# PROJET INFORMATIQUE
Bilan première séance

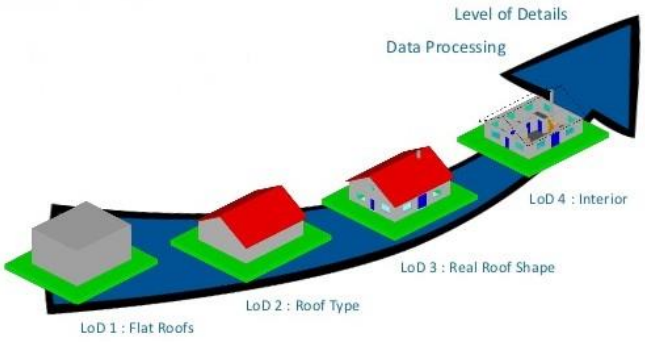
Sujet P05 : Développement d'une interface graphique pour améliorer des classifications de manière interactive
Commanditaires : Clément MALLET & Oussama ENNAFII – IGN | LaSTIG | équipe MATIS

## I. Contextualisation du sujet

Ce projet s'inscrit dans le cadre des recherches en reconstruction 3D des bâtiments à partir de MNS.

En effet, grâce au LiDAR et aux prises de vues aériennes et satellite, il est possible de réaliser des nuages de points, mais également des Modèles Numériques de Surface (MNS). Les MNS sont des modèles de surface topographique représentant le terrain mais aussi les constructions (toits, …), la végétation, … .

Pour reconstruire les bâtiments en 3D, on cherche à associer un modèle prédéfini à une entité, reconstituée par le MNS + l'emprise au sol du bâtiment (utilisation de la corrélation). Suivant le niveau de détail recherché, différents modèles sont possibles : Manhattan (= toits plats), Biplan (= toits pentus), Modélisation précise (à partir de la triangulation du MNS), … Plus le modèle est évolué, plus le niveau de détail est important mais plus la généralisation est compliquée et lourde à coder.




Cependant, aucune méthode de reconstruction n'est encore fiable à 100%. Ainsi, des erreurs sont relevés lors de la classification des bâtiments (sous-découpage des parcelles, erreur de pentes du toit, mauvaise forme, …).

## II. Analyse des besoins

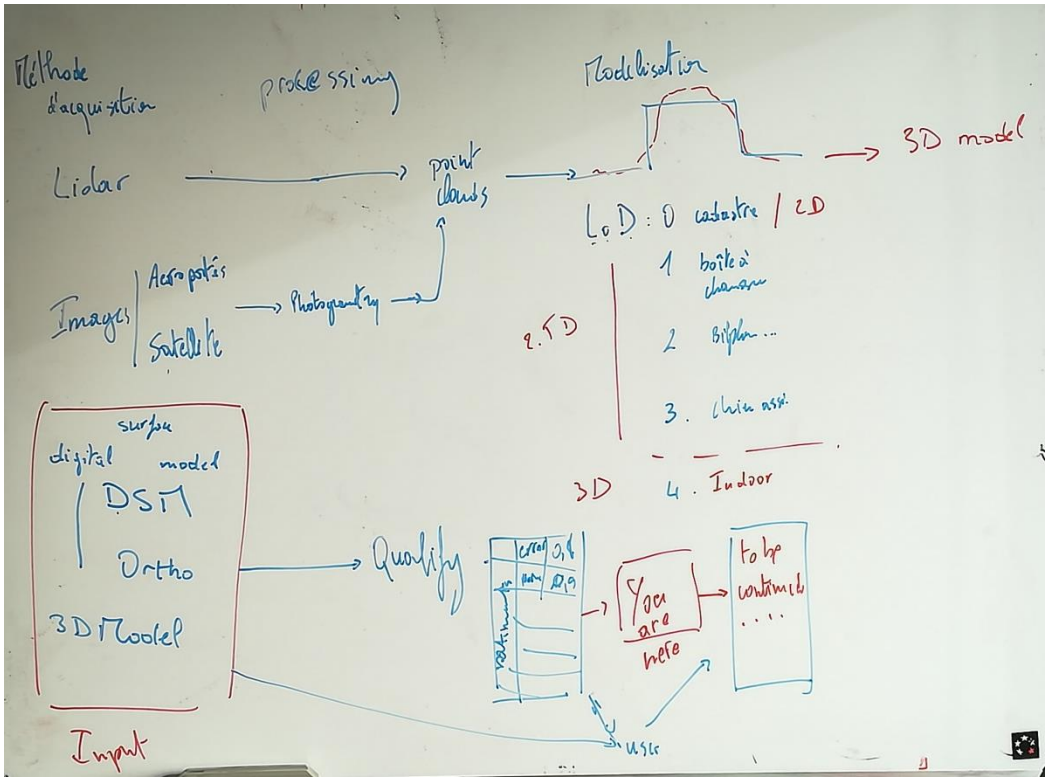



*Figure 1 – Organigramme général du projet*

L'équipe du MATIS s'intéresse principalement à la reconstruction des bâtiments en 2.5D. Cela signifie que les « irrégularités » des façades ne sont pas prises en compte (balcons, dévers et occlusions, …), et que l'on projette orthogonalement la toiture.

Selon les choix réalisés par l'utilisateur, la reconstruction 2.5D est réalisée et qualifiée. Cela signifie que chaque bâtiment modélisé est associé à une classe d'erreur (None, Building error, …). Un pourcentage est également renseigné et correspond à la probabilité d'appartenance à la classe.

Pour valider et améliorer cette classification, l'objectif est de faire intervenir l'utilisateur : on parle donc de classification supervisée. Pour faciliter l'échange avec l'utilisateur, on cherche à développer une interface graphique. Cette interface utiliserait les résultats de la modélisation et de la classification en affichant les emprises sur un fond cartographique. Sur une sélection d'entités, l'utilisateur pourrait confirmer ou non la classification, et reclasser les erreurs.

## III. Organisation générale et analyse (succincte) des données

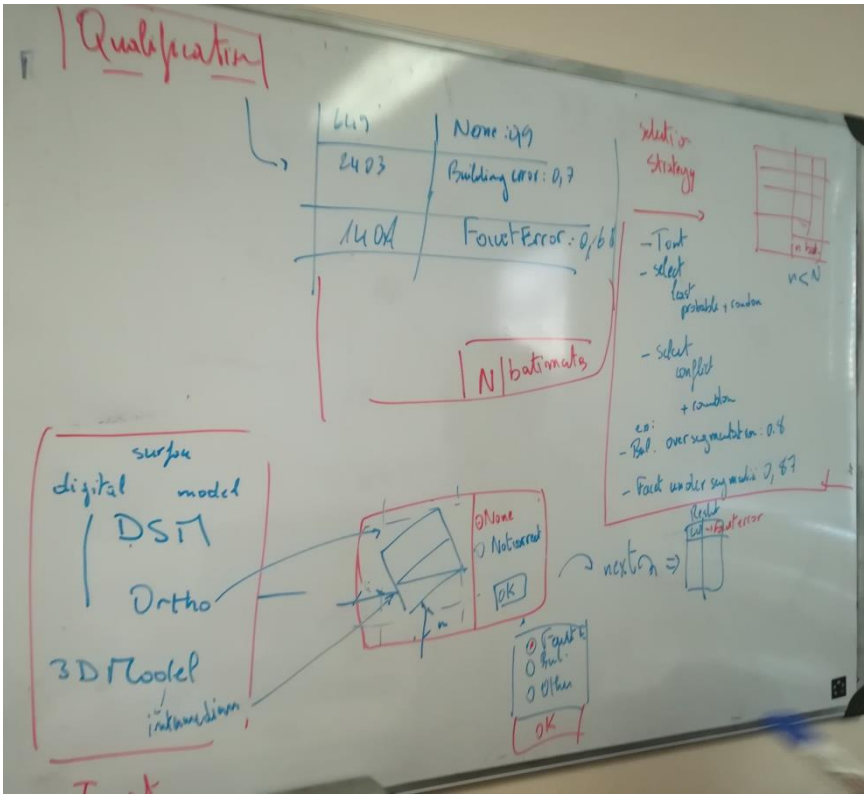



*Figure 2 - Organisation générale de la solution et données utilisées*

Données en entrée : liste de bâtiments et de leur type d'erreur + probabilité
= OUTPUT de la classification

Interface : affichage de l'emprise et du fond de carte (Ortho ou DSM) + fenêtre de choix (bon classement/mauvais classement) + (si besoin) fenêtre de reclassification (avec le choix des autres classes disponibles)

Données en sortie : liste des seuls bâtiments modifiés (ou liste originale modifiée) et leur type d'erreur + probabilité

Choix des données à faire traiter par l'utilisateur : plusieurs stratégies de sélection sont possibles :

- Montrer toutes les entités
- Sélectionner les entités présentant les plus petites probabilités + valeurs random
- Sélectionner les entités présentant des conflits de classification + valeurs random

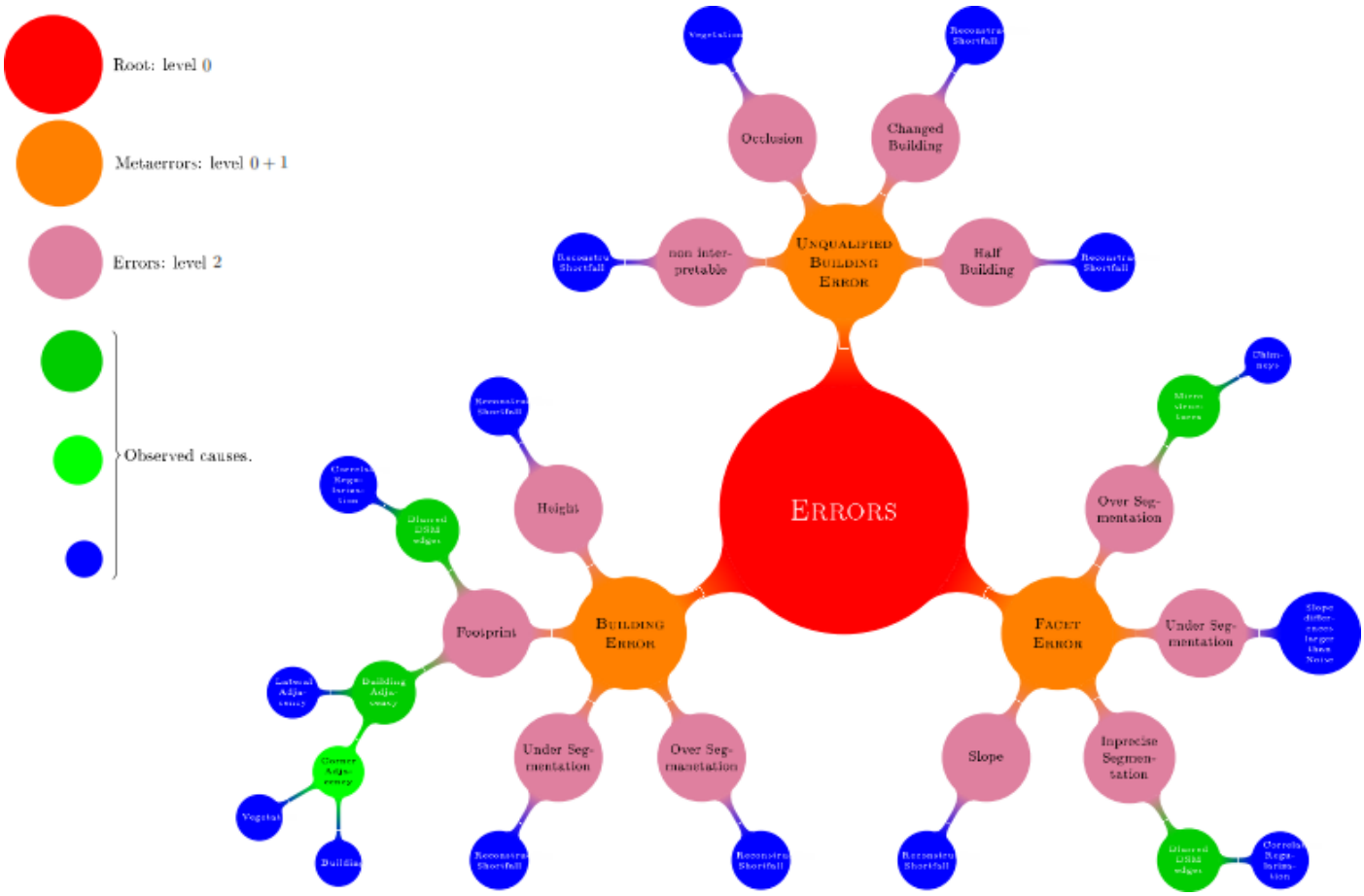



*Figure 3 - Types d'erreur rencontrée (= classes)*

## IV. Outils utilisés

1. Programmation orientée objet

https://openclassrooms.com/courses/apprenez-a-programmer-en-python/premiere-approche-des-classes
http://www.courspython.com/classes-et-objets.html
(…)

Mots clés : classes, objets, attributs, méthodes, fonction, héritage, héritage en diamant
Exemples : animal (chien/chat), Personnage (Mario 1-2-3/Luigi), Strategie_selection (tous/faible_proba, …)

2. PyQt

https://www.youtube.com/watch?v=JBME1ZyHiP8&list=PLQVvvaa0QuDdVpDFNq4FwY9APZPGSUyR4

Mots clés : interface, PyQt

3. Git

Git est un logiciel permettant de gérer les versions de programmes de manière décentralisé : cela permet d'avoir un code dupliqué sur un serveur distinct, et de pouvoir revenir en arrière lors de la programmation.

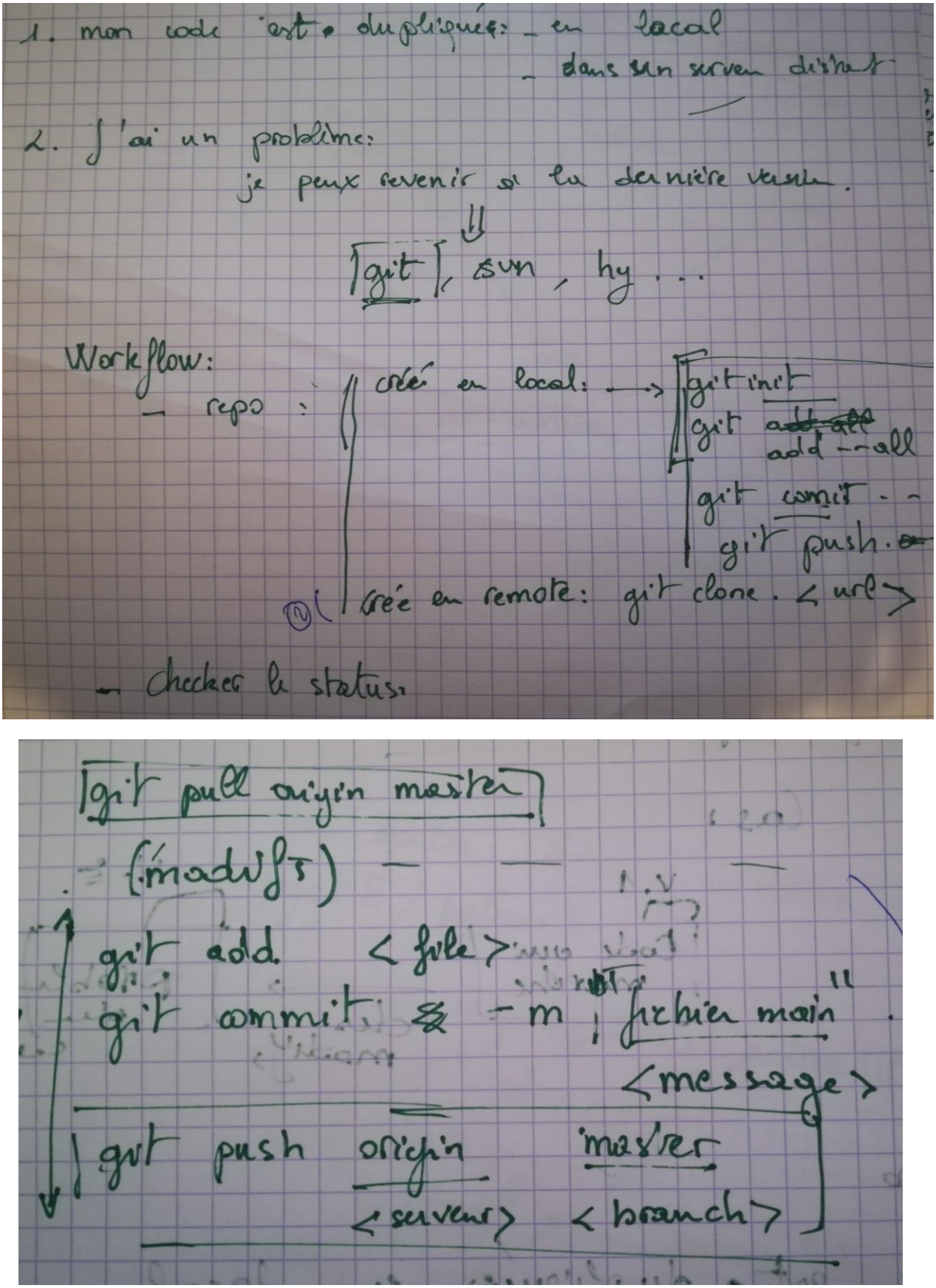



*Figure 4 – Organisation et méthode d'utilisation de Git*

## V. **Premières réflexions** *(organisation du travail, organisation du code pour l'interface, …)*

### 1. Organisation du code de l'interface

Pour faciliter l'écriture et la compréhension du code de l'interface, on utilisera la méthode Model/Vue/Controller :

- Le fichier « Model » contiendra les données à afficher en entrée et les résultats attendus en sortie
- Le fichier « Vue » contiendra le code de l'interface graphique
- Le fichier « Controller » contiendra les définitions informatiques des actions effectuées par l'utilisateur dans l'interface (ex : action à exécuter lors de la validation de la classification, …). Il va alors modifier les données contenues dans « Model », ce qui va entrainer un changement dans la « Vue ».

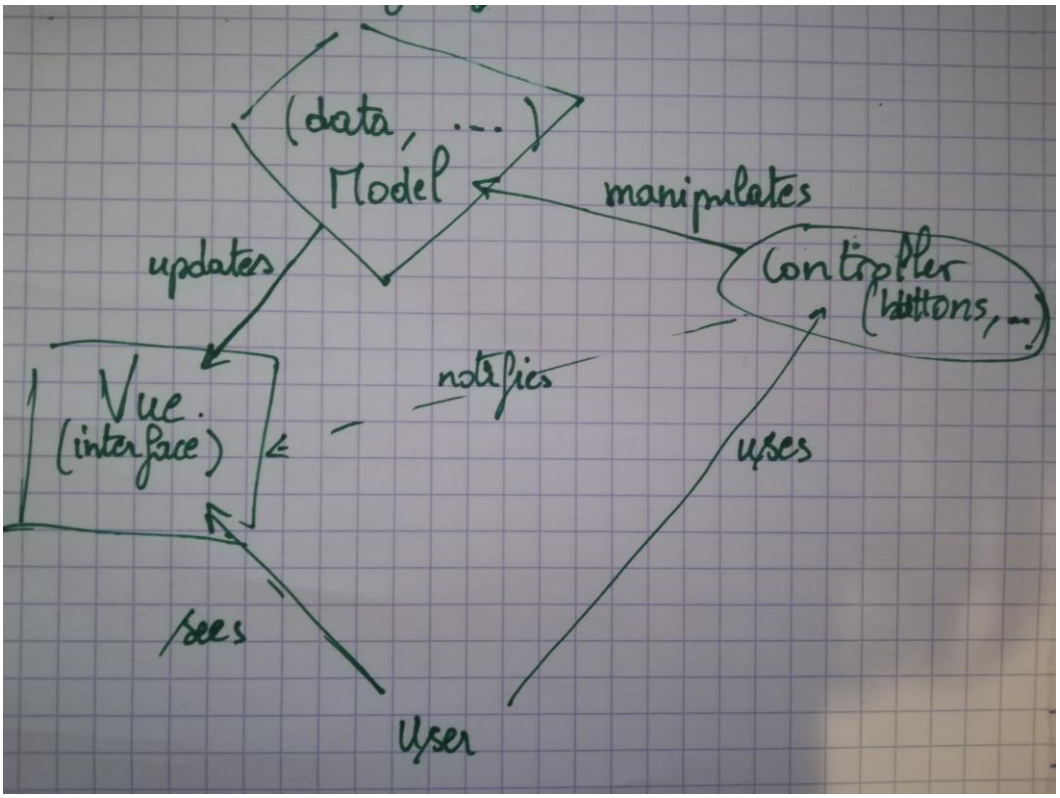



*Figure 5 – Organisation de la méthode MVC*