modèles.

L'OMG (Object Management Group), l'organisation qui standardise UML et autres outils, a émis les bases de cette approche appelée MDA (Model Driven Architecture), l'**ingénierie logicielle guidée par les modèles**.

Le but de cette approche est d'avoir un **modèle**, à partir duquel nous générons le code.
Un exemple simple est une classe UML qui peut être **transformée en code** d'une classe java.

Mais une application ne se résume pas à une seule classe, l'approche MDA est plus compliquée que cela.

Transformation de modèles

La transformation de modèles s'appuie sur les concepts de méta-modèles.

Un méta-modèle définit ce qu'il est possible d'exprimer dans un modèle, une sorte de contrat que doit respecter le modèle ; les règles de transformation sont définies entre les méta-modèles source et cible.

La transformation s'effectue à partir du modèle. Voici un schéma qui récapitule ce qui vient d'être dit :

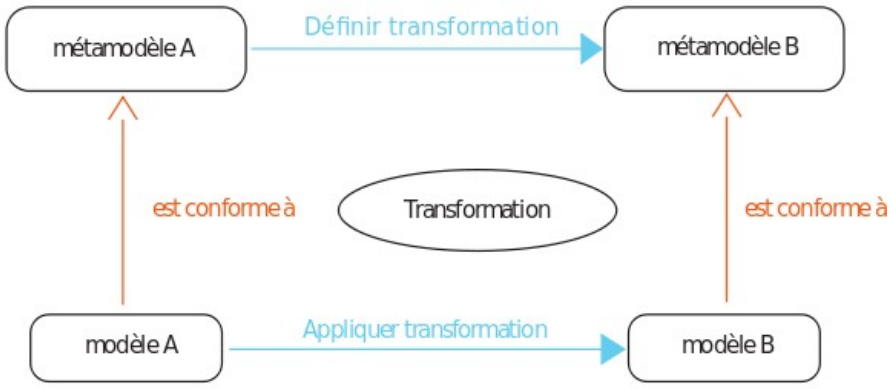


Illustration 25 : Schéma de transformation de modèles

La transformation de Modèle à Modèle se fait par l'utilisation de QVTO en travaillant au niveau des MétaModèles. Pour faire un parallèle avec les langues, c'est comme de créer un dictionnaire de traduction pour la syntaxe et une correspondance grammaticale entre deux langue. C'est très générique, ça s'applique à quasiment tout les textes, mais de temps à autre il arrive qu'il faille ajouter une règle spécifique de traduction pour une phrase ou un mot donné, car il peut ne pas exister de traduction univoque.
Ainsi avec QVTO, on élabore les règles de transformation en réfléchissant au niveau MétaModèle et on applique ces règle pour transformé un modèle conforme au premier MétaModèle vers un modèle conforme au MétaModèle destination.

Détails des produits développés

Méta-Modèle

Chaque client possède une organisation qui lui est propre, mais dont on peut abstraire des fonctions, des actions, des rôles, et des résultats, communs avec l'organisation d'autres clients.

L'idée est de regrouper ce qui est commun au sein d'un modèle abstrait.

Ce modèle de modèle, que l'on appelle méta-modèle, facilite l'identification des rôles, des fonctions, des objectifs, des résultats et des tâches, présentes dans toute organisation.

Le méta-modèle évolue après chaque intégration de la solution développée pour un client, ceci afin de réutiliser au maximum ce qui a été produit.

Le méta-modèle est soit produit en utilisant la notion de « Profile » fourni par UML, car cette notion permet d'enrichir le méta-modèle d'UML et ainsi de pouvoir partager ces innovations entre plusieurs outils UML, soit d'utiliser MOF (Méta Object Facility) ou EMF. (Eclipse Méta Object Facility).

La Société exploite un juste milieu entre la méthodologie d'Ossad qui a déjà réalisé une dichotomie de l'organisation des entreprises, et celle du BPM qui, à travers l'organisation « Business Process Management Initiative », a explicitement demandé l'enrichissement d'UML 2.0 pour faciliter la modéliser les processus d'entreprise.

Une première version du méta-modèle a été publiée en 2009.

Modeleur

La réalisation de modèles à partir du méta modèle passe par l'utilisation d'un outil appelé modeleur.

La Société travaille avec l'outil Papyrus, modeleur BPMN intégré à Eclipse.

Générateurs

Le but de ces outils est de réaliser la transformation du résultat de la modélisation vers un intranet et vers un moteur de workflow.

La solution retenue repose sur la technologie de transformation d'un document XML en un autre document XML, via le langage de transformation QVT (Query/View/Transformation).

Les modeleurs UML savent exporter au format XMI, qui est un format XML dédié à la modélisation, et certains serveurs d'applications à la base des portails collaboratifs, transforment un document XML spécifique en composants exécutables.

Idem pour les moteurs de workflow pour lesquels le document XML est au format WfML.

Approche par composants

Un composant est un bloc logiciel indépendant qui réalise une fonction ou une sous-fonction d'un système informatique.

Cette approche permet de capitaliser au maximum les développements logiciels effectués, et autorise la livraison des fonctionnalités de l'intranet au fur et à mesure de son développement.

Livrer les blocs réalisés de façon progressive permet au client d'appréhender peu à peu son système, selon ses besoins et ses capacités financières.

L'intérêt de cette démarche réside dans la possibilité d'utiliser des composants réalisés et éprouvés.

L'intégration de ses composants « sur étagère » avec le produit consiste à les rajouter au méta-modèle.

Serveurs à base de GNU/Linux

Les serveurs intranet et les workflows de chacun des clients sont hébergés chez eux ou en data center ou sur le cloud par un ou plusieurs serveurs Linux.

Ces serveurs Linux sont reliés à Internet, ainsi la Société peut, à distance, livrer les composants développés et réaliser la maintenance.

Le choix de Linux a été effectué pour des raisons de coût, de sécurité et d'évolutivité.

Caractère innovant de la technologie

OMEGSI est constituée d'un ensemble d'outils open source déjà existants ou développés par la Société. Ils permettent la modélisation des entreprises et la génération de portails collaboratifs adaptés.

Les portails collaboratifs générés par OMEGSI utilisent de très nombreux composants « disponibles sur étagères » offrant ainsi un haut niveau de qualité et de sécurité.

L'innovation est portée à la fois par la démarche de modélisation des entreprises, les outils de génération du système d'informations, et sur une volonté de réaliser une chaîne complète de Business Process Management (BPM).

La Société met au point les outils et technologies suivants :

- Réalisation d'un ensemble de méta-modèles (modèle de modèle) selon le mécanisme d'extension « Profile » du formalisme Unified Model Language (UML). Le but de cet ensemble de méta-modèles est de permettre de modéliser le plus efficacement possible les procédures du client .

- Modification d'un modeleur UML open source permettant de créer rapidement des modèles à partir du méta-modèle.

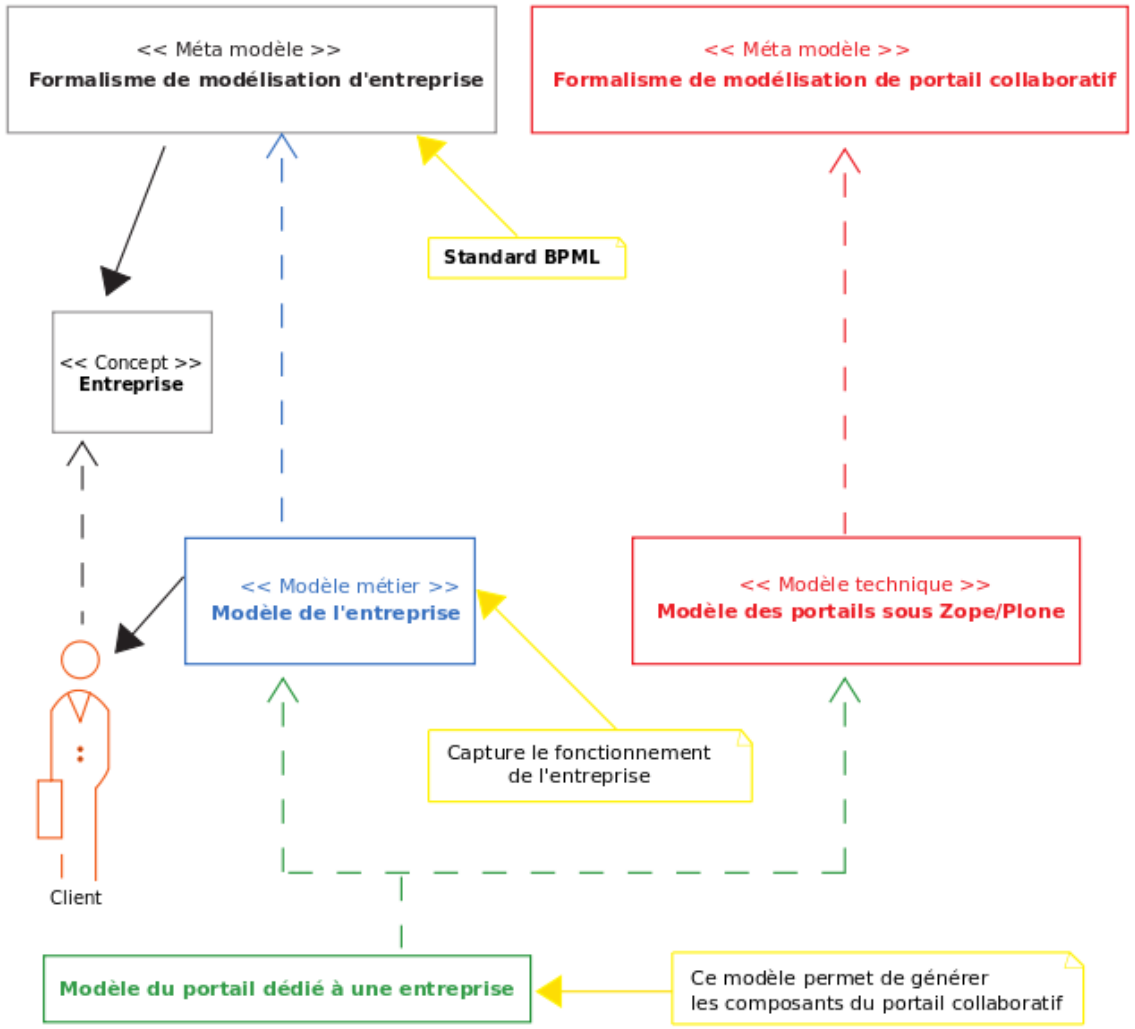
- Réalisation d'outils informatiques nommés Générateurs permettant, à partir du modèle d'une organisation, de générer l'intranet et le système de workflow adéquat.

- Utilisation de l'approche par composants, qui permet de livrer à distance via Internet les blocs fonctionnels du système réalisé, et ceci selon les priorités définies en partenariat avec le client.

Cette découpe de l'outil OMEGSI en différents « petits » outils simplifie la réalisation et permet la planification de l'ensemble des tâches nécessaires à la mise au point de la chaîne complète.

L'ensemble de la chaîne est résumé par le schéma suivant, où sont en rouge les outils OMEGSI, en vert et bleu les modèles produits par la prestation de réalisation et d'adaptation des systèmes d'information collaboratifs.

Le serveur d'applications, le moteur de workflow et une grande majorité des composants nécessaires aux systèmes d'informations existent sous forme de logiciels libres.



Dessin 1 : La chaîne de production de portails collaboratifs

Les développements utilisent les méthodes Agiles telle le « pair programming ».

Liberté d'exploitation et risques de contrefaçon

Les produits réalisés sont des logiciels ne faisant pas partie d'un procédé visant à produire un bien matériel.

En conséquence et pour les logiciels, la propriété des droits revient à l'employeur lorsque ceux-ci sont créés par ses employés dans l'exercice de leurs fonctions.

L'acquisition des droits est automatique à la création de l'œuvre.

La durée d'application de ces droits est valable jusqu'à 70 ans après le décès des auteurs.

La protection d'un logiciel consiste donc à prouver que l'on est bien l'auteur du logiciel, ce qui peut facilement être obtenu par envoi de lettre Soleau à l'INPI (durée 6 ans).

Néanmoins, avec les outils de reverse engineering il est illusoire de penser qu'un quelconque moyen technique protégera les produits créés.

De plus, une grande partie des technologies nécessaires aux outils peut être satisfaite par l'usage de logiciels libres, les produits dérivant de ces logiciels doivent à leur tour être libre (propagation de la licence GPL et CECILL).

En conséquence, et afin de pouvoir se protéger contre toute contestation de droits d'auteur, il a été décidé de produire des logiciels libres selon la licence CECILL (licence libre respectant le droit français, mis au point par le CNRS, le CEA, et l'INRIA).

La protection du produit s'effectue alors par la somme des compétences à maîtriser.

Dès lors, ce qui est vendu n'est plus le logiciel, mais les services autour du logiciel (garantie, packaging, formation, utilisation, etc.).

Aspects réglementaires

Les produits installés (l'équipement informatique) par la Société sont soumis à la garantie légale en cas de défauts ou de vices cachés (art.1641 et suivant du code Civil).

Le cas des logiciels est particulier, car le droit considère qu'en tant qu'œuvre, ils ne peuvent être cédés, seuls les droits voisins peuvent l'être (droit de diffusion et de représentation), en conséquence le droit considère que le logiciel est « loué ».

La Loi du 5 janvier 1988 dite Godfrain qui traite « des atteintes aux systèmes de traitement de données » s'applique en ce qui concerne toute tentative d'atteinte à l'intégrité des systèmes d'informations SICAE produits.

Il est à noter que les systèmes produits par la Société respecteront l'article L 121-8 du code du travail ayant trait aux dispositifs de surveillance des employés, ainsi que la loi du 6 août 2004 relative à la protection des personnes physiques à l'égard des traitements de données à caractère personnel (Nouvelle Loi Informatique et Liberté).

Positionnement la Société

Chaque organisation s'appuie sur des modes de collaboration spécifiques pour fonctionner et se développer. Ces modes de collaboration sont un élément essentiel de leur savoir-faire et de leur avantage concurrentiel.

La Société définit et conçoit à partir de vos modes de collaboration spécifiques, le système d'information collaboratif adapté à la logique de fonctionnement et au métier du client. Accessible par un simple navigateur Internet, le système d'information collaboratif peut être utilisé quel que soit la machine et l'environnement informatique des utilisateurs sous la forme d'un Intranet, un Extranet ou une plateforme collaborative.

Pour répondre en permanence à l'attente des utilisateurs du client, la réalisation de son système d'information collaboratif se fait sous la forme de cycles de développement courts avec des livraisons régulières de lots de fonctionnalités. Pour lui permettre d'interagir à tout moment, en temps réel et en direct avec l'équipe de développement, la Société met à disposition un extranet projet avec un tracker.

Développé sur la base des systèmes de gestion de contenu Pyramid/Substance D ou Plone, ce votre système d'information collaboratif ne dépend d'aucun éditeur et peut être repris, le cas échéant, par un autre prestataire, membre de la communauté Python. La pérennité de l’investissement du client est garantie par la capacité la Sociétéde la Société à maintenir dans le temps la compatibilité des versions de Plone et Substance D et à étendre le système d'information collaboratif du client à de nouveaux usages et de nouveaux contextes sans régression fonctionnelle.

L'utilisation de Docker permettant la mise en production des applications

Depuis janvier 2015, la Société s’investit pleinement sur l'utilisation de la technologie Docker qui permet des mises en production rapides et parallèles d'applications sur un serveur. Elle a développé une forte compétence.

Le développement d’applications ReactJS et ReactNative depuis 2016

La société a publié deux applications développées en ReactNative sur les stores d’Apple et de Google.

Depuis 2016, elle développe ses interfaces homme machine en ReactJS.

L'application des méthodes Agiles depuis 2009

Depuis 2009, la Société applique systématiquement les méthodes Agiles pour gérer les projets de ses clients.

La mise en œuvre des méthodes Agiles a conduit la Société :

- à mettre systématiquement à disposition de ses clients des équipes de développeurs expertes, stables dans le temps, capable de les accompagner du début à la fin de leurs projets,
- à proposer des forfaits Agiles sous forme de cycles de développement ou itérations courts, généralement inférieurs à un mois, tout en garantissant un haut niveau de qualité,
- à commencer, pour chaque itération, par des spécifications détaillées mais limitées à un périmètre précis, pour livrer rapidement des évolutions significatives et visibles par les utilisateurs,
- à permettre aux clients d'intervenir à tout moment sur l'itération en cours via un extranet projet avec un tracker et visualiser en permanence l'état d'avancement des itérations,
- à faire pour chaque itération une démonstration des développements réalisées et permettre aux clients d'effectuer des tests de recette utilisateur réguliers avant chaque livraison en production.

Chaque projet dispose d'un extranet projet mis à la disposition des clients qui permet de consigner l'ensemble des documents et contacts relatifs au projet et d'avoir un système de suivi de projet, tracker, permettant aux clients de pouvoir notifier à tout moment une demande de fonctionnalité, une question ou un problème rencontré.

L'extranet projet permet également d'échanger en temps réel entre le client et le prestataire, d'avoir l'historique des demandes formulées et de suivre en temps réel l'état d'avancement de chaque itération.

Pour assurer une fiabilité maximale dans les développements livrés, la Société travaille pour chaque projet client avec deux serveurs dédiés : un serveur de recette et de secours pour livrer les développements réalisés en recette avant validation par les clients et sauvegarder l'ensemble des données du système d'information collaboratif du client et un serveur d'exploitation pour les déployer en production.

Architecte de l'information depuis 2005

Depuis la création de la Société, celle-ci a développé un savoir faire dans l'architecture de l'information permettant de créer des systèmes d'information multi-canaux et transcanaux.

Elle a suivis les nombreuses évolutions des méthodes de ce domaine, passant des cycles en V au méthodes Agiles, Du design selon J.J Garett à l'approche transcanal de Resmini et Rosati selon les cinq principes suivants :

1) Rendre explorable et appropriable (« place-making »)

1.1) Explorable

Un “nouvel arrivant” doit pouvoir s’orienter, même sans expérience préalable.

Il doit trouver une aide adéquate à l’orientation, implicite ou explicite (info-bulles, FAQ ou foire aux questions)

1.2) Appropriable

Permettre à l’utilisateur d’adapter son environnement : identification, cookies…

2) Rendre cohérent (« consistency »)

Revient à chercher un ordre, et utiliser des classements qui font sens.

Il n’y a pas de classification juste ou fausse.

La classification doit être adaptée à un ensemble d’utilisateurs donnés pour une tâche spécifique.

Cela revient à :

- Identifier les catégories de base

- pour les utilisateurs visés

- avec les utilisateurs visés, via le tri par cartes, par exemple

- Organiser l’interaction autour de ces catégories

- Dégager ensuite les catégories soit plus générales soit plus spécifiques qui sont effectivement nécessaires.

3) Rendre souple et robuste (« resilience »)

- Un espace informationnel souple s’adapte à chaque contexte d’usage et à différents besoins ou stratégies de recherche des utilisateurs

- Un espace informationnel robuste maintient en même temps son organisation d’ensemble

- L’accès à l’information se fait le plus souvent de manière passive et sans but explicitement formulé afin de favoriser les trouvailles (sérendipité – serendipity)

- Proposer directement à l’utilisateur des informations possiblement pertinentes pour lui laisser le choix.

4) Rendre simple (« reduction »)

- « plus de choix » ne doit pas se transformer en « trop de choix »
- Ne pas diminuer le nombre de choix possibles mais ordonner les choix pour faciliter les parcours

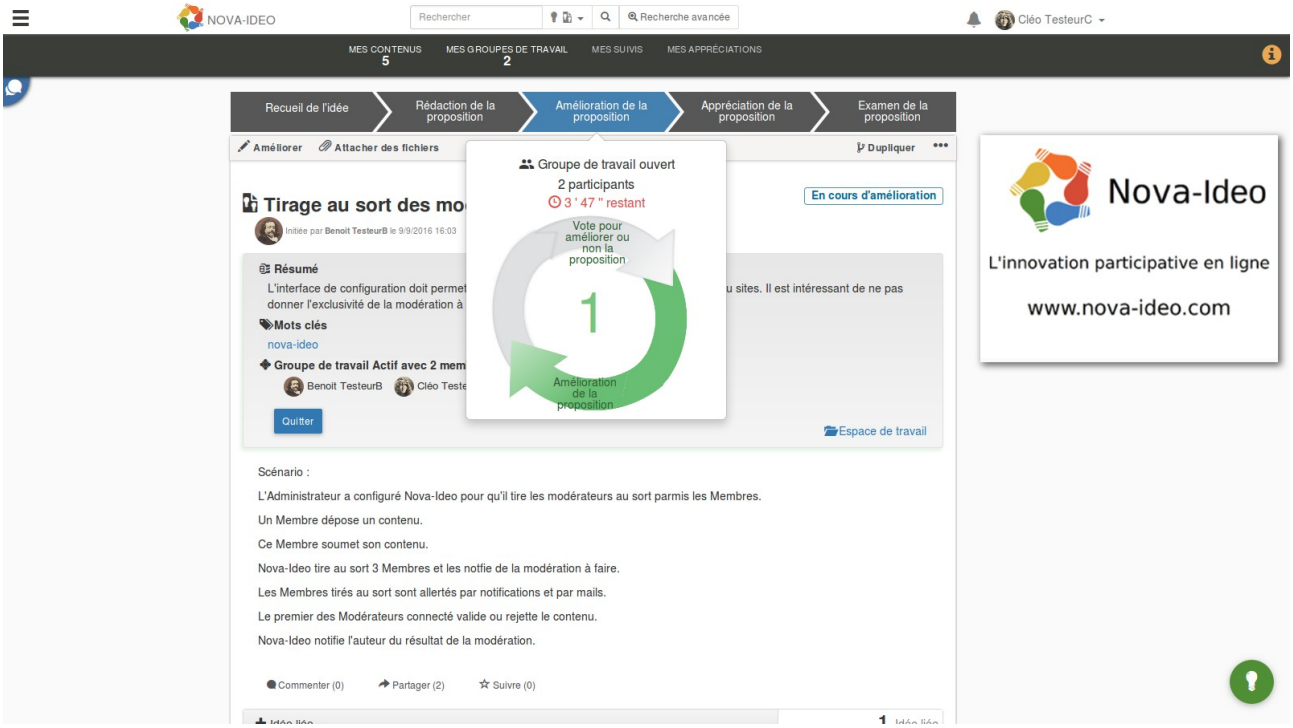
5) Coordonner (« correlation »)

C'est favoriser les regroupements d’informations selon des principes de continuité et de prolongement.

C'est établir des liens pertinents entre éléments.

Coordonner répond aux buts explicites des utilisateurs et à certains de leurs besoins latents afin de faciliter les trouvailles.

Exemple de projet Nova-Ideo.com



La Société est éditeur du projet libre de management des idées Nova-Ideo.

Nova-Ideo est la fusion de la boite à idées et du portail collaboratif.

Il répond aux nouveaux enjeux d'innovation participative, de démocratie participative et de co-création.

Il met en œuvre des processus cycliques d'amélioration collaborative des idées.

Nova-Ideo est Responsive Design et en conséquence s'adapte à la taille de l'écran.

Ainsi nous pouvons voir sur la capture précédente l'affichage de la page de saisie des idées sur un mobile.

Nova-Ideo est actuellement en cours de réécriture complète, la nouvelle interface graphique de la version 3 est entièrement réalisée sous React.

Toutes les sources du projet sont consultables sur https://github.com/ecreall/nova-ideo